Mixed Linear and Semidefinite Programming for Combinatorial and Quadratic 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.

February 13, 1998; revised April 1999

Abstract

We use the semidefinite relaxation to approximate combinatorial and quadratic optimization problems subject to linear, quadratic, as well as boolean constraints. We present a dual potential reduction algorithm and show how to exploit the sparse structure of various problems. Coupled with randomized and heuristic methods, we report computational results for approximating graph-partition and quadratic problems with dimensions 800 to 10,000. This finding, to the best of our knowledge, is the first computational evidence of the effectiveness of these approximation algorithms for solving large-scale problems.

Key words. Semidefinite relaxation, dual potential reduction algorithm, randomized algorithm.

*This work was 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

Many combinatorial and quadratic optimization problems are NP hard, and even finding their approximate solutions is a difficult problem. One approximation method, the positive semidefinite programming (SDP) relaxation, was originally proposed by Lovász and Shrijver [15], and further developed by many authors, see, e.g., Alizadeh [1], Fujie and Kojima [10], Karisch, Rendl, and Clausen [14], and Polijak, Rendl and Wolkowicz [19]. Recently, there were several theoretical results on the effectiveness of approximating combinatorial and quadratic optimization problems by solving semidefinite programs, see Goemans and Williamson [12] (also see, e.g., Frieze and Jerrum[9], Nesterov [16], and Ye [22]).

Although they are "polynomially" solvable, semidefinite programs with dimension above 1000 have been extremely hard to solve in practice. Thus, exploiting the structure and sparsity characteristic of large-scale semidefinite programs becomes critical to the efficient computation of their solution. In this paper we show that high quality approximations to a variety of large-scale combinatorial and quadratic problems can be found by coupling a dual-scaling algorithm, applied to their positive semidefinite relaxations, with a randomized or heuristic procedure. We present positive test results for problems with dimensions of 800 to 10,000. To the best of our knowledge, our finding is the first computational evidence of the effectiveness of these approximations in solving large-scale problems.

A variety of combinatorial and quadratic optimization problems can be posed as:

$$\begin{array}{ll} \text{Minimize} & \hat{C} \bullet (vv^T) + \hat{c}^T x := v^T \hat{C} v + \hat{c}^T x \\ \text{Subject to} & \hat{A}_i \bullet (vv^T) + \hat{a}_i^T x := v^T \hat{A}_i v + \hat{a}_i^T x = b_i, \quad i = 1, \dots, m, \\ & x \geq 0, \end{array} \tag{1}$$

where $\hat{C}, \hat{A}_i$ are given symmetric matrices, $\hat{c}$ and $\hat{a}_i$ are given column vectors, and v and x are the unknown variable vectors. The operation $A \bullet B = \text{tr } A^T B = \sum_{jk} A_{jk} B_{jk}$. In most applications, $\hat{A}_i$ is a sparse rank-one matrix, $\hat{C}$ is sparse, and $\hat{a}_i$ is null or the unit i^{th} unit vector. The combinatorial problems can then be relaxed by replacing the matrix vv^T with any positive semidefinite matrix and posed as a semidefinite program in standard form.

$$\text{(SDP)} \qquad \begin{array}{ll} \text{Minimize} & C \bullet X \\ \text{Subject to} & A_i \bullet X = b_i, \quad i = 1, \dots, m, \\ & X \in K \end{array} \tag{2}$$

where $K = K_1 \oplus K_2 \oplus \cdots \oplus K_r$ and K_l is the cone of $n_l \times n_l$ symmetric positive semidefinite matrices, and $C, A_i \in \Re^{n \times n}$ are symmetric.

The dual of (SDP) can be written as:

$$
\text{(DSP)} \qquad \begin{array}{ll} \text{Maximize} & b^T y \\ \text{Subject to} & \sum_{i=1}^{m} y_i A_i + S = C, \quad S \succeq 0, \end{array} \tag{3}
$$

where y_i, $i = 1, \ldots, m$.

Thus, we have the following well-known duality theorem:

Theorem 1 *(Strong Duality) Provided there is a strictly interior point to (SDP) and (DSP) there exist primal and dual optimal solutions, (X^*, y^*, S^*), and $C \bullet X^* = b^T y^*$.*

There are actually several polynomial algorithms that can solve a pair of semidefinite programs. One is the primal-scaling algorithm (Nesterov and Nemirovskii [18], Alizadeh [1], Vandenberghe and Boyd [21], and Ye [23]), which is the analogue of the primal potential reduction algorithm for linear programming. This algorithm uses X to generate the iterate direction. Another is the dual-scaling algorithm (Vandenberghe and Boyd [21], Anstreicher and Fampa [2], and Ye [23]), which is the analogue of the dual-scaling algorithm for linear programming. The dual-scaling algorithm uses only S to generate the iterate direction. The third is the primal-dual scaling algorithm which uses both X and S to generate iterate directions, including Alizadeh-Haeberly-Overton, Helmberg-Rendl-Vanderbei-Wolkowicz/Kojima-Shida-Hara/Monteiro, Nesterov-Todd, Gu, and Toh directions, as well as directions called the MTW and Half directions (see Todd [20] and references therein). All these algorithms possess $O(\sqrt{n}\log(1/\epsilon))$ iteration complexity to yield accuracy ϵ.

2 Dual Scaling Algorithm

We briefly describe the dual-scaling algorithm, which is a modification of the linear programming algorithm. For simplicity, consider (SDP) and (DSP) without vector variables (x, s). The algorithm reduces the Tanabe-Todd-Ye primal-dual potential function

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

where $\rho \geq n + \sqrt{n}$. This algorithm, along with other SDP algorithms, is described in Ye [23]. We will use notations defined there.

Let

$$
\mathcal{A}(X) = \begin{pmatrix} A_1 \bullet X \\ \vdots \\ A_m \bullet X \end{pmatrix} \quad \text{and} \quad \mathcal{A}^T y = \sum_{i=1}^{m} A_i y_i,
$$

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

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

Its gradient is:

$$\nabla\psi(y,\bar{z}) = -\frac{\rho}{\bar{z}-b^T y}b + \mathcal{A}(S^{-1}). \tag{4}$$

Each step in the algorithm minimizes the linearized dual potential function subject to an ellipsoidal constraint that keeps $S \succ 0$ and the quadratic error term small.

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

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

where α is a constant in $(0,1)$. (The matrix norm used here will be the Frobenius norm and all vector norms will be the 2-norm.)

Letting $\hat{A}_i = (S^k)^{-.5}A_i(S^k)^{-.5}$, the first order Karusch-Kuhn-Tucker conditions state that the optimal point, y^{k+1}, of this convex problem satisfies

$$\begin{pmatrix} \hat{A}_1 \bullet \hat{A}_1 & \cdots & \hat{A}_1 \bullet \hat{A}_m \\ \vdots & \ddots & \vdots \\ \hat{A}_m \bullet \hat{A}_1 & \cdots & \hat{A}_m \bullet \hat{A}_m \end{pmatrix} (y^{k+1}-y^k) + \beta\nabla\psi(y^k,\bar{z}^k) = 0 \tag{6}$$

for a positive value of β.

The matrix in (6) is a Gram matrix and is positive definite when $S \succ 0$ and the constraint matrices A_i are linearly independent. In this paper, it will be referred to as M^k.

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

$$y^{k+1} - y^k = \frac{\alpha}{\sqrt{\nabla\psi^T(y^k,\bar{z}^k)(M^k)^{-1}\nabla\psi(y^k,\bar{z}^k)}} d(\bar{z}^k)_y. \tag{7}$$

The step direction $d(\bar{z}^k)_y$ of (7) is comprised of two parts:

$$dy_1 = (M^k)^{-1}b \tag{8}$$

$$dy_2 = (M^k)^{-1}\mathcal{A}((S^k)^{-1}) \tag{9}$$

so that

$$d(\bar{z}^k)_y = \frac{\rho}{\bar{z}^k - e^T y^k} dy_1 - dy_2 \tag{10}$$

Since the dual step depends upon the upper bound $\bar{z}$ and the parameter ρ, splitting the step direction into these two parts allows the algorithm more flexibility.

The dual step direction can also be used to find a primal matrix X near the central path. Consider the least squares problem:

$$\begin{array}{ll} \text{Minimize} & \|(S^k)^{.5}X(S^k)^{.5} - \frac{\bar{z}^k - b^T y^k}{\rho} I\| \\ \text{Subject to} & \mathcal{A}(X) = b. \end{array} \tag{11}$$

The solution to this problem is

$$X(\bar{z}^k) = \frac{\bar{z}^k - b^T y^k}{\rho}(S^k)^{-1}\left(\mathcal{A}^T d(\bar{z}^k)_y + S^k\right)(S^k)^{-1}. \tag{12}$$

Creating the primal matrix may be costly. However, the evaluation of the primal objective value $C \bullet X(\bar{z}^k)$ requires much less work.

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

Since the vectors $\mathcal{A}(S^k)^{-1}$ and $d(\bar{z}^k)_y$ were found previously in the calculation of the dual step direction, the cost of computing a primal objective value is the cost of a dot product.

Under suitable conditions, the following lemma guarantees the feasibility of $X(\bar{z}^k)$. If $\mathcal{A}^T d(\bar{z}^k)_y + S^k \succeq 0$ and $z^k > C \bullet X(\bar{z}^k)$, let $X^{k+1} = X(\bar{z}^k)$. Otherwise let $X^{k+1} = X^k$. In fact, more than one value of ρ may be used to calculate different directions $d(\bar{z}^k)_y$ and different matrices $X(\bar{z}^k)$. Some of these matrices will be positive definite while others will not and some will have have better objective values than the others. Only the objective value of the matrix is needed by the algorithm; since it does not require the primal matrix to compute the step direction, the algorithm saves time and memory by not explicitly computing $X(\bar{z}^k)$.

Defining

$$P(\bar{z}^k) = \frac{\rho}{\bar{z}^k - b^T y^k}(S^k)^{.5} X(\bar{z})(S^k)^{.5} - I, \tag{13}$$

we have the following lemma, adapted from Ye's [23] linear programming potential reduction algorithm.

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

$$\|P(\bar{z}^k)\| < min(\alpha\sqrt{\frac{n}{n+\alpha^2}}, 1-\alpha), \tag{14}$$

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 can be found in [5]. An equivalent lemma for positive semidefinite programming can be found in [23]. ■

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

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

which by (5) and (7), makes

$$\begin{aligned} \|P(\bar{z}^k)\| &= \tfrac{\sqrt{\nabla\psi^T(y^k,\bar{z}^k)(M^k)^{-1}\nabla\psi(y^k,\bar{z}^k)}}{\alpha}\|(S^k)^{-.5}\left(\mathcal{A}^T(y-y^k)\right)(S^k)^{-.5}\| \\ &= \sqrt{\nabla\psi^T(y^k,\bar{z}^k)(M^k)^{-1}\nabla\psi(y^k,\bar{z}^k)} \\ &= \sqrt{-\nabla\psi^T(y^k,\bar{z}^k)d(\bar{z}^k)_y}. \end{aligned} \tag{15}$$

Update the dual variables according to

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

Furthermore,

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

and the reduction in the potential function satisfies the inequality

$$\psi(y^{k+1},\bar{z}^k) - \psi(y^k,\bar{z}^k) \le -\alpha\|P(\bar{z}^k)\| + \frac{\alpha^2}{2(1-\alpha)}. \tag{18}$$

Lemma 1 and (17) lead to the following theorem which assures that either the primal of dual point sufficiently reduces the potential function.

Theorem 2 *If $\rho \ge n + \sqrt{n}$,*

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

for a positive constant δ.

The theoretical algorithm can be stated as follows.

DUAL ALGORITHM. Given X^0, y^0, S^0 such that $\mathcal{A}(X^0) = b, X^0 \succ 0$, $\bar{z}^0 = C \bullet X^0$, and $S^0 = C - \mathcal{A}^T y^0 \succ 0$, set $k = 0$ and do the following:

while $\bar{z}^k - b^T y^k \ge \epsilon$ **do**

1. Compute $\nabla\psi^T(y^k,\bar{z}^k)$ and the matrix M^k of (6).
2. Solve (8) and (9) for the step directions dy_1 and dy_2.

3. Select ρ and compute $d(\bar{z}^k)_y$.

   if $\bar{z}^{k+1} > C \bullet X(\bar{z}^k)$, **and** $\mathcal{A}^T d(\bar{z}^k)_y + S^k \succ 0$,

   then $X^{k+1} = X(\bar{z}^k)$, and $\bar{z}^{k+1} = C \bullet X(\bar{z}^k)$

   else $X^{k+1} = X^k$, and $\bar{z}^{k+1} = \bar{z}^k$ **end if**.

4. Select ρ, compute $d(\bar{z}^k)_y$ and $\|P(\bar{z}^{k+1})\|$, and select α such that $y^{k+1} = y^k + \frac{\alpha}{\|P(\bar{z}^{k+1})\|} d(\bar{z}^{k+1})_y$ and $S^{k+1} = C - \mathcal{A}^T y^{k+1} \succ 0$.

5. $k := k + 1$.

end

Corollary 1 *Let $\rho \geq n + \sqrt{n}$. Then, the algorithm terminates in at most $O((\rho - n)\log(1/\epsilon))$ iterations.*

From the algorithm and (12) we can see that X^k can be generated from a dual matrix S and has a structure that can exploit sparsity in the data. While solving the positive semidefinite problem, it is only used for proving convergence and complexity, and it is not needed in generating the iterate direction or any other part of the algorithm. Benson, Ye, and Zhang [5] discussed in detail how the dual scaling algorithm takes advantage of the sparsity for the maximum cut problem.

3 Combinatorial Optimization

Examples of combinatorial optimizations problems that can be approximated with the positive semidefinite relaxation are graph partitioning problems and bound constrained quadratic problems. Graph partitioning problems often involve finding a cut that maximizes an objective function. Given a proper subset of $\mathcal{S}$ of vertices, the cut corresponding to $\mathcal{S}$ is the subset of edges with one end in $\mathcal{S}$ and the other end in the complement of $\mathcal{S}$. The weight of a cut is the sum of the weights of the edges of the cut. Several combinatorial problems ask to partition the vertices of a graph into sets that maximize its cut. The constraints placed upon the vertices will define the optimal solution and determine the appropriate techniques for solving it.

Represent the vertices and edges in the graph by the weighted adjacency matrix W and formulate the maximum cut problem by placing the vertices in one of two sets by assigning each vertex a $+1$ or -1.

$$\begin{array}{ll} \text{Maximize} & \frac{1}{2}\sum_{i<j} W_{i,j}\,(1 - v_i v_j) \\ \text{Subject to} & v \in \{-1, 1\}^n, \end{array} \tag{19}$$

Using the Laplacian matrix

$$\hat{C} = \frac{1}{4}\left(\text{Diag}(We) - W\right), \tag{20}$$

the problem can be rewritten as:

$$\text{MAX-CUT} \qquad \begin{array}{ll} \text{Maximize} & \hat{C} \bullet (vv^T) \\ \\ \text{Subject to} & \operatorname{diag}(vv^T) = e \end{array} \tag{21}$$

The operator $\operatorname{diag}(\cdot)$ takes the diagonal of a matrix. The vector $e \in \Re^n$ has each element equal to one. This formulation has the form of (1) since matrices the form $(\hat{A}_i)_{i,i} = 1$ can describe the constraints upon the diagonal. Since the constraint matrices for this problem contain only one nonzero element along the diagonal, the dual matrix S is sparse when the adjacency matrix W is sparse.

Derive the positive semidefinite relaxation of the maximum cut problem by letting $X = vv^T$ and relaxing the set of feasible matrices X to include all positive semidefinite matrices. This relaxation can be posed as:

$$\begin{array}{ll} \text{Maximize} & \hat{C} \bullet X \\ \\ \text{Subject to} & \operatorname{diag}(X) = e, \quad X \succeq 0 \end{array} \tag{22}$$

As discussed in the introduction, many algorithms exist for solving this positive semidefinite programs. However, the original combinatorial problem (21) requires a rank-one matrix of the form $X = vv^T$. The technique for using the solution of (22) to find an approximate solution v to (21) was first analyzed by Geomans and Williamson [12].

In their randomized algorithm, let $V \in \Re^{n \times n}$ such that $X = VV^T$, select a vector $u \in \Re^n$ from a uniform distribution of unit vectors, and let

$$\hat{v} = Vu. \tag{23}$$

and define

$$v_i = sign(\hat{v}_i), \ i = 1, \ldots, n.$$

Geomans and Williamson proved that if all the edge weights $W_{ij} \geq 0$, the expectation of $q(v) = v^T \hat{C} v$ is at least 0.878 of maximum cut solution.

For another measure of optimality, let $\overline{q}$ and $\underline{q}$ be the upper and lower bound of $q(v)$. Furthermore, let $E[\cdot]$ be the expectation operator. Nesterov [16] proved that for arbitrary adjacency matrices, the randomized technique above gives an approximate solution v such that

$$E\left[\frac{q(v) - \underline{q}}{\overline{q} - \underline{q}}\right] \geq \frac{4}{7}.$$

By the strong law of large numbers, as the number of applications of this randomized algorithm increases to infinity, the probability of generating a solution whose objective value equals or exceeds the expectation increases to one. In practice, this algorithm is very quick and can be repeated many times.

Since the primal solution matrix is symmetric, has diagonal elements equal to one, and has off diagonal elements between -1 and 1, it can be interpreted as the covariance matrix of $\hat{v}$ (Bertsimas and Ye [6]). The rank reduction techniques can be applied to $\hat{v} = X_{:,i}$, the ith column of the solution matrix. In fact, this approach amounts to a deterministic choice for the vector u. If $u = V_{:,i}^T$, the ith row of V, then $X_{:,i} = V(V^T)_{:,i}$.

Other problems may impose additional constraints upon the feasible set. For example, the problem may specify the size of the two subsets of vertices. Another problem may require that two particular vertices be placed in different subsets. Alternatively, the problem may allow the elements in vector v to assume any value in the interval $[-1, 1]$. For these problems, the max-cut randomized algorithm can be modified to provide answers that approximate the optimal solution and satisfy the additional constraints.

3.1 Equal/Unequal-Cut Problem

For $\kappa > 0$, the unequal-cut problem asks for the maximum cut such that $|e^T v| = \kappa$. Its positive semidefinite relaxation can be posed in the standard form (2) as:

$$\text{(UNEQUAL-CUT)} \qquad \begin{array}{lrl} \text{Minimize} & \hat{C} \bullet X & \\ \text{Subject to} & \operatorname{diag}(X) & = e, \\ & (ee^T) \bullet X & = \kappa^2, \\ & X \succeq 0, & \end{array} \tag{24}$$

Momentarily separating the dual variables for the first n constraints from the last constraint, the dual program can be expressed as:

$$\begin{array}{ll} \text{Maximize} & e^T y + \kappa^2 \lambda \\ \text{Subject to} & \operatorname{Diag}(y) + \lambda(ee^T) + S = \hat{C}, \quad S \succeq 0. \end{array} \tag{25}$$

In this problem, the additional constraint makes the matrix S totally dense, but nonetheless, very structured.

$$\nabla_y \psi(y^k, \lambda^k, \bar{z}^k) = -\frac{\rho}{\bar{z}^k - e^T y^k - \kappa^2 \lambda^k} e + \operatorname{diag}((S^k)^{-1})$$

$$\nabla_\lambda \psi(y^k, \lambda^k, \bar{z}^k) = -\frac{\rho}{\bar{z}^k - e^T y^k - \kappa^2 \lambda^k} \kappa^2 - e^T (S^k)^{-1} e$$

and

$$M^k = \begin{pmatrix} (S^k)^{-1} \circ (S^k)^{-1} & (S^k)^{-1} e \circ (S^k)^{-1} e \\ e^T (S^k)^{-1} \circ e^T (S^k)^{-1} & (e^T (S^k)^{-1} e)^2 \end{pmatrix}. \tag{26}$$

where $\circ$ represents componentwise multiplication.

The two parts of the step direction $d(\bar{z}^k)_y$ are:

$$\begin{bmatrix} dy_1 \\ d\lambda_1 \end{bmatrix} = (M^k)^{-1} \begin{bmatrix} e \\ k^2 \end{bmatrix} \tag{27}$$

$$\begin{bmatrix} dy_2 \\ d\lambda_2 \end{bmatrix} = (M^k)^{-1} \begin{bmatrix} \operatorname{diag}((S^k)^{-1}) \\ e^T (S^k)^{-1} e \end{bmatrix} \tag{28}$$

so that

$$\begin{bmatrix} d(\bar{z}^k)_y \\ d(\bar{z}^k)_\lambda \end{bmatrix} = \frac{\rho}{\bar{z}^k - e^T y^k - \kappa^2 \lambda} \begin{bmatrix} dy_1 \\ d\lambda_1 \end{bmatrix} - \begin{bmatrix} dy_2 \\ d\lambda_2 \end{bmatrix}. \tag{29}$$

The primal objective value and measure $\|P(\bar{z}^k)\|$ are given by

$$\hat{C} \bullet X(\bar{z}^k) = e^T y^k + \kappa^2 \lambda^k + \frac{\bar{z}^k - e^T y^k - \kappa^2 \lambda^k}{\rho} \left(d(\bar{z}^k)_y^T diag((S^k)^{-1}) + d(\bar{z}^k)_\lambda^k e^T (S^k)^{-1} e + n \right)$$

and

$$\|P(\bar{z}^k)\| = \sqrt{-\nabla \psi_y^T (y^k, \bar{z}^k) d(\bar{z}^k)_y - \nabla \psi_\lambda d(\bar{z}^k)_\lambda}.$$

The dual variables will be updated for some step-size $\beta \leq 1$:

$$y^{k+1} = y^k + \beta d(\bar{z}^k)_y, \qquad \lambda^{k+1} = \lambda^k + \beta d(\bar{z}^k)_\lambda,$$

and

$$S^{k+1} = \hat{C} - \operatorname{Diag}(y^{k+1}) - \lambda^{k+1}(ee^T).$$

The equal-cut problem requires a partition that places an equal number of vertices in each of the two sets ($e^T v = 0$). It is like the unequal cut problem discussed in the previous section with $\kappa = 0$. But since the only matrices X that satisfy $(ee^T) \bullet X = 0$ are singular, the dual solution set may be unbounded. We create an interior to the primal feasible set by adding an artificial variable x and a large positive penalty parameter Λ to the objective function.

These technical reasons revise the positive semidefinite relaxation to:

$$\text{(EQUAL-CUT)} \qquad \begin{array}{lrl} \text{Minimize} & \hat{C} \bullet X + \Lambda x & \\ \text{Subject to} & \operatorname{diag}(X) & = e, \\ & (ee^T) \bullet X - x & = 0. \\ & X \succeq 0, x \geq 0 & \end{array} \tag{30}$$

Its dual program can be expressed as:

$$\begin{array}{lrll} \text{Maximize} & e^T y & & \\ \text{Subject to} & \operatorname{Diag}(y) + \lambda(ee^T) + S & = \hat{C}, & S \succeq 0, \\ & -\lambda + s & = \Lambda, & s \geq 0. \end{array} \tag{31}$$

Notice that due to the presence of x, s, and Λ, the dimension n of this problem is one higher that the dimension in the unequal-cut problem for the same graph.

To generate a feasible cut for the unequal and equal cut problem, the randomized algorithm for obtaining a vector v can be modified by sorting the vector $\hat{v}$. Let $\sigma(i)$ be the permutation function that sorts the vector in ascending order. Let $\hat{v}_{\sigma(1)}$ be the smallest component in the vector and $\hat{v}_{\sigma(n)}$ the largest. For the equal cut problem, define an approximate solution

$$v_{\sigma(i)} = \begin{cases} -1 & \text{if } i \leq \frac{n}{2} \\ 1 & \text{if } i > \frac{n}{2} \end{cases}$$

This procedure is also discussed by Bertsimas and Ye [6]. For the unequal cut problem, denote the median component $\hat{v}_{med}$. If $\hat{v}_{med} \leq 0$,

$$v_{\sigma(i)} = \begin{cases} -1 & \text{if } i \leq \frac{n+\kappa}{2} \\ 1 & \text{if } i > \frac{n+\kappa}{2} \end{cases}$$

Otherwise, when $\hat{v}_{med} > 0$,

$$v_{\sigma(i)} = \begin{cases} -1 & \text{if } i \leq \frac{n-\kappa}{2} \\ 1 & \text{if } i > \frac{n-\kappa}{2} \end{cases}$$

3.2 S-T Cut problem

The s-t cut problem asks for the maximum cut that places vertices s and t in different sets. Defining the vector e_{st} to have a 1 in indices s and t, and 0's elsewhere, the problem requires that $e_{st}^T v = 0$. When the constraint is relaxed to $(e_{st}e_{st}^T) \bullet X = 0$, the only matrices X that satisfy the second constraint lie on the boundary. Like in the equal-cut problem, we add an artificial variable x and a large positive penalty parameter Λ to the objective function. The positive semidefinite relaxation for the s-t cut problem is:

$$\text{(S-T CUT)} \qquad \begin{array}{lll} \text{Minimize} & \hat{C} \bullet X + \Lambda x & \\ \text{Subject to} & \operatorname{diag}(X) & = e, \\ & (e_{st}e_{st}^T) \bullet X - x & = 0, \\ & X \succeq 0, x \geq 0 & \end{array} \tag{32}$$

The same randomization algorithm that is applied to the maximum cut problem can also be applied to the s-t cut problem. When $x = 0$, this procedure will place vertices s and t in different sets with probability 1. Thus, this algorithm will provide a feasible vector v expected to be at least $\frac{4}{7}$ optimal.

4 Box-Constrained Quadratic Problem

The positive semidefinite relaxation also applies problems not related to the partitioning of vertices in a graph. Minimizing a quadratic function over a "box" around the origin can be posed as:

$$\begin{array}{ll} \text{Minimize} & v^T \hat{C} v \\ \\ \text{Subject to} & -e \le v \le e. \end{array} \tag{33}$$

This objective function can be rewritten as $\hat{C} \bullet (vv^T)$ and the problem can be relaxed in the same manner used by the graph partitioning problems:

$$(\text{BOX}) \qquad \begin{array}{ll} \text{Minimize} & \hat{C} \bullet X \\ \\ \text{Subject to} & \operatorname{diag}(X) + x = e, \\ & X \succeq 0, x \ge 0. \end{array} \tag{34}$$

To approximate solutions to nonconvex quadratic functions, Ye [22] (see Nesterov [17] for a generalization) adapted the randomized algorithm to the box-constrained QP problem. He defined

$$v_i = \sqrt{X_{i,i}} sign(\hat{v}_i), \; i = 1, \ldots, n.$$

and proved that

$$E\left[\frac{\overline{q} - q(v)}{\overline{q} - \underline{q}}\right] \ge \frac{4}{7}.$$

5 DSDP: An Implementation

To solve the combinatorial problems discussed in Section 3, we developed a software package, DSDP [4], that implements the dual scaling algorithm and the randomized algorithm for finding approximate solutions to these combinatorial problems. It assumes the mixed linear and positive semidefinite problems have the form

$$(\text{DSDP}) \qquad \begin{array}{ll} \text{Minimize} & \hat{C} \bullet X + \hat{c}^T x \\ \\ \text{Subject to} & (a_i a_i^T) \bullet X + \hat{a}_i^T x = b_i, \quad i = 1, \ldots, m. \\ & X \succeq 0, \; x \ge 0. \end{array} \tag{35}$$

where $\hat{C} \in \Re^{n_1 \times n_1}, a_i \in \Re^{n_1}, \hat{c} \in \Re^{n_2}$ and $\hat{a}_i \in \Re^{n_2}$. Note that this form is more restrictive than the standard form (2), where A_i is replaced by the rank-one matrix $a_i a_i^T$. On the other hand, the vector variable x is explicitly added and handled. All problems mentioned earlier can be described in this form. The assumption of this form is just one of the ways the program was designed exploit the structure and sparsity of these problems. Therefore, to describe more about the implementation of the algorithm, we will use the notation in (35) in highlighting the rank-one nature.

The dual of (35) is

$$\begin{array}{lll} \text{Maximize} & b^T y & \\ \text{Subject to} & \sum_{i=1}^m y_i (a_i a_i^T) + S & = \hat{C}, \quad S \succeq 0 \\ & \sum_{i=1}^m y_i \hat{a}_i + s & = \hat{c}, \quad s \geq 0. \end{array} \tag{36}$$

The most expensive stage of the algorithm is the computation of the linear system (6) and the factorization of M^k. The elements of M^k are defined by

$$M_{i,j}^k = (a_i^T (S^k)^{-1} a_j)^2 + \hat{a}_i^T (\text{Diag}(s^k))^{-2} \hat{a}_j.$$

The DSDP software package calculates these values by solving m linear systems of the form $S^k w = a_i$ and then computing the inner products $w^T a_j$. This construction of M^k is simple compared with primal dual methods of computing Schur complement matrices, and it uses relatively few floating point operations. To exploit any sparsity offered by the problem, DSDP reorders the dual matrix before the first iterate to reduce fill-in during factorization. Since the sparsity pattern of the dual matrix remains static for all iterates, one ordering applies to all iterates. When its Cholesky factor is sparse, the algorithm realizes additional savings during the factorization of S^k and triangular solves involved in computing M^k.

After computing the linear system (6), DSDP factors M^k and directly solves for dy_1 and dy_2. Using $n = n_1 + n_2$, DSDP computes four dual step directions corresponding to four different values of ρ:

$$\rho \in \left\{ 0.8(\rho^{k-1} - n) + n, \rho^{k-1} 1.2(\rho^{k-1} - n) + n), 1.2n \right\}. \tag{37}$$

Using each step direction, the program looks for a positive definite primal matrix and an improved primal objective value. The program checks if $X(\bar{z}) \succeq 0$ by factoring the matrix

$$\mathcal{A}^T d(\bar{z}^k)_y + S^k \succeq 0 \tag{38}$$

and checking for positive pivots. Similarly,

$$x(\bar{z}) = \frac{\bar{z} - b^T y^k}{\rho} \text{Diag}(s)^{-2} \left(\sum_{i=0}^m \hat{a}_i d(\bar{z}^k)_{y_i} + s^k \right)$$

and $x(\bar{z}) \geq 0$ if and only if $\sum \hat{a}_i d(\bar{z}^k)_{y_i} + s^k \geq 0$. Since (38) has the same nonzero structure as S, it can exploit sparsity and use the same data structure as S. The program conserves memory since it creates only two matrices: one dense structure for M and one sparse structure for the S and (38).

For the dual step, DSDP uses a dynamic selection of ρ. Experience has shown that appropriate values of ρ at one point may be different than appropriate values at another point. Our experience suggests that the best value of ρ for the dual step is the near the value that provides the feasible primal matrix with the best objective value. Multiple values of ρ were used to find primal matrices primarily as a means of determining a good value for the dual step. If one or more of the four values of ρ in (37) provided a positive definite primal matrix $X(\bar{z})$, the algorithm defines ρ^k to be the one that provided the best primal objective value. This heuristic for choosing ρ^k is a one dimensional optimization problem; each value of ρ the program tests may cost a matrix factorization, so the we only solve this problem approximately. Experience has also shown that very high values of ρ often lead to points far from the central path, which hurts convergence in future iterates, so DSDP bounds ρ^k based on the relative gap and $\|P(\bar{z}^k)\|$. If the relative gap $\frac{\bar{z}^k - b^T y^k}{(|b^T y^k| + |\bar{z}^k|)/2+1} > 1.5$, the program bounds ρ above by $1.2n$. As the relative gap decreases, we let the the upper bound increase to allow a faster rate of convergence. In early iterates, points are generally not as well centered and $\|P(\bar{z}^k)\|$ tends to be large. If $\|P(\bar{z}^k)\| > \sqrt{n}/10 + 1$, the program bounds the rho^k above by $1.2n$.

If none of the four values of ρ provided a positive definite X, DSDP uses another heuristic. In this case the program usually selects $\rho^k = \rho^{k-1}$. However, if no feasible primal matrix has been found for two or more consecutive iterates, it selects $\rho^k = 0.95\rho^{k-1}$. This heuristic reduces ρ^k from the previous iterate to favor the centering direction and improve the likelihood of finding a positive definite primal matrix in the next iterate. To satisfy the convergence criteria of Corollary 1, the program bounds ρ^k below by $n + \sqrt{n}$.

The program then begins the dual step line search with the nearly full step length of $\alpha = 0.99 \times \|P(\bar{z})\|$. The program checks for a positive s^k and attempts a Cholesky factorization of the dual matrix to check if S^k is positive definite. If successful, the point is accepted. Otherwise, the program reduces α by a factor of 0.8 until a feasible point is found. Theoretically, the new point will be feasible if $\alpha < 1$, but experience has shown that occasionally, $\|P(\bar{z})\|$ is several orders of magnitude larger that 1. Only rarely can the algorithm accept such a large value for α and remain in the cone of positive definite dual matrices. To prevent an inordinate amount of backtracking, the largest value of α that the program will use is $\sqrt{n}/2 + 1$. The Cholesky factorization of the dual matrix is needed in the next iterate to compute M^{k+1}, so when the first value of α provides a feasible point, the line search essentially cost nothing. Exact line searches, on the other hand, require expensive eigenvalue computations.

5.1 Rank Reduction

Recall from Section 3 that the randomized algorithms for finding approximate solutions to combinatorial problems require a matrix V such that $X = VV^T$. Even though the matrix $X(\bar{z}^k)$ is totally dense, the dual scaling algorithm provides a structure that allows sparsity in the data to reduce the computation time.

Since

$$X(\bar{z}^k) = \frac{\bar{z}^k - b^T y^k}{\rho}(S^k)^{-1}\left(\mathcal{A}^T d(\bar{z}^k)_y + S^k\right)(S^k)^{-1},$$

DSDP lets $LL^T = \left(\mathcal{A}^T d(\bar{z}^k)_y + S^k\right)$ and uses

$$V = \sqrt{\frac{\bar{z}^k - b^T y^k}{\rho}}(S^k)^{-1}L. \tag{39}$$

The computation of $\hat{v} = Vu$ requires one matrix vector multiplication and two triangular solves. Sparsity in the problem can accelerate the computations.

DSDP also allows the columns of $X(\bar{z}^k)$ to be used for $\hat{v}$. It computes these columns, $X(\bar{z}^k)_{:,j}$ by

$$X(\bar{z}^k)_{:,j} = \frac{\bar{z}^k - b^T y^k}{\rho}(S^k)^{-1}\left(\sum_{i=1}^{m} d(\bar{z}^k)_{y_i}(a_i a_i^T)(S^k)^{-1} + I\right) e_j.$$

where e_j is column j of the identity matrix. This technique requires four triangular solves and m vector inner products, vector scalar multiplications, and vector sums.

5.2 Dense Constraints

In problems like the equal and unequal cut problem, one constraint can make the dual matrix entirely dense. When relatively few constraints make Cholesky factor of the matrix dense, the problems still contains some sparsity that DSDP can utilize.

Instead of factoring a dense matrix to check if S is positive definite, DSDP separates the dense constraints from the sparse ones and factors a sparse matrix. Without loss of generality, let the last m_d constraints be the dense ones. Assuming a nonsingular matrix, let

$$L\tilde{I}L^T = \hat{C} - \sum_{i=1}^{m-m_d} y_i a_i a_i^T,$$

for a lower triangular matrix L and matrix $\tilde{I}$ containing only 1's and -1's along the diagonal. Let $\tilde{m}_d$ be the sum of m_d and the number of negative diagonal elements in $\tilde{I}$. For $i = 1, \ldots, m_d$, define $\tilde{a}_i = L^{-1}a_i$ and for $i = m_d + 1, \ldots, \tilde{m}_d$, let $\tilde{a}_i$ be the unit coordinate vectors corresponding to the negative diagonal elements of $\tilde{I}$. Furthermore, define $\gamma \in \Re^{\tilde{m}_d}$ by $\gamma_i = -y_i$ for $i = 1, \ldots, m_d$, and $\gamma_i = -2$ for $i = m_d + 1, \ldots, \tilde{m}_d$. Then

$$S = \hat{C} - \sum_{i=1}^{m} y_i a_i a_i^T = L\left(\tilde{I} - \sum_{i=m-m_d+1}^{m} L^{-1} y_i a_i a_i^T L^{-T}\right) L^T = L\left(I + \sum_{i=1}^{\tilde{m}_d} \gamma_i \tilde{a}_i \tilde{a}_i^T\right) L^T \tag{40}$$

where $\tilde{a}_i = L^{-1}a_i$.

Let $W = I + \sum_{i=1}^{\tilde{m}_d} \gamma_i \tilde{a}_i \tilde{a}_i^T$. For any nonzero $w \in R^n$ that is orthogonal to the vectors $\tilde{a}_i$, $w^T W w = w^T w > 0$. Therefore, W is positive definite if and only if

$$(\sum_{k=1}^{\tilde{m}_d} \zeta_i \tilde{a}_i)^T W (\sum_{k=1}^{\tilde{m}_d} \zeta_i \tilde{a}_i) > 0 \qquad \forall \zeta \in \Re^{\tilde{m}_d}, \zeta \neq 0.$$

Note that

$$(\sum_{k=1}^{\tilde{m}_d} \zeta_i \tilde{a}_i)^T W (\sum_{k=1}^{\tilde{m}_d} \zeta_i \tilde{a}_i) = \zeta^T \tilde{W} \zeta.$$

where $\tilde{W} \in \Re^{\tilde{m}_d \times \tilde{m}_d}$ and

$$\tilde{W}_{i,j} = \sum_{k=1}^{\tilde{m}_d} \gamma_k (\tilde{a}_i^T \tilde{a}_k)(\tilde{a}_j^T \tilde{a}_k) + \tilde{a}_i^T \tilde{a}_j$$

The matrix S is positive definite if $\tilde{W}$ has a Cholesky factorization. Since $\tilde{m}_d \ll n$, the cost of this procedure is little more than the cost of computing of the matrix factorization.

Furthermore, the factorization L and vectors $\tilde{a}_i$ computed in this procedure can be used in the next iterate to compute the matrix M^k. The Sherman-Morrison-Woodbury formula allows us to solve the dense linear systems $S^k w = a_i$ with two sparse triangular systems and some lower order computations.

Recall that the dual scaling algorithm produces an X of the form

$$X(\bar{z}^k) = \frac{\bar{z}^k - b^T y^k}{\rho} (S^k)^{-1} \left(\mathcal{A}^T d(\bar{z}^k)_y + S^k \right) (S^k)^{-1}. \tag{41}$$

The same technique can be applied to the middle term of (41) to check if the matrix $X(\bar{z})$ is positive semidefinite.

An additional technique for handling dense constraints arises in the randomized algorithm for finding approximate solutions to the combinatorial problem (1). These algorithms require a matrix V such that $X = VV^T$. As in (40), let

$$\left(\mathcal{A}^T d(\bar{z}^k)_y + S^k \right) = \hat{C} - \sum_{i=1}^{m} (y_i - dy_i) a_i a_i^T = L \left(I + \sum_{i=1}^{\tilde{m}_d} \gamma_i \tilde{a}_i \tilde{a}_i^T \right) L^T$$

Since $I + \sum_{i=1}^{\tilde{m}_d} \gamma_i \tilde{a}_i \tilde{a}_i^T \succ 0$, there exists a unique square root of the matrix that is symmetric and positive definite. In DSDP we select

$$V = \sqrt{\frac{\bar{z}^k - b^T y^k}{\rho}} (S^k)^{-1} L \left(I + \sum_{i=1}^{\tilde{m}_d} \hat{\gamma}_i \tilde{a}_i \tilde{a}_i^T \right).$$

for appropriate constants $\tilde{\gamma}$ such that

$$\left(I + \sum_{i=1}^{\tilde{m}_d} \hat{\gamma}_i \tilde{a}_i \tilde{a}_i^T \right)^2 = I + \sum_{i=1}^{\tilde{m}_d} \gamma_i \tilde{a}_i \tilde{a}_i^T \tag{42}$$

When $\tilde{m}_d = 0$, (42) coincides with (39). The computation of $\hat{v} = Vu$ requires two triangular solves, one triangular matrix vector multiplication, and $\tilde{m}_d$ vector inner products, vector scalar multiplications, and vector sums. The latter operations have lower complexity and cost little since $\tilde{m}_d \ll n$.

To compute the constants $\hat{\gamma}$, DSDP first computes an orthonormal matrix E and diagonal matrix D such that

$$EDE^T = I + \sum_{i=1}^{m_d} \gamma_i P \tilde{a}_i \tilde{a}_i^T P^T \tag{43}$$

for an orthonormal matrix P. The matrix E contains the eigenvalues of (43) and the matrix D contains its eigenvalues. A proper choice of P makes the matrix $I + \sum \gamma_i P\tilde{a}_i\tilde{a}_i^T P^T$ very sparse and the matrices E and D very quick to compute. The DSDP software applies the Gram-Schmidt process to the vectors $\tilde{a}_i$ to get the first $\tilde{m}_d$ rows of P. The remaining rows can be any set of unit vectors orthogonal to the other rows; since their inner product with $\tilde{a}_i$ equals zero, they do not have to be explicitly computed.

Since $(EDE^T)^{0.5} = ED^{0.5}E^T$,

$$ED^{0.5}E^T = I + \sum_{i=1}^{\tilde{m}_d} \hat{\gamma}_i P\tilde{a}_i\tilde{a}_i^T P^T \tag{44}$$

This matrix equals the identity matrix everywhere except for one $\tilde{m}_d \times \tilde{m}_d$ block. The $\tilde{m}_d$ diagonal elements of this submatrix provide the system of equations the DSDP uses to solve for the $\tilde{m}_d$ unknowns $\hat{\gamma}$. Premultiplication of (42) by P and postmuliplication by P^T shows that the vector $\hat{\gamma}$ that satisfies (44) also satisfies (42).

5.3 Initial Points

DSDP attempts to generate an initial feasible dual matrix by searching for constraints that bound the diagonal of the matrix X. It ensures that $S^0 \succ 0$ by using these constraints to make S^0 diagonally dominant. The first n constraints in each of our problems bound the diagonal elements above by 1. The constraint matrices $a_i a_i^T$ are positive definite, so as a corollary of the Gerschgorin's circle theorem, for any positive γ_1 and nonnegative γ_2, selecting

$$y_i^0 = \begin{cases} \hat{C}_{i,i} - \sum_{i \neq j} |\hat{C}_{i,j}| - \gamma_1 & i = 1, \ldots, n \\ -\gamma_2 & \text{otherwise} \end{cases}$$

makes $S^0 \succ 0$. DSDP uses $\gamma_1 = \gamma_2 = 1.0$. In many combinatorial problems, the constraint vectors $\hat{a}_i$ are either null or the i^{th} unit vector. For the problem we considered, this choice of y^0 satisfies $s^0 = \hat{c} - \sum_{i=1}^m y_i^0 \hat{a}_i > 0$.

Recall that the construction of $d(\bar{z}^k)_y$ and primal matrices $X(\bar{z}^k)$ use the upper bound, and primal objective value, $\bar{z}$. To initialize $\bar{z}^0$, define $\mu = \frac{\rho}{\bar{z}^k - b^T y^0}$, so that

$$d(\bar{z})_y = \frac{1}{\mu} dy_1 - dy_2$$

and

$$\hat{C} \bullet X(\bar{z}^k) + \hat{c}^T x(\bar{z}^k) = b^T y^k + \mu \left(d(\bar{z})_y^T \mathcal{A}(S^k)^{-1} + n \right)$$

One or more values of μ can be used to locate an initial upper bound. DSDP chooses these values by first calculating a $\bar{\mu}$ such that $b^T(\frac{1}{\bar{\mu}} dy_1 - dy_2) = 0.0$. Since the matrix M is positive definite, $b^T dy_1 > 0$. Since $-dy_2$ is the centering direction, it is likely that $b^T dy_2 > 0$. Therefore, $\bar{\mu} =$

$(b^T dy_1)/(b^T dy_2)$. For points y near the central path, $\bar{\mu}$ approximates nearby centrality parameters μ. In the case $b^T dy_2 \leq 0$, we let $\bar{\mu} = 0.001$.

DSDP uses $\mu = 0.8\bar{\mu}, 1.2\bar{\mu}$, and $2.0\bar{\mu}$ to locate a positive semidefinite matrix X and upper bound $\bar{z}^0$. If none of these values provide a positive semidefinite matrix, the program accepts the highest of the three primal objective values, and notes that this value of $\bar{z}$ is just an guess for an upper bound and should be replaced when any positive matrix $X(\bar{z})$ and positive vector $x(\bar{z})$ has been found.

6 Computational Results

We tested our implementation of the dual scaling algorithm on five different problems: max-cut, equal-cut, unequal-cut, s-t -cut, and box-constrained QP. These problems were selected because they had positive semidefinite relaxations that fit into the form (35) and they had quick, effective methods for obtaining approximate solutions to the combinatorial problem (1) from the positive semidefinite relaxation.

In previous studies, Helmberg and Rendl [13] used a spectral bundle method to solve the positive semidefinite relaxations for max-cut problems and stable set problems with up to 3000 vertices. Their method, however, does not generate a primal solution matrix X and their study did not generate cuts to the underlying combinatorial problems. Fujisawa et. al. [11] solved the relaxations of maximum cut and equal cut problems with up to 1250 vertices and maximum clique problems with up to 300 vertices. They used a primal dual interior point algorithm to solve the relaxations, but they did not generate cuts to the combinatorial problems either. After the initial release of this report, Burer and Monteiro [7] applied a projected gradient method to the the relaxation of maximum cut problems and applied the Geomans and Williamson randomized algorithm to generate cuts.

Our 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. Furthermore, this is the first computational study using the positive semidefinite relaxation to find approximate solutions to combinatorial problems with over 1000 vertices.

The test problems came from the set of graphs developed by Helmberg and Rendl and have also been used by Burer and Monteiro to test their algorithm. The collection of graphs was created from a machine independent graph generator written by G. Rinaldi, contains a mixture of toroidal, planar, and randomly generated graphs of varying sparsity, and is emerging as a benchmark set of graphs for combinatorial optimization. We selected three graphs of each type in the collection for a total of 27 graphs. The weights of the nonzero edges equal one unless otherwise specified. Of these graphs G1-G3 are randomly generated graphs with 800 vertices and nonzeros weights on 6.12% of the edges; G11 - G13 are 2 dimensional toroidal graphs with 800 vertices and nonzero edges randomly weighted to either negative or positive one; G14-G16 are planar graphs with 800 vertices and 1.59% nonzero edges; G22-G24 are randomly generated graphs with 2000 vertices and 1.05% nonzero edges; G32-G34 are toroidal graphs with 2000 vertices and nonzero edges randomly weighted to either negative or positive one; G35-G37 are planar graphs with 2000 vertices and 0.64% nonzero edges; G43 - G45 are randomly generated graphs with 1000 vertices and 2.10% nonzero edges; G48 - G50 are toroidal graphs with 3000 vertices nonzero edges equal to one; and G51-G53 are planar graphs with 1000 vertices and 1.28% nonzero edges.

The program was run on a PC (400MHZ, 512MB) running the Linux operating system. The semidefinite program stopped when $\frac{\bar{z}^k - b^T y^k}{(|b^T y^k| + |\bar{z}^k|)/2+1} < 10^{-3}$. For the equal cut and s-t cut problems, we used the penalty parameter $\Lambda = n - 1$. The unequal-cut problem was specified by $|e^T v| = n/10$ and the s-t cut problem used $s = n/4$ and $t = 3n/4$.

The results are contained in Tables 4 - 8. The first column of the tables is the problem index. The second column (n) shows the number of vertices in each graph, and the third column (S) is the percentage of nonzero weighted edges in the graphs. The fourth column (L) gives the percentage of nonzeros in the Cholesky factor the the dual matrix for this problem. This statistic is particular to our matrix ordering, but it gives a better indication of whether there exists sparsity in the problem that the program can exploit.

The fifth and sixth columns show the number of iterations (It) required to solve the positive semidefinite relaxation and the bound provided by the dual objective value (SDP). After solving the positive definite program, we used n different random vectors and each of the n rows of the primal matrix to generate approximate solutions to the combinatorial problem. Columns 7 an 8 represent the best answer provided by randomized (Rand) and X column (Xcol) techniques. The ratio $\max(Rand, Xcol)/SDP$ appears in the ninth column. If this ratio equals 1, then the approximate solution is 100% optimal. The seconds (SDP) required to solve the positive semidefinite program appears in the tenth column. The seconds required for each rank reduction technique appears in columns 11 and 12 (Rtime) (Xtime) and include the time spent factoring the one or two matrices needed to generate vectors $\hat{v}$.

Graph	n	Sparsity of S_{chol}	Seconds per Iterate				
			Max	s-t	Equal	Unequal	Box
G1	800	73.6%	21.4	21.5	21.6	22.0	21.4
G11	800	4.2%	2.6	2.6	2.6	2.6	2.6
G14	800	14.3%	4.9	5.0	4.9	5.1	5.1
G22	2000	47.8%	254.8	247.1	245.9	257.5	240.4
G32	2000	1.6%	33.5	33.6	34.1	34.1	33.6
G35	2000	11.8%	74.8	75.9	75.7	76.4	74.4

Table 1: Exploiting sparsity while solving SDP

Table 1 demonstrates how well the algorithm exploits sparsity. Graphs G1 and G11 have the same number of vertices and their maximum cut formulations have the same dimensions, but the Cholesky factor used when solving G11 is much more sparse than the Cholesky factor used to solve G1. Because sparsity can reduce the cost of matrix factorizations and triangular solves, G11 required only 2.6 seconds per iterate while G1 used 21.4 seconds per iterate.

Graph G22 has almost the same number of edges as G1, but it requires far more time. For these problems, edges affect the sparsity of the dual matrix, but the dimension of the problem is strictly a function of the number of vertices. Solving G22 as a max-cut problem requires the computation of matrix M with four times as many elements as the M created for G1. The matrix factorizations are third order operations whose cost dominates the computational complexity of each iterate.

Table 1 also demonstrates the value of treating dense constraints separately when the rest of the data in the problem has a sparse structure. If we had treated all the constraints in the equal cut problem G11 in an identical manner, the Cholesky factor would be completely dense and the time per iterate would exceed the 21.6 seconds required by a G1 iterate. Treating the dense constraint separately reduced the computation time to 2.6 seconds per iterate – almost the same time required to solve G11 as a maximum cut problem.

Graph	n	Sparsity	Max		Equal		Box	
		of S_{chol}	Rtime	Xtime	Rtime	Xtime	Rtime	Xtime
G1	800	73.6%	33	35	36	38	33	35
G11	800	4.2%	2	2	4	4	2	2
G14	800	14.3%	7	6	10	9	7	7
G22	2000	47.8%	344	389	363	408	346	390
G32	2000	1.6%	19	17	37	35	19	17
G35	2000	11.8%	84	93	102	111	84	93

Table 2: Rank reduction times (seconds)

We generated n approximate solutions using the randomized algorithm and n approximate solutions using the rows of the primal matrix. The times required by each of the two techniques were very close to one another and were about the same as one or two iterates of the positive semidefinite program. The quality of the solutions generated by the two techniques were also very similar. Each technique generated a better solution than the other in about half the problems. The answers for most problems were within 90% of the bound offered by the semidefinite relaxation. The ratios for a few problems (G11-G13, G32-G34), fell below 0.9 because all of these problems have negative edge weights. Since the bound is not necessarily tight, these approximate answers could even closer to the optimal answer than the ratio suggests.

Table 2 demonstrates several features of the rank reduction. First, it shows how our implementation of the randomized algorithm uses sparsity in the Cholesky factors. Even though the matrices X are totally dense, they have a special structure that can be exploited. Second, it shows that times to generate the various cuts are relatively similar. The randomized algorithms for equal cut require sorting a vector, but this procedure cost little. For the graphs with 800 vertices, sorting the vectors required about 2.5 seconds. This cost doubled the time for generating cuts for G11 but had relatively little effect on more dense problems like G1.

Graphs G11-G13, G32-G35, and G48-G50 are the toroidal graphs. These graphs are very sparse (with about $2n$ edges), and their special structure produces very sparse Cholesky factors. Because our implementation of the dual algorithm can take advantage of this sparse structure, toroidal graphs are solved much faster that other types of graphs. Furthermore, the cuts generated are of very high quality. For G48 and G49, the algorithm generated solutions that are 100% optimal. The structure of toroidal graphs has also been recognized by other researchers. De Simone et al. solve toroidal problems with 10,000 variables to optimality using an LP based branch and cut algorithm [8]. On a workstation will sufficient memory, we can also solve problems with 10,000 nodes. The "B" graphs in Table 6 are toroidal graphs. For these graphs, B1-B4 had positive edges while B5 had both positive and negative edges. Although it is not part of the Gset test problems, we used them to create a max-cut problem. On the toroidal graphs with positive edges, the problems were solved to optimality since the value of the best cut equaled the bound provided by the positive semidefinite relaxation. Problems B2, B3, and B5 were solved on a Hewlett-Packard computer (300 MHz) with 1 GB of RAM plus additional virtual memory. For these problems, the times for the two rank reduction techniques are the average of the two techniques. For graphs with 10,000 vertices, the matrix M alone uses over 800 MB RAM. The computing and factoring such a large matrix each iteration increased the computation time for this problem to over a day. Nevertheless, to our knowledge, these problems are some of the biggest maximum cut problems ever solved.

Problem				Objective Values					Seconds		
Name	n	S	L	It	SDP	Rand	Xcol	Ratio	SDP	Rand	Xcol
B1	5000	0.10%	1.16%	14	10,000.0	10,000	10,000	1.000	7,284	167	176
B2	6000	0.08%	1.01%	12	12,000.0	12,000	12,000	1.000	18,142	227	227
B3	7000	0.07%	0.96%	13	14,000.0	14,000	14,000	1.000	39,469	347	347
B4	8000	0.06%	0.86%	15	16,000.0	16,000	16,000	1.000	32,956	480	536
B5	10000	0.05%	0.69%	23	31,261.8	25,580	25,604	0.819	158,672	837	837

Table 3: Large toroidal graph maximum cut problems.

Problem				Objective Values					Seconds			
Name	n	S	L	It	SDP	Rand	Xcol	Ratio	SDP	Rand	Xcol	Total
G1	800	6.12%	73.71%	16	12083.2	11425	11421	0.946	343	33	35	415
G2	800	6.12%	74.10%	16	12089.5	11443	11399	0.947	347	34	35	420
G3	800	6.12%	74.18%	18	12084.4	11378	11416	0.945	395	34	36	468
G11	800	0.62%	2.60%	18	629.2	532	530	0.846	46	2	2	50
G12	800	0.62%	3.64%	21	623.9	530	528	0.850	57	2	2	62
G13	800	0.62%	4.37%	17	647.1	558	550	0.862	49	3	2	54
G14	800	1.59%	14.49%	24	3191.6	2981	2978	0.934	118	7	6	133
G15	800	1.58%	13.88%	31	3171.6	2959	2966	0.935	152	7	6	166
G16	800	1.58%	14.05%	24	3175.0	2970	2962	0.935	115	7	6	129
G22	2000	1.05%	47.88%	18	14136.0	12995	12991	0.919	4332	344	389	5082
G23	2000	1.05%	47.68%	17	14145.5	12970	12972	0.917	4051	340	384	4792
G24	2000	1.05%	48.29%	16	14140.9	12953	12963	0.917	3930	347	392	4685
G32	2000	0.25%	1.70%	17	1567.6	1306	1306	0.833	569	19	17	606
G33	2000	0.25%	1.85%	17	1544.3	1284	1276	0.831	575	20	18	614
G34	2000	0.25%	2.28%	17	1546.7	1290	1288	0.834	596	22	20	641
G35	2000	0.64%	11.86%	35	8014.8	7440	7447	0.929	2617	84	93	2800
G36	2000	0.64%	12.70%	30	8006.0	7438	7428	0.929	2360	91	101	2557
G37	2000	0.64%	12.24%	31	8018.7	7445	7458	0.930	2386	89	97	2577
G43	1000	2.10%	50.34%	15	7032.2	6492	6534	0.929	445	44	47	539
G44	1000	2.10%	50.46%	15	7027.9	6476	6501	0.925	442	44	47	535
G45	1000	2.10%	50.63%	20	7024.8	6490	6484	0.924	590	44	47	684
G48	3000	0.17%	1.73%	14	6000.0	6000	6000	1.000	1610	54	53	1722
G49	3000	0.17%	1.57%	14	6000.0	6000	6000	1.000	1585	51	50	1691
G50	3000	0.17%	1.28%	14	5988.2	5880	5880	0.982	1522	46	44	1617
G51	1000	1.28%	13.38%	23	4006.3	3749	3731	0.936	215	13	12	242
G52	1000	1.28%	13.65%	22	4009.7	3737	3739	0.932	206	13	12	232
G53	1000	1.28%	13.53%	24	4009.7	3754	3747	0.936	224	13	12	250

Table 4: Maximum-cut G-set problems.

Problem				Objective Values					Seconds			
Name	n	S	L	It	SDP	Rand	Xcol	Ratio	SDP	Rand	Xcol	Total
G1	800	6.12%	73.71%	16	12083.0	11402	11425	0.946	346	36	38	423
G2	800	6.12%	74.10%	18	12089.3	11396	11402	0.943	394	36	38	472
G3	800	6.12%	74.18%	18	12083.8	11353	11418	0.945	393	36	38	470
G11	800	0.62%	2.60%	17	629.2	534	532	0.849	44	4	4	53
G12	800	0.62%	3.64%	17	623.8	530	528	0.850	48	5	4	57
G13	800	0.62%	4.37%	17	647.1	554	550	0.856	49	5	5	59
G14	800	1.59%	14.49%	23	3189.9	2982	2986	0.936	112	10	9	131
G15	800	1.58%	13.88%	24	3171.1	2975	2971	0.938	116	9	8	134
G16	800	1.58%	14.05%	22	3174.6	2969	2965	0.935	110	9	9	129
G22	2000	1.05%	47.88%	19	14135.8	12959	12989	0.919	4673	363	408	5460
G23	2000	1.05%	47.68%	17	14145.4	12966	12970	0.917	4144	359	404	4924
G24	2000	1.05%	48.29%	21	14140.5	12977	12966	0.918	5223	366	410	6016
G32	2000	0.25%	1.70%	17	1567.6	1304	1308	0.834	580	37	35	653
G33	2000	0.25%	1.85%	17	1544.3	1282	1286	0.833	587	37	35	662
G34	2000	0.25%	2.28%	17	1546.7	1282	1284	0.830	612	40	38	692
G35	2000	0.64%	11.86%	31	8014.1	7449	7460	0.931	2346	102	111	2565
G36	2000	0.64%	12.70%	30	8005.0	7433	7419	0.929	2398	109	119	2631
G37	2000	0.64%	12.24%	30	8017.4	7447	7443	0.929	2339	106	116	2567
G43	1000	2.10%	50.34%	15	7032.1	6484	6533	0.929	453	47	51	554
G44	1000	2.10%	50.46%	20	7027.7	6488	6501	0.925	604	48	51	706
G45	1000	2.10%	50.63%	16	7024.6	6507	6481	0.926	477	48	51	579
G48	3000	0.17%	1.73%	14	6000.0	6000	6000	1.000	1659	94	94	1852
G49	3000	0.17%	1.57%	14	6000.0	6000	6000	1.000	1629	90	90	1814
G50	3000	0.17%	1.28%	14	5988.2	5880	5880	0.982	1566	82	81	1734
G51	1000	1.28%	13.38%	26	4005.6	3741	3736	0.934	255	17	17	290
G52	1000	1.28%	13.65%	23	4009.1	3729	3735	0.932	229	17	17	264
G53	1000	1.28%	13.53%	22	4009.2	3743	3747	0.935	217	17	16	251

Table 5: Equal-cut G-set problems.

Problem				Objective Values					Seconds			
Name	n	S	L	It	SDP	Rand	Xcol	Ratio	SDP	Rand	Xcol	Total
G1	800	6.12%	73.71%	21	11986.4	11316	11309	0.944	461	36	38	538
G2	800	6.12%	74.10%	21	11988.5	11322	11297	0.944	466	37	38	544
G3	800	6.12%	74.18%	21	11990.4	11344	11332	0.946	467	37	38	545
G11	800	0.62%	2.60%	36	629.0	524	528	0.839	95	4	4	104
G12	800	0.62%	3.64%	33	623.4	524	522	0.841	94	5	4	104
G13	800	0.62%	4.37%	33	646.8	556	548	0.860	99	5	5	109
G14	800	1.59%	14.49%	29	3190.2	2979	2975	0.934	147	10	9	166
G15	800	1.58%	13.88%	31	3169.4	2964	2960	0.935	153	10	9	172
G16	800	1.58%	14.05%	28	3172.9	2957	2955	0.932	139	9	9	158
G22	2000	1.05%	47.88%	45	14033.4	12866	12871	0.917	11568	361	406	12352
G23	2000	1.05%	47.68%	37	14041.3	12819	12873	0.917	9431	358	402	10209
G24	2000	1.05%	48.29%	40	14040.9	12862	12876	0.917	10340	364	411	11131
G32	2000	0.25%	1.70%	53	1566.8	1288	1294	0.826	1805	36	34	1878
G33	2000	0.25%	1.85%	55	1543.3	1274	1276	0.827	1900	37	35	1974
G34	2000	0.25%	2.28%	52	1545.7	1286	1272	0.832	1883	40	38	1963
G35	2000	0.64%	11.86%	62	8007.6	7432	7435	0.928	4735	102	111	4953
G36	2000	0.64%	12.70%	55	7997.6	7428	7425	0.929	4423	109	119	4656
G37	2000	0.64%	12.24%	57	8012.8	7449	7458	0.931	4464	106	116	4691
G43	1000	2.10%	50.34%	25	6980.7	6446	6463	0.926	766	47	51	868
G44	1000	2.10%	50.46%	29	6977.7	6395	6438	0.923	892	48	51	994
G45	1000	2.10%	50.63%	25	6974.1	6423	6431	0.922	768	48	51	871
G48	3000	0.17%	1.73%	41	5940.0	5400	5400	0.909	4704	94	94	4897
G49	3000	0.17%	1.57%	41	5940.0	5400	5400	0.909	4643	91	90	4828
G50	3000	0.17%	1.28%	38	5928.3	5400	5400	0.911	4303	82	81	4471
G51	1000	1.28%	13.38%	35	4002.6	3742	3733	0.935	344	17	16	378
G52	1000	1.28%	13.65%	57	4006.5	3724	3732	0.931	566	17	16	601
G53	1000	1.28%	13.53%	34	4005.7	3738	3733	0.933	338	17	16	372

Table 6: Unequal-cut G-set problems.

Problem				Objective Values					Seconds			
Name	n	S	L	It	SDP	Rand	Xcol	Ratio	SDP	Rand	Xcol	Total
G1	800	6.12%	73.71%	20	-9980.8	-7294.1	-7330.1	0.734	427	33	35	499
G2	800	6.12%	74.10%	21	-10005.7	-7337.9	-7238.0	0.733	459	34	35	531
G3	800	6.12%	74.18%	20	-9985.4	-7329.9	-7345.9	0.736	433	34	35	506
G11	800	0.62%	2.60%	15	-2448.7	-2059.5	-2059.5	0.841	39	2	2	43
G12	800	0.62%	3.64%	15	-2503.5	-2103.6	-2119.6	0.847	42	2	2	47
G13	800	0.62%	4.37%	15	-2520.5	-2139.6	-2131.6	0.849	43	3	2	48
G14	800	1.59%	14.49%	31	-3378.3	-2551.1	-2523.1	0.755	158	7	7	172
G15	800	1.58%	13.88%	31	-3364.2	-2589.1	-2545.1	0.770	151	7	6	165
G16	800	1.58%	14.05%	39	-3356.1	-2587.3	-2515.3	0.771	189	7	6	203
G22	2000	1.05%	47.88%	22	-16563.8	-11945.3	-11989.3	0.724	5289	346	390	6040
G23	2000	1.05%	47.68%	21	-16602.1	-11925.1	-11909.1	0.718	4973	343	385	5717
G24	2000	1.05%	48.29%	23	-16583.4	-11959.5	-11855.6	0.721	5774	349	394	6534
G32	2000	0.25%	1.70%	14	-6226.6	-5177.8	-5169.8	0.832	471	19	17	508
G33	2000	0.25%	1.85%	16	-6237.3	-5169.8	-5153.8	0.829	545	20	18	585
G34	2000	0.25%	2.28%	19	-6282.8	-5245.9	-5253.9	0.836	666	22	20	711
G35	2000	0.64%	11.86%	43	-8503.0	-6230.4	-6230.4	0.733	3200	84	93	3382
G36	2000	0.64%	12.70%	44	-8491.9	-6233.6	-6177.6	0.734	3429	91	101	3625
G37	2000	0.64%	12.24%	47	-8504.5	-6223.6	-6251.6	0.735	3573	89	98	3764
G43	1000	2.10%	50.34%	18	-8148.9	-6074.4	-6154.4	0.755	522	43	46	614
G44	1000	2.10%	50.46%	19	-8131.6	-6037.2	-6017.2	0.742	561	44	47	654
G45	1000	2.10%	50.63%	18	-8119.1	-5978.4	-5954.4	0.736	530	44	47	623
G48	3000	0.17%	1.73%	13	-12000.0	-11996.7	-11996.7	1.000	1468	55	55	1582
G49	3000	0.17%	1.57%	13	-12000.0	-11996.7	-11996.7	1.000	1452	52	50	1558
G50	3000	0.17%	1.28%	13	-11952.7	-11515.3	-11515.3	0.963	1403	47	44	1498
G51	1000	1.28%	13.38%	36	-4207.0	-3157.4	-3105.4	0.750	342	13	12	369
G52	1000	1.28%	13.65%	35	-4206.6	-3139.1	-3127.1	0.746	340	13	12	367
G53	1000	1.28%	13.53%	30	-4210.9	-3187.1	-3167.1	0.757	282	13	12	308

Table 7: Box QP G-set problems.

Problem				Objective Values					Seconds			
Name	n	S	L	It	SDP	Rand	Xcol	Ratio	SDP	Rand	Xcol	Total
G1	800	6.12%	73.71%	17	12081.8	11376	11414	0.945	365	33	35	437
G2	800	6.12%	74.10%	16	12088.0	11413	11407	0.944	355	34	35	428
G3	800	6.12%	74.18%	19	12083.4	11409	11429	0.946	414	34	36	488
G11	800	0.63%	2.60%	17	629.2	534	528	0.849	44	2	2	48
G12	800	0.63%	3.66%	17	623.8	526	530	0.850	46	2	2	51
G13	800	0.63%	4.25%	17	647.0	554	552	0.856	47	3	2	53
G14	800	1.59%	14.49%	21	3190.9	2982	2973	0.935	104	7	6	118
G15	800	1.58%	13.89%	24	3171.4	2979	2967	0.939	116	7	6	130
G16	800	1.59%	14.07%	24	3174.8	2982	2963	0.939	116	7	6	129
G22	2000	1.05%	47.88%	16	14133.8	12963	12989	0.919	3954	341	387	4699
G23	2000	1.05%	47.68%	17	14144.5	12967	12975	0.917	4157	341	386	4901
G24	2000	1.05%	48.29%	16	14140.2	12981	12964	0.918	3942	348	394	4702
G32	2000	0.25%	1.71%	18	1567.6	1300	1304	0.832	604	19	17	642
G33	2000	0.25%	1.78%	18	1544.3	1280	1284	0.831	609	20	17	648
G34	2000	0.25%	2.32%	18	1546.6	1286	1282	0.831	638	23	20	683
G35	2000	0.64%	11.86%	33	8014.1	7443	7443	0.929	2504	85	94	2689
G36	2000	0.64%	12.70%	34	8005.8	7439	7439	0.929	2722	92	101	2920
G37	2000	0.64%	12.22%	33	8018.2	7454	7460	0.930	2563	89	98	2756
G43	1000	2.10%	50.34%	16	7030.6	6491	6521	0.928	477	44	47	571
G44	1000	2.10%	50.46%	17	7024.5	6495	6498	0.925	514	44	47	608
G45	1000	2.10%	50.79%	15	7023.5	6494	6482	0.925	455	44	47	550
G48	3000	0.17%	1.64%	15	6000.0	6000	6000	1.000	1727	54	52	1837
G49	3000	0.17%	1.48%	15	6000.0	6000	6000	1.000	1682	50	48	1785
G50	3000	0.17%	1.37%	14	5987.9	5870	5868	0.980	1557	49	46	1657
G51	1000	1.28%	13.38%	25	4005.9	3742	3745	0.935	242	13	12	268
G52	1000	1.28%	13.65%	23	4009.4	3734	3746	0.934	219	13	12	246
G53	1000	1.28%	13.53%	24	4009.6	3754	3747	0.936	226	13	12	253

Table 8: $s-t$-cut G-set problems ($s = n/4,\ t = 3n/4$).

References

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

[2] 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.

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

[4] Steven J. Benson, Yinyu Ye, and Xiong Zhang. "DSDP user's guide," http://dollar.biz.uiowa.edu/col/, January 1999.

[5] S. Benson, Y. Ye, and X. Zhang, "Solving sparse, large scale positive semidefinite programs," Department of Management Science, The University of Iowa, Iowa City, 1997, To appear in *SIAM J. of Optimization.*

[6] D. Bertsimas and Y. Ye, "Semidefinite relaxations, multivariate normal distributions, and order statistics," Working Paper, Sloan School, MIT, December 1997.

[7] S. Burer and R. D. C. Monteiro. "An efficient algorithm for solving the MAXCUT SDP relaxation", Working Paper, School of ISyE, Georgia Tech, submitted to *Mathematical Programming*, December 1998.

[8] De Simone and et al. "Exact ground states of Ising spin glasses: new experimental results with branch-and-cut algorithm," *Journal of Statistical Physics* 80(1/2):487–486, 1995.

[9] A. Frieze and M. Jerrum, "Improved approximation algorithms for max k-cut and max bisection," *Proc. 4th IPCO Conference* (1995) 1-13.

[10] 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.*

[11] 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.

[12] 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.

[13] 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.

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

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

[17] Yu. E. Nesterov, "Semidefinite relaxation and nonconvex quadratic optimization," *Optimization Methods and Software*, 9:141–160, 1998.

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

[19] 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.

[20] 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, 1997.

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

[22] Y. Ye, "Approximating quadratic programming with bound and quadratic constraints," *Mathematical Programming* 84 (1999) 219-226.

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