Math. Program., Ser. B 95: 359–379 (2003)

Digital Object Identifier (DOI) 10.1007/s10107-002-0353-7

Samuel Burer[⋆] · Renato D.C. Monteiro[⋆⋆] · Yin Zhang[⋆⋆⋆]

A computational study of a gradient-based log-barrier algorithm for a class of large-scale SDPs[†]

Received: June 22, 2001 / Accepted: January 20, 2002
Published online: December 9, 2002 – © Springer-Verlag 2002

Abstract. The authors of this paper recently introduced a transformation [4] that converts a class of semidefinite programs (SDPs) into nonlinear optimization problems free of matrix-valued constraints and variables. This transformation enables the application of nonlinear optimization techniques to the solution of certain SDPs that are too large for conventional interior-point methods to handle efficiently. Based on the transformation, we proposed a globally convergent, first-order (i.e., gradient-based) log-barrier algorithm for solving a class of linear SDPs. In this paper, we discuss an efficient implementation of the proposed algorithm and report computational results on semidefinite relaxations of three types of combinatorial optimization problems. Our results demonstrate that the proposed algorithm is indeed capable of solving large-scale SDPs and is particularly effective for problems with a large number of constraints.

Key words. semidefinite program – semidefinite relaxation – nonlinear programming – interior-point methods – limited memory quasi-Newton methods.

1. Introduction

It is well-known in optimization that first-order methods, i.e., those that use only gradient information to calculate their iterates, typically require a large number of iterations to reach a high accuracy, while second-order methods, i.e., those that also use Hessian information, attain the same accuracy in far fewer iterations. On the other hand, iterations of first-order methods are typically much faster than those of the second-order methods.

For many problems, second-order approaches are favored over first-order approaches since a small number of expensive iterations may be less expensive in total than a large

S. Burer: Department of Management Sciences, University of Iowa, Iowa City, IA 52242-1000, USA, e-mail: samuel-burer@uiowa.edu

R.D.C. Monteiro: School of ISyE, Georgia Institute of Technology, Atlanta, Georgia 30332, USA, e-mail: monteiro@isye.gatech.edu

Y. Zhang: Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005, USA, e-mail: zhang@caam.rice.edu

Mathematics Subject Classification (1991): 90C06, 90C27, 90C30.

⋆ This author was supported in part by NSF Grants CCR-9902010, INT-9910084 and CCR-0203426

⋆⋆ This author was supported in part by NSF Grants CCR-9902010, INT-9910084 and CCR-0203113

⋆⋆⋆ This author was supported in part by DOE Grant DE-FG03-97ER25331, DOE/LANL Contract 03891-99-23 and NSF Grant DMS-9973339.

† Computational results reported in this paper were obtained on an SGI Origin2000 computer at Rice University acquired in part with support from NSF Grant DMS-9872009.

number of inexpensive iterations. For other problems, the reverse is true. Clearly, the relative advantages and disadvantages of the two should be determined on a case-by-case basis.

For semidefinite programming, the second-order interior-point methods (either primal-dual or dual-scaling) have proven to be very robust for solving small- to medium-sized problems to high accuracy. On large-scale problems, however, their performance has been mostly discouraging because the cost per iteration for those methods increases dramatically with the problem size. In fact, on many problems these methods are inappropriate for obtaining even low accuracy solutions. In contrast, first-order methods have proven capable of obtaining moderate accuracy in a reasonable amount of time for large-scale problems [2, 15, 16].

Based on a nonlinear transformation, we recently proposed a first-order, log-barrier method for solving a class of large-scale SDPs and established its global convergence [4]. The main purpose of this paper is to study the implementation issues for this algorithm and to report our computational results.

This paper is organized as follows. In Section 2, we introduce the class of SDPs to be considered and describe three types of such SDPs that will be used to test the performance of our algorithm. In Section 3, we will introduce the aforementioned nonlinear transformation, describe the log-barrier algorithm, and state essential theoretical results obtained in [4] that are necessary for understanding the properties of the algorithm. In Section 4, we discuss issues involved in our implementation and experimentation, and report our numerical results. Finally, we conclude the paper in Section 5.

1.1. Preliminary notation and terminology

In this paper, $\Re$, $\Re^n$, and $\Re^{n\times n}$ denote the space of real numbers, real n-dimensional column vectors, and real $n \times n$ matrices, respectively. By $\mathcal{S}^n$ we denote the space of real $n \times n$ symmetric matrices, and we define $\mathcal{S}^n_+$ and $\mathcal{S}^n_{++}$ to be the subsets of $\mathcal{S}^n$ consisting of the positive semidefinite and positive definite matrices, respectively. We write $A \succeq 0$ and $A \succ 0$ to indicate that $A \in \mathcal{S}^n_+$ and $A \in \mathcal{S}^n_{++}$, respectively. We let $\mathrm{tr}(A)$ denote the trace of a matrix $A \in \Re^{n\times n}$, namely $\mathrm{tr}(A)$ denotes the sum of the diagonal elements of A. Moreover, for $A, B \in \Re^{n\times n}$, we define $A \bullet B \equiv \mathrm{tr}(A^T B)$. If $\mathcal{I}$ is a finite set, we let $|\mathcal{I}|$ denote its cardinality, that is, the number of elements of $\mathcal{I}$. $\mathcal{L}^n$ denotes the space of real $n \times n$ lower triangular matrices, and $\mathcal{L}^n_{++}$ are the subset of $\mathcal{L}^n$ consisting of those matrices with positive diagonal entries. In addition, we define $\mathcal{L}^n_0 \subset \mathcal{L}^n$ to be the set of all $n \times n$ strictly lower triangular matrices.

2. The SDP problem and three examples

In this section, we introduce the class of SDP problems to be studied in this paper and describe three subclasses arising from semidefinite relaxations of combinatorial optimization problems. Test instances will be chosen from these three subclasses to test the performance of our algorithm. The key characteristic of the class under consideration is that the diagonal of the primal variable X is fixed. We note that the SDP problem considered here is slightly more general than the class studied in [4]. With some minimal adjustments, however, all the results in [4] still apply in the current context.

2.1. The primal-dual SDP pair

Consider the primal SDP

$$(P) \qquad \max \{C \bullet X : \operatorname{diag}(X) = d,\ \mathbf{A}(X) = b,\ \mathbf{G}(X) \leq h,\ X \succeq 0\},$$

where the variable is $X \in \mathcal{S}^n$ and the data consist of the matrix $C \in \mathcal{S}^n$, the vectors $d \in \Re^n_+$, $b \in \Re^m$, and $h \in \Re^p$, and the linear maps $\mathbf{A} : \mathcal{S}^n \to \Re^m$ and $\mathbf{G} : \mathcal{S}^n \to \Re^p$. It is well-known that there exist unique matrices $A_1, \ldots, A_m \in \mathcal{S}^n$ and $G_1, \ldots, G_p \in \mathcal{S}^n$ such that, for all $X \in \mathcal{S}^n$, there hold $[\mathbf{A}(X)]_i = A_i \bullet X$ for $i = 1, \ldots, m$ and $[\mathbf{G}(X)]_j = G_j \bullet X$ for $j = 1, \ldots, p$. Aside from the primal inequality constraints, (P) differs from the standard form primal SDP only by the constraint $\operatorname{diag}(X) = d$.

The dual to (P) is the problem

$$(D) \qquad \min \left\{ d^T z + b^T y + h^T u : \operatorname{Diag}(z) + \mathbf{A}^*(y) + \mathbf{G}^*(u) - C = S,\ u \geq 0,\ S \succeq 0 \right\},$$

where $(z, y, u, S) \in \Re^n \times \Re^m \times \Re^p \times \mathcal{S}^n$ are the dual variables and where $\mathbf{A}^* : \Re^m \to \mathcal{S}^n$ and $\mathbf{G}^* : \Re^p \to \mathcal{S}^n$ are the adjoints of the operators $\mathbf{A}$ and $\mathbf{G}$, which in terms of the matrices $\{A_i\}_{i=1}^m$ and $\{G_j\}_{j=1}^p$ are given by $\mathbf{A}^*(y) = \sum_{i=1}^m y_i A_i$ for all $y \in \Re^m$ and $\mathbf{G}^*(u) = \sum_{j=1}^p u_j G_j$ for all $u \in \Re^p$.

We denote by $\mathcal{F}^0(P)$ and $\mathcal{F}^0(D)$ the sets of interior feasible solutions for problems (P) and (D), respectively, i.e.,

$$\begin{aligned} \mathcal{F}^0(P) &\equiv \{X \in \mathcal{S}^n_{++} : \operatorname{diag}(X) = d,\ \mathbf{A}(X) = b,\ \mathbf{G}(X) < h\}, \\ \mathcal{F}^0(D) &\equiv \{(z, y, u, S) \in \Re^n \times \Re^m \times \Re^p_{++} \times \mathcal{S}^n_{++} : \\ &\qquad \operatorname{Diag}(z) + \mathbf{A}^*(y) + \mathbf{G}^*(u) - C = S\}. \end{aligned}$$

Note that $\mathcal{F}^0(D)$ is nonempty since, for any y and any $u > 0$, z can be chosen so that the resulting matrix S is positive definite. In addition, the term $\operatorname{Diag}(z)$ found in the equality constraint of problem (D) can be rewritten as $\sum_{i=1}^n z_i (e_i e_i^T)$, where $e_i \in \Re^n$ is the vector having all zeros except a one in the i-th position. We make the following assumptions throughout our presentation.

Assumption A.1. $\mathcal{F}^0(P) \neq \emptyset$.

Assumption A.2. The matrices $\{e_i e_i^T\}_{i=1}^n \cup \{A_j\}_{j=1}^m \cup \{G_\ell\}_{\ell=1}^p$ are linearly independent.

As mentioned above, problem (P) is specialized from the usual standard form problem by the constraint $\operatorname{diag}(X) = d$. Nonetheless, this constraint arises naturally from semidefinite relaxations of quadratic integer programs in binary (or ± 1) variables. Given an n-dimensional variable x such that $x_i \in \{-1, 1\}$ for each $i = 1, \ldots, n$, the products $x_i x_j$ for all pairs (i, j) can be conveniently represented as the elements of the rank-one matrix $xx^T \in \mathcal{S}^n_+$ for which $\operatorname{diag}(xx^T) = e$, where $e \in \Re^n$ is the vector of all ones.

A semidefinite relaxation is then obtained by replacing xx^T with the variable matrix $X \in \mathcal{S}^n_+$ and requiring $\mathrm{diag}(X) = e$, i.e., by dropping the rank-one restriction.

Numerous combinatorial optimization problems can be cast as quadratic integer programs in ± 1 variables, and in particular, many graph theoretic optimization problems can be stated in this way. Each such problem thus has a semidefinite relaxation in the form of (P), and in this paper, we focus on three specific examples of SDP relaxations of graph optimization problems.

For the examples below, let $G = (V, E)$ be an undirected graph with vertex set $V = \{1, \ldots, r\}$ and edge set $E \subseteq V \times V$, and let $W \in \mathcal{S}^r$ be a weight matrix for G such that $w_{ij} = w_{ji}$ is the weight associated with edge $(i, j) \in E$. For those edges $(i, j) \notin E$, we assume $w_{ij} = w_{ji} = 0$. In addition, we define the Laplacian matrix of G with weight matrix W as

$$L(G, W) \equiv \mathrm{Diag}(We) - W. \tag{1}$$

2.2. *Maximum cut relaxation*

The maximum cut problem on G is to find a bipartition (V_1, V_2) of V that maximizes the sum of the weights of the edges with one vertex in V_1 and the other in V_2. The maximum cut problem (or simply "maxcut," for short) is a well-known NP hard combinatorial optimization problem that can be cast as a quadratic integer program in ± 1 variables. Its SDP relaxation, which was first given by Goemans and Williamson in [13], is

$$\max \left\{ \frac{1}{4} L(G, W) \bullet X : \mathrm{diag}(X) = e,\, X \succeq 0 \right\}, \tag{2}$$

which is in the form of (P) with $n = r$ and $\mathbf{A}$ and $\mathbf{G}$ nonexistent. Note that Assumption A.1 is satisfied for the maxcut SDP relaxation since the $n \times n$ identity matrix I is strictly feasible. In addition, Assumption A.2 is trivially satisfied.

2.3. *Maximum stable set relaxation*

The maximum stable set problem on G is to find a subset $V_0 \subseteq V$ of maximum size such that no two vertices in V_0 are adjacent. The maximum stable set problem can also be formulated as an NP hard quadratic integer program in ± 1 variables. The number of binary variables in the integer program is $r + 1$, i.e., one more than the number of vertices in the graph, and hence the SDP relaxation has a matrix variable X of size $(r + 1) \times (r + 1)$. Let

$$C = \frac{1}{4} \begin{bmatrix} 2I_r & e \\ e^T & 0 \end{bmatrix} \in \mathcal{S}^{r+1},$$

where I_r is the identity matrix of dimension r and e is the vector of all ones in $\Re^r$, and let $A_{ij} = (e_i + e_j + e_{r+1})(e_i + e_j + e_{r+1})^T$ where e_ℓ is the ℓ-th coordinate vector in $\Re^{r+1}$. Then the SDP relaxation of the maximum stable set problem is given by

$$\max\{C \bullet X : \mathrm{diag}(X) = e,\, A_{ij} \bullet X = 1 \;\forall\, (i, j) \in E,\, X \succeq 0\}, \tag{3}$$

which is in the form of (P) with $n = r + 1$ and $m = |E|$ (see [18]). The optimal value of the above SDP relaxation is called the Lovász theta number of G and is denoted by $\vartheta(G)$. Therefore, the SDP (3) is called a Lovász theta SDP (there are other equivalent forms). The SDP also satisfies Assumption A.1 (see Theorem 1 of [1]) and clearly satisfies A.2.

2.4. Frequency assignment relaxation

Frequency assignment problems arise in wireless communication networks (see [9, 10], for example). Given a network represented by a graph G and an edge-weight matrix W, a certain type of frequency assignment problem on G can be formulated as the following maximum k-cut problem (see [11, 20]):

$$\begin{aligned}\max\ & \left[\left(\frac{k-1}{2k}\right) L(G, W) - \frac{1}{2}\,\mathrm{Diag}(We)\right] \bullet X \\ \text{s.t.}\ & -E^{ij} \bullet X \le 2/(k-1)\ \ \forall\ (i,j) \\ & -E^{ij} \bullet X = 2/(k-1)\ \ \forall\ (i,j) \in U \subseteq E \\ & \mathrm{diag}(X) = e,\ \ X \succeq 0,\ \ \mathrm{rank}(X) = k,\end{aligned} \tag{4}$$

where $k > 1$ is an integer, $L(G, W)$ is the Laplacian matrix as defined in (1), and $E^{ij} = e_i e_j^T + e_j e_i^T$. In this k-cut problem, nodes incident to a given edge subset U must be partitioned into different sets.

The above maximum k-cut problem can be relaxed into the following SDP:

$$\begin{aligned}\max\ & \left[\left(\frac{k-1}{2k}\right) L(G) - \frac{1}{2}\,\mathrm{Diag}(We)\right] \bullet X \\ \text{s.t.}\ & -E^{ij} \bullet X \le 2/(k-1)\ \ \forall\ (i,j) \in E \setminus U \\ & -E^{ij} \bullet X = 2/(k-1)\ \ \forall\ (i,j) \in U \subseteq E \\ & \mathrm{diag}(X) = e,\ \ X \succeq 0,\end{aligned} \tag{5}$$

where the rank restriction and the inequality constraints for the non-edges are dropped. The resulting SDP (5) is in the form of (P) with $n = |V|$, $m = |U|$ and $p = |E \setminus U|$. It is a property of this SDP that one may assume $w_{ij} = 0$ for all $(i, j) \in U$. Moreover, it is clear that (5) satisfies Assumption A.2, and it is also known that Assumption A.1 is not satisfied in general, that is, (5) does not necessarily have an interior feasible solution. It can be proven [8], however, that if the subgraph $H = (V, U)$ of G is $(k-1)$-colorable, then an interior feasible solution does exist. We note that all the instances of the frequency assignment problem used in our computational experiments in Section 4 do have interior feasible points.

3. A first-order log-barrier algorithm

In this section, we introduce the framework of our first-order, log-barrier algorithm based on a special nonlinear transformation that converts the dual SDP to a nonlinear program with very simple constraints. We will also state the gradient formulas and a global convergence result for this algorithm. All the stated theoretical results have essentially been proven in [4]. We include them here in order to make the present paper reasonably self-contained.

3.1. Standard log-barrier subproblems

Under Assumptions A.1 and A.2, it is well-known that for any $\nu > 0$ the standard dual log-barrier subproblem for (D),

$$(D_\nu) \quad \min \left\{ d^T z + b^T y + h^T u - \nu \log(\det S) - \nu \sum_{j=1}^{p} \log u_j \,:\, (z, y, u, S) \in \mathcal{F}^0(D) \right\},$$

and the standard primal log-barrier subproblem for (P),

$$(P_\nu) \quad \max \left\{ C \bullet X + \nu \log(\det X) + \nu \sum_{j=1}^{p} \log(h_j - G_j \bullet X) \,:\, X \in \mathcal{F}^0(D) \right\},$$

have unique optimal solutions $(z_\nu, y_\nu, u_\nu, S_\nu) \in \mathcal{F}^0(D)$ and $X_\nu \in \mathcal{F}^0(P)$, respectively. Moreover, together they satisfy

$$X_\nu S_\nu = \nu I \quad \text{and} \quad u_\nu * (h - \mathbf{G}(X_\nu)) = \nu e,$$

where the operation $*$ defines the Hadamard product and $e \in \Re^p$ is the vector of all ones. The paths $\{(z_\nu, y_\nu, u_\nu, S_\nu) : \nu > 0\}$ and $\{X_\nu : \nu > 0\}$ are called dual and primal central paths, respectively, and each tends to a dual and a primal optimal solution, respectively, as ν goes to zero.

A classic dual log-barrier algorithm is one that approximately solves a sequence of dual log-barrier subproblems (D_ν) corresponding to a set of decreasing ν values.

3.2. The nonlinear programming formulation

Our log-barrier algorithm is a dual algorithm. However, instead of solving (D) directly, we will first employ a nonlinear transformation to map the interior feasible set $\mathcal{F}^0(D) \subset \Re^n \times \Re^m \times \Re^p_{++} \times \mathcal{S}^n_{++}$ into the set $\Re^n_{++} \times \Re^m \times \Re^p_{++}$, and then apply the log-barrier approach to the resulting nonlinear optimization problem in the transformed space. These ideas were first introduced in [4] and [3]. Recall the definition of the interior feasible set for (D):

$$\mathcal{F}^0(D) \equiv \{(z, y, u, S) \in \Re^n \times \Re^m \times \Re^p_{++} \times \mathcal{S}^n_{++} : \operatorname{Diag}(z) + \mathbf{A}^*(y) + \mathbf{G}^*(u) - C = S\}.$$

The transformation from (D) to a nonlinear optimization problem consists of two stages. The first stage is derived from the well-known fact that every $S \in \mathcal{S}^n_{++}$ can be uniquely factored into the product LL^T, where L is an $n \times n$ lower triangular matrix with a positive diagonal, i.e., $L \in \mathcal{L}^n_{++}$. Applying this idea to the equality constraint of (D), we easily see that $\mathcal{F}^0(D)$ is in bijective correspondence with the set

$$\{(z, y, u, L) \in \Re^n \times \Re^m \times \Re^p_{++} \times \mathcal{L}^n_{++} : \mathrm{Diag}(z) + \mathbf{A}^*(y) + \mathbf{G}^*(u) - C = LL^T\}, \tag{6}$$

where the numbers of variables and equations remain unchanged after the change of variables from S to L. In order to describe the second stage of the transformation, we decompose the variable $L \in \mathcal{L}^n_{++}$ into

$$L = \mathrm{Diag}(w) + L_0,$$

where $w \in \Re^n_{++}$ and $L_0 \in \mathcal{L}^n_0$ (i.e., it is strictly lower triangular). As such, the set (6) can be rewritten as

$$\begin{aligned}\{(z, y, u, w, L_0) \in \Re^n \times \Re^m \times \Re^p_{++} \times \Re^n_{++} \times \mathcal{L}^n_0 :\\ \mathrm{Diag}(z) + \mathbf{A}^*(y) + \mathbf{G}^*(u) - C = (\mathrm{Diag}(w) + L_0)(\mathrm{Diag}(w) + L_0)^T\}.\end{aligned} \tag{7}$$

Counting the number of variables and equations in the symmetric equality system of (7), we see that there are a total of $n + m + p + n(n + 1)/2$ variables and $n(n + 1)/2$ equations. A fundamental observation is that the $n(n+1)/2$ equations in (7) can be used to eliminate $n(n + 1)/2$ variables, leaving $n + m + p$ variables and no equations. More specifically, through the equations in (7) the variables z and L_0 can be explicitly defined as functions of the variables w, y and u. Consequently, the sets (7) and $\mathcal{F}^0(D)$ are in bijective correspondence with the set $\Re^n_{++} \times \Re^m \times \Re^p_{++}$.

The key results of the two-stage transformation just described are given in the following theorem.

Theorem 1. *The following statements hold:*

(a) for each $(w, y, u) \in \Re^n_{++} \times \Re^m \times \Re^p_{++}$, there exists a unique $(z, L_0) \in \Re^n \times \mathcal{L}^n_0$ such that

$$\mathit{Diag}(z) + \mathbf{A}^*(y) + \mathbf{G}^*(u) - C = (\mathit{Diag}(w) + L_0)\,(\mathit{Diag}(w) + L_0)^T\;; \tag{8}$$

(b) the functions $L_0(w, y, u)$ and $z(w, y, u)$ defined according to (8) are each infinitely differentiable on their domain $\Re^n_{++} \times \Re^m \times \Re^p_{++}$;

(c) the sets $\Re^n_{++} \times \Re^m \times \Re^p_{++}$ and $\mathcal{F}^0(D)$ are in bijective correspondence according to the assignment $(w, y, u) \mapsto (z, y, u, S)$ where $z \equiv z(w, y, u)$ and $S \equiv S(w, y, u)$ and where $S(w, y, u) = L(w, y, u)L(w, y, u)^T$ and $L(w, y, u) \equiv \mathit{Diag}(w) + L_0(w, y, u)$.

As an immediate consequence of Theorem 1, the problem obtained from (D) by restricting the feasible region to the set $\mathcal{F}^0(D)$ can be recast as the nonlinear program

$$(NLD) \qquad \inf\{f(w, y, u) : (w, y, u) \in \Re^n_{++} \times \Re^m \times \Re^p_{++}\},$$

where $f : \Re^n_{++} \times \Re^m \times \Re^p_{++} \to \Re$ is defined by

$$f(w, y, u) = d^T z(w, y, u) + b^T y + h^T u. \tag{9}$$

The transformed dual problem (NLD) differs from the original dual problem (D) in that the nonlinearity has been shifted from the constraints to the objective function. As a result, the feasible region for the transformed problem becomes extremely simple. This shift is likely to have a significant impact on the behavior of the log-barrier approach.

We note that problems (D) and (NLD) have the same optimal values as was shown in [4], but (NLD) has an open feasible set and in general does not have an optimal solution. In fact, it can be easily seen that if $(d, b, h) \neq 0$, then all optimal solutions of (D) lie in the boundary of $\mathcal{F}^0(D)$, and in this case (NLD) does not have an optimal solution. This is not a practical limitation, however, since the algorithm for (NLD) described in this paper maintains its iterates in the open set $\Re^n_{++} \times \Re^m \times \Re^p_{++}$, only approaching the boundary of the feasible set in the limit.

3.3. The transformed dual log-barrier subproblem

After the transformation, the dual log-barrier subproblem (D_ν) becomes the following "nonlinear" dual log-barrier subproblem:

$$(NLD_\nu) \quad \min$$
$$\left\{ f(w, y, u) - 2\nu \sum_{i=1}^{n} \log w_i - \nu \sum_{j=1}^{p} \log u_j \,:\, (w, y, u) \in \Re^n_{++} \times \Re^m \times \Re^p_{++} \right\},$$

where we have used the equalities $\det(S) = \det(LL^T) = (\det(L))^2 = (\prod_{i=1}^n w_i)^2$.

The following theorem is proven in [4] which details the relationship between the transformed dual log-barrier subproblem (NLD_ν) and the original dual log-barrier subproblem (D_ν). It indicates that, although convexity may have been lost during the transformation, the benefits of convexity are basically intact.

Theorem 2. *For each $\nu > 0$, problem (NLD_ν) has a unique minimum (w_ν, y_ν, u_ν), which is also its unique stationary point. This minimum (w_ν, y_ν, u_ν) is equal to the inverse image of the minimum $(z_\nu, y_\nu, u_\nu, S_\nu)$ of (D_ν) under the bijective correspondence of Theorem 1. In particular, we have $z(w_\nu, y_\nu, u_\nu) = z_\nu$ and $S(w_\nu, y_\nu, u_\nu) = S_\nu$.*

Theorem 2 ensures in a theoretical sense that a log-barrier algorithm based on solving a sequence of (NLD_ν) will be well-behaved.

3.4. Gradient formulas and more

Since our algorithm will need to use the first derivative information of the function f, we restate the formulas for the gradient of f which were derived in [4]. In particular, we show that, for each $(w, y, u) \in \Re^n_{++} \times \Re^m \times \Re^p_{++}$, the gradient formula $\nabla f(w, y, u)$ is based on a certain symmetric matrix $X(w, y, u)$ that serves as a primal estimate for the problem (P).

Associated with a point $(w, y, u) \in \Re^n_{++} \times \Re^m \times \Re^p_{++}$, as in Theorem 1 we define

$$L(w, y, u) \equiv \mathrm{Diag}(w) + L_0(w, y, u) \in \mathcal{L}^n_{++}, \tag{10}$$

$$S(w, y, u) \equiv L(w, y, u)L(w, y, u)^T \in \mathcal{S}^n_{++}. \tag{11}$$

In the following theorem, we summarize the main results of [4] concerning the first derivative of $f(w, y, u)$.

Theorem 3. *Let* $(w, y, u) \in \Re^n_{++} \times \Re^m \times \Re^p_{++}$ *be given and define* $L \equiv L(w, y, u)$*. Then the system of linear equations*

$$diag(X) = d, \quad [XL]_{ij} = 0 \ \ \forall i > j,$$

has a unique solution in $\mathcal{S}^n$*, which we denote by* $X(w, y, u)$*. Moreover, the matrix* $X \equiv X(w, y, u)$ *satisfies*

(a) $\nabla_w f(w, y, u) = 2\, diag(XL)$*,*
(b) $\nabla_y f(w, y, u) = b - \mathbf{A}(X)$*,*
(c) $\nabla_u f(w, y, u) = h - \mathbf{G}(X)$*.*

The following corollary of Theorem 3, which is a slight adaptation of Lemma 5 and Theorem 5 of [4], shows that the matrix $X(w, y, u)$ plays the role of a (possibly infeasible) primal estimate for any feasible point (w, y, u).

Corollary 1. *Let* $(w, y, u) \in \Re^n_{++} \times \Re^m \times \Re^p_{++}$*, and define* $L \equiv L(w, y, u)$*,* $S \equiv S(w, y, u)$*,* $X \equiv X(w, y, u)$ *and* $\nabla f \equiv \nabla f(w, y, u)$*. Then:*

(a) X *is positive semidefinite (definite) if and only if* $\nabla_w f$ *is nonnegative (positive);*
(b) $\mathbf{A}(X) = b$ *if and only if* $\nabla_y f = 0$*;*
(c) $h - \mathbf{G}(X)$ *is nonnegative (positive) if and only if* $\nabla_u f$ *is nonnegative (positive);*
(d) $2\, X \bullet S = w^T \nabla_w f$*;*
(e) $u^T(h - \mathbf{G}(X)) = u^T \nabla_u f$*.*

Moreover, if (w_ν, y_ν, u_ν) *solves* (NLD_ν)*, then* $X(w_\nu, y_\nu, u_\nu)$ *solves* (P_ν)*.*

For each $(w, y, u) \in \Re^n_{++} \times \Re^m \times \Re^p_{++}$, parts (a), (b) and (c) of Corollary 1 clearly give necessary and sufficient conditions for $X(w, y, u)$ to be a feasible or strictly feasible solution for (P), and these conditions are based entirely on the gradient $\nabla f \equiv \nabla f(w, y, u)$. Moreover, if $\nabla_w f \geq 0$ and $\nabla_u f \geq 0$, then the quantities $X \bullet S$ and $u^T(h - \mathbf{G}(X))$ are both nonnegative, and hence one can measure the closeness to optimality of (w, y, u) by the magnitude of $\nabla_y f$, $w^T \nabla_w f$ and $u^T \nabla_u f$, which, according to parts (c), (d) and (e) of Corollary 1, are a measure of the complementarity of the primal-dual solution (X, z, y, u, S), where $X \equiv X(w, y, u)$, $z \equiv z(w, y, u)$ and $S \equiv S(w, y, u)$.

The reader may have noticed that the definition of $X \equiv X(w, y, u)$ implies that X will be dense in general. Since we have claimed that the algorithm of this paper is designed for solving large-scale SDPs, it is reasonable to question how the gradient $\nabla f(w, y, u)$ can be computed efficiently when its computation is dependent upon the dense matrix X. As it turns out, X is not necessary for computing the gradient. In fact, we have proved in [4] the existence of a sparse analogue of X which can be used as an alternative to X in the computation of the gradient.

We briefly describe the sparse analogue of X as follows. Let $V \equiv \{1, \ldots, n\}$, and define

$$\mathcal{F} \equiv \{(i, j) \in V \times V : i \geq j \text{ and } L_{ij} \neq 0 \text{ for some } L \equiv L(w, y, u)\}.$$

In other words, $\mathcal{F}$ is the collection of nonzero elements of the function $L(\cdot,\cdot,\cdot)$. Alternatively, $\mathcal{F}$ can be described as the fill-in resulting from the Cholesky factorization of $S(w, y, u)$. Defining

$$\bar{\mathcal{F}} \equiv \{(i, j) \in V \times V : i \geq j \text{ and } (i, j) \notin \mathcal{F}\},$$

we have the following theorem that describes the sparse analogue of X.

Theorem 4. *Let $(w, y, u) \in \Re^n_{++} \times \Re^m \times \Re^p_{++}$ be given and define $L \equiv L(w, y, u)$. Then the system of linear equations*

$$diag(\hat{X}) = d, \quad \hat{X}_{ij} = 0 \ \ \forall\, (i, j) \in \bar{\mathcal{F}}, \quad [\hat{X}L]_{ij} = 0 \ \ \forall\, (i, j) \in \mathcal{F},$$

has a unique solution $\hat{X} \in \mathcal{S}^n$, which we denote by $\hat{X}(w, y, u)$. Moreover, the matrix $\hat{X}$ satisfies

(a) $\nabla_w f(w, y, u) = 2\, diag(\hat{X}L)$,
(b) $\nabla_y f(w, y, u) = b - \mathbf{A}(\hat{X})$,
(c) $\nabla_u f(w, y, u) = h - \mathbf{G}(\hat{X})$.

3.5. A log-barrier framework and global convergence

We now give a generic log-barrier algorithm based on the nonlinear programming formulation (LND), followed by a global convergence result.

```
Algorithm GLB:
    Let $\sigma \in (0, 1)$ and $\nu_1 > 0$ be given, and set $k = 1$.
    For $k = 1, 2, 3, \ldots,$ do
        1. Use an unconstrained minimization method to solve $(NLD_{\nu_k})$
           approximately, obtaining $(w^k, y^k, u^k) \approx (w_{\nu_k}, y_{\nu_k}, u_{\nu_k})$.
        2. Set $\nu_{k+1} = \sigma \nu_k$, increment $k$ by 1, and return to step 1.
    End
```

In order to state a global convergence result for the above algorithm, we need to specify in Step 1 of the algorithm how accurate the iterate (w^k, y^k, u^k) should be as an approximate solution to (NLD_{ν_k}). For this, let constants $\gamma_1 \in [0, 1)$, $\gamma_2 > 1$ and $\gamma_3 > 0$ be given, and for each $\nu > 0$ define $\mathcal{N}(\nu) \subset \Re^n_{++} \times \Re^m \times \Re^p_{++}$ to be the set of all points (w, y, u) satisfying

$$2\gamma_1 \nu e \leq w * \nabla_w f \leq 2\gamma_2 \nu e, \qquad \|\nabla_y f\| \leq \gamma_3 \nu, \qquad \gamma_1 \nu e \leq u * \nabla_u f \leq \gamma_2 \nu e, \tag{12}$$

where $\nabla f \equiv \nabla f(w, y, u)$ and e is the vector of all ones of appropriate dimension. It was shown in [4] that for any $\nu > 0$ the unique minimizer (w_ν, y_ν, u_ν) of (NLD_ν) is in $\mathcal{N}(\nu)$, and we will require that our approximate solution (w^k, y^k, u^k) be in $\mathcal{N}(\nu_k)$ for k sufficiently large.

The following global convergence result for the generic nonlinear log-barrier algorithm was proven in [4] in a slightly different form. For each k, we define $\nabla f^k \equiv \nabla f(w^k, y^k, u^k)$, $z^k \equiv z(w^k, y^k, u^k)$, $L^k \equiv L(w^k, y^k, u^k)$, $S^k \equiv S(w^k, y^k, u^k)$ and $X^k \equiv X(w^k, y^k, u^k)$.

Theorem 5. *Let $\{(w^k, y^k, u^k)\}_{k\geq 1}$ be the sequence of points produced by the log-barrier algorithm. If there exists some $k_0 > 0$ such that $(w^k, y^k, u^k) \in \mathcal{N}(\nu_k)$ for all $k > k_0$, then*

(a) $\nabla_w f^k \geq 0$ *and* $\nabla_u f^k \geq 0$ *for all* $k \geq k_0$;
(b) $\lim_{k\to\infty}(w^k)^T \nabla_w f^k = 0$, $\lim_{k\to\infty} \nabla_y f^k = 0$ *and* $\lim_{k\to\infty}(u^k)^T \nabla_u f^k = 0$;
(c) the sequences $\{X^k\}$, $\{(z^k, y^k, u^k, S^k)\}$, $\{L^k, w^k\}$ *and* $\{\nabla_w f^k, \nabla_u f^k\}$ *are bounded;*
(d) any accumulation points of $\{X^k\}$ *and* $\{(z^k, y^k, u^k, S^k)\}$ *are optimal solutions of* (P) *and* (D)*, respectively.*

3.6. The implemented algorithm

In practice, we of course stop the iterations once some stopping criterion is met. Moreover, we will use a first-order, or gradient based, unconstrained minimization algorithm for the task of solving the log-barrier subproblems in Step 1. The following is a more specific and more realistic version of our algorithm, where we define

$$\phi_k(w, y, u) \equiv f(w, y, u) - 2\nu_k \sum_{i=1}^{n} \log w_i - \nu_k \sum_{j=1}^{p} \log u_j \tag{13}$$

to be the objective function of (NLD_{ν_k}).

Algorithm LB:

```
Let $\sigma \in (0,1)$, $\epsilon, \nu_1 > 0$ be given, and set $k = 1$.
**For** $k = 1, 2, 3, \ldots$, until $\nu_k \leq \epsilon$, **do**
    1. Use a version of the limited-memory BFGS algorithm to
       solve $(NLD_{\nu_k})$ approximately, obtaining $(w^k, y^k, u^k)$
       such that $\|\nabla\phi_k(w^k, y^k, u^k)\| \leq \epsilon_k$ for a selected $\epsilon_k > 0$.
    2. Set $\nu_{k+1} = \sigma\nu_k$, increment $k$ by 1, and return to step 1.
**End**
```

It is easy to verify that for any γ_1, γ_2 and γ_3 in the definition of the neighborhood $\mathcal{N}(\nu)$ (see (12)), there exists $\epsilon_k > 0$ such that $\|\nabla\phi_k(w, y, u)\| \leq \epsilon_k$ implies that $(w, y, u) \in \mathcal{N}(\nu_k)$. Consequently, global convergence will be ensured by Theorem 5. In our implementation, however, we will select ϵ_k based primarily on practical considerations rather than theoretical ones.

4. Computational results

In this section, we describe our computational experiences with the first-order log-barrier algorithm of Section 3.6. In particular, we discuss our method for exploiting sparsity of the problem data within the evaluations of f and ∇f. We then consider a number of implementation details and conclude with some computational results showing the effectiveness of our method on a variety of large-scale SDPs.

4.1. Function and gradient evaluations

Effectively exploiting sparsity in the data of semidefinite programming is a major concern for any algorithm designed to solve large-scale instances of (P) and (D), and we now discuss how Algorithm LB proposed in Section 3.6 is able to use the sparsity of the problem to its advantage when evaluating f and ∇f.

Let us fist consider the evaluation of $f(w, y, u)$ for any $(w, y, u) \in \Re^n_{++} \times \Re^m \times \Re^p_{++}$. As was shown in [4], the main work in calculating $f(w, y, u)$ is the computation of $L_0(w, y, u)$, and this provides a key opportunity to exploit the sparsity of the problem. Indeed, using the standard symbolic Cholesky factorization (see [12], for example), it is possible to determine in polynomial time the nonzero positions of $L_0(w, y, u)$ by analyzing the nonzeros of $S(w, y, u)$. Since the nonzeros of $S(w, y, u)$ do not exceed the aggregate nonzeros of the data matrices $\{C\} \cup \{A_i\}_{i=1}^m \cup \{G_j\}_{j=1}^p$, disregarding possible cancellations the nonzeros of $L_0(w, y, u)$ can be considered as independent of (w, y, u) so that each $L_0(w, y, u)$ shares the same set, say $\mathcal{F}$, of nonzeros. Assuming that $\mathcal{F}$ has been computed once and is readily available for each evaluation of $L_0(\cdot, \cdot, \cdot)$, it is not difficult to see that the computation of $L_0(w, y, u)$ can be performed in a similar fashion to a sparse Cholesky factorization that accesses only the nonzeros of $L_0(w, y, u)$ and the off-diagonal nonzeros of $S(w, y, u)$. (See also Lemma 1 of [4].)

The evaluation of the objective function serves as the primary motivation for exploiting sparsity in Algorithm LB. This sparsity, however, carries over into the evaluation of the gradient as well, as exemplified by Theorem 4 in which the formula for ∇f is expressed in terms of the sparse matrix $\hat{X} \equiv \hat{X}(w, y, u)$. From Theorem 4, it is easy to see that, once $\hat{X}$ has been computed, forming the gradient is a simple computation that can exploit the sparsity of the data. So it seems sensible (perhaps necessary) to compute $\hat{X}$ first and then to compute the gradient. Hence, our implementation efforts have focused on an efficient method for computing $\hat{X}$.

From the definition of $\hat{X}$ in Theorem 4, it is not difficult to develop a straightforward technique for computing $\hat{X}$ in a right-to-left column-by-column fashion. (See the proof of Lemma 3 in [4], for example.) Such an algorithm may not necessarily be the most efficient, however, since the computations should allow for the exploitation of both the sparsity of $\hat{X}$ and the sparsity of $L(w, y, u)$ (upon which $\hat{X}$ is defined via the linear system of Theorem 4). In our implementation, we have arranged the computation of $\hat{X}$ in such a way that sparsity is exploited to the fullest extent. As a result the overall computation of ∇f is often quicker than the evaluation of f.

4.2. Special considerations

There are, of course, many details that contribute to the successful implementation of a given algorithm, and so in this subsection we list and discuss several details concerning our implementation of Algorithm LB.

As with most iterative algorithms, the choice of starting point is a crucial element affecting the performance of Algorithm LB. We believe that the point $(w^0, y^0, u^0) = (e, 0, e)$, where — with a slight abuse of notation — $e \in \Re^n$ and $e \in \Re^p$ are vectors of all ones, is a good generic starting point for Algorithm LB. (If either of the vectors y or u are nonexistent, then the corresponding components are dropped.) The primary motivation for this starting point is that, with $w^0 = e$ and $u^0 = e$, the value of the function $b^T y$ and the value of the log-barrier function are both equal to zero, thus ensuring a good initial balance between the two objective values.

Before we discuss the choice of the initial barrier parameter ν_1, we first discuss a slight variation of the log-barrier algorithm that we have found carries with it some practical advantages. Recall that in the theoretical development of Algorithm LB, we have introduced the nonlinear program (NLD_ν), where $\nu > 0$ is the "weight," or barrier parameter, assigned to the barrier terms

$$-2\sum_{i=1}^{n} \log w_i \quad \text{and} \quad -\sum_{j=1}^{p} \log u_j .$$

In particular, we have assigned a single barrier parameter to both terms. There is, however, no reason why two separate barrier parameters, say $\nu^w > 0$ for the first term and $\nu^u > 0$ for the second, cannot be used. In fact, it is not difficult to prove that the resulting nonlinear program, called $(NLD_{\nu^w \nu^u})$, has a unique minimizer and that this minimizer corresponds to the unique minimizers of the analogous SDP problem $(D_{\nu^w \nu^u})$ (whose definition should be self-evident) in a manner similar to that described by Theorem 2. A straightforward, convergent variant of Algorithm LB which takes into account the different choices of barrier parameters ν^w and ν^u can then be easily developed.

To help ensure strong practical performance, we suggest that the above scheme be implemented with different values for ν^w and ν^u. In particular, in our implementation we set $\nu_1^w = 1$ and then let ν_1^u be calculated as the minimizer of the strictly convex program

$$\min_{\nu^u \in \Re} \|\nabla_u f^0 - \nu^u (u^0)^{-1}\|^2,$$

where (w^0, y^0, u^0) is the initial starting point and $\nabla f^0 \equiv \nabla f(w^0, y^0, u^0)$. In other words, ν_1^u should be chosen so as to minimize the initial norm of the gradient of the barrier with respect to u. We remark, however, that since the unique minimizer of the above program may not be positive, it may be necessary to override the above choice and set ν_1^u to a safe value such as 1. Nonetheless, in our experiments with the frequency assignment problems, the minimizer was positive in every case and consequently helped to ensure good convergence of the overall algorithm.

The separation of the barrier parameter ν into two parameters ν^w and ν^u brings up another issue besides just that of initial values. Namely, we must decide when and how to update each of the parameters. Recall that, after the presentation of Algorithm LB, we defined a neighborhood $\mathcal{N}(\nu_k)$ and updated the parameter ν once the iterates entered the neighborhood (at least for the later stages of the algorithm). In the case of two parameters, the definition of the neighborhood can certainly be generalized to a new neighborhood $\mathcal{N}(\nu_k^w, \nu_k^u)$, and so we have the following theoretical update rule: once the iterates have entered the neighborhood $\mathcal{N}(\nu_k^w, \nu_k^u)$, update the parameters ν^w and ν^u by multiplying them with the same parameter $\sigma = 0.1$.

The above update rule ensures the theoretical convergence of the log-barrier algorithm, but in our experimentation, we found that it was advantageous to replace the theoretical neighborhood $\mathcal{N}(\nu_k^w, \nu_k^u)$ with a more practical neighborhood, one consisting of all points (w, y, u) for which $\|\nabla\phi_k(w, y, u)\|$ is "small" as given in Algorithm LB. (Note that this practical neighborhood certainly contains the unique minimizer of the log-barrier function.) In other words, with this practical neighborhood we update the parameters by σ once the norm of the gradient of the barrier function passes below a certain threshold (usually 10^{-1} for early stages of the algorithm and 10^{-2} for later stages of the algorithm).

4.3. Results of the log-barrier algorithm

We implemented the log-barrier algorithm as described in the two previous subsections, and the results are presented in this section. To solve the log-barrier subproblems, we chose a limited-memory BFGS algorithm with strong Wolfe-Powell line-search and three limited-memory vector updates (see [5] for example). Our sparse data structures were similar to those commonly found in the literature, and in order to increase the sparsity of the Cholesky factor $L(w, y, u)$ in our algorithm, we employed the sparse symmetric matrix reordering subroutines of the external code Metis 4.0.1 [17].

We implemented our code in ANSI C and compared it with the first-order spectral bundle method (SBmethod v1.1.1) of Helmberg, Rendl, and Kiwiel [15, 14]. The default parameters for the spectral bundle method were used, and all tests were run on an SGI Origin2000 with 16 300MHz R12000 processors and 10 Gigabytes of RAM at Rice University, although each code used only one processor.

We decided to compare our method with the spectral bundle method mainly based on two considerations: (1) it is the only other first-order method currently available that can solve a wide range of combinatorial SDPs; and (2) like our method, it is a dual-feasible and descent method, and therefore is highly comparable with our method. Also, we chose not to compare with any second-order method since our goal was to show the performance of our method on truly large-scale problems (which is the case for nearly all test problems presented in this paper) that are currently out of reach for second-order methods.

Since both methods solve the dual problems of the maximization problems given in Sections 2.2, 2.3 and 2.4, we note that, in evaluating the quality of an approximate solution, the smaller the objective value is, the better.

4.3.1. Results on maxcut relaxations The first set of test problems consists of thirteen instances of the maxcut SDP relaxation (see Section 2.2). Of these thirteen, the first nine come from the so-called Gset of randomly generated graphs first introduced in [15], and the last four come from the recent Seventh DIMACS Implementation Challenge on Semidefinite and Related Optimization Problems[7].

In Table 1, we give information concerning the thirteen problems, namely the problem name, the value of n for our formulation (note that $m = p = 0$), the density of non-zeros in the lower part of the matrices $S(\cdot, \cdot, \cdot)$ and in $L(\cdot, \cdot, \cdot)$ (including the diagonal entries), and the optimal value of the SDP relaxation wherever it is known to us. We note that, for the maxcut SDP relaxation, n equals the number of vertices in the underlying graph, and the density of S represents roughly the density of edges present in the graph.

Table 1. Problem Statistics for the Maxcut Comparison

name	n	dens% S	dens% L	SDP opt
G01	800	6.23	76.80	—
G11	800	0.75	2.64	—
G14	800	1.71	16.32	—
G22	2000	1.10	52.53	—
G32	2000	0.30	1.78	—
G35	2000	0.69	14.11	—
G43	1000	2.20	54.61	—
G48	3000	0.20	1.39	6000
G51	1000	1.38	15.70	—
toruspm3-08-50	512	1.56	14.09	527.808663
toruspm3-15-50	3375	0.24	6.02	—
torusg3-08	512	1.56	14.09	457.358179
torusg3-15	3375	0.24	6.02	—

Table 2 gives the performance of the spectral bundle method (SB) and our method (BMZ) on the thirteen maxcut test graphs. Each method was given an upper bound of ten hours (or 36,000 seconds) of computation time on each instance, although on only one problem (G22 for our method) was this time limit relevant. Our method was terminated once the barrier parameter ν^w achieved the value 10^{-4}. (Again, note that $p = 0$ and so ν^u is nonexistent for this class of problems.) We remark that both methods solve the dual formulation of the maxcut SDP and moreover that each method is a feasible descent method. Hence, the objective values given in the second and third columns are directly comparable. The times given in the fourth and fifth columns are in seconds, and the iterations given in the last two columns represent the total number of "serious steps" for the spectral bundle method (see [15]) and the number of inner iterations of our algorithm.

From the table, it is not difficult to see that, for most problems, SB achieves higher accuracy in less time than BMZ. Interestingly, for those problems in which the times for BMZ are relatively close to those for SB, the density of L is not much larger than the density of S, i.e., for these case, the fill-in of the Cholesky factorization is small. On the other hand, when the fill-in is great, there is a great disparity between the times of the two methods. Since the spectral bundle method works only with S while our method works with both S and L, these results seem to indicate that BMZ is most negatively affected by the fill-in of the Cholesky factorization. Of course, the results also indicate that SB converges very well on the maxcut problems, obtaining good accuracy in just a few iterations on most problems.

4.3.2. Results on maximum stable set relaxations The next set of test problems consists of thirty-one Lovász theta SDPs (see Section 2.3), with the first twelve coming from the same Gset collection as above, the next thirteen coming from the Second DIMACS Challenge on the Maximum Clique problem [6] (for which the Lovász theta number represents an upper bound on the size of a maximum clique), and the final six coming from the Seventh DIMACS Implementation Challenge as mentioned above. The data in Table 3 are similar to those in Table 1, though here we point out that n (the size of the matrices in our formulation) equals one plus the number of vertices of the underlying

Table 2. Comparison of the Two Methods on the Maxcut Graphs

Problem	Obj Value		Time		Iter	
name	SB	BMZ	SB	BMZ	SB	BMZ
G01	12083.2730	12083.2390	21	15831	14	1737
G11	629.1730	629.1761	68	20	31	567
G14	3191.5894	3191.5887	31	1214	29	1895
G22	14136.0390	14136.2130	89	36010	19	1944
G32	1567.6548	1567.6601	286	264	45	1017
G35	8014.7961	8014.7731	211	12495	35	1419
G43	7032.2542	7032.2482	22	17252	16	1740
G48	6000.0000	6000.0010	0	68	1	106
G51	4006.2862	4006.2727	48	1683	25	1500
toruspm3-08-50	527.8130	527.8177	10	85	22	711
toruspm3-15-50	3475.1585	3475.1557	288	13550	41	1605
torusg3-08	457.3611	457.3657	9	144	27	1139
torusg3-15	3134.5923	3134.5894	392	10241	53	1167

graph, m (the number of additional primal constraints) equals the number of edges, and the density of S roughly represents the density of edges in the graph. Also, $p = 0$ for this formulation.

Table 4 gives the performance of SB and BMZ on the thirty-one theta test graphs. As before, each method was given an upper bound of ten hours of computation time on each instance. Our method was terminated once ν^w became 10^{-6}, and again, both methods are dual feasible descent methods. The objective values, times, and iterations are given just as in Table 2 for the maxcut instances. However, under each category, there are three columns: the first two columns, SB and BMZ, corresponding to SB and BMZ applied to the maximum stable set relaxation (3), respectively, and the third column SB2 corresponding to SB applied to a different formulation of the Lovász theta SDP; that is,

$$\max\{(ee^T) \bullet X : \mathrm{tr}(X) = 1,\ X_{ij} = 0\ \forall\ (i, j) \in E,\ X \succeq 0\}, \tag{14}$$

where $e \in \Re^n$ is the vector of all ones. The spectral bundle method is applicable to this formulation, but our method is not (see [3], however, for an extension of our method). We note that although the two formulations are different, the methods are working with approximately the same size of matrices and the same number of constraints.

A comparison of the results in the columns SB and BMZ in Table 4 indicates that BMZ outperforms SB on the formulation (3), obtaining a better objective value in less time on all but two problems. On these two problems (G50 and hamming-9-8), SB seems to have terminated prematurely.

We now compare the results in the columns BMZ and SB2, i.e., BMZ applied to (3) and SB applied to (14). It is worth noting that on ten of the tested problems, SB computed the theta function in just one iteration. This seems to indicate that SB's routine for choosing a starting point performed extraordinarily well on these problems. As such, these results may not reflect the general performance of SB. Also, on one problem, namely G50, SB seems not to have converged, attaining a value of 1497.0372 when the optimal value is clearly no greater than the value obtained by BMZ, 1494.0997. On the remaining twenty problems, it is difficult to draw clear conclusions from the data as

Table 3. Problem Statistics for the Theta Comparison

name	n	m	dens% S	dens% L	SDP opt
G43	1001	9990	2.39	55.10	—
G44	1001	9990	2.39	54.42	—
G45	1001	9990	2.39	56.51	—
G46	1001	9990	2.39	54.46	—
G47	1001	9990	2.39	55.30	—
G48	3001	6000	0.27	1.46	1500
G49	3001	6000	0.27	1.34	1500
G50	3001	6000	0.27	1.30	—
G51	1001	5909	1.58	15.58	—
G52	1001	5916	1.58	15.32	—
G53	1001	5914	1.58	15.91	—
G54	1001	5916	1.58	15.64	—
MANN-a27.co	379	702	2.03	3.12	—
brock200-1.co	201	5066	26.93	84.68	—
brock200-4.co	201	6811	35.53	90.31	—
brock400-1.co	401	20077	25.90	91.33	—
c-fat200-1.co	201	18366	92.44	99.82	—
hamming-6-4.co	65	1312	67.18	97.48	—
hamming-8-4.co	257	11776	37.07	92.42	—
johnson8-4-4.co	71	1680	27.43	75.23	14
johnson16-2-4.co	121	560	26.03	75.34	8
keller4.co	172	5100	36.58	83.51	—
p-hat300-1.co	301	33917	75.95	98.67	—
san200-0.7-1.co	201	5970	31.38	68.68	—
sanr200-0.7.co	201	6032	31.69	88.51	—
hamming-9-8	513	2304	2.53	17.98	224
hamming-10-2	1025	23040	4.77	38.54	102.4
hamming-11-2	2049	56320	2.88	36.92	$170\frac{2}{3}$
hamming-7-5-6	129	1792	24.44	73.71	$42\frac{2}{3}$
hamming-8-3-4	257	16128	50.19	93.34	25.6
hamming-9-5-6	513	53760	41.55	92.74	$85\frac{1}{3}$

to which method performed better. On the Gset problems, SB seems to have the edge, achieving a better objective value in ten hours for problems G43–G47 and calculating a slightly worse objective value in less time for problems G51–G54. On the Second DIMACS Challenge graphs, however, BMZ performs better with higher accuracy in less time. Interestingly, the disparity between the densities of S and L still exist for these problems just as they did for the maxcut problems, but in these cases, there seems to be less of a consequence on the running times of the two methods. Perhaps this is a reflection of the higher number of iterations that SB tends to perform on the theta problems as compared with the maxcut problems.

We mention that since the SB and BMZ methods apply different transformations to an SDP problem, it is not possible to start the two methods from an identical or even "equivalent" initial point, thus making a more rigorous comparison difficult. We also note that for the test problems c-fat200-1.co and hamming8-4.co, the stability numbers are 12 and 16, respectively. For these two problems, by solving the SDP relaxations we obtained sufficiently tight upper bounds that deliver the stability numbers due to their integrality, even though we are still uncertain of the exact optimal values for these two SDP relaxations.

Table 4. Comparison of the Two Methods on Theta Graphs

Problem	Obj Value			Time			Iter		
name	SB	BMZ	SB2	SB	BMZ	SB2	SB	BMZ	SB2
G43	313.0560	280.6894	280.6294	36002	36009	36000	85	3277	54
G44	312.9946	280.6140	280.5877	36001	36007	36000	89	3594	55
G45	312.7823	280.2505	280.1899	36002	36003	36001	84	3202	56
G46	313.2684	279.8673	279.8427	36001	36006	36000	88	3500	49
G47	315.2982	281.9578	281.8987	36002	36008	36000	87	3433	52
G48	1507.0761	1500.0000	1500.0000	3600	564	1	386	610	1
G49	1504.5145	1500.0000	1500.0000	5317	361	1	385	546	1
G50	1686.4673	1494.0997	1497.0372	206	35058	3	11	46818	6
G51	349.1676	349.0034	349.0235	36000	7239	5683	72	6322	67
G52	348.6703	348.4026	348.5148	36001	14015	3400	74	12441	57
G53	348.7891	348.3755	348.3856	36001	36000	27718	75	26955	60
G54	341.1159	341.0013	341.0135	36000	6136	1313	82	5403	61
MANN-a27.co	132.7672	132.7632	132.7642	13562	15	1689	117	1655	44
brock200-1.co	28.5799	27.4584	27.4585	36000	3596	36000	72	20944	61
brock200-4.co	23.9297	21.2959	21.2955	36000	4865	36000	102	24142	58
brock400-1.co	47.0865	39.7045	39.7104	36001	36000	36000	63	24362	64
c-fat200-1.co	19.0317	12.0003	12.0033	36001	3657	36000	65	13811	57
hamming-6-4.co	5.4111	5.3349	5.3335	36000	113	7590	123	10170	51
hamming-8-4.co	25.2189	16.0013	16.0013	36000	6526	36000	72	15967	57
johnson8-4-4.co	14.0022	14.0009	14.0000	30047	24	0	83	3435	1
johnson16-2-4.co	8.7894	8.0010	8.0000	36000	72	3	140	2345	1
keller4.co	17.2862	14.0164	14.0135	36000	6156	36000	98	52737	57
p-hat300-1.co	25.4711	10.0735	10.1091	36000	20159	36000	52	25130	71
san200-0.7-1.co	30.0002	30.0000	30.0001	164	108	35	36	1033	20
sanr200-0.7.co	25.8951	23.8377	23.8379	36000	4187	36000	94	22891	59
hamming-9-8	228.6400	224.0025	224.0000	286	3179	1	108	17546	1
hamming-10-2	121.1150	102.4255	102.4000	36002	36004	51	60	7927	1
hamming-11-2	223.0563	171.6680	170.6667	36006	36003	159	62	947	1
hamming-7-5-6	42.6676	42.6680	42.6667	15133	322	0	57	8613	1
hamming-8-3-4	30.3151	25.6055	25.6000	36000	15200	10	43	31467	1
hamming-9-5-6	85.6523	85.3504	85.3333	36000	36001	5	30	9815	1

4.3.3. Results on frequency assignment relaxations Finally, we compare the two methods on a set of twelve frequency assignment relaxations (see Section 2.4) that were obtained from A. Eisenblätter, one of which, fap09, was used in the Seventh DIMACS Implementation Challenge. The statistics for the problems are given in Table 5, and we point out that n is the number of edges in the underlying graph and that the sum of m and p is the number of edges. In addition, the density of S corresponds to the density of the edges in the graph.

We recall that in deriving the SDP relaxation (5) we dropped constraints on the (i, j) pairs for non-edges. To distinguish this type of relaxations with the ones that keep a constraint for each (i, j) pair (whether edge or non-edge), we append the symbol ".sup" to the problem names where "sup" stands for "support", meaning that the constraints are supported only on the edge set.

Table 6 gives the performance of both methods on the twelve FAPs. Each method was given an upper bound of fifty hours computation time (or 180,000 seconds) on each problem, and our method was stopped once ν^w had reached the value 10^{-7}. (Recall that ν^w and ν^u are linked in that they are always updated simultaneously and by the same factor $\sigma = 0.1$, and so the given rule is equivalent to stopping once ν^u reaches a value

Table 5. Problem Statistics for the Frequency Assignment Comparison

name	n	m	p	dens% S	dens% L
fap01.sup	52	166	1160	100.00	100.00
fap02.sup	61	204	1601	98.68	100.00
fap03.sup	65	243	1837	100.00	100.00
fap04.sup	81	194	3046	100.00	100.00
fap05.sup	84	223	3263	100.00	100.00
fap06.sup	93	281	3997	100.00	100.00
fap07.sup	98	614	4139	100.00	100.00
fap08.sup	120	472	6668	100.00	100.00
fap09.sup	174	1026	14025	100.00	100.00
fap10.sup	183	542	13754	86.00	97.99
fap11.sup	252	765	23275	76.20	93.88
fap12.sup	369	1683	24410	38.76	60.12

Table 6. Comparison of the Two Methods on Frequency Assignment Problems

Problem	Obj Value		Time		Iter	
name	SB	BMZ	SB	BMZ	SB	BMZ
fap01.sup	−0.0327907	−0.0328360	180000	117	91	19370
fap02.sup	−0.0005313	−0.0007109	180000	85	89	8756
fap03.sup	−0.0491999	−0.0493589	180000	274	85	26191
fap04.sup	−0.1746284	−0.1746702	180000	844	86	47285
fap05.sup	−0.3080811	−0.3081169	180000	659	89	31225
fap06.sup	−0.4590581	−0.4591657	180000	921	88	32013
fap07.sup	−2.1165139	−2.1165397	180000	1903	94	53940
fap08.sup	−2.4354640	−2.4360202	180000	1742	80	25940
fap09.sup	−10.7942300	−10.7971610	180000	11904	79	54897
fap10.sup	−0.0029943	−0.0095547	180000	18194	89	93791
fap11.sup	−0.0118932	−0.0296136	180000	39038	87	86135
fap12.sup	−0.2151594	−0.2733099	180000	44984	82	81119

that is 10^{-7} times its initial value.) Once again, both methods are dual-feasible, descent methods.

The objective values reported in the table for the FAP problems are calculated under the assumption that the weights corresponding to $(i, j) \in U$ are zero. This represents a constant shift in the objective values if those weight are not all zeros, but otherwise will not change the solution of the problems. Since both the SB code and our code apply this same scheme, the objective values from the two codes are completely comparable.

In each of the twelve problems, BMZ terminated before the fifty-hour time limit with an objective value that was better than that obtained by SB after fifty hours. In some cases, the time taken and the objective value achieved by BMZ were considerably better than those by SB. Overall, the results seem to indicate much stronger performance by BMZ on FAPs of these sizes. Note also that the issue of the differences between the densities between S and L are not relevant for most of the problems but also do not seem to affect fap12, a problem for which the density of S is notably less than the density of L. As with the theta problems, this may be a consequence of the relatively large number of iterations performed by SB.

The seventh DIMACS Implementation Challenge problem set contains two larger instances of the FAP relaxations, called fap25 and fap36. We did not run our code on these two problems. An independent benchmarking by Mittelmann [19] showed that both our code and the SB code did not solve either of these two instances to prescribed accuracies within a large amount of allocated time, though both codes made substantial and similar progresses towards optima. We refer to [20] for the information on these two problems, and to [19] for the detailed benchmarking results.

5. Final remarks

Our numerical results have shown that the transformation introduced in [4] is indeed a viable approach to solving some large-scale SDPs from combinatorial optimization. In fact, at present many of the tested problems in our numerical experiments can only be solved, within a reasonable amount of time, by our method and the spectral bundle method [15]. These two methods both have their advantages and disadvantages, depending on problem characteristics, and our method appears to be particularly effective for problems with a large number of constraints.

Recently, we have extended the application of our transformation to general SDP problems [3] where the diagonal of the primal matrix variable need not to be fixed. Preliminary numerical results in [3] indicate that the approach also holds promise for general SDP problems.

Our software package used in this study is currently undergoing further development. A public release is planned in a near future through the internet.

Acknowledgements. We thank Christoph Helmberg for his help in running the code SBmethod, and Andreas Eisenblätter for providing us with the frequency assignment test problems. We also thank the guest editor Gabor Pataki and an anonymous referee for their valuable comments that have helped improve the paper.

References

1. S. Benson and Y. Ye. Approximating maximum stable set and minimum graph coloring problems with the positive semidefinite relaxation. In M. Ferris and J. Pang, editors, *Applications and Algorithms of Complementarity*, pages 1–18. Kluwer, 2000.
2. S. Burer and R.D.C. Monteiro. A projected gradient algorithm for solving the Maxcut SDP relaxation. *Optimization Methods and Software*, 15:175–200, 2001.
3. S. Burer, R.D.C. Monteiro, and Y. Zhang. Interior-point algorithms for semidefinite programming based on a nonlinear programming formulation. *Computational Optimization and Applications*, 22:49–79, 2002.
4. S. Burer, R.D.C. Monteiro, and Y. Zhang. Solving a class of semidefinite programs via nonlinear programming. *Mathematical Programming*, 93:97–122, 2002.
5. R.H. Byrd, J. Nocedal, and R.B. Schnabel. Representation of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming*, 63:129–156, 1994.
6. Second DIMACS implementation challenge on maximum clique, graph coloring, and satisfiability, 1992–1993. See the website: `http://dimacs.rutgers.edu/Challenges/`.
7. Seventh DIMACS implementation challenge on semidefinite and related optimization problems, November 2000. See the website: `http://dimacs.rutgers.edu/Challenges/`.
8. A. Eisenbläter and C. Helmberg, May 2000. Private communication.
9. A. Eisenblätter. *Frequency assignment in GSM Networks: Models, Heuristics, and Lower Bounds*. PhD thesis, Technische Universität Berlin, Berlin, Germany, 2000.
10. A. Eisenblätter, M. Grötschel, and A.M.C.A. Koster. Frequency planning and ramifications of coloring. *Discussiones Mathematicae Graph Theory*, 22(1):51–88, 2002.

11. A. Frieze and M. Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. *Algorithmica*, 18:61–77, 1997.
12. A. George and J. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1981.
13. M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, pages 1115–1145, 42 (1995).
14. C. Helmberg and K.C. Kiwiel. A spectral bundle method with bounds. ZIB Preprint SC-99-37, Konrad-Zuse-Zentrum, Berlin, Germany, December 1999. To appear in *Mathematical Programming*.
15. C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10:673–696, 2000.
16. S. Homer and M. Peinado. Design and performance of parallel and distributed approximation algorithms for maxcut. *Journal of Parallel and Distributed Computing*, 46:48–61, 1997.
17. G. Karypis and V. Kumar. Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, version 4.0. Manuscript, Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, USA, 1998.
18. M. Laurent, S. Poljak, and F. Rendl. Connections between semidefinite relaxations of the max-cut and stable set problems. *Mathematical Programming*, 77:225–246, 1997.
19. H.D. Mittelmann. An independent benchmarkingof SDP and SOCP solvers. DOI 10.1007/s10107-002-0355-5, this volume.
20. G. Pataki and S. Schmieta. The DIMACS library of semidefinite-quadratic-linear programs. Technical report, 1999. See `http://dimacs.rutgers.edu/Challenges/Seventh/Instances/`.