RANK-TWO RELAXATION HEURISTICS FOR MAX-CUT AND OTHER BINARY QUADRATIC PROGRAMS*

SAMUEL BURER†, RENATO D.C. MONTEIRO‡, AND YIN ZHANG§

Abstract.

The Goemans-Williamson randomized algorithm guarantees a high-quality approximation to the Max-Cut problem, but the cost associated with such an approximation can be excessively high for large-scale problems due to the need for solving an expensive semidefinite relaxation. In order to achieve better practical performance, we propose an alternative, rank-two relaxation and develop a specialized version of the Goemans-Williamson technique. The proposed approach leads to continuous optimization heuristics applicable to Max-Cut as well as other binary quadratic programs, for example the Max-Bisection problem.

A computer code based on the rank-two relaxation heuristics is compared with two state-of-the-art semidefinite programming codes that implement the Goemans-Williamson randomized algorithm, as well as with a purely heuristic code for effectively solving a particular Max-Cut problem arising in physics. Computational results show that the proposed approach is fast and scalable and, more importantly, attains a higher approximation quality in practice than that of the Goemans-Williamson randomized algorithm. An extension to Max-Bisection is also discussed as well as an important difference between the proposed approach and the Goemans-Williamson algorithm, namely that the new approach does not guarantee an upper bound on the Max-Cut optimal value.

Key words. Binary quadratic programs, Max-Cut and Max-Bisection, semidefinite relaxation, rank-two relaxation, continuous optimization heuristics.

AMS subject classifications. 90C06, 90C27, 90C30

1. Introduction.

Many combinatorial optimization problems can be formulated as quadratic programs with binary variables, a simple example being the Max-Cut problem. Since such problems are usually NP-hard, which means that exact solutions are difficult to obtain, different heuristic or approximation algorithms have been proposed, often based on continuous relaxations of the original discrete problems. A relatively new relaxation scheme is called the semidefinite programming relaxation (or SDP relaxation) in which a vector-valued binary variable is replaced by a matrix-valued continuous variable, resulting in a convex optimization problem called a semidefinite program (SDP) that can be solved to a prescribed accuracy in polynomial time. Some early ideas related to such a relaxation can be found in a number of works, including [10, 23, 24, 26, 27].

Based on solving the SDP relaxation, Goemans and Williamson [18] proposed a randomized algorithm for the Max-Cut problem and established the celebrated 0.878 performance guarantee. Since then, SDP relaxation has become a powerful and popular theoretical tool for devising polynomial-time approximation algorithms for hard combinatorial optimization problems, and even in cases where performance guarantees are not known, randomized algorithms based on the SDP relaxation can often give good-quality approximate solutions in practice. It is important to note that such Goemans-Williamson-

*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.

†School of Mathematics, Georgia Institute of Technology, Atlanta, Georgia 30332, USA. This author was supported in part by NSF Grant CCR-9902010 and INT-9910084. (Email: burer@math.gatech.edu).

‡School of ISyE, Georgia Institute of Technology, Atlanta, Georgia 30332, USA. This author was supported in part by NSF Grant CCR-9902010 and INT-9910084. (Email: monteiro@isye.gatech.edu).

§Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005, USA. This author was supported in part by DOE Grant DE-FG03-97ER25331, DOE/LANL Contract 03891-99-23 and NSF Grant DMS-9973339. (Email: zhang@caam.rice.edu).

type approaches produce both upper and lower bounds on the optimal value of the underlying discrete problem.

In the meantime, there have been hopes that the SDP relaxation would eventually lead to practically efficient algorithms for solving large-scale combinatorial optimization problems by producing tight lower and upper bounds. In this regard, however, results thus far have not always been encouraging. The main difficulty lies in the fact that the number of variables and/or constraints in a SDP relaxation is one order of magnitude higher than that of the original problem. Hence, the cost of solving such SDP problems grows quickly as the size of the problems increases. In other words, a key issue here is the scalability of the SDP relaxation approach with respect to the problem size.

There have been a great deal of research efforts towards improving the efficiency of SDP solvers, including works on exploiting sparsity in more traditional interior-point methods [1, 9, 16, 17, 29] and works on alternative methods [5, 6, 7, 20, 21, 30, 31]. Indeed, the efficiency of SDP solvers has been improved significantly in the last few years. Nevertheless, the scalability problem still remains.

On the other hand, computational studies have continued to affirm that the quality of bounds produced by the SDP relaxation is quite high. For example, the Goemans-Williamson approximation algorithm produces lower bounds (i.e., discrete solutions) that are better than or at least comparable to that of a number of heuristics (see [11], for example). It is thus natural to investigate whether the quality of the SDP relaxation can be preserved while somehow extending its use to problems of very large size.

Can the approaches inspired by Goemans and Williamson, which rely on solving the SDP relaxation, ever become competitive in attacking large-scale problems? In this paper, we hope to provide a partial answer to this question. We will argue that in terms of producing a lower bound, the answer seems to be negative at least for some problem classes including the familiar Max-Cut problem. In other words, if one is only interested in obtaining a high-quality approximate solution, then the SDP relaxation does not seem to hold much promise. Our argument is based on strong empirical evidence showing that on a large set of test problems the performance of the SDP relaxation approach is clearly inferior to that of a new rank-two relaxation approach that we will propose and study in this paper. The advantages of this rank-two approach are not only in terms of computational costs but, more notably, also in terms of the approximation quality.

Based on the proposed rank-two relaxation and a specialized version of the Goemans-Williamson technique, we construct a continuous optimization heuristic for approximating the Max-Cut problem and establish some properties for this approach that are useful in designing algorithms. We then compare a code based on our heuristic with some state-of-the-art SDP-based approximation codes on a set of Max-Cut test problems. We also compare our code with a well-established, heuristic code for Max-Cut on a set of test problems from physics. Finally, we consider extensions to other related problems—in particular, to the Max-Bisection problem.

This paper is organized as follows. Section 2 briefly introduces the Max-Cut problem and its corresponding SDP relaxation. In Section 3, we present the rank-two relaxation scheme and study its properties, including a useful characterization for a maximum cut. In Section 4, we present our heuristic algorithm for the Max-Cut problem, and computational results on Max-Cut are given in Section 5. We extend the heuristic to the Max-Bisection problem in Section 6 and give numerical results as well. Lastly, we conclude the paper in Section 7.

2. Max-Cut and the Semidefinite Relaxation. Let an undirected and connected graph $G = (V, E)$, where $V = \{1, 2, \cdots, n\}$ and $E \subset \{(i, j) : 1 \leq i < j \leq n\}$, be given. Let the edge weights $w_{ij} = w_{ji}$ be given such that $w_{ij} = 0$ for $(i, j) \notin E$, and in particular, let $w_{ii} = 0$. The Max-Cut problem is to find a bipartition (V_1, V_2) of V so that the sum of the weights of the edges between V_1 and V_2 is maximized. It is well-known that the Max-Cut problem can be formulated as

$$\begin{array}{ll} \max & \frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij}(1 - x_i x_j) \\ \text{s.t.} & |x_i| = 1, \;\; i = 1, \ldots, n, \end{array} \tag{2.1}$$

which has the same solution as the following binary quadratic program:

$$\begin{array}{ll} \min & \sum_{1 \leq i < j \leq n} w_{ij} x_i x_j \\ \text{s.t.} & |x_i| = 1, \; i = 1, \ldots, n. \end{array} \tag{2.2}$$

Moreover, it is easy to verify that (2.2) can be rewritten into the matrix optimization problem

$$\begin{array}{rl} \min & \frac{1}{2} W \bullet X, \\ \text{s.t.} & \mathrm{diag}(X) = e, \\ & \mathrm{rank}(X) = 1, \\ & X \succeq 0, \end{array} \tag{2.3}$$

where $W = [w_{ij}]$, $W \bullet X = \sum_{i,j=1}^{n} w_{ij} x_{ij}$, $\mathrm{diag}(X)$ is the vector in $\Re^n$ consisting of the diagonal elements of X, e is the vector of all ones, and $X \succeq 0$ means that X is symmetric positive semidefinite.

Since the Max-Cut problem is NP-hard, various heuristics and approximation algorithms have been proposed to attack the problem. A recent ground-breaking work is due to Goemans and Williamson [18] who replace the "unit scalars" x_i in (2.2) by unit vectors $v_i \in \Re^n$ and the scalar products $x_i x_j$ by the inner products $v_i^T v_j$. The resulting problem is the following relaxation of the Max-Cut problem:

$$\begin{array}{ll} \min & \sum_{1 \leq i < j \leq n} w_{ij} v_i^T v_j \\ \text{s.t.} & \|v_i\|_2 = 1, \; i = 1, \ldots, n, \end{array} \tag{2.4}$$

where $v_i \in \Re^n$. Furthermore, a change of variables $X = [v_i^T v_j] \in \Re^{n \times n}$ leads to the following so-called SDP relaxation for the Max-Cut problem

$$\begin{array}{rl} \min & \frac{1}{2} W \bullet X, \\ \text{s.t.} & \mathrm{diag}(X) = e, \\ & X \succeq 0. \end{array} \tag{2.5}$$

It is well-known that such a SDP problem is approximately solvable in polynomial time (see [25], for example). Comparing (2.5) with (2.3), we clearly see that the SDP relaxation is nothing but the problem obtained from (2.3) by dropping the rank-one restriction on X.

It is worth observing that a solution $(v_1, \ldots, v_n)$ of (2.4) consists of n points on the surface of the unit sphere in $\Re^n$, each representing a node in the graph. Goemans and Williamson [18] proposed the following randomized algorithm for generating cuts in the graph: after a solution of (2.4) is obtained, one randomly partitions the unit sphere into two half-spheres H_1 and H_2 (the boundary in between can be on either side) and forms the bipartition consisting of $V_1 = \{i : v_i \in H_1\}$ and $V_2 = \{i : v_i \in H_2\}$. Furthermore, Goemans and Williamson established the celebrated result that if all the weights are nonnegative, then the expected value of such randomly generated cuts is at least 0.878 times the maximum cut value. This result gives a strong performance guarantee for this randomization procedure. In fact, it has recently been shown in [13] that the factor 0.878 is indeed the best possible in several senses.

3. A Rank-Two Relaxation. In this section, we present an alternative rank-two relaxation scheme that leads to a nonlinear optimization problem having only n variables but also a nonconvex objective function. Since the number of variables is not increased from the Max-Cut problem, this approach possesses scalability for relaxing large-scale problems. On the other hand, since the relaxation is nonconvex, we cannot expect to find an optimal solution in practice, and so we can no longer ensure a computable upper bound on the original problem. For solving this problem to gain information about the underlying Max-Cut problem, the trade-off is obviously between computational efficiency and a theoretical guarantee. When the main objective is to obtain high-quality approximate solutions, however, we hope to demonstrate through computational experiments that the gain clearly outweighs the loss.

We replace the "unit scalar" variables x_i in (2.2) by unit vectors $v_i \in \Re^2$ (not $\Re^n$) and the scalar products $x_i x_j$ by the inner products $v_i^T v_j$. As before, the constraint $|x_i| = 1$ becomes $\|v_i\|_2 = 1$; namely, all the vectors v_i should be on the unit circle. In this way, we obtain a relaxation of the Max-Cut problem that has exactly the same form as (2.4) except that now all vectors v_i are in $\Re^2$ instead of $\Re^n$. Alternatively, this relaxation can also be viewed as replacing the rank-one restriction on X in (2.3) by the rank-two restriction $\mathrm{rank}(X) \le 2$; hence we call it a rank-two relaxation.

Using polar coordinates, we can represent a set of n unit vectors $v_1, \ldots, v_n$ in $\Re^2$ by means of a vector $\theta = (\theta_1, \ldots, \theta_n)^T \in \Re^n$ consisting of n angles, namely

$$v_i = \begin{pmatrix} \cos\theta_i \\ \sin\theta_i \end{pmatrix}, \quad \forall\, i = 1, \ldots, n.$$

In this case, we have

$$v_i^T v_j \equiv \cos(\theta_i - \theta_j), \quad \forall\, i, j = 1, \ldots, n.$$

Let $T(\theta)$ be the skew-symmetric matrix-valued function of θ defined by

$$T_{ij}(\theta) = \theta_i - \theta_j, \quad \forall\, i, j = 1, \ldots, n,$$

and let $f : \Re^n \to \Re$ be the function defined as

$$f(\theta) \equiv \frac{1}{2} W \bullet \cos(T(\theta)), \quad \forall\, \theta \in \Re^n, \tag{3.1}$$

where $\cos(T(\theta))$ is the $n \times n$ matrix whose entries are the cosine of the corresponding entries of $T(\theta)$. Then, in terms of the polar coordinates, we obtain the following relaxation for the Max-Cut problem:

$$\min_{\theta \in \Re^n} f(\theta) \tag{3.2}$$

This is an unconstrained optimization problem with a nonconvex objective function. In general, it has multiple local, non-global minima.

The derivatives of the function $f(\theta)$ can be easily computed. Indeed, the first partial derivatives of $f(\theta)$ are given by

$$\frac{\partial f(\theta)}{\partial \theta_j} = \sum_{k=1}^{n} w_{kj} \sin(\theta_k - \theta_j), \quad \forall\, j = 1, \ldots, n,$$

and hence,

$$g(\theta) \equiv \nabla f(\theta) = [W \circ \sin(T(\theta))]^T e, \tag{3.3}$$

where the notation "$\circ$" indicates the Hadamard, i.e., entry-wise, product of W and $\sin(T(\theta))$. The second partial derivatives of $f(\theta)$ are given by

$$\frac{\partial^2 f(\theta)}{\partial\theta_i \partial\theta_j} = \begin{cases} w_{ij}\cos(\theta_i - \theta_j), & \text{if } i \neq j, \\ -\sum_{k\neq j} w_{kj}\cos(\theta_k - \theta_j), & \text{if } i = j, \end{cases}$$

for every $i, j = 1, \ldots, n$, and hence the Hessian of $f(\theta)$ is given by

$$H(\theta) \equiv \nabla^2 f(\theta) = W \circ \cos(T(\theta)) - \mathrm{Diag}\left([W \circ \cos(T(\theta))]e\right), \tag{3.4}$$

where for any vector v, $\mathrm{Diag}(v)$ is the diagonal matrix with v on its diagonal. Note that the major effort in the evaluation of f, g and H is the computation of the quantities $W \circ \cos(T(\theta))$ and $W \circ \sin(T(\theta))$.

There are interesting relationships between cuts in the graph and the function $f(\theta)$, which we will now describe. We call a vector $\bar{\theta} \in \Re^n$ an *angular representation* of a cut, or simply a cut, if there exist integers k_{ij} such that

$$\bar{\theta}_i - \bar{\theta}_j = k_{ij}\pi, \quad \forall\ i, j = 1, \ldots, n. \tag{3.5}$$

Clearly, in this case $\cos(\bar{\theta}_i - \bar{\theta}_j) = \pm 1$ and there exists a binary vector $\bar{x} \in \{-1, 1\}^n$ such that

$$\cos(\bar{\theta}_i - \bar{\theta}_j) \equiv \bar{x}_i \bar{x}_j = \pm 1, \quad \forall\ i, j = 1, \ldots, n.$$

Moreover, the cut value corresponding to a cut $\bar{\theta}$ is

$$\psi(\bar{\theta}) \equiv \frac{1}{2}\sum_{i>j} w_{ij}[1 - \cos(\bar{\theta}_i - \bar{\theta}_j)]. \tag{3.6}$$

We note that the function $f(\theta)$ is invariant with respect to simultaneous, uniform rotation on every component of θ, i.e., $f(\theta) \equiv f(\theta + \tau e)$ for any scalar τ, and is periodic with a period of 2π with respect to each variable θ_i. Modulo the uniform rotation and the periodicity for each variable, there is an obvious one-to-one correspondence between the binary and angular representations of a cut; namely,

$$\bar{\theta}_i = \begin{cases} 0, & \text{if } \bar{x}_i = +1, \\ \pi, & \text{if } \bar{x}_i = -1, \end{cases}$$

and vice versa. With the above correspondence in mind, in the sequel we will use $\bar{\theta}$ and $\bar{x}$ interchangeably to represent a cut. Moreover, given an angular representation of a cut $\bar{\theta}$ (or a binary one $\bar{x}$), we will use the notation $x(\bar{\theta})$ (or $\theta(\bar{x})$) to denote the corresponding binary (or angular) representation of the same cut.

Since $\sin(\bar{\theta}_i - \bar{\theta}_j) = 0$ for any $\bar{\theta}$ satisfying (3.5), it follows directly from (3.3) that $g(\bar{\theta}) = 0$ at any cut $\bar{\theta}$. We state this simple observation in the following proposition.

PROPOSITION 3.1. *Every cut $\bar{\theta} \in \Re^n$ is a stationary point of the function $f(\theta)$.*

We will now derive a characterization of a maximum (minimum) cut in the lemma below which will be useful in the later development. We first need the following definition. A matrix $M \in \Re^{n\times n}$ is called *nonnegatively summable* if the sum of the entries in every principal submatrix of M is nonnegative, or equivalently, if $u^T M u \geq 0$ for every binary vector $u \in \{0, 1\}^n$. Clearly, every positive semidefinite matrix is nonnegatively summable. On the other hand, the matrix $ee^T - I$ is nonnegatively summable, but not positive semidefinite.

LEMMA 3.2. *Let $\bar{x} \in \{-1,1\}^n$ be given and consider the matrix $M(\bar{x}) \in \Re^{n\times n}$ defined as*

$$M(\bar{x}) = W \circ (\bar{x}\bar{x}^T) - \mathrm{Diag}([W \circ (\bar{x}\bar{x}^T)]e). \tag{3.7}$$

Then, $\bar{x}$ is a maximum (respectively, minimum) cut if and only if $M(\bar{x})$ (respectively, $-M(\bar{x})$) is nonnegatively summable.

Proof. Let $q : \Re^n \to \Re$ be the quadratic function defined as $q(x) = (x^T W x)/2$ for all $x \in \Re^n$ and note that $\bar{x}$ is a maximum cut if and only if $\bar{x}$ minimizes $q(x)$ over the set of all $x \in \{-1,1\}^n$. Now, let $x \in \{-1,1\}^n$ be given and observe that

$$\bar{x} - x = 2\,\delta \circ \bar{x},$$

where "$\circ$" represents the Hadamard product, and $\delta \in \Re^n$ is defined as

$$\delta_i \equiv \begin{cases} 0, & \text{if } x_i = \bar{x}_i, \\ 1, & \text{if } x_i \neq \bar{x}_i. \end{cases} \tag{3.8}$$

Using this identity and the fact that $\delta^T v = \delta^T \mathrm{Diag}(v)\delta$ for any $v \in \Re^n$, we obtain

$$\begin{aligned} q(x) - q(\bar{x}) &= (W\bar{x})^T(x-\bar{x}) + \frac{1}{2}(x-\bar{x})^T W (x-\bar{x}) \\ &= -2\bar{x}^T W(\delta \circ \bar{x}) + 2(\delta\circ\bar{x})^T W(\delta\circ\bar{x}) \\ &= -2\delta^T([W \circ \bar{x}\bar{x}^T]e) + 2\delta^T[W\circ\bar{x}\bar{x}^T]\delta \\ &= -2\delta^T \mathrm{Diag}\left([W\circ\bar{x}\bar{x}^T]e\right)\delta + 2\delta^T[W\circ\bar{x}\bar{x}^T]\delta \;=\; 2\delta^T M(\bar{x})\delta \end{aligned}$$

Noting that every $x \in \{-1,1\}^n$ corresponds to a unique vector $\delta \in \{0,1\}^n$ via (3.8) and vice versa, we conclude from the above identity that $\bar{x}$ minimizes $q(x)$ over $x \in \{-1,1\}^n$ if and only if $\delta^T M(\bar{x})\delta \geq 0$ for all $\delta \in \{0,1\}^n$, or equivalently, $M(\bar{x})$ is nonnegatively summable.

The proof of the second equivalence is analogous. Hence, the result follows. □

Although every cut is a stationary point of $f(\theta)$, the following theorem guarantees that only the maximum cuts can possibly be local minima of $f(\theta)$. In fact, the theorem gives a complete classification of cuts as stationary points of the function $f(\theta)$.

THEOREM 3.3. *Let $\bar{\theta}$ be a cut and let $\bar{x} \equiv x(\bar{\theta})$ be the associated binary cut. If $\bar{\theta}$ is a local minimum (respectively, local maximum) of $f(\theta)$, then $\bar{x}$ is a maximum (respectively, minimum) cut. Consequently, if $\bar{x}$ is neither a maximum cut nor a minimum cut, then $\bar{\theta}$ must be a saddle point of $f(\theta)$.*

Proof. Since $\bar{x}_i\bar{x}_j = \cos(\bar{\theta}_i - \bar{\theta}_j)$, we have $\nabla^2 f(\bar{\theta}) \equiv M(x(\bar{\theta}))$ due to (3.4) and (3.7). If $\bar{\theta}$ is a local minimum of f, then the Hessian $\nabla^2 f(\bar{\theta})$ is positive semi-definite, and hence nonnegatively summable. The first implication of the theorem then follows from the first equivalence of Lemma 3.2. The second implication of the theorem can be proved in a similar way using the second equivalence of Lemma 3.2. Hence, the result follows. □

The converses of the two implications in the above theorem do not hold. Indeed, consider the unweighted graph K_3 (the complete graph with three nodes) for which the cut $\bar{x} = [1 \;\; -1 \;\; -1]^T$ is maximum. From (3.7), we have

$$M(\bar{x}) = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 0 & 1 \\ -1 & 1 & 0 \end{bmatrix},$$

which is indeed nonnegatively summable, but not positive semi-definite. Hence, the corresponding angular representation $\bar{\theta}$ is not a local minimum of the function $f(\theta)$ in view of the fact that $M(\bar{x}) \equiv \nabla^2 f(\bar{\theta})$.

There are indeed instances where maximum cuts are local minima of $f(\theta)$, as indicated by the following observation.

PROPOSITION 3.4. *For a bipartite graph with nonnegative weights, the global minimum value of $f(\theta)$ is attained by a maximum cut.*

Proof. A maximum cut is one that cuts through all the edges in the bipartite graph. For this cut, $\cos(\theta_i - \theta_j) = -1$ for all edges $(i,j) \in E$. Hence the global minimum value of $f(\theta)$ is attained at $-e^T We/2$. □

Obviously, for problems where a maximum cut $\bar{x}$ corresponds to a local minimum of $f(\theta)$, the optimality of $\bar{x}$ can be checked in polynomial time by determining whether $M(\bar{x})$ is positive semidefinite or not.

Since non-maximum cuts cannot possibly be local minima of $f(\theta)$, a good minimization algorithm would not be attracted to stationary points corresponding to non-maximum cuts that are either local maxima or saddle points of $f(\theta)$. This fact will play an important role in the construction of our algorithms.

4. A Heuristic Algorithm for Max-Cut. To produce an approximate solution to the Max-Cut problem, we first minimize the function $f(\theta)$ and obtain a local minimum θ corresponding to a distribution of points on the unit circle. Using periodicity, we may easily assume that $\theta_i \in [0, 2\pi)$ for each $i = 1, \ldots, n$. Any partition of the unit circle into two equal halves gives a cut as follows: pick an angle $\alpha \in [0, \pi)$ and let

$$x_i = \begin{cases} +1, & \text{if } \theta_i \in [\alpha, \alpha + \pi), \\ -1, & \text{otherwise.} \end{cases} \tag{4.1}$$

The corresponding value of the cut x is given by

$$\gamma(x) \equiv \frac{1}{2} \sum_{i>j} w_{ij}(1 - x_i x_j). \tag{4.2}$$

An advantage of the rank-two relaxation over the SDP relaxation is that it is straightforward and inexpensive to examine all possible cuts generated in the above fashion, making it easy to find the best one. The following, deterministic (rather than random) procedure finds a best possible Goemans-Williamson-type cut associated with a given θ. Without loss of generality, let us assume that θ satisfies $\theta_i \in [0, 2\pi)$, $i = 1, \ldots, n$, and that

$$\theta_1 \le \theta_2 \le \ldots \le \theta_n,$$

after a reordering if necessary.

Procedure-CUT (input θ, output x^*):

Let $\alpha = 0$, $\Gamma = -\infty$, $i = 1$. Let j be the smallest index such that $\theta_j > \pi$ if there is one; otherwise set $j = n + 1$. Set $\theta_{n+1} = 2\pi$.

While $\alpha \le \pi$

1. Generate cut x by (4.1) and compute $\gamma(x)$.
2. If $\gamma(x) > \Gamma$, then let $\Gamma = \gamma(x)$ and $x^* = x$.

3. If $\theta_i \le \theta_j - \pi$, let $\alpha = \theta_i$ and increment i by 1;
   otherwise let $\alpha = \theta_j - \pi$ and increment j by 1.

End

Since our rank-two relaxation has the same form as Goemans and Williamson's relaxation (2.4) except ours has variables in $\Re^2$ rather than $\Re^n$, the same analysis of Goemans and Williamson, with minimal changes, can be applied to show that the cut value generated by the above procedure is at least 0.878 times the relaxed cut value $\psi(\theta)$ as is defined in (3.6). That is,

$$\gamma(x^*) \ge 0.878\, \psi(\theta).$$

However, since we cannot guarantee that $\psi(\theta)$ is an upper bound on the maximum cut value, there is no performance guarantee. Nevertheless, we do have the property that, in a weak sense and to some extent, the better local maximum of $\psi(\theta)$ (or, equivalently, local minimum of $f(\theta)$) we obtain, the better cut will likely be produced. To see this, let x_a^* and x_b^* be two binary cuts generated by Procedure-CUT from θ_a and θ_b, respectively. If $\gamma(x_a^*) \le \psi(\theta_a)$ and $\psi(\theta_b) > \frac{1}{0.878}\psi(\theta_a)$, then since

$$\gamma(x_b^*) \ge 0.878\ \psi(\theta_b) > \psi(\theta_a) \ge \gamma(x_a^*),$$

x_b^* is a better cut than x_a^*.

After we minimize the function $f(\theta)$ and obtain a local minimum θ^1, we will call Procedure-CUT to produce a best possible cut x^1 associated with θ^1. At this point, we may stop and return the generated cut x^1. On the other hand, if we are willing to spend more time, we may try to improve the quality of our approximation.

We know that the angular representation of the cut x^1, $\theta(x^1)$, is a stationary point—most likely a saddle point—of the function $f(\theta)$, but not a minimizer unless it is already a maximum cut. Assuming that $\theta(x^1)$ is in fact a saddle point, it is probable that close by there are local minima of f that are deeper than θ^1 is. Although we cannot restart the minimization directly from the stationary point $\theta(x^1)$, we can certainly restart from a slight perturbation of $\theta(x^1)$ and hopefully escape to a better local minimum θ^2 which in turn would hopefully lead to a better cut x^2 or $\theta(x^2)$. We can continue this process until we reach a cut from which we deem that further improvement seems unlikely. We state this heuristic as the following algorithm:

Algrithm-1: (input N, θ^0, output x^*)

Given $\theta^0 \in \Re^n$, integer $N \ge 0$, let $k = 0$ and $\Gamma = -\infty$.

While $k \le N$

1. Starting from θ^0, minimize f to get θ.
2. Compute a best cut x associated with θ by Procedure-CUT.
3. If $\gamma(x) > \Gamma$, let $\Gamma = \gamma(x)$, $x^* = x$ and $k = 0$; otherwise set $k = k + 1$.
4. Set θ^0 to a random perturbation of the angular representation of x.

End

The parameter N controls how many consecutive, non-improving random perturbations are allowed before we stop the algorithm. If so desired, the algorithm can be run M times with multiple starting points θ^0 to increase the chances of achieving better cuts. Generally speaking, the larger N and M, the longer time the algorithm will take to run, and the better cut it will return.

A geometric interpretation of Algorithm-1 is as follows. After we arrive at a local minimum of f, we search around this local minimum for a nearby saddle point (i.e., a cut) that has the lowest f-value in the neighborhood. We then move to the saddle point and restart the minimization to locate a nearby local minimum that, hopefully, has a smaller f-value than the previous one. We repeat this process until we deem that the search has become unfruitful.

5. Computational Results for Max-Cut. We have implemented Algorithm-1 in a Fortran90 code named "CirCut." For the minimization of $f(\theta)$, we use a simple gradient algorithm with a back-tracking Armijo line-search. Since numerical experiments indicate that the accuracy of the minimization is not crucial, we stop the minimization when the relative change in the function value is less than 10^{-4}.

In CirCut, we also include an option for a simple local search in the cut space; that is, after a cut is returned from Procedure-CUT, one has the option to improve it through a quick local search that moves one or two nodes at a time, producing a so-called locally 2-optimal solution. This feature can often slightly improve the quality of a cut and is therefore set to be a default feature unless specified otherwise.

We compare our code CirCut with two SDP codes SBmethod and DSDP, both implementing the Goemans-Williamson randomized algorithm (along with other features). Since these codes produce both an upper bound and a lower bound while our code only gives the latter, the comparisons should not be taken at face value. In carrying out such comparisons, we have two objectives in mind. Firstly, since our heuristic is derived from the Goemans-Williamson randomized algorithm by a rank restriction, we want to see how our modifications affect the performance, both time-wise and quality-wise, of generating lower bounds. Secondly, since the approximation quality of Goemans-Williamson randomized algorithm has been shown to be at least as good as a number of heuristics [11], through the comparisons we hope to get a good picture on the approximation quality of our heuristic. We select the codes SBmethod and DSDP for our comparisons because they represent the state of the art in solving large-scale SDP problems.

We also compare our code with a state-of-the-art heuristic code for Max-Cut problems from the Ising spin glass model in physics, developed by Hartmann [19]. The purpose of this comparison is self-evident.

5.1. Comparison with SBmethod. We first report numerical results on the Max-Cut problem in comparison with SBmethod, an SDP code developed by Helmberg and Rendl [20]. SBmethod solves a large class of semidefinite programs using a specialized bundle method, the so-called spectral bundle method, and in particular is one of the fastest codes for solving Max-Cut SDP relaxations.

The first set of test problems comes from the DIMACS library of mixed semidefinite-quadratic-linear programs [12]. This set contains four Max-Cut problems, called the torus problems, which originated from the Ising model of spin glasses in physics (see Section 5.3 for details). In Table 5.1, we give statistics for this set of problems; note that the sizes of the graphs are given as $(|V|, |E|)$. In the table, the columns "Lower Bound" and "Upper Bound" give the best lower and upper bounds on the maximum cut value known to us to date, and the column "SDP bound" gives the SDP upper bounds on the maximum cut values. All the lower and upper bounds were supplied to us by Michael Jünger and Frauke Liers [22] except for the lower bounds 2988 for `pm3-15-50` and 285790637 for `g3-15` which were the best cut values so far obtained by CirCut on these two problems, respectively. We mention that for `pm3-8-50` and `g3-8`, the best cut values so far obtained by CirCut are, respectively, 454 and 41684814, and the latter value is optimal.

In Table 5.2, we present a comparison between the code SBmethod and our code CirCut. Since the latest version of SBmethod does not include the functionality of generating cuts by the Goemans-Williamson randomized procedure, we used an earlier version that does. It is quite likely that the latest

TABLE 5.1
Statistics for the Torus Set of Max-Cut Problems

Graph Name	Size	Lower Bound	Upper Bound	SDP Bound
pm3-8-50	(512, 1536)	456	461	527
pm3-15-50	(3375, 10125)	2988	3069.51	3474
g3-8	(512, 1536)	41684814	41684814	45735817
g3-15	(3375, 10125)	2.85790e+8	2.87725e+8	3.1346e+8

version of SBmethod would produce better timings than those presented in the table.

We ran both SBmethod and CirCut on an SGI Origin2000 machine with sixteen 300MHZ R12000 processors at Rice University. Since neither code is parallel, however, only one processor was used at a time. For both codes, the cut values were obtained without any post-processing heuristics, i.e., the simple local search feature of CirCut was not invoked. The default parameter settings were used for SBmethod. In Table 5.2, the cut value and computation time are reported for each problem. For CirCut, the value of M is the number of times Algorithm-1 was run with random starting points, and the value of N is the parameter required by Algorithm-1. The average time per run, the average cut value, and the best value in the M runs are reported in the last three columns of the table, respectively. All the reported times are in seconds. From the table, it is clear that an average run of CirCut is much faster and produces better quality cuts on all four test problems.

TABLE 5.2
Comparison with SBmethod on Max-Cut Problems from the Torus Set

Graph	SBmethod		CirCut ($N = 4$, $M = 100$)		
Name	Value	Time	Avg. Value	Avg. Time	Best Value
pm3-8-50	434	28.72	443	0.218	452
pm3-15-50	2728	2131.89	2888	2.332	2936
g3-8	4.04736e+7	36.03	4.09098e+7	0.298	4.13946e+7
g3-15	2.73412e+8	3604.54	2.74357e+8	2.835	2.77917e+8

More results are reported in Table 5.3 for CirCut using different values of N. These results indicate that the variations between the average and best cut values are quite moderate, and they also show that even with $N = 0$ (no further improvement attempted after minimization), CirCut gives quite respectable cuts in a minimal amount of time on the average. As N increases, CirCut produces better quality cuts, and uses more time of course. However, even for $N = 8$, CirCut is still faster by orders of magnitude.

However, we should bear in mind that in every run SBmethod also produces an upper bound, hence the running times for CirCut and SBmethod are not exactly comparable. They become totally comparable only when the sole objective of the computation is to obtain approximate solutions. These comments also apply to the comparisons presented in the next subsection and in Section 6.

5.2. Comparison with DSDP. The second set of test problems are from the so-called G-set graphs which are randomly generated. Recently, Choi and Ye reported computational results on a subset of G-set graphs that were solved as Max-Cut problems using their SDP code COPL-DSDP [9], or simply DSDP. The code DSDP uses a dual-scaling interior-point algorithm and an iterative linear-

TABLE 5.3
More CirCut Results on Max-Cut problems from the Torus Set

Graph	CirCut ($N = 0$, $M = 100$)			CirCut ($N = 8$, $M = 100$)		
Name	Avg. Val	Avg. Time	Best Val	Avg. Val	Avg. Time	Best Val
pm3-8-50	430	0.031	444	448	0.386	454
pm3-15-50	2792	0.212	2834	2937	4.272	2964
g3-8	37870328	0.024	40314704	40917332	0.538	41684814
g3-15	253522848	0.154	264732800	277864512	7.880	281029888

equation solver. It is currently one of the fastest interior-point codes for solving SDP problems.

We ran CirCut on a subset of G-set graphs as Max-Cut problems and compare our results with those reported in Choi and Ye [9]. The comparison is given in Table 5.4, along with graph name and size information. We emphasize that the timing for DSDP was obtained on an HP 9000/785/C3600 machine with a 367 MHZ processor [8], while ours was on the aforementioned SGI Origin2000 machine at Rice University. These two machines seem to have comparable processing speeds. We did not run DSDP on the same computer at Rice University for several reasons: (1) the latest version of DSDP with an iterative linear-equation solver has not yet been made publicly available; (2) since the speeds of DSDP and CirCut are orders of magnitude apart, a precise timing is unnecessary in a qualitative comparison, and (3) it would be excessively time-consuming to rerun DSDP on all the tested problems (as can be see from Table 5.4).

The first two columns of Table 5.4 contain information concerning the tested graphs, followed by timing (in seconds) and cut value information. The DSDP results were given as reported in [9]. We ran CirCut using two sets of parameters: C1 results were for $N = 0$ and $M = 1$ (no further improvement after minimization and a single starting point); and C2 for $N = 10$ and $M = 5$. Note that in this table the running times listed for C2 include all $M = 5$ runs, i.e., the times are not averaged as in the previous tables.

We observe that C1 took less than 11 seconds to return approximate solutions to all the 27 test problems with a quality that, on average, is nearly as good as that of the DSDP cuts which required more than 5 days of computation. On the other hand, C2 took more time to generate the cuts, but the quality of the C2 cuts is almost uniformly better than those of DSDP with one exception. Only on problem G50 did DSDP produce a slightly better cut. We mention, however, that CirCut can easily find a cut of the same value on G50 if M is set to a larger value.

5.3. Comparison with a Heuristic Algorithm from Physics. An area of great interest in modern physics is the study of spin glasses [3, 14], and the particular problem of computing the so-called groundstate of an Ising spin glass can be cast as the problem of finding a maximum cut in a edge-weighted graph. In this section, we compare our heuristic CirCut with a successful heuristic by Hartmann [19] for finding an approximation to the groundstate of specially structured spin glasses.

Roughly speaking, a spin glass is a collection of n magnetic spins that possesses various interactions between the spins and also exhibits disorder in its frozen, or low-energy, state. In the collection, each spin can take on one of a finite number of positions. For example, when there are exactly two possible positions, the two positions are imagined as "up" and "down" (or $+1$ and -1). In addition, the interactions between the spins describe how the the positions of a given spin and its "neighbor" spins affect

TABLE 5.4
Comparison with DSDP on Max-Cut Problems from the G-set

Graph		Time			Value		
Name	Size	DSDP	C1	C2	DSDP	C1	C2
G11	(800, 1600)	16.6	0.06	3.88	542	524	554
G12	(800, 1600)	17.7	0.06	3.76	540	512	552
G13	(800, 1600)	18.2	0.06	3.45	564	536	572
G14	(800, 4694)	35.2	0.09	5.53	2922	3016	3053
G15	(800, 4661)	32.1	0.09	5.91	2938	3011	3039
G20	(800, 4672)	32.0	0.11	5.56	838	901	939
G21	(800, 4667)	37.6	0.08	5.56	841	887	921
G22	(2000, 19990)	4123.3	0.36	22.31	12960	13148	13331
G23	(2000, 19990)	3233.5	0.37	18.85	13006	13197	13269
G24	(2000, 19990)	3250.7	0.30	27.30	12933	13195	13287
G30	(2000, 19990)	3718.9	0.32	23.77	3038	3234	3377
G31	(2000, 19990)	3835.7	0.33	19.61	2851	3146	3255
G32	(2000, 4000)	142.6	0.18	13.09	1338	1306	1380
G33	(2000, 4000)	132.5	0.14	12.62	1330	1290	1352
G34	(2000, 4000)	156.7	0.12	9.82	1334	1276	1358
G50	(3000, 6000)	264.6	0.17	15.71	5880	5748	5856
G55	(5000, 12498)	1474.8	0.54	39.73	9960	10000	10240
G56	(5000, 12498)	15618.6	0.46	33.52	3634	3757	3943
G57	(5000, 10000)	1819.8	0.48	32.23	3320	3202	3412
G60	(7000, 17148)	58535.1	0.73	56.75	13610	13765	14081
G61	(7000, 17148)	52719.6	0.51	63.57	5252	5429	5690
G62	(7000, 14000)	5187.2	0.47	47.04	4612	4486	4740
G64	(7000, 41459)	102163.9	0.94	67.56	7624	8216	8575
G70	(10000, 9999)	33116.2	0.37	94.39	9456	9280	9529
G72	(10000, 20000)	12838.1	0.72	86.59	6644	6444	6820
G77	(14000, 28000)	32643.4	0.95	109.41	9418	9108	9670
G81	(20000, 40000)	131778.2	1.49	140.46	13448	12830	13662

the overall energy of the spin glass. For example, in Table 5.5, we show the energy contributed by two interacting spins i and j for a spin glass in which (i) there are two possible positions for a spin, (ii) all interactions act pair-wise between spins, and (iii) each interaction is either positive or negative.

The groundstate, or low-energy state, of a spin glass occurs when the positions of the n spins are chosen so as to minimize the overall energy of the spin glass, and spin glasses are characterized by the fact that their groundstate is disordered, that is, all interactions cannot be satisfied with zero energy and hence the overall energy of the system is positive. (Note that the standard physics terminology differs somewhat from—but is equivalent to—our terminology.)

A special subclass of spin glasses, called the Ising spin glasses, has been studied extensively. Ising spin glasses satisfy items (i) and (ii) of the previous paragraph, and the so-called $\pm J$ model of Ising

TABLE 5.5
Energy levels of two interacting spins

i	j	interaction	energy
up	up	$+$	0
up	down	$+$	1
down	up	$+$	1
down	down	$+$	0
up	up	$-$	1
up	down	$-$	0
down	up	$-$	0
down	down	$-$	1

spin glasses also satisfies item (iii). It is not difficult to see that this model can be represented by an edge-weighted graph $\bar{G} = (V, E, \bar{W})$, where the vertex set V consists of the n spins, the edge set E describes the pair-wise interactions, and the symmetric weight matrix $\bar{W} = (\bar{w}_{ij})$ has $\bar{w}_{ij}$ equal to 1, -1, or 0, respectively, if i and j interact positively, negatively, or not at all. Moreover, if a variable x_i that can take on values $+1$ or -1 is used to represent the position of spin i, then the groundstate of the Ising spin glass can be seen to be the optimal solution of the optimization

$$\begin{array}{ll} \min & \sum_{(i,j)\in E} \frac{1}{2}(1 - \bar{w}_{ij} x_i x_j) \\ \text{s.t.} & |x_i| = 1, \quad i = 1, \ldots, n. \end{array} \tag{5.1}$$

After some immediate simplifications, (5.1) can be written in the equivalent form (2.2), where $w_{ij} = -\bar{w}_{ij}$, that is, (5.1) is equivalent to the maximum cut problem on the graph $G = (V, E, W)$, where $W = -\bar{W}$.

Many approaches for solving (5.1) have been investigated in both the physics community as well as the optimization community (see [2, 28]). Recently, one of the most successful heuristic approaches for solving (5.1) has been the approach of Hartmann [19], which in particular focuses on finding the groundstates of $\pm J$ Ising spin glasses that can be embedded as square or cubic lattices in two or three dimensions, respectively. The interactions are of the type "nearest neighbor" so that each vertex (or spin) has four neighbors in two dimensions and six in three dimensions. Such lattice graphs lead to regular graphs having a great deal of structure. In addition, Hartmann considers interactions in which negative interactions occur as many times as positive interactions, that is, $\sum_{(i,j)\in E} \bar{w}_{ij} = 0$. Hartmann reported strong computational results with square lattices having side length $L = 4, 5, \ldots, 30$ and cubic lattices having length $L = 4, 5, \ldots, 14$. Note that the square lattices have a total of L^2 vertices and that the cubic lattices have a total of L^3 vertices.

Although we refer the reader to [19] for a full description of Hartmann's algorithm, we summarize the basic idea of the method here. Given a feasible solution x to (5.1), the algorithm tries to find a new feasible solution $\hat{x}$ having less energy by using x to randomly build up a set of nodes $\hat{V}$ for which the groundstate $x_{\hat{V}}$ of the induced graph on $\hat{V}$ can be found in polynomial time using a max-flow min-cut algorithm. Then $\hat{x}$ is formed from x by setting $\hat{x}_i = (x_{\hat{V}})_i$ if $i \in \hat{V}$ and $\hat{x}_i = x_i$ if $i \notin V$. The energy of $\hat{x}$ is guaranteed to be no worse than that of x, and so this procedure can be iterated until the energy exhibits no strict improvement from iteration to iteration. Various parameters of the algorithm can affect its running time and also the quality of solution that is returned; these parameters determine the number of iterations allowed with no improvement, the number of independent times the overall algorithm is run, and, more generally, the exhaustiveness of the search performed by the algorithm.

We ran both CirCut and the algorithm of Hartmann on the same SGI Origin 2000 used for the computational results in the previous subsections. Hartmann's code is written in ANSI C and uses only one processor. In addition, we compiled both codes with the same compiler optimization option. In Table 5.6, we compare CirCut with the algorithm of Hartmann on twenty graphs arising from twenty cubic lattices having randomly generated interaction magnitudes; these problems are of the same type that Hartmann investigated in [19]. Ten of the graphs have $(L, n, |E|) = (10, 1000, 3000)$; and ten have $(L, n, |E|) = (14, 2744, 8232)$. We note that, for comparison purposes, the output of each algorithm is in terms of the equivalent maximum cut problem. Two versions of CirCut corresponding to the parameter choices $(N, M) = (10, 5)$ and $(N, M) = (50, 10)$ were run on all thirty graphs; the versions are named C1 and C2, respectively. Similarly, two versions H1 and H2 of Hartmann's algorithm were run such that H1 performed a less exhaustive search than H2. We remark that H2 represented the default parameters supplied to us by Hartmann.

TABLE 5.6
Comparison of CirCut and Hartmann's algorithm

Graph			Cut Values				Times			
#	$\|V\|$	$\|E\|$	C1	C2	H1	H2	C1	C2	H1	H2
1	1000	3000	874	880	882	896	5	39	69	9528
2	1000	3000	894	892	892	900	7	47	68	9605
3	1000	3000	878	882	878	892	6	45	68	9537
4	1000	3000	888	894	890	898	7	54	68	9583
5	1000	3000	878	880	876	886	6	48	69	9551
6	1000	3000	866	876	874	888	6	47	68	9555
7	1000	3000	882	894	890	900	8	57	69	9564
8	1000	3000	872	874	870	882	7	53	69	9629
9	1000	3000	884	896	888	902	6	48	68	9551
10	1000	3000	876	888	884	894	5	56	69	9629
11	2744	8232	2396	2410	2382	2446	22	219	236	33049
12	2744	8232	2398	2426	2390	2458	20	170	236	32836
13	2744	8232	2382	2404	2370	2442	20	165	235	33171
14	2744	8232	2398	2418	2394	2450	19	173	236	33136
15	2744	8232	2382	2412	2370	2446	20	177	235	32851
16	2744	8232	2404	2416	2384	2450	23	183	236	33129
17	2744	8232	2390	2406	2384	2444	19	166	234	32999
18	2744	8232	2412	2414	2386	2446	28	171	236	33089
19	2744	8232	2382	2390	2356	2424	31	187	235	32963
20	2744	8232	2410	2422	2388	2458	19	166	236	33140

Table 5.6 contains data corresponding to the four algorithms' performance on each of the twenty graphs. The first three columns give the graph number, the size of V, and the size of E. The next four columns give the cut value found by the algorithms, and the final four columns give the times (in

seconds) required by each of the algorithms.

It can be seen from the table that on the first ten graphs 1-10 C1 had the fastest speed, but the cuts it returned were in a few cases inferior to those produced by H1. On the other hand, C2 was able to produce better cuts than H1 in a considerably less amount of time. The overall winner in terms of cut values on the graphs 1–10 was H2, but this performance was achieved at the expense of very large computation times. For the second set of graphs 11–20, we see that both C1 and C2 outperformed H1 in terms of cut values and that C1 was much faster than H1 and C2 was notably faster than H1 as well. Again, H2 returned the best cuts but took a very long time. In all the cases, the differences in the quality of cuts generated by the algorithms are small percentage-wise. For example, on average C1 attained over 98 percent of the cut value of H2 in an amount of time less than one-tenth of a percent of that used by H2.

Overall, the results seem to indicate that C2 is a good choice when quality cuts are needed in a short amount of time. In particular, C2 is at least as effective as H1. In addition, C1 is a good alternative, especially when the size of the graph becomes large. When high quality cuts are needed and time is not an issue, H2 is the best choice. Moreover, we remark that, based on some unreported experimentation, CirCut does not seem to be able to achieve the same cut values as H2 even if CirCut is allowed to search for a very long time.

6. Some Extensions. Conceptually, there is little difficulty in extending the rank-two relaxation idea to other combinatorial optimization problems in the form of a binary quadratic program, especially to those arising from graph bipartitioning. For a given problem, however, whether or not the rank-two relaxation will lead to high-performance algorithms, like the one we have demonstrated for Max-Cut, must be determined by an individual investigation and a careful evaluation. Close attention must also be paid to the specific structure of each problem in order to obtain good algorithms.

In this section, we focus on extending the rank-two relaxation idea to a close relative of Max-Cut — the Max-Bisection problem. Max-Bisection is the same as Max-Cut except that it has the additional constraint $e^T x = 0$ (i.e., the number of positive ones in x must equal the number of negative ones, hence implying that n should be even), which can also be written as

$$(e^T x)^2 = (ee^T) \bullet (xx^T) = 0.$$

After the removal of the rank-one restriction, one obtains the following SDP relaxation of the Max-Bisection problem (comparable to (2.5)):

$$\begin{array}{rc} \min & \frac{1}{2} W \bullet X \\ \text{s.t.} & \operatorname{diag}(X) = e, \\ & ee^T \bullet X = 0, \\ & X \succeq 0. \end{array} \tag{6.1}$$

Randomized procedures similar to the Goemans-Williamson technique for Max-Cut have been proposed with different performance guarantees for Max-Bisection; see [15, 32], for example.

In a similar fashion as with Max-Cut, using the rank-two relaxation and polar coordinates, we obtain a new relaxation for Max-Bisection:

$$\begin{array}{rc} \min & f(\theta) \\ \text{s.t.} & e^T \cos(T(\theta)) e = 0. \end{array} \tag{6.2}$$

Suppose that we have obtained a (local or global) minimizer θ for (6.2). How do we generate a bisection? Without loss of generality, let us assume that n is even and that θ satisfies $\theta_i \in [0, 2\pi)$, $i = 1, \ldots, n$. We may also assume that, after a reordering,

$$\theta_1 \leq \theta_2 \leq \ldots \leq \theta_n.$$

Then, to generate a bisection, we pick any integer $k \in [1, n/2)$ and let

$$x_i = \begin{cases} 1, & \text{if } i \in [k, k + n/2), \\ -1, & \text{otherwise.} \end{cases} \tag{6.3}$$

The following procedure efficiently considers all possible values of k in (6.3) and saves the best resultant bisection:

Procedure-BIS (input θ, output x^*):

Given $\theta \in \Re^n$ such that $0 \leq \theta_1 \leq \cdots \leq \theta_n < 2\pi$, let $\Gamma = -\infty$.

For $k = 1, \ldots, n/2 - 1$

1. Generate a cut x by (6.3) and compute $\gamma(x)$.
2. If $\gamma(x) > \Gamma$, then let $\Gamma = \gamma(x)$ and $x^* = x$.

End

Instead of solving the constrained relaxation (6.2), we have found through numerical experiments that solving the unconstrained relaxation (3.2) can generate the same or better quality bisections while taking less time. Intuitively, this is not hard to understand since the best bisection generated by Procedure-BIS for a given θ is dependent only on the ordering of the points along the circle and independent of the actual locations of the points. In fact, it is easy to verify that the constraint in (6.2) is equivalent to $\|[v_1 \ \cdots \ v_n]e\|^2 = 0$ where $v_i = [\cos(\theta_i) \ \sin(\theta_i)]^T$; that is, the n vectors on the unit circle must sum up to zero. So by itself, the constraint puts a restriction on the locations of points, but has nothing to do with their ordering. Hence, whether a given θ satisfies the constraint or not has no bearing on the quality of the bisection x^* generated by Procedure-BIS. On the other hand, the quality of x^* depends greatly on the objective value $f(\theta)$. Since it is more likely to obtain lower function values at unconstrained local minima than at constrained ones, we are thus more likely to obtain better bisections without the constraint.

In view of this, we construct our heuristic algorithm based on minimizing $f(\theta)$ without the additional constraint. We simply replace Procedure-CUT in Algorithm-1 by Procedure-BIS, and obtain a heuristic algorithm for the Max-Bisection problem, which we call Algorithm-2. In Algorithm-2, we also have the option of improving a cut by a minimal local search that allows swapping only a pair of nodes at a time, and is set to be a default feature.

We ran Algorithm-2 of CirCut on a subset of the G-set problems plus two additional test problems. These extra problems were contained in a test set used in Choi and Ye [9] and are publicly available.

In Table 6.1, we compare the results of CirCut with the results of DSDP reported in [9]. Again, we mention that the timing for DSDP was obtained on an HP 9000/785/C3600 computer with a 367 MHZ processor, while ours was on an SGI Origin2000 machine with sixteen 300 MHZ processors at Rice University. (Note, however, that both codes always use a single processor.)

Again, the first two columns of Table 6.1 contain the information on the tested graphs, followed by timing (in second) and cut value information. We ran CirCut using two sets of parameters: C1 results were for $N = 0$ and $M = 1$ (no further improvement after minimization and a single starting point); and C2 for $N = 5$ and $M = 1$.

C1 took less than 22 seconds to return approximate solutions to all 13 test problems with a quality that is superior to that of DSDP on average. While C2 took more time to generate the bisections, the quality of the bisections generated by C2 is better than that of DSDP on all but one problem: G50. Again, we mention that if N and M are set to larger values, CirCut is able to produce a bisection of the same value on G50 as that of DSDP's, within a time still much shorter than that required by DSDP.

TABLE 6.1
Comparison with DSDP on Max-Bisection Problems

Graph		Time			Value		
Name	Size	DSDP	C1	C2	DSDP	C1	C2
G50	(3000, 6000)	462.2	0.29	2.29	5878	5690	5830
G55	(5000,12498)	1793.4	0.46	4.32	9958	10007	10171
G56	(5000,12498)	20793.5	0.44	3.36	3611	3672	3835
G57	(5000,10000)	2090.8	0.32	2.98	3322	3146	3382
G60	(7000,17148)	48949.9	0.54	4.66	13640	13759	13945
G61	(7000,17148)	42467.2	0.62	7.16	5195	5312	5545
G62	(7000,14000)	5446.0	0.50	4.98	4576	4402	4706
G64	(7000,41459)	123409.7	0.92	12.05	7700	8056	8431
G72	(10000,20000)	15383.9	0.76	7.34	6628	6314	6736
G77	(14000,28000)	36446.7	1.15	11.38	6560	8980	9638
G81	(20000,40000)	334824.2	1.54	26.87	9450	12582	13618
bm1	(882,4711)	33.9	0.08	0.65	848	857	863
biomedp	(6514,629839)	46750.7	13.89	37.55	5355	5575	5593

6.1. Maximization versus Minimization. So far, we have only presented computational results on maximization problems, i.e., the Max-Cut and Max-Bisection problems which are equivalent to minimizing $f(\theta)$. Moreover, all of the graphs in the test sets have either all positive edge weights or a combination of both positive and negative weights.

Now let us consider the corresponding minimization problems on these graphs, equivalent to maximizing $f(\theta)$. For those graphs having both positive and negative weights, one can apply the same algorithms to the minimization problems by simply minimizing $-f(\theta)$ instead of $f(\theta)$. Things are not so simple, however, if all the weights are positive. In this case, it is easy to see that the global minimum of $-f(\theta)$ is attained whenever all n points coincide on the unit circle such that $\cos(\theta_i - \theta_j) \equiv 1$. This result makes sense for the Min-Cut problem in that the minimum cut in a graph with all positive weights is to have all nodes on one side of the cut (i.e., to have no cut at all). On the other hand, this result does not have a meaningful interpretation for Min-Bisection, creating a challenge for generating a bisection whenever a global minimum of $-f(\theta)$ is attained (although actually finding a global minimum may not happen often). An obvious, possible remedy to this problem is to reinstall the bisection constraint back into the formulation. Further investigation is clearly needed for the Min-Bisection problem.

7. Concluding Remarks. The computational results indicate that the proposed rank-two relaxation heuristics are effective in approximating the Max-Cut and Max-Bisection problems. Being able to return high-quality approximate solutions in a short amount of time, they are particularly useful in situations where either the problem is very large or time is at a premier.

Several factors have contributed to the performance of the rank-two relaxation approach: (1) the costs of local optimization are extremely low; (2) desirable properties relate the discrete problem to its rank-two relaxation enabling us to locate high-quality local minima; and (3) good local minima of the rank-two relaxation appear to be sufficient for generating good approximate solutions to the discrete problem.

The proposed heuristics consistently produce better quality approximate solutions while taking only a tiny fraction of time in comparison to the SDP relaxation approach, particularly on larger problems. This fact suggests that as a practical technique for producing lower bounds, the SDP relaxation approach does not seem to hold much promise, at least for the Max-Cut and the Max-Bisection problems. In addition, the rank-two relaxation heuristic compares favorably to other heuristics, i.e., ones that are not based on the SDP relaxation.

It is known that, besides Max-Cut, a number of other combinatorial optimization problems can also be formulated as unconstrained binary quadratic programs in the form of (2.2), such as the Max-Clique problem (see [4], for example). These are potential candidates for which the rank-two relaxation approach may also produce high-performance heuristic algorithms. Further investigation in this direction will be worthwhile.

Acknowledgment. We are grateful to an anonymous referee for valuable comments and suggestions that have helped us to improve the paper. We would like also thank Alexander Hartmann for sharing his computer code with us.

REFERENCES

[1] S. Benson, Y. Ye, and X. Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM Journal on Optimization.*, 10:443–461, 2000.

[2] B. A. Berg and T. Celik. Phys. Rev. Lett., 69, 2292–2295, 1992.

[3] K. Binder and A. P. Young. Rev. Mod. Phys., 58, 801–977, 1986.

[4] I. Bomze and M. Budinich and P. Pardalos and M. Pelillo. The maximum clique problem. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, volume 4. Kluwer Academic Publishers, Boston, MA, 1999.

[5] S. Burer and R. D. C. Monteiro. A Projected Gradient Algorithm for Solving the Maxcut SDP Relaxation. Working paper, School of ISyE, Georgia Tech, USA, December 1998. (To appear in *Optimization Methods and Software*.)

[6] S. Burer, R. D. C. Monteiro, and Y. Zhang. Solving a Class of Semidefinite Programs via Nonlinear Programming. Submitted to *Mathematical Programming A*. (Also Technical Report TR99-17, Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005, USA, September 1999.)

[7] S. Burer, R. D. C. Monteiro, and Y. Zhang. Interior-Point Algorithms for Semidefinite Programming Based on A Nonlinear Programming Formulation. Technical Report TR99-27, Department of Computational and Applied Mathematics, Rice University,

[8] C. Choi. Private communication. October, 2000.

[9] C. Choi and Y. Ye. Solving Sparse Semidefinite Programs Using the Dual Scaling Algorithm with an Iterative Solver. Working paper, Department of Management Science, University of Iowa, Iowa, 2000.

[10] C. Delorme and S. Poljak. Laplacian Eigenvalues and the Maximum Cut Problem. *Mathematical Programming*, 62:557–574, 1993.

[11] O. Dolezal, T. Hofmeister and H. Lefmann. A Comparison of Approximation Algorithms for the Maxcut-Problem. Manuscript, Universität Dortmund, Lehrstuhl Informatik 2, 44221 Dortmund, Germany. 1999.

[12] See the website: `http://dimacs.rutgers.edu/Challenges/Seventh/Instances/`.

[13] U. Feige and G. Schechtman. On the optimality of the random hyperplane rounding technique for MAX CUT. Manuscript, Faculty of Mathematics and Computer Science, Weizmann Institute, Rehovot 76100, Israel, Oct. 2000.

[14] K. H. Fisher and J. A. Hertz. *Spin Glasses*. Cambridge University Press, 1991.

[15] A. Frieze and M. Jerrum. Improved algorithms for Max K-cut and Max bisection. *Algorithmica*, 18 (1997), 67-81.

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

[17] M. Fukuda, M. Kojima, K. Murota, and K. Nakata. Exploiting Sparsity in Semidefinite Programming via Matrix Completion I: General Framework. *SIAM Journal on Optimization*, 11:647–674, 2001.

[18] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, 42:1115–1145, 1995.

[19] A. K. Hartmann. Cluster-Exact Approximation of Spin Glass Groundstates. Working paper, Institut für Theoretische Physik, Universität Heidelberg, Heidelberg, July 1998.

[20] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10:673–696, 2000.

[21] M. Peinado and S. Homer. Design and performance of parallel and distributed approximation algorithms for maxcut. *Journal of Parallel and Distributed Computing*, 46:48–61, 1997.

[22] M. Jünger and Frauke Liers. Private communication. November, 2000.

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

[24] L. Lovász and A. Schrijver. Cones of matrices and set functions, and 0-1 optimization. *SIAM Journal on Optimization*, 1:166–190, 1991.

[25] Y. E. Nesterov and A. S. Nemirovskii. *Interior Point Methods in Convex Programming: Theory and Applications.* Society for Industrial and Applied Mathematics, Philadelphia, 1994.

[26] S. Poljak, and F. Rendl. Nonpolyhedral Relaxations of Graph-Bisection Problems. *SIAM Journal on Optimization*, 5:467–487, 1995.

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

[28] C. De Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt and G. Rinaldi. Exact Ground States of Ising Spin Glasses: New Experimental Results with a Branch and Cut Algorithm. *Journal of Statistical Physics*, 80, 487–496, 1995.

[29] K.-C. Toh and M. Kojima. Solving some large scale semidefinite programs via the conjugate residual method. Research Report, Department of Mathematics, National University of Singapore, August 2000.

[30] R. J. Vanderbei and H. Yurttan Benson. On Formulating Semidefinite Programming Problems as Smooth Convex Nonlinear Optimization Problems. Technical Report ORFE 99-01, Dept. of Operations Research and Financial Engineering, Princeton University, Princeton NJ, November 1999.

[31] S. Vavasis. A Note on Efficient Computation of the Gradient in Semidefinite Programming. Working paper, Department of Computer Science, Cornell University, September 1999.

[32] Y. Ye. A .699-Approximation Algorithm for Max-Bisection. Working paper, Department of Management Science, University of Iowa, Iowa, 2000.