Solving Large-Scale Sparse Semidefinite Programs for Combinatorial Optimization

Article *in* SIAM Journal on Optimization · February 1970

DOI: 10.1137/S1052623497328008 · Source: CiteSeer

Yinyu Ye
Stanford University
396 PUBLICATIONS **23,343** CITATIONS

Solving Large-Scale Sparse Semidefinite Programs for Combinatorial Optimization[*]

Steven J. Benson
Applied Mathematics and Computational Sciences
The University of Iowa
Iowa City, Iowa 52242, U.S.A.

Yinyu Ye
Department of Management Sciences
The University of Iowa
Iowa City, Iowa 52242, U.S.A.

Xiong Zhang[†]
School of Mechanical Engineering
Huazhong University of Science and Technology
Wuhan, Hubei, China.

September 15, 1997 (Revised May 10, 1998)

Abstract

We present a dual-scaling interior-point algorithm and show how it exploits the structure and sparsity of some large scale problems. We solve the positive semidefinite relaxation of combinatorial and quadratic optimization problems subject to boolean constraints. We report the first computational results of interior-point algorithms for approximating the maximum cut semidefinite programs with dimension up-to 3000.

Key words. Semidefinite programming, dual potential reduction algorithm, maximum cut problem.

[*]This work is partially supported by NSF grants DMI-9522507 and DMS-9703490.

[†]This author is currently visiting the Computational Optimization Lab, Department of Management Sciences, The University of Iowa, Iowa City, Iowa 52242, U.S.A.

1 Introduction

Recently, there were several theoretical results on the effectiveness of approximating combinatorial and nonconvex quadratic optimization problems by using semidefinite programming (see, e.g., Goemans and Williamson [11], Nesterov [24], and Ye [33]). These results raise the hope that some hard optimization problems could be tackled by solving large-scale semidefinite relaxation programs. The positive semidefinite relaxation was early considered by Lovász [18] and Shor [29], and the field had received further contributions by many other researchers (e.g., see Lovász and Shrijver [19], Alizadeh [2], Sherali and Adams [28], and references therein).

The approximate solution to the quadratic optimization problem is obtained by solving a semidefinite relaxation, i.e., a semidefinite program (SDP) of the form

$$\text{(SDP)} \qquad \begin{array}{ll} \text{Minimize} & C \bullet X \\ \\ \text{Subject to} & A_i \bullet X = b_i, \ i = 1, \ldots, m, \\ \\ & X \succeq 0. \end{array} \tag{1}$$

Here, the given matrices $C, A_i \in \mathcal{S}^n$, the set of n-dimensional symmetric matrices; vector $b \in \Re^m$; and unknown $X \in \mathcal{S}^n$. Furthermore, the A_i's are linearly independent, meaning that $\sum_{i=1}^m y_i A_i = 0$ implies $y_1 = \ldots = y_m = 0$; $C \bullet X = \text{tr } C^T X = \sum_{jk} C_{jk} X_{jk}$; and $X \succeq 0$ means that X is positive semidefinite.

In this paper, one additional assumption will be made: the constraint matrices have a rank one form, $A_i = a_i a_i^T$, $a_i \in \Re^n$. This structure arises in many large scale problems and results in considerable simplifications.

The dual of (SDP) can be written as:

$$\text{(DSDP)} \qquad \begin{array}{ll} \text{Maximize} & b^T y \\ \\ \text{Subject to} & \sum_{i=1}^m y_i A_i + S = C, \quad S \succeq 0, \end{array} \tag{2}$$

where y_i, $i = 1, \ldots, m$ are scalar variables.

We have the following well known duality theorem [25]:

Theorem 1 *(Strong Duality) Provided that (SDP) and (DSDP) are both feasible and there is a strictly interior point to either (SDP) or (DSDP), there is no duality gap.*

Thus, if both (SDP) and (DSDP) are well behaved or a primal and dual optimal solution pair (X^*) and (y^*, S^*) exists, then $C \bullet X^* = b^T y^*$.

A well behaved pair of semidefinite programs can be solved in "polynomial time". There are actually several interior-point polynomial algorithms. One is the primal-scaling algorithm (Nesterov and Nemirovskii [25], Alizadeh [2], Vandenberghe and Boyd [31], and Ye [34]), which is the analogue of the primal path-following and potential reduction algorithm for linear programming. This algorithm uses only X to generate the iterate direction. In other words,

$$\begin{pmatrix} X^{k+1} \\ S^{k+1} \end{pmatrix} = F_p(X^k),$$

where F_p is the primal algorithm iterative mapping.

Another is the dual-scaling algorithm (Vandenberghe and Boyd [31], Anstreicher and Fampa [4], and Ye [34]), which is the analogue of the dual path-following and potential reduction algorithm for linear programming. The dual-scaling algorithm uses only S to generate the new iterate:

$$\begin{pmatrix} X^{k+1} \\ S^{k+1} \end{pmatrix} = F_d(S^k),$$

where F_d is the dual algorithm iterative mapping.

The third is the primal-dual scaling algorithm which uses both X and S to generate the new iterate (see Todd [30] and references therein):

$$\begin{pmatrix} X^{k+1} \\ S^{k+1} \end{pmatrix} = F_{pd}(X^k, S^k),$$

where F_{pd} is the primal-dual algorithm iterative mapping.

All these algorithms generate primal and dual iterates simultaneously, and possess $O(\sqrt{n}\ln(1/\epsilon))$ iteration complexity to yield the duality gap accuracy ϵ. Other scaling algorithms have been proposed in the past. For example, an SDP equivalent of Dikin's affine-scaling algorithm could be very fast. However this algorithm may not even converge. Muramatsu [22] and Muramatsu and Vanderbei [23] showed an example in which these affine scaling algorithms will not converge to an optimal answer.

There are also quite a few computational results and implementations of these interior algorithms, see Anstreicher and Fampa [4], Alizadeh, Haeberly, and Overton [3], Fujisawa, Kojima and Nakata [8], Helmberg, Rendl, Vanderbei, and Wolkowicz [13], Karisch, Rendl, and Clausen [14], Vandenberghe and Boyd [31], Wolkowicz and Zhao [32], Zhao, Karisch, Rendl, and Wolkowicz [35][36]. To the best of our knowledge, the largest problem that could be solved was at $n = 900$ from their reports. (After the initial version of this paper was submitted, one more implementation came out: Fujisawa, Fukuda, Kojima and Nakata [10] reported that they could solve a maximum cut semidefinite program with $n = 1250$, using a powerful work-station.)

The practical winner of solving semidefinite programs was Helmberg and Rendl [12], an implementation of a non-interior-point algorithm called the bundle method. They reported the solutions of a set of dual semidefinite programs with n up-to 3000. The bundle method enables them to take advantage of the sparsity structure of these problems. The (minor) weakness of their method is that the method does not simultaneously solve the primal problem and cannot guarantee or verify the optimality accuracy at its termination, and it is not a polynomial time algorithm.

Therefore, an open question is how to exploit the sparsity structure by polynomial interior-point algorithms so that they can solve large-scale problems in practice. In this paper we try to answer this question. We show that many large-scale semidefinite programs arisen from combinatorial and quadratic optimization have features which make the dual-scaling interior-point algorithm the most suitable choice:

1. The computational cost of each iteration in the dual algorithm is less than the cost of a primal-dual iterations. Although primal-dual algorithms may possess superlinear convergence, the approximation problems under consideration require less accuracy than some other applications. Therefore, the

superlinear convergence exhibited by primal-dual algorithms may not be utilized in our applications. The dual-scaling algorithm has been shown to perform equally well when only a lower precision answer is required, see, e.g., Adler et al. [1] and Vandenberghe and Boyd [31].

2. In most combinatorial applications we need only a lower bound for the optimal objective value of (SDP). Solving (DSDP) alone would be sufficient to provide such a lower bound. Thus, we may not need any X at all. Even if an optimal primal solution is necessary, our dual-scaling algorithm can generate an optimal X at the termination of the algorithm with little additional cost.

3. For large scale problems, S tends to be very sparse and structured since it is the linear combination of C and the A_i's. This sparsity allows considerable savings in both memory and computation time. The primal matrix, X, may be much less sparse and have a structure unknown beforehand. Consequently, primal and primal-dual algorithms may not fully exploit the sparseness and structure of the data.

These problems include the semidefinite relaxations of the graph-partition problem, the box-constrained quadratic optimization problem, the $0-1$ integer set covering problem, etc. We will use the maximum cut problem to illustrate our points later, where we report our computational result, using a PC machine, on solving the maximum cut semidefinite relaxations of the Helmberg and Rendl set of graph problems for n up-to 3000.

2 Dual Scaling Algorithm

The dual-scaling algorithm, which is a modification of the dual-scaling linear programming algorithm, reduces the Tanabe-Todd-Ye primal-dual potential function

$$\Psi(X,S) = \rho \ln(X \bullet S) - \ln\det X - \ln\det S.$$

The first term decreases the duality gap, while the second and third terms keep X and S in the interior of the positive semidefinite matrix cone. When $\rho > n$, the infimum of the potential function occurs at an optimal solution. Also note that, using the arithmetic-geometric mean inequality, we have (e.g., [34])

$$n \ln(X \bullet S) - \ln\det X - \ln\det S \geq n \ln n.$$

The algorithm, along with other SDP algorithms, is described in Ye [34], so that we shall use notations defined there.

Let operator $\mathcal{A}(X) : \mathcal{S}^n \to \Re^m$ be defined as

$$\mathcal{A}(X) = \begin{pmatrix} A_1 \bullet X \\ A_2 \bullet X \\ \vdots \\ A_m \bullet X \end{pmatrix}.$$

Since $\mathcal{A}(X)^T y = \sum_{i=1}^m y_i (A_i \bullet X) = (\sum_{i=1}^m y_i A_i) \bullet X$, the adjoint operator $\mathcal{A}^T : \Re^m \to \mathcal{S}^n$ is

$$\mathcal{A}^T(y) = \sum_{i=1}^m y_i A_i.$$

Let $\bar{z} = C \bullet X$ for some feasible X and consider the dual potential function

$$\psi(y, \bar{z}) = \rho \ln(\bar{z} - b^T y) - \ln \det S.$$

Its gradient is

$$\nabla \psi(y, \bar{z}) = -\frac{\rho}{\bar{z} - b^T y} b + \mathcal{A}(S^{-1}). \tag{3}$$

To estimate the reduction in the potential function from a current iterate $(y^k, \bar{z}^k)$ to the next, we will use a lemma from linear programming that can be found in [34] and is essentially do to Karmarkar [15].

Lemma 1 *Let $X \in \mathcal{S}^n$ and $\|X - I\|_\infty < 1$. Then*

$$\ln \det(X) \geq \text{tr } (X - I) - \frac{\|X - I\|}{2(1 - \|X - I\|_\infty)}$$

where I denotes the identity matrix and $\|\cdot\|$ denotes the Frobenius norm, and

$$\|A\|_\infty = \max_{i=1,\ldots,n} \{|\lambda_i(A)|\} \leq \|A\|$$

and $\lambda_i(A)$ is the ith eigenvalue of $A \in \mathcal{S}^n$.

Proof. We have $0 < \lambda_i := \lambda_i(X) < 2$ for all $i = 1, \ldots, n$, since $\|X - I\|_\infty < 1$. Moreover,

$$\begin{aligned}
\ln \lambda_i &= \ln(1 + \lambda_i - 1) \\
&= (\lambda_i - 1) - \tfrac{(\lambda_i - 1)^2}{2} + \tfrac{(\lambda_i - 1)^3}{3} - \tfrac{(\lambda_i - 1)^4}{4} + \ldots \\
&\geq (\lambda_i - 1) - \tfrac{(\lambda_i - 1)^2}{2}(1 + |\lambda_i - 1| + |\lambda_i - 1|^2 + \ldots) \\
&= (\lambda_i - 1) - \tfrac{(\lambda_i - 1)^2}{2(1 - |\lambda_i - 1|)} \geq (\lambda_i - 1) - \tfrac{(\lambda_i - 1)^2}{2(1 - \|X - I\|_\infty)}.
\end{aligned}$$

Summing the inequality over i, we have the desired result. ■

For any given y and $S = C - \mathcal{A}^T(y)$ such that $S \succ 0$ and $\|(S^k)^{-.5} \left(\mathcal{A}^T(y - y^k)\right) (S^k)^{-.5}\| < 1$, using the above lemma, the concavity of the first term in the potential function, and the fact that

$$(S^k)^{-.5} S (S^k)^{-.5} - I = (S^k)^{-.5}(S - S^k)(S^k)^{-.5} = -(S^k)^{-.5} \left(\mathcal{A}^T(y - y^k)\right) (S^k)^{-.5},$$

we establish an over-estimator for the potential reduction:

$$\begin{aligned}
&\psi(y, \bar{z}^k) - \psi(y^k, \bar{z}^k) \\
&= \rho \ln(\bar{z}^k - b^T y) - \rho \ln(\bar{z}^k - b^T y^k) - \ln \det((S^k)^{-.5} S (S^k)^{-.5}) \\
&\leq \rho \ln(\bar{z}^k - b^T y) - \rho \ln(\bar{z}^k - b^T y^k) - \text{tr } ((S^k)^{-.5} S (S^k)^{-.5} - I) + \tfrac{\|(S^k)^{-.5}\left(\mathcal{A}^T(y - y^k)\right)(S^k)^{-.5}\|}{2(1 - \|(S^k)^{-.5}\left(\mathcal{A}^T(y - y^k)\right)(S^k)^{-.5}\|_\infty)} \\
&= \rho \ln(\bar{z}^k - b^T y) - \rho \ln(\bar{z}^k - b^T y^k) + \mathcal{A}((S^k)^{-1})^T (y - y^k) + \tfrac{\|(S^k)^{-.5}\left(\mathcal{A}^T(y - y^k)\right)(S^k)^{-.5}\|}{2(1 - \|(S^k)^{-.5}\left(\mathcal{A}^T(y - y^k)\right)(S^k)^{-.5}\|_\infty)} \\
&\leq \nabla \psi(y^k, \bar{z}^k)^T (y - y^k) + \tfrac{\|(S^k)^{-.5}\left(\mathcal{A}^T(y - y^k)\right)(S^k)^{-.5}\|}{2(1 - \|(S^k)^{-.5}\left(\mathcal{A}^T(y - y^k)\right)(S^k)^{-.5}\|_\infty)}.
\end{aligned} \tag{4}$$

Therefore, beginning with a strictly feasible dual point (y^k, S^k) and upper bound $\bar{z}^k$, each iteration solves the following problem.

$$\begin{array}{ll} \text{Minimize} & \nabla\psi^T(y^k, \bar{z}^k)(y - y^k) \\ \text{Subject to} & \|(S^k)^{-.5}\left(\mathcal{A}^T(y - y^k)\right)(S^k)^{-.5}\| \leq \alpha, \end{array} \tag{5}$$

where α is a positive constant less than 1. For simplicity, in what follows we let

$$\Delta^k = \bar{z}^k - b^T y^k.$$

The first order Karusch-Kuhn-Tucker conditions state that the minimum point, y^{k+1}, of this convex problem satisfies

$$M^k(y^{k+1} - y^k) + \beta\nabla\psi(y^k, \bar{z}^k) = M^k(y^{k+1} - y^k) + \beta(-\frac{\rho}{\bar{z}^k - b^T y^k} b + \mathcal{A}((S^k)^{-1})) = 0 \tag{6}$$

for a positive value of β, where

$$M^k = \begin{pmatrix} A_1(S^k)^{-1} \bullet (S^k)^{-1}A_1 & \cdots & A_1(S^k)^{-1} \bullet (S^k)^{-1}A_m \\ \vdots & \ddots & \vdots \\ A_m(S^k)^{-1} \bullet (S^k)^{-1}A_1 & \cdots & A_m(S^k)^{-1} \bullet (S^k)^{-1}A_m \end{pmatrix} \quad \text{and} \quad \mathcal{A}((S^k)^{-1}) = \begin{pmatrix} A_1 \bullet (S^k)^{-1} \\ \vdots \\ A_m \bullet (S^k)^{-1} \end{pmatrix}.$$

The matrix M^k is a Gram matrix and is positive definite when $S^k \succ 0$ and the A_is are linearly independent. In this paper, it will sometimes be referred to as M.

Using the ellipsoidal constraint, the minimal solution, y^{k+1}, of (5) is given by

$$y^{k+1} - y^k = \beta d(\bar{z}^k)_y \tag{7}$$

where

$$\begin{array}{r} d(\bar{z}^k)_y = -(M^k)^{-1}\nabla\psi(y^k, \bar{z}^k) \\ \beta = \frac{\alpha}{\sqrt{-\nabla\psi^T(y^k, \bar{z}^k) d(\bar{z}^k)_y}}. \end{array} \tag{8}$$

Unlike linear programming, positive semidefinite programming requires a significant amount of the time to compute the system of equations that determines the step direction. For arbitrary symmetric matrices A_i, Monteiro and Zanjácomo [20] demonstrated an efficient implementation of several primal-dual step directions. The AHO direction [3] can be computed in $5nm^3 + n^2m^2 + O(\max\{m,n\}^3)$ operations. The HRVW/KSH/M direction [13][16][21] uses $2nm^3 + n^2m^2 + O(\max\{m,n\}^3)$ operations, and the NT direction [26] uses $nm^3 + n^2m^2/2 + O(\max\{m,n\}^3)$ operations. The complexity of computing the matrix is a full order of magnitude higher than any other step of the algorithm. Fujisawa, Kojima and Nakata [8] explored another technique for computing primal-dual step directions that exploit the sparsity of the data matrices. However, it is our belief that only the dual-scaling algorithm can fully exploit the structure and sparsity of many problems, as explained below.

Generally, $M^k_{ij} = A_i(S^k)^{-1} \bullet (S^k)^{-1}A_j$. When $A_i = a_i a_i^T$, the Gram matrix can be rewritten in the form

$$M^k = \begin{pmatrix} (a_1^T(S^k)^{-1}a_1)^2 & \cdots & (a_1^T(S^k)^{-1}a_m)^2 \\ \vdots & \ddots & \vdots \\ (a_m^T(S^k)^{-1}a_1)^2 & \cdots & (a_m^T(S^k)^{-1}a_m)^2 \end{pmatrix} \quad \text{and} \quad \mathcal{A}((S^k)^{-1}) = \begin{pmatrix} a_1^T(S^k)^{-1}a_1 \\ \vdots \\ a_m^T(S^k)^{-1}a_m \end{pmatrix}. \tag{9}$$

This matrix can be computed very quickly without computing, or saving, $(S^k)^{-1}$. Instead, S^k can be factored, and then we can use

Algorithm M: To compute M^k and $\mathcal{A}((S^k)^{-1})$, factor $S^k = L^k(L^k)^T$ and do the following:

```
For $i = 1 : m$;
    Solve $L^k w_i = a_i$;
    $\mathcal{A}((S^k)^{-1})_i = w_i^T w_i$ and $M^k_{ii} = (\mathcal{A}((S^k)^{-1})_i)^2$;
    For $j = 1 : i-1$;    $M^k_{ij} = (w_i^T w_j)^2$;    end;
end.
```

Solving each of the m systems of equations uses $n^2 + O(n)$ floating point operations. Since there are $m(m+1)/2$ vector multiplications, Algorithm M uses $nm^2 + n^2m + O(nm)$ operations after factoring S^k. Note that these operations can be significantly reduced if S^k is structured and sparse. In applications like the maximum cut problem, discussed in Section 3, the matrix S^k is indeed very sparse while its inverse is usually dense, so working with S^k is faster than working with its inverse. Using matrices of the form $A_i = a_i a_i^T$ also reduces the complexity of primal-dual algorithms by a factor of n, but even the quickest direction to compute takes about twice as long as our dual-scaling direction. Furthermore, they all need to handle dense X.

Algorithm M needs to store all vectors $w_1, ..., w_m$ and they are generally dense. To save storage and exploit the sparsity of $a_i, ..., a_m$, an alternative algorithm is

Algorithm M': To compute M^k and $\mathcal{A}((S^k)^{-1})$, factor $S^k = L^k(L^k)^T$ and do the following:

```
For $i = 1 : m$;
    Solve $S^k w_i = a_i$;
    $\mathcal{A}((S^k)^{-1})_i = w_i^T a_i$ and $M^k_{ii} = (\mathcal{A}((S^k)^{-1})_i)^2$;
    For $j = i+1 : m$;    $M^k_{ij} = (w_i^T a_j)^2$;    end;
end.
```

Algorithm M' does not need to store w_j but uses one more back-solve for w_i.

To find a feasible primal point X, we solve the least squares problem

$$\begin{array}{ll} \text{Minimize} & \|(S^k)^{.5} X (S^k)^{.5} - \frac{\Delta^k}{\rho} I\| \\ \text{Subject to} & \mathcal{A}(X) = b. \end{array} \tag{10}$$

This problem looks for a matrix $X(\bar{z}^k)$ near the central path. Larger values of ρ generally give a lower objective value, but provide a solution matrix that is not positive definite more frequently. The answer to (10) is a by-product of computing (8), given explicitly by

$$X(\bar{z}^k) = \frac{\Delta^k}{\rho}(S^k)^{-1}\left(\mathcal{A}^T(d(\bar{z}^k)_y) + S^k\right)(S^k)^{-1}. \tag{11}$$

Creating the primal matrix may be costly. However, the evaluation of the primal objective value

$C \bullet X(\bar{z}^k)$ requires drastically less work.

$$\begin{aligned} C \bullet X(\bar{z}^k) &= b^T y^k + X(\bar{z}^k) \bullet S^k \\ &= b^T y^k + \text{tr } \left(\tfrac{\Delta^k}{\rho} (S^k)^{-1} \left(\mathcal{A}^T (d(\bar{z}^k)_y) + S^k \right) (S^k)^{-1} S^k \right) \\ &= b^T y^k + \tfrac{\Delta^k}{\rho} \text{tr } \left((S^k)^{-1} \mathcal{A}^T (d(\bar{z}^k)_y) + I \right) \\ &= b^T y^k + \tfrac{\Delta^k}{\rho} \left(d(\bar{z}^k)_y^T \mathcal{A}((S^k)^{-1}) + n \right). \end{aligned}$$

Since the vectors $\mathcal{A}((S^k)^{-1})$ and $d(\bar{z}^k)_y$ were previously found in calculating the dual step direction, the cost of computing a primal objective value is the cost of a vector dot product! The matrix $X(\bar{z}^k)$ never gets computed during the iterative process, saving time and memory. On the other hand, primal-dual methods require far more resources to compute the primal variables X.

Defining

$$P(\bar{z}^k) = \frac{\rho}{\Delta^k} (S^k)^{.5} X(\bar{z}^k) (S^k)^{.5} - I, \tag{12}$$

we have the following lemma:

Lemma 2 *Let* $\mu^k = \frac{\Delta^k}{n} = \frac{\bar{z}^k - b^T y^k}{n}$, $\mu = \frac{X(\bar{z}^k) \bullet S^k}{n} = \frac{C \bullet X(\bar{z}^k) - b^T y^k}{n}$, $\rho \geq n + \sqrt{n}$, *and* $\alpha < 1$. *If*

$$\|P(\bar{z}^k)\| < min(\alpha \sqrt{\frac{n}{n + \alpha^2}}, 1 - \alpha), \tag{13}$$

then the following three inequalities hold:

1. $X(\bar{z}^k) \succ 0$;
2. $\|(S^k)^{.5} X(\bar{z}^k)(S^k)^{.5} - \mu I\| \leq \alpha \mu$;
3. $\mu \leq (1 - .5\alpha/\sqrt{n})\mu^k$.

Proof. The proofs are by contradiction. If the first inequality is false, then $(S^k)^{.5} X(\bar{z}^k)(S^k)^{.5}$ has at least one nonpositive eigenvalue, which by (12) implies that $\|P(\bar{z}^k)\| \geq 1$.

If the second does not hold, then

$$\begin{aligned} \|P(\bar{z}^k)\|^2 &= \|\tfrac{\rho}{n\mu^k} (S^k)^{.5} X(\bar{z}^k)(S^k)^{.5} - I\|^2 \\ &= \|\tfrac{\rho}{n\mu^k} (S^k)^{.5} X(\bar{z}^k)(S^k)^{.5} - \tfrac{\rho\mu}{n\mu^k} I + \tfrac{\rho\mu}{n\mu^k} I - I\|^2 \\ &= \|\tfrac{\rho}{n\mu^k} (S^k)^{.5} X(\bar{z}^k)(S^k)^{.5} - \tfrac{\rho\mu}{n\mu^k} I\|^2 + \|\tfrac{\rho\mu}{n\mu^k} I - I\|^2 \\ &> \left(\tfrac{\rho\mu}{n\mu^k}\right)^2 \alpha^2 + \left(\tfrac{\rho\mu}{n\mu^k} - 1\right)^2 n \\ &\geq \alpha^2 \left(\tfrac{n}{n+\alpha^2}\right) \end{aligned}$$

where the last inequality is true because the quadratic term has a minimum at $\frac{\rho\mu}{n\mu^k} = \frac{n}{n+\alpha^2}$.

If the third inequality does not hold, then

$$\frac{\rho\mu}{n\mu^k} > \left(1 + \frac{1}{\sqrt{n}}\right)\left(1 - \frac{.5\alpha}{\sqrt{n}}\right) \geq 1.$$

which leads to

$$\begin{aligned}
\|P(\bar{z}^k)\|^2 &\geq \left(\frac{\rho\mu}{n\mu^k}-1\right)^2 n \\
&\geq \left(\left(1+\frac{1}{\sqrt{n}}\right)\left(1-\frac{\alpha}{2\sqrt{n}}\right)-1\right)^2 n \\
&= \left(1-\frac{\alpha}{2}-\frac{\alpha}{2\sqrt{n}}\right)^2 \\
&\geq (1-\alpha)^2.
\end{aligned}$$

■

Focusing on the expression $P(\bar{z}^k)$, it can be rewritten as

$$\begin{aligned}
P(\bar{z}^k) &= \tfrac{\rho}{\Delta^k}(S^k)^{.5}\left(\tfrac{\Delta^k}{\rho}(S^k)^{-1}\left(\mathcal{A}^T(d(\bar{z}^k)_y)+S^k\right)(S^k)^{-1}\right)(S^k)^{.5}-I \\
&= (S^k)^{-.5}\mathcal{A}^T\left(d(\bar{z}^k)_y\right)(S^k)^{-.5} \\
&= (S^k)^{-.5}\mathcal{A}^T\left(\tfrac{y^{k+1}-y^k}{\beta}\right)(S^k)^{-.5}
\end{aligned}$$

which by (7) and (8), makes

$$\nabla\psi^T(y^k,\bar{z}^k)d(\bar{z}^k)_y = -\|P(\bar{z}^k)\|^2 \tag{14}$$

and

$$\nabla\psi^T(y^k,\bar{z}^k)(y^{k+1}-y^k) = -\alpha\|P(\bar{z}^k)\|. \tag{15}$$

Updating the dual variables according to

$$y^{k+1} = y^k+\beta d(\bar{z})_y = y^k+\frac{\alpha}{\|P(\bar{z}^{k+1})\|}d(\bar{z})_y \qquad \text{and} \qquad S^{k+1} = C-\mathcal{A}^T(y^{k+1}) \tag{16}$$

assures the positive definiteness of S^{k+1} when $\alpha < 1$, which implies that they are feasible. Using (15) and (4), the reduction in the potential function satisfies the inequality

$$\psi(y^{k+1},\bar{z}^k)-\psi(y^k,\bar{z}^k) \leq -\alpha\|P(\bar{z}^k)\|+\frac{\alpha^2}{2(1-\alpha)}. \tag{17}$$

The theoretical algorithm can be stated as follows.

DUAL ALGORITHM. Given an upper bound $\bar{z}^0$ and a dual point (y^0, S^0) such that $S^0 = C-\mathcal{A}^Ty^0 \succ 0$, set $k=0$, $\rho > n+\sqrt{n}$, $\alpha \in (0,1)$, and do the following:

while $\bar{z}^k - b^Ty^k \geq \epsilon$ **do**

begin

1. Compute $\mathcal{A}((S^k)^{-1})$ and the Gram matrix M^k (9) using Algorithm M or M'.
2. Solve (8) for the dual step direction $d(\bar{z}^k)_y$.
3. Calculate $\|P(\bar{z}^k)\|$ using (14).

4. **If** (13) is true, **then** $X^{k+1} = X(\bar{z}^k)$, $\bar{z}^{k+1} = C \bullet X^{k+1}$, and $(y^{k+1}, S^{k+1}) = (y^k, S^k)$;
   else $y^{k+1} = y^k + \frac{\alpha}{\|P(\bar{z}^k)\|} d(\bar{z}^{k+1})_y$, $S^{k+1} = C - \mathcal{A}^T(y^{k+1})$, $X^{k+1} = X^k$, and $\bar{z}^{k+1} = \bar{z}^k$.
   endif
5. $k := k + 1$.

end

We can derive the following potential reduction theorem based on the above lemma:

Theorem 2

$$\Psi(X^{k+1}, S^{k+1}) \leq \Psi(X^k, S^k) - \delta$$

where $\delta > 1/50$ for a suitable α.

Proof.

$$\Psi(X^{k+1}, S^{k+1}) - \Psi(X^k, S^k) = \left(\Psi(X^{k+1}, S^{k+1}) - \Psi(X^{k+1}, S^k)\right) + \left(\Psi(X^{k+1}, S^k) - \Psi(X^k, S^k)\right).$$

In each iteration, one of the differences is zero. If $\|P(\bar{z}^k)\|$ does not satisfy (13), the dual variables get updated and (17) shows sufficient improvement in the potential function when $\alpha = 0.4$.

On the other hand, if the primal matrix gets updated, then using Lemma 1 and the first two parts of Lemma 2,

$$\begin{aligned}
& n \ln\left(X^{k+1} \bullet S^k\right) - \ln\det\left(X^{k+1}\right) - \ln\det\left(S^k\right) \\
&= n \ln\left(X^{k+1} \bullet S^k\right) - \ln\det\left(X^{k+1} S^k\right) \\
&= n \ln\left(X^{k+1} \bullet S^k/\mu\right) - \ln\det\left(X^{k+1} S^k/\mu\right) \\
&= n \ln n - \ln\det\left((S^k)^{.5} X^{k+1} (S^k)^{.5}/\mu\right) \\
&\leq n \ln n + \tfrac{\|(S^k)^{.5} X^{k+1} (S^k)^{.5}/\mu - I\|}{2\left(1 - \|(S^k)^{.5} X^{k+1} (S^k)^{.5}/\mu - I\|_\infty\right)} \\
&\leq n \ln n + \tfrac{\alpha^2}{2(1-\alpha)} \\
&\leq n \ln\left(X^k \bullet S^k\right) - \ln\det\left(X^k\right) - \ln\det\left(S^k\right) + \tfrac{\alpha^2}{2(1-\alpha)}.
\end{aligned}$$

Additionally, by the third part of Lemma 2

$$\sqrt{n}\left(\ln(X^{k+1} \bullet S^k) - \ln(X^k \bullet S^k)\right) = \sqrt{n} \ln \frac{\mu}{\mu^k} \leq -\frac{\alpha}{2}.$$

Adding the two inequalities gives

$$\Psi(X^{k+1}, S^k) \leq \Psi(X^k, S^k) - \frac{\alpha}{2} + \frac{\alpha^2}{2(1-\alpha)}.$$

By choosing $\alpha = 0.4$ again, we have the desired result. ■

This theorem leads to

Corollary 1 *Let $\rho \geq n + \sqrt{n}$ and $\Psi(X^0, S^0) \leq (\rho - n)\ln(X^0 \bullet S^0)$. Then, the algorithm terminates in at most $O((\rho - n)\ln(X^0 \bullet S^0/\epsilon))$ iterations.*

Proof. In $O((\rho - n)\ln(X^0 \bullet S^0/\epsilon))$ iterations,

$$\Psi(X^k, S^k) \leq (\rho - n)\ln(\epsilon).$$

Also,

$$(\rho - n)\ln(C \bullet X^k - b^T y^k) = (\rho - n)\ln(X^k \bullet S^k) \leq \Psi(X^k, S^k) - n\ln n \leq \Psi(X^k, S^k).$$

Combining the two inequalities,

$$C \bullet X^k - b^T y^k = X^k \bullet S^k < \epsilon.$$

■

Again, from (11) we see that the algorithm can generate an X^k as a by-product. However, it is not needed in generating the iterate direction, and it is only explicitly used for proving convergence and complexity.

The computation cost in each iteration of the algorithm can be summarized as follows. First, updating S or $S + \mathcal{A}^T(d(\bar{z}^k))$ uses matrix additions and mn^2 operations, and factoring it uses $O(n^3)$ operations. Secondly, creating the Gram matrix uses $nm^2 + 2n^2m + O(nm)$ operations, and factoring and solving the system of equations uses $O(m^3)$ operations. Finally, dot products for $\bar{z}^{k+1}$ and $\|P(\bar{z}^k)\|$ and the calculation of y^{k+1} use only $O(m)$ operations. These give the total $O(m^3 + nm^2 + n^2m + n^3)$ floating point operations. Note that the procedure uses only the Cholesky factorization.

In contrast, each iteration of primal-dual methods requires several additional computations. First, the various Schur complement matrices used to compute the step directions cost significantly more to compute than the matrices used in this dual-scaling algorithm. Second, primal-dual algorithms must compute a primal step direction. This step direction involves the product of three matrices, which can be very costly. Third, the primal-dual algorithms does use line searches in both the primal and dual problems. Such a search requires additional dense matrix factorizations.

3 Maximum Cut Problem

The maximum cut problem asks to partition the vertices of a graph into two sets that maximize the sum of the weighted edges connecting vertices in one set with vertices in the other. The positive semidefinite relaxation of the maximum cut problem can be expressed as (e.g., [11][27])

$$\text{(MAX-CUT)} \qquad \begin{array}{ll} \text{Minimize} & C \bullet X \\ \text{Subject to} & diag(X) = e, \\ & X \succeq 0. \end{array} \tag{18}$$

The operator $diag(\cdot)$ takes the diagonal of a matrix and makes it a vector. In other words, $A_i = e_i e_i^T$, $i = 1, ..., n$, where e_i is the vector with 1 for the ith component and 0 for all others.

The dual program can be expressed as:

$$\text{(DMAX)} \qquad \begin{array}{ll} \text{Maximize} & e^T y \\ \text{Subject to} & Diag(y) + S = C, \quad S \succeq 0, \end{array} \tag{19}$$

The operator $Diag(\cdot)$ forms a diagonal matrix from a vector.

Many examples of the maximum cut problem have a very sparse matrix C. Since S is a linear combination of C and a diagonal matrix, it possesses the same sparse structure of C that remains constant for all iterations. This sparsity can be exploited by reordering S to reduce fill-in during the Cholesky factorization. A good reordering will drastically speed up the factorization and the many forward and back substitutions required to compute the Gram matrices.

Applying the dual-scaling algorithm to this relaxation,

$$\nabla \psi(y^k, \bar{z}^k) = -\frac{\rho}{\Delta^k} e + diag((S^k)^{-1})$$

and

$$M^k = (S^k)^{-1} \circ (S^k)^{-1}, \tag{20}$$

where $\circ$ represents the Hadamard product and $\Delta^k = \bar{z}^k - e^T y^k$. That is, $M^k_{ij} = \left((S^k)^{-1}_{ij}\right)^2$. When the graph represented by C is connected, M^k is generally dense—even when C is sparse.

The direction $d(\bar{z}^k)_y$ of (8) is comprised of two parts:

$$dy_1 = (M^k)^{-1} e \tag{21}$$

$$dy_2 = (M^k)^{-1} diag((S^k)^{-1}) \tag{22}$$

so that

$$d(\bar{z}^k)_y = \frac{\rho}{\Delta^k} dy_1 - dy_2 \tag{23}$$

Since the dual direction depends upon the upper bound $\bar{z}^k$, splitting the direction into these two parts allows the algorithm to take advantage of a possibly improved upper bound.

To determine the stepsize and measure the improvement in the potential function, we again compute

$$\|P(\bar{z}^k)\| = \sqrt{-\nabla \psi^T (y^k, \bar{z}^k)^T d(\bar{z}^k)_y}. \tag{24}$$

If $\|P(\bar{z}^k)\|$ is sufficiently small, Lemma 2 guarantees an improved primal solution, $X(\bar{z}^k)$ with $C \bullet X(\bar{z}^k) < \bar{z}^k$, where from (11),

$$X(\bar{z}^k) = \frac{\Delta^k}{\rho} (S^k)^{-1} \left(Diag(d(\bar{z}^k)_y) + S^k \right) (S^k)^{-1}.$$

Frequently, an improved primal objective value $\bar{z}$ can be found for even larger values of $\|P(\bar{z}^k)\|$. We may first compute

$$\bar{z} := C \bullet X(\bar{z}^k) = e^T y^k + \frac{\Delta^k}{\rho} \left(diag((S^k)^{-1})^T d(\bar{z}^k)_y + n \right). \tag{25}$$

If $\bar{z} < \bar{z}^k$, then we go on to check if $X(\bar{z}^k) \succ 0$. But from the above expression, $X(\bar{z}^k) \succ 0$ if and only if

$$\left(Diag(d(\bar{z}^k)_y) + S^k\right) \succ 0. \tag{26}$$

To check if $Diag(d(\bar{z}^k)_y) + S^k \succ 0$, we use the Cholesky factorization and simply check if its pivots are all positive. We stop the factorization process as soon as we encounter a negative or zero pivot, and conclude that the matrix is not positive definite. Note that $Diag(d(\bar{z}^k)_y) + S^k$ has the same sparse structure as S^k or C, allowing it to be stored in the same data structure. If $Diag(d(\bar{z}^k)_y) + S^k \succ 0$, we set $\bar{z}^{k+1} = \bar{z} < \bar{z}^k$. Otherwise, $\bar{z}^{k+1} = \bar{z}^k$.

An improved upper bound $\bar{z}^{k+1}$ results in a smaller $\Delta^k := \bar{z}^{k+1} - e^T y^k$ and will modify the dual step direction calculated in (23), which is why the step direction was divided into two parts. Finally, the dual variables will be updated by

$$y^{k+1} = y^k + \frac{\alpha}{\|P(\bar{z}^{k+1})\|} d(\bar{z}^{k+1})_y \quad \text{and} \quad S^{k+1} = C - Diag(y^{k+1}).$$

If $\alpha < 1$, $S(\alpha) = C - Diag\left(y^k + \frac{\alpha}{\|P(\bar{z}^{k+1})\|} d(\bar{z}^{k+1})_y\right) \succ 0$. Larger values of α increase the stepsize which can speed up the convergence of the algorithm. Larger stepsizes, however, can also step outside the cone of positive semidefinite matrices. If a larger step is used, a Cholesky factorization can check the positive definiteness of $S(\alpha)$. Note that this factorization is needed in the next iteration anyway. Since the matrix $S(\alpha)$ is sparse and well ordered, an unsuccessful attempt to increase the stepsize costs very little. In general, these factorizations cost far less than a factorization of the dense M^k, but allow large stepsizes to significantly reduce the number of iterations required to achieve the desired accuracy.

We now state the specialized dual-scaling algorithm for solving the maximum cut semidefinite program.

DUAL ALGORITHM. Reorder C to reduce fill-in during Cholesky factorization. Set $\bar{z}^0 = C \bullet I$ and choose y^0 such that $S^0 = C - Diag(y^0) \succ 0$. Set $k = 0$, $\alpha = .99$ and do the following:

```
while $\frac{\bar{z}^k - e^T y^k}{|e^T y^k| + 1} \geq \epsilon$ do
begin
    1. Compute $diag((S^k)^{-1})$ the matrix $M^k$ (20) using Algorithm M or M'.
    2. Solve (21), (22), and (23) for the dual step direction.
    3. Use (25) to compute a new upper bound $\bar{z}$.
    4. If $\bar{z} < \bar{z}^k$ and $\left(Diag(d(\bar{z}^k)_y) + S^k\right) \succ 0$,
       then let $\bar{z}^{k+1} = \bar{z}$ and recompute $d(\bar{z}^{k+1})_y$ using (23);
       else let $\bar{z}^{k+1} = \bar{z}^k$. endif
    5. Compute $\|P(\bar{z}^{k+1})\|$ using (24).
    6. Select $\beta \geq \alpha/\|P(\bar{z}^{k+1})\|$, so that $y^{k+1} = y^k + \beta d(\bar{z}^{k+1})_y$ and $S^{k+1} = C - Diag(y^{k+1}) \succ 0$.
    7. $k := k + 1$.
end
```

4 Computational Results

We implemented the dual-scaling algorithm in ANSI C and ran the program on a PC with 233 MHz, 64 MB RAM and 256 K cache memory. (The code and its User-Guide are available for public download at the web-side *http://dollar.biz.uiowa.edu/col/*)

To accelerate convergence of the algorithm, the implementation used more aggressive stepsizes. It used values of α equal to $.99, 1.5, 3$, and 6. Initially, we set $\alpha = 3$. When the value of α was successful for three consecutive iterates, we tried the next larger value. If we stepped out of the feasible region, we tried the next smaller value of α. We found that that larger stepsizes were frequently used and this strategy yields a significant improvement in the total number of iterations.

In addition, we initialized the value of ρ to be $5n$. Larger values of ρ more aggressively seek the optimal solution, but are also more likely to yield infeasible points. After a couple of iterates, ρ was dynamically selected using the following criteria:

$$\rho = 1.6n * \sqrt{(rgap^{k-1}/rgap^k)}$$

where $rgap^{k-1}$ and $rgap^k$ are the relative duality gaps at the previous and current iteration

$$rgap^k := \frac{\bar{z}^k - b^T y^k}{1 + |b^T y^k|}.$$

We let the initial point

$$X^0 = I \quad \text{and} \quad y_i^0 = C_{ii} - \sum_{j \neq i} |C_{ij}| - 1, \quad i = 1, ..., n,$$

which by Gerschgorin's Theorem, guarantees $S^0 \succ 0$ (see Atkinson [5]). This value generally provides a reasonable starting point. We have used the minimum degree ordering algorithm to reorder C.

We have stopped the iteration process when the relative duality gap

$$rgap^k = \frac{\bar{z}^k - b^T y^k}{1 + |b^T y^k|} \leq 10^{-6}.$$

Most combinatorial applications ask for a reasonable bound to be found very quickly. Therefore, the precision required in the semidefinite program is far less than required by other applications. In addition, the original maximum cut problem has only simple, binary variables. For these problems, we believe that a precision of 10^{-4} is sufficient, so we have recorded the number of iterations and seconds needed to compute that level of precision.

Our experiments used a machine independent graph generator, **rudy**, created by G. Rinaldi. We tested the maximum cut semidefinite program on the G set of graphs used by Helmberg and Rendl [12]. This set of problems becomes a standard test set for graph optimization. These maximum cut problems range in size from $n = 800$ to $n = 3000$. Many of these problems, like G1-G10, G22-G31, and G43-G47 have a randomly created structure. Problems G11-G13, G32-G34, and G48-G50 come from a 2 dimensional toroidal grid, while the others represent planar type graphs. (Helmberg and Rendl [12] actually solved G53-G54 semidefinite programs for another graph problem, the ϑ-function [18], instead of the maximum cut problem.)

Name	Sparsity of S_{chol}	Factor S (sec.)	Compute M (sec.)	Factor M (sec.)
G1	73.6%	1.412	12.856	1.983
G11	4.2%	0.010	1.272	1.863
G14	14.3%	0.140	3.105	1.863
G22	47.8%	11.076	129.917	32.046
G32	1.6%	0.030	9.864	30.744
G35	11.8%	1.352	41.113	30.764

Table 1: Seconds used for the three most expensive computations.

Table 1 shows the cost of key steps of the algorithm for six different problems. It shows the seconds required to factor S, create M, and factor M. The sparsity statistic in the second column gives the percentage of nonzero entries in the factor after reordering.

This table shows that when S is sparse, the factorization of M dominates the computation time. Since M is generally dense, regardless of the sparsity of S, its computation time is constant for problems of equal size. For more dense problems, the creation of M dominates the computation time. This is not surprising since it uses $3n^3$ floating point operations, while the Cholesky factorization uses a sixth of that amount. Most large scale applications, however, will contain a certain sparse structure, and the table shows how this dual-scaling algorithm exploits that structure to save computation time.

The table also emphasizes the importance of a good ordering of the matrix in the beginning of the algorithm. The reordering of matrices has been studied for years [6], but to illustrate its importance, we include a few figures. The following figures in Figure 1 show the structure of S and its Cholesky factor in the 800×800 example G14.

The objective matrix G14 has about 1.58% nonzero entries. Figure 1(a) shows the sparsity structure of the matrix before the minimum degree ordering and Figure 1(c) presents the structure after the minimum degree ordering. The dual solution S has the same sparse structure. Figure 1(b) and Figure 1(d) show the sparsity patterns of the Cholesky factors before and after the minimum degree ordering. The Cholesky factor of the unordered matrix is very dense. Comparing the running times shows that the reordering matrix would reduce the running time approximately by a factor of 4.

Table 2 shows the performance of the code on solving the G set maximum cut semidefinite programs for stopping tolerances of $rgap \leq 10^{-4}$ and $rgap \leq 10^{-6}$. *PriObj*, *DualObj*, and *Rgap* are the primal and dual objective values and the relative duality gap at termination. Also shown is the dimension and the percentage of nonzero entries in the objective matrix, as well as the time (in seconds) and number of iterations required by the program.

Most of the previous numerical tests [7][17][32][35][36], were conducted on smaller problem data sets where the dimension n was only a few hundred or less, so that no available computation result could be compared to ours. After our results reported, a study of using a primal-dual algorithm for solving relative larger problems, including the maximum cut problem, was conducted by Fujisawa, Fukuda, Kojima, and Nakata [10]. They tested solving sparse maximum cut semidefinite programs with dimension up-to 1250. On a sparse problem with dimension of 1000, they required 63,130 seconds; on a problem of dimension

1250, they used 111,615 seconds. Their computations were performed on a DEC AlphaServer 8400 with a processing speed of 437 MHz and 8GB memory, which is much superior to the PC machine used in our test.

As we mentioned before, Helmberg and Rendl [12] used a spectral bundle method to solve the same set of G1-G42 maximum cut problems. Their computations were performed on a UltraSPARC station with 64 MB memory. One advantage of the spectral bundle method is that it uses considerably less memory since it does not create or store a matrix as large as M. On problems with a randomly created structure, the bundle method appears slightly faster than ours. In these problems, the Cholesky factor of the slack matrix is relatively dense, despite the sparsity of the objective matrix. For the toroidal and planar graphs (such as G14), the dual matrix has a much better structure. In these problems, a minimum degree ordering kept the Cholesky factor sparse and the back and forward substitutions quick. In problems with a more structured objective matrix, the dual SDP algorithm outperformed the bundle method.

Finally, our implementation of the dual-scaling algorithm appears to be the first algorithm to converge to an optimal point in polynomial time, to use the characteristics inherent in many large scale problems to its advantage, and to verify the optimality by solving both the primal and dual problems simultaneously. Its success of even relative dense examples shows that the algorithm is generally efficient, while the improved performance on more sparse examples shows how it exploits the structure of most large scale problems.

Acknowledgement The authors would like to thank the two referees for their valuable comments and suggestions on improving the presentation of the paper.

						rgap= 10^{-4}		rgap= 10^{-6}	
Name	Dim	Spars	Pobj	Dobj	Rgap	Iter	Time	Iter	Time
G1	800	6.12%	-4.833276e+04	-4.833279e+04	5.452359e-07	20	616.08	24	741.15
G2	800	6.12%	-4.835767e+04	-4.835772e+04	9.502824e-07	19	592.69	23	719.25
G3	800	6.12%	-4.833732e+04	-4.833733e+04	3.401351e-07	19	589.56	24	746.54
G4	800	6.12%	-4.844576e+04	-4.844581e+04	9.824817e-07	19	595.43	23	723.28
G5	800	6.12%	-4.839953e+04	-4.839955e+04	3.552550e-07	19	594.71	24	752.24
G6	800	6.12%	-1.062463e+04	-1.062464e+04	8.541375e-07	21	646.12	25	769.41
G7	800	6.12%	-9.957042e+03	-9.957051e+03	8.420776e-07	21	654.40	25	779.28
G8	800	6.12%	-1.002773e+04	-1.002773e+04	9.678249e-07	21	655.63	24	780.46
G9	800	6.12%	-1.011492e+04	-1.011493e+04	7.775437e-07	21	654.09	25	778.95
G10	800	6.12%	-9.940246e+03	-9.940253e+03	7.465415e-07	20	620.63	24	742.82
G11	800	0.63%	-2.516658e+03	-2.516659e+03	3.719002e-07	18	64.40	23	82.05
G12	800	0.63%	-2.495497e+03	-2.495498e+03	5.198500e-07	19	71.03	24	89.50
G13	800	0.63%	-2.588544e+03	-2.588546e+03	9.570427e-07	20	76.70	24	91.88
G14	800	1.59%	-1.276625e+04	-1.276627e+04	9.986387e-07	29	166.43	33	189.11
G15	800	1.58%	-1.268623e+04	-1.268623e+04	3.450954e-07	33	188.68	39	222.87
G16	800	1.59%	-1.270007e+04	-1.270007e+04	4.674935e-07	27	154.31	31	177.01
G17	800	1.58%	-1.268530e+04	-1.268531e+04	4.548353e-07	26	149.60	30	172.44
G18	800	1.59%	-4.664038e+03	-4.664040e+03	4.708559e-07	47	276.82	51	299.92
G19	800	1.58%	-4.328040e+03	-4.328042e+03	3.345754e-07	33	193.66	38	221.98
G20	800	1.59%	-4.445567e+03	-4.445570e+03	7.261350e-07	33	193.80	37	216.54
G21	800	1.58%	-4.417133e+03	-4.417136e+03	8.094406e-07	38	223.30	42	246.10
G22	2000	1.05%	-5.654376e+04	-5.654378e+04	3.957778e-07	23	8215.71	28	9997.62
G23	2000	1.05%	-5.658202e+04	-5.658204e+04	3.910436e-07	23	8146.30	28	9920.25
G24	2000	1.05%	-5.656340e+04	-5.656342e+04	4.981024e-07	23	8323.22	27	9759.06
G25	2000	1.05%	-5.657696e+04	-5.657698e+04	3.876964e-07	23	8306.01	28	10106.27
G26	2000	1.05%	-5.653145e+04	-5.653148e+04	5.027876e-07	23	8323.72	27	9756.52
G27	2000	1.05%	-1.656663e+04	-1.656664e+04	6.729729e-07	25	8851.74	29	10268.68
G28	2000	1.05%	-1.640315e+04	-1.640316e+04	7.335225e-07	25	8862.04	29	10278.55
G29	2000	1.05%	-1.683555e+04	-1.683556e+04	6.531697e-07	25	9034.53	29	10483.11
G30	2000	1.05%	-1.686152e+04	-1.686153e+04	6.413532e-07	26	9386.62	30	10843.05
G31	2000	1.05%	-1.646672e+04	-1.646673e+04	6.898466e-07	25	9047.75	29	10493.99
G32	2000	0.25%	-6.270553e+03	-6.270559e+03	9.633737e-07	23	1070.39	27	1255.70
G33	2000	0.25%	-6.177246e+03	-6.177250e+03	6.333926e-07	25	1175.24	29	1362.61
G34	2000	0.25%	-6.186747e+03	-6.186750e+03	4.510949e-07	24	1182.73	28	1381.23
G35	2000	0.64%	-3.205895e+04	-3.205896e+04	3.752328e-07	46	5167.17	51	5716.93
G36	2000	0.64%	-3.202383e+04	-3.202386e+04	9.011525e-07	38	4381.39	42	4841.52
G37	2000	0.64%	-3.207448e+04	-3.207449e+04	3.820329e-07	42	4836.22	47	5400.13
G38	2000	0.64%	-3.205987e+04	-3.205990e+04	8.761401e-07	47	5392.57	52	5952.48
G39	2000	0.64%	-1.151058e+04	-1.151059e+04	7.376637e-07	59	6615.01	63	7056.50
G40	2000	0.64%	-1.145915e+04	-1.145916e+04	3.728182e-07	52	6123.85	57	6703.57
G41	2000	0.64%	-1.146087e+04	-1.146087e+04	4.158517e-07	58	6629.09	63	7194.15
G42	2000	0.64%	-1.178500e+04	-1.178501e+04	7.308258e-07	45	5049.62	49	5495.55
G43	1000	2.10%	-2.812887e+04	-2.812889e+04	7.416858e-07	18	767.17	22	939.50
G44	1000	2.10%	-2.811152e+04	-2.811154e+04	7.511560e-07	18	770.22	22	939.07
G45	1000	2.10%	-2.809911e+04	-2.809913e+04	4.530243e-07	21	900.51	25	1075.49
G46	1000	2.10%	-2.811972e+04	-2.811973e+04	3.978937e-07	18	782.46	23	999.66
G47	1000	2.10%	-2.814662e+04	-2.814664e+04	7.609519e-07	18	751.57	22	920.39
G48	3000	0.17%	-2.399999e+04	-2.400000e+04	3.699426e-07	14	2878.85	19	3861.30
G49	3000	0.17%	-2.399999e+04	-2.400000e+04	3.718535e-07	14	2873.69	19	3826.62
G50	3000	0.17%	-2.395268e+04	-2.395269e+04	3.543263e-07	14	2389.73	19	3192.86
G51	1000	1.28%	-1.602501e+04	-1.602502e+04	7.759416e-07	29	341.45	33	388.06
G52	1000	1.28%	-1.603854e+04	-1.603856e+04	7.427919e-07	36	441.94	41	489.28
G53	1000	1.28%	-1.603886e+04	-1.603887e+04	8.122974e-07	26	306.85	30	353.86
G54	1000	1.28%	-1.602476e+04	-1.602478e+04	7.421491e-07	29	340.43	33	387.51

Table 2: Performance on solving the G-set semidefinite programs.

References

[1] I. Adler, N. K. Karmarkar, M. G. C. Resende, and G. Veiga, "An implementation of Karmarkar's algorithm for linear programming," *Mathematical Programming*, 44:297–335, 1989, Errata in *Mathematical Programming*, 50:415, 1991.

[2] F. Alizadeh, "Combinatorial optimization with interior point methods and semi–definite matrices," Ph.D. Thesis, University of Minnesota, Minneapolis, MN, 1991.

[3] F. Alizadeh, J. P. A. Haeberly, and M. L. Overton, "Primal-dual interior point methods for semidefinite programming," Technical Report 659, Computer Science Department, Courant Institute of Mathematical Sciences, New York University, 1994.

[4] K. M. Anstreicher and M. Fampa, "A long-step path following algorithm for semidefinite programming problems," Working Paper, Department of Management Science, The University of Iowa, Iowa City, 1996.

[5] K. E. Atkinson, *An Introduction to Numerical Analysis* (Second Edition John Wiley & Sons, New York, 1989).

[6] I. S. Duff, " Sparse Numerical Linear Algebra: Direct Methods and Preconditioning," *The State of the Art in Numerical Analysis* (Clarendon Press, Oxford, 1997).

[7] J. Falkner, F. Rendl, and H. Wolkowicz, "A computational study of graph partitioning," *Mathematical Programming*, 66(2):211-240, 1994.

[8] K. Fujisawa, M. Kojima and K. Nakata, "Exploiting Sparsity in Primal-Dual Interior Point Methods for Semidefinite Programming" Research Report B-324, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro-ku, Tokyo 152, September 1997.

[9] T. Fujie and M. Kojima, "Semidefinite programming relaxation for nonconvex quadratic programs," Research Report B-298, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Meguro, Tokyo 152, May 1995, to appear in *Journal of Global Optimization.*

[10] K. Fujisawa, M. Fukuda, M. Kojima and K. Nakata, "Numerical Evaluation of SDPA (SemiDefinite Programming Algorithm)," Research Report B-330, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro-ku, Tokyo 152, September 1997.

[11] M. X. Goemans and D. P. Williamson, " Improved approximation algorithms for Maximum Cut and Satisfiability problems using semidefinite programming," *Journal of ACM* 42 (1995) 1115–1145.

[12] C. Helmberg and F. Rendl, "A spectral bundle method for semidefinite programming," ZIB Preprint SC 97-37, Konrad-Zuse-Zentrum fuer Informationstechnik Berlin, Takustrasse 7, D-14195 Berlin, Germany, August 1997.

[13] C. Helmberg, F. Rendl, R. J. Vanderbei, H. Wolkowicz, "An interior point method for semidefinite programming," *SIAM Journal of Optimization,* 6:342-361, 1996.

[14] S. E. Karisch, F. Rendl, and J. Clausen, "Solving graph bisection problems with semidefinite programming," Technical Report DIKU-TR-97/9, Department of Computer Science, University of Copenhagen, July 1, 1997.

[15] N. Karmarkar, A new polynomial-time algorithm for linear programming, *Combinatorica* 4, (1984), 373-395.

[16] M. Kojima, S. Shindoh, and S. Hara, "Interior point methods for the monotone semidefinite linear complementarity problem in symmetric matrices." Research Reports on Information Sciences, Ser. B: Operations Research B-282, Dept. of Information Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152, Japan, April 1994, to appear in *SIAM Journal of Optimization.*

[17] C. Lin and R. Saigal, "On solving large scale semidefinite programming problems - a case study of quadratic assignment problem," Technical Report, Dept. of Ind. and Oper. Eng., University of Michigan, Ann Arbor, MI, 1997.

[18] L. Lovász, "On the Shannon Capacity of a graph," *IEEE Transactions of Information Theory* IT-25(1):1-7, Jan. 1979.

[19] L. Lovász and A. Shrijver, "Cones of matrices and setfunctions, and $0-1$ optimization," *SIAM Journal on Optimization* 1 (1990) 166-190.

[20] R. D. C. Monteiro and P. R. Zanjácomo, "Implementation of primal-dual methods for semidefinite programming based on Monteiro and Tsuchiya directions and their variants," Technical Report, School of Ind. and Systems Engineering, Georgia Institute of Technology, Atlanta, 1997.

[21] R. D. C. Monteiro and Y. Zhang, "A unified analysis for a class of path-following primal-dual interior-point algorithms for semidefinite programming," manuscript, School of Ind. and Systems Engineering, Georgia Institute of Technology, Atlanta, 1996.

[22] M. Muramatsu, "Affine scaling algorithm fails for semidefinite programming," Technical Report, Department of Mechanical Engineering, Sophia University, Tokyo, Japan, 1996.

[23] M. Muramatsu and R. Vanderbei, "Primal-dual affine scaling algorithms fail for semidefinite programming," Technical Report, SOR, Princeton University, Princeton, NJ, 1997.

[24] Yu. E. Nesterov, "Quality of semidefinite relaxation for nonconvex quadratic optimization," CORE Discussion Paper, #9719, Belgium, March 1997.

[25] Yu. E. Nesterov and A. S. Nemirovskii, *Interior Point Polynomial Methods in Convex Programming : Theory and Algorithms* (SIAM Publications, SIAM, Philadelphia, 1993).

[26] Y. E. Nesterov and M. Todd, "Primal-dual interior point methods for self scaled cones, " Technical Report 1125, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York, 14853-3801m 1995, to appear in *SIAM J. Optimization.*

[27] S. Polijak, F. Rendl and H. Wolkowicz, "A recipe for semidefinite relaxation for 0-1 quadratic programming," *Journal of Global Optimization* 7 (1995) 51-73.

[28] H. D. Sherali and W. P. Adams, "Computational advances using the reformulation-linearization technique (rlt) to solve discrete and continuous nonconvex problems. *Optima*, 49:1-6, 1996.

[29] N. Z. Shor, "Quadratic optimization problems," *Soviet Journal of Computer and Systems Sciences*, 25:1-11, 1987. Originally published in Tekhnicheskaya Kibernetika, No. 1, 1987, pp. 128-139.

[30] M. J. Todd, “On search directions in interior-point methods for semidefinite programming,” Technical Report No. 1205, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853-3801, October 1997.

[31] L. Vandenberghe and S. Boyd, “Semidefinite programming,” *SIAM Review* 38(1) (1996) 49–95.

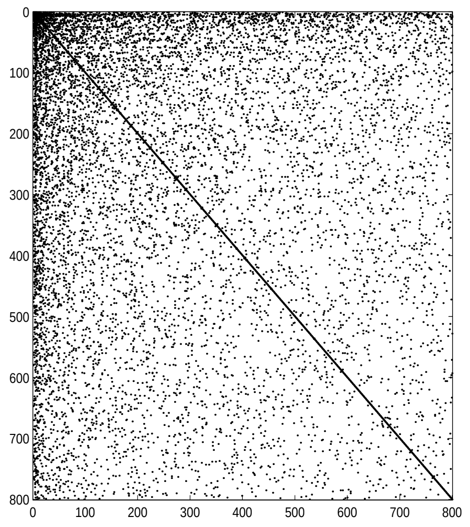
[32] H. Wolkowicz and Q. Zhao, “Semidefinite programming for the graph partitioning problem,” manuscript, 1996, to appear *Discrete Applied Math.*.

[33] Y. Ye, “Approximating quadratic programming with bound and quadratic constraints,” Working Paper, Department of Management Science, The University of Iowa, Iowa City, 1997, to appear in *Mathematical Programming.*

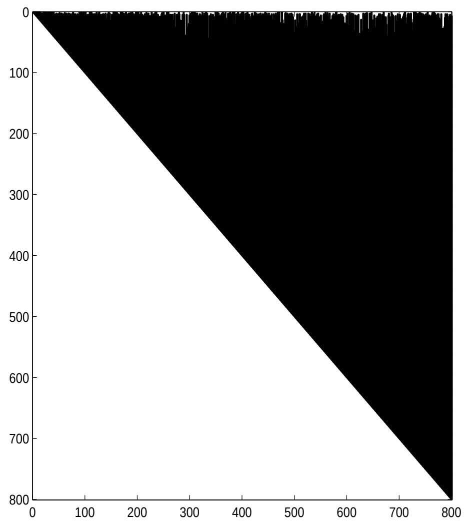
[34] Y. Ye, *Interior Point Algorithms : Theory and Analysis* (Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, New York, 1997).

[35] Q. Zhao, “Semidefinite Programming for Assignment and Partitioning Problems” PhD thesis, University of Waterloo, 1996.

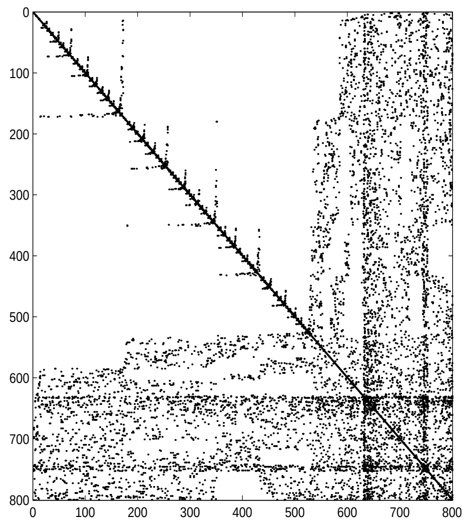
[36] Q. Zhao, S. Karisch, F. Rendl, and H. Wolkowicz, “Semidefinite programming relaxations for the quadratic assignment problems,” *J. Combinatorial Optimization*, 2.1, 1998, CORR 95-27.



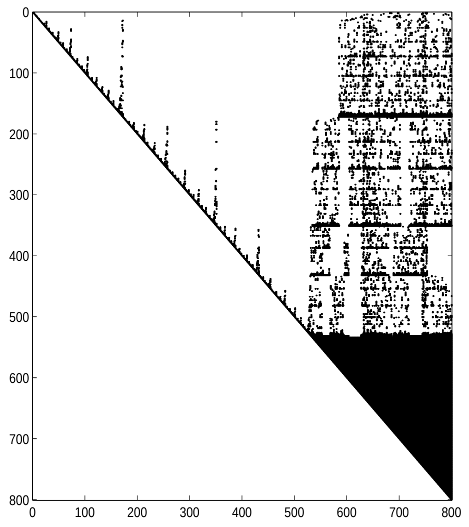
(a) Sparsity pattern before reordering.



(b) Cholesky factor before reordering.



(c) Sparsity pattern after reordering.



(d) Cholesky factor after reordering.

Figure 1: Sparsity patterns and Cholesky factors of G14.