CS-01-01

A New Second-order Cone Programming Relaxation for MAX-CUT Problems

Masakazu Muramatsu
Department of Computer Science
The University of Electro-Communications
1-5-1 Chofugaoka, Chofu-shi, Tokyo, 182-8585 JAPAN.

Tsunehiro Suzuki
Department of Mechanical Engineering
Sophia University
7-1 Kioi-cho, Chiyoda-ku, Tokyo, 102 JAPAN.

March, 2001, Revised: May 2001.

Abstract

We propose a new relaxation scheme for the MAX-CUT problem using second-order cone programming. We construct relaxation problems to reflect the structure of the original graph. Numerical experiments show that our relaxation approaches give better bounds than those based on the spectral decomposition proposed by Kim and Kojima [11], and that the efficiency of the branch-and-bound method using our relaxation is comparable to that using semidefinite relaxation in some cases.

Key words: Second-order cone programming, MAX-CUT, Primal-dual Interior-Point Method, Relaxation, Branch-and-Bound

1. Introduction

In branch-and-bound methods for solving integer programming problems or nonconvex quadratic problems, the choice of relaxation problem significantly affects total performance of the algorithm. It is now popular to use semidefinite programming (SDP) ([4, 6, 7, 16, 19, 25]) for relaxation problems whenever possible. Although SDP relaxation gives a good bound, the computational cost of solving SDP is so expensive, despite efforts to develop better algorithms to solve SDP (e.g., [9]), that it is still difficult to use SDP problems in branch-and-bound methods for solving large problems. On the other hand, the so-called lift-and-project (reformulation-linearization) method ([2, 20]) has been developed to use linear programming (LP) for relaxation. Generally, LP can be solved much more easily than SDP. However, the bounds obtained by the lift-and-project method are worse than those of SDP relaxation unless other effective constraints are added.

Second-order cone programming (SOCP) is an optimization problem having linear constraints and second-order cone constraints. SOCP is a special case of symmetric cone programming ([3]), which also includes SDP and LP as special cases. Recently, primal-dual interior-point algorithms were developed for both SOCP ([14, 23, 24]) and symmetric cone programming ([17, 21, 15]). Several programs have been implemented to solve SOCP (e.g., [1]) and symmetric cone programming (e.g., [22]). Numerical experiments show that the computational cost of solving SOCP is much less than that of SDP, and similar to LP. It is natural to consider the use of SOCP for relaxation of integer programming or nonconvex quadratic problems, and this is the subject of this paper.

Kim and Kojima [11] first pointed out that SOCP can be used to relax nonconvex quadratic problems. The bounds their relaxation problems provided are worse than those of SDP relaxation, but better than those of lift-and-project LP formulations. It was reported in [11] that their problems consumed much less CPU time than SDP in practice. Their way of constructing SOCP is based on the spectral decomposition of indefinite matrices. There is no difficulty in solving general nonconvex quadratic problems. However, when we want to solve problems based on graphs, such as MAX-CUT problems, spectral decomposition destroys the graph's (possibly sparse) structure, and it is difficult to use more information about the graph. For example, it is not obvious with their method how to use the 'triangle inequalities' (see [8] for example) of MAX-CUT problems.

The aim of this paper is to find an efficient SOCP relaxation suitable for graph-based problems. We propose new relaxation problems for the MAX-CUT problem using SOCP. Our relaxation uses the same framework as [11], but our strategy for choosing effective inequalities is different. We obtain effective convex quadratic inequalities that reflect the structure of the original graph. Numerical experiments show that, while consuming slightly more CPU time, our relaxation problems always give better bounds than those of Kim and Kojima's method. Furthermore, we obtain the 'triangle inequalities', which restrict the feasible region of the relaxed problem efficiently.

We compare the efficiency of the relaxation methods in the context of the branch-and-bound method. Specifically, we implemented the branch-and-bound method to solve the MAX-CUT problems by using three SOCP relaxations ((i) Kim and Kojima's method, (ii) our original method, (iii) (ii) + triangle inequalities), and the SDP relaxation. The results show that the total performance of our methods is always much better than that of Kim and Kojima's method. Furthermore, our SOCP relaxation outperforms the SDP relaxation in sparse graphs.

This paper is organized as follows. In the next section, we describe Kim and Kojima's SOCP relaxation scheme for nonconvex quadratic problems, because we use the same framework of relaxation. Section 3 introduces the MAX-CUT problem and our SOCP relaxation, together with our version of the 'triangle inequalities'. Section 4 is devoted to showing the results of the numerical experiments. In Section 5, we give some concluding remarks.

We denote by $\mathcal{S}(n)$ the set of $n \times n$ real symmetric matrices. Also $\mathcal{S}(n)^+$ denotes the set of $n \times n$ positive semidefinite matrices. For $X, Y \in \mathcal{S}(n)$,

$$X \bullet Y := \sum_{i,j} X_{ij} Y_{ij}$$

and $X \succeq Y$ if and only if $X - Y \in \mathcal{S}(n)^+$. The second-order cone $\mathcal{K}(r)$ is defined by

$$\mathcal{K}(r) = \left\{ x \in \mathbb{R}^r \;\middle|\; x_1 \geq \sqrt{\sum_{j=2}^{r} x_j^2} \right\}.$$

The vector $e_j \in \mathbb{R}^n$ is the zero vector except for the j-th component, which is 1.

2. A nonconvex quadratic problem and its relaxation problems

In this section, we consider the following nonconvex quadratic problem:

$$\langle QP \rangle \begin{cases} \text{minimize} & c^t x \\ \text{subject to} & x^T Q_p x + q_p^T x + \gamma_p \leq 0, \quad p = 1, \ldots, m, \end{cases}$$

where $Q_p \in \mathcal{S}(n)$, $c \in \mathbb{R}^n$, $q_p \in \mathbb{R}^n$, and $\gamma_p \in \mathbb{R}$. We assume that $Q_p,\ p = 1, \ldots, m$ are indefinite matrices in general. Because $x^T Q_p x = Q_p \bullet xx^T$, $\langle QP \rangle$ can also be written as

$$\begin{cases} \text{minimize} & c^t x \\ \text{subject to} & Q_p \bullet X + q_p^T x + \gamma_p \leq 0, \quad p = 1, \ldots, m, \\ & X = xx^T. \end{cases}$$

This problem is NP-hard because of the last constraint. We now consider relaxing the problem by replacing this constraint by some other relations between X and xx^T.

If we simply ignore the constraint $X = xx^T$, we obtain the following LP:

$$\langle LP - QP \rangle \begin{cases} \text{minimize} & c^t x \\ \text{subject to} & Q_p \bullet X + q_p^T x + \gamma_p \leq 0 \quad (p = 1, \ldots, m), \end{cases}$$

This type of relaxation problem is often called 'lift-and-project' relaxation or the 'reformulation-linearization' technique.

The second idea is to use the property $X \succeq xx^T$ instead of $X = xx^T$. This constraint is called a semidefinite constraint, and using this we obtain the SDP relaxation:

$$\langle SDP - QP \rangle \begin{cases} \text{minimize} & c^t x \\ \text{subject to} & Q_p \bullet X + q_p^T x + \gamma_p \leq 0 \quad (p = 1, \ldots, m), \\ & X \succeq xx^T. \end{cases}$$

Obviously, $\langle SDP - QP \rangle$ gives a bound no worse than $\langle LP - QP \rangle$. On the other hand, the computational cost of $\langle LP - QP \rangle$ is much less than that of $\langle SDP - QP \rangle$.

The third relaxation using SOCP proposed by Kim and Kojima [11] is as follows. First, suppose that we are given $\mathcal{C} \subseteq \mathcal{S}(n)^+$. It is easy to see that for $Z \in \mathcal{S}(n)$,

$$Z \succeq 0 \;\Rightarrow\; \forall C \in \mathcal{C},\ C \bullet Z \geq 0. \tag{1}$$

Using this relation, we relax the constraint $X \succeq xx^T$ to $(X - xx^T) \bullet C \geq 0$ for $C \in \mathcal{C}$, which are convex quadratic constraints. Note that if $\mathcal{C} = \mathcal{S}(n)^+$, then the right-hand side of (1) also implies the left-hand side.

A convex quadratic constraint can easily be transformed into a second-order cone constraint. To do this, for $C \in \mathcal{C}$, we first decompose $C = UU^T$, where $U \in \mathbb{R}^{n \times k}$ and $k = \mathrm{Rank}(C)$. Such a decomposition is always possible, as C is symmetric and positive semidefinite. The constraint $C \bullet X \geq x^T C x$ is equivalent to

$$x^T UU^T x \leq C \bullet X. \tag{2}$$

Observe that for any $w \in \mathbb{R}^n$, $\eta, \xi \in \mathbb{R}$,

$$w^T w \leq \xi\eta,\ \xi \geq 0,\ \eta \geq 0 \Leftrightarrow \begin{pmatrix} \xi + \eta \\ \xi - \eta \\ 2w \end{pmatrix} \in \mathcal{K}(n+2).$$

Therefore, (2) is equivalent to

$$\begin{pmatrix} 1 + C \bullet X \\ 1 - C \bullet X \\ 2U^T x \end{pmatrix} \in \mathcal{K}(k+2).$$

This is the basic idea of the SOCP relaxation for nonconvex quadratic programming. The final form of the SOCP is as follows:

$$\langle SOCP-QP\rangle \left\{ \begin{array}{ll} \text{minimize} & c^t x \\ \text{subject to} & Q_p \bullet X + q_p^T x + \gamma_p \le 0, \quad p=1,\ldots,m, \\ & \begin{pmatrix} 1+C\bullet X \\ 1-C\bullet X \\ 2U^T x \end{pmatrix} \in \mathcal{K}(\mathrm{Rank}(C)+2), \quad i=1,\ldots,r, \\ & C\in\mathcal{C}, C=UU^T. \end{array} \right.$$

The problem $\langle SOCP-QP\rangle$ has $O(n^2)$ variables. This number of variables makes it difficult to solve the resulting SOCP when n is large. Kim and Kojima [11] proposed a technique to reduce the number of variables. In [11], they demonstrated that with this technique, the SOCP relaxation could have total performance as good as that of LP relaxation and SDP relaxation. We now describe their method.

For the sake of simplicity, we omit the subscript p and consider the linear inequality

$$Q\bullet X + q^T x + \gamma \le 0. \tag{3}$$

Let

$$Q=\sum_{j=1}^{n}\lambda_j u_j u_j^T$$

be the spectral decomposition of Q, where λ_j are eigenvalues and u_j are corresponding unit eigenvectors. Without loss of generality, we assume that

$$\lambda_1 \ge \ldots \ge \lambda_l \ge 0 > \lambda_{l+1} \ge \ldots \ge \lambda_n,$$

and put $Q^+ := \sum_{j=1}^{l}\lambda_j u_j u_j^T$. We choose Q^+ and $u_j u_j^T$, $j=l+1,\ldots,n$ for $\mathcal{C}$ to obtain the following inequalities:

$$x^T Q^+ x - Q^+ \bullet X \le 0 \tag{4}$$

$$x^T u_j u_j^T x - u_j u_j^T \bullet X \le 0 \quad j=l+1,\ldots,n. \tag{5}$$

Then, combining (3) and (4), we produce a new (weaker) inequality:

$$x^T Q^+ x + \sum_{j=l+1}^{n}\lambda_j u_j u_j^T \bullet X + q^T x + \gamma \le 0. \tag{6}$$

If (x,X) satisfies (3) and (4), then it also satisfies (6), but the converse is not generally true. Putting $z_j = u_j u_j^T \bullet X$, we obtain the following convex quadratic constraints that do not contain X:

$$x^T Q^+ x + \sum_{j=l+1}^{n}\lambda_j z_j + q^T x + \gamma \le 0, \tag{7}$$

$$x^T u_j u_j^T x - z_j \le 0, \quad j=l+1,\ldots,n. \tag{8}$$

We will call this parameter-reducing technique Kim and Kojima's SOCP relaxation, or Kim and Kojima's method.

A substantial advantage of Kim and Kojima's method is that we can reduce the number of variables from $O(n^2)$ to the total number of negative eigenvalues of Q_ps. On the other hand, the inequalities (7) and (8) are weaker than the original constraints (3),(4), and (5). In fact, if we do not impose any upper bound to z_j, then any x can satisfy (7) and (8) with large z_js (note that $\lambda_j < 0$ for $j > l$). Therefore, we require some restriction on z_j in advance.

3. The MAX-CUT problem and its relaxation

3.1. **The MAX-CUT problem.** Let $\mathcal{G}=(V,\mathcal{E})$ be an undirected graph where $V=\{1,\ldots,n\}$ and $\mathcal{E}$ are the sets of vertices and edges, respectively. We assume that a weight w_{ij} is attached to each edge $[i,j]\in\mathcal{E}$. For a partition $(S,\bar{S})$ of V, we define

$$w(S,\bar{S}) := \sum_{[i,j]\in\mathcal{E}, i\in S, j\in\bar{S}} w_{ij}.$$

The MAX-CUT problem is to find a partition maximizing $w(S, \bar{S})$.

For each $i \in V$, we put

$$x_i = \begin{cases} 1 & \text{if } i \in S \\ -1 & \text{if } i \in \bar{S}. \end{cases}$$

Because $(x_i - x_j)^2 = 4$ if i and j belong to different sets and 0 otherwise, we see that

$$w(S, \bar{S}) = \frac{1}{4} \sum_{[i,j] \in \mathcal{E}} w_{ij}(x_i - x_j)^2 = \frac{1}{2} \sum_{[i,j] \in \mathcal{E}} w_{ij} + \frac{1}{2} \sum_{[i,j] \in \mathcal{E}} w_{ij} x_i x_j.$$

Let us now define $L \in \mathcal{S}(n)$ by

$$L_{ij} = L_{ji} = \begin{cases} \sum_{[i,k] \in \mathcal{E}} w_{ik} & \text{if } i = j \\ -w_{ij} & \text{if } [i,j] \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases}$$

Then the objective function can be written as $x^T Lx/4$. Therefore, we can write the MAX-CUT problem as

$$\langle MC \rangle \begin{cases} \text{maximize} & x^T Lx/4 \\ \text{subject to} & x \in \{-1, 1\}^n. \end{cases} \tag{9}$$

Because

$$x_j = 1 \text{ or } -1 \Leftrightarrow x_j^2 = 1 \Leftrightarrow x_j^2 \leq 1 \text{ and } x_j^2 \geq 1 \Leftrightarrow x^T e_j e_j^T x \leq 1 \text{ and } x^T(-e_j e_j^T)x \leq -1,$$

and

$$\text{maximize } x^T Lx/4 \Leftrightarrow \text{ minimize } \theta \text{ subject to } -x^T Lx/4 \leq \theta,$$

$\langle MC \rangle$ is among the nonconvex quadratic problems introduced in Section 2.

3.2. **An SOCP relaxation for MAX-CUT problem.** We can now state our new SOCP relaxation for $\langle MC \rangle$ based on the general framework $\langle SOCP - QP \rangle$. Our aim is to use the structure of L. To do this, we first put

$$\begin{aligned} u_{ij} &:= e_i + e_j, \\ v_{ij} &:= e_i - e_j. \end{aligned}$$

Our choice of $\mathcal{C}$ consists of the following:

$$e_i e_i^T, \qquad i = 1, \ldots, n, \tag{10}$$

$$u_{ij} u_{ij}^T, \qquad [i,j] \in \mathcal{E}, \tag{11}$$

$$v_{ij} v_{ij}^T, \qquad [i,j] \in \mathcal{E}, \tag{12}$$

The corresponding convex quadratic constraints are:

$$x^T e_i e_i^T x - e_i e_i^T \bullet X \leq 0, \quad i = 1, \ldots, n, \tag{13}$$

$$x^T u_{ij} u_{ij}^T x - u_{ij} u_{ij}^T \bullet X \leq 0, \quad [i,j] \in \mathcal{E}, \tag{14}$$

$$x^T v_{ij} v_{ij}^T x - v_{ij} v_{ij}^T \bullet X \leq 0, \quad [i,j] \in \mathcal{E}, \tag{15}$$

We show that, like Kim and Kojima's SOCP relaxation, we can reduce the number of variables in a simpler and more efficient way by using the structure of the MAX-CUT problem. From (13) and the fact that $X_{ii} = 1$, we have

$$x^T e_i e_i^T x \leq 1, \quad i = 1, \ldots, n, \tag{16}$$

or $x_i^2 \leq 1$. By introducing new variables

$$s_{ij} := u_{ij}^T X u_{ij}, \quad [i,j] \in \mathcal{E}, \tag{17}$$

$$z_{ij} := v_{ij}^T X v_{ij}, \quad [i,j] \in \mathcal{E}, \tag{18}$$

we obtain convex quadratic inequalities from (14) and (15):

$$(x_i + x_j)^2 \leq s_{ij}, \quad [i,j] \in \mathcal{E}, \tag{19}$$

$$(x_i - x_j)^2 \leq z_{ij}, \quad [i,j] \in \mathcal{E}. \tag{20}$$

For those variables, we have the following bound:

$$s_{ij} + z_{ij} = X \bullet (u_{ij}u_{ij}^T + v_{ij}v_{ij}^T) = 2(X_{ii} + X_{jj}) = 4. \tag{21}$$

Furthermore, we have the following proposition:

Proposition 1.

$$L = -\sum_{[i,j]\in\mathcal{E}} L_{ij}v_{ij}v_{ij}^T. \tag{22}$$

Proof. Let us define

$$\delta_{ij} := e_i^T e_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

The (k, l) component of the negative of the right-hand side of (22) is:

$$\begin{aligned} e_k^T \left(\sum_{[i,j]\in\mathcal{E}} L_{ij}v_{ij}v_{ij}^T \right) e_l &= \sum_{[i,j]\in\mathcal{E}} L_{ij} e_k^T (e_i - e_j)(e_i - e_j)^T e_l \\ &= \sum_{[i,j]\in\mathcal{E}} L_{ij} \left(\delta_{ki}\delta_{il} + \delta_{kj}\delta_{jl} - \delta_{ki}\delta_{jl} - \delta_{kj}\delta_{il}\right) \\ &= \begin{cases} \sum_{[k,j]\in\mathcal{E}} L_{kj} & \text{if } k = l \\ -L_{kl} & \text{if } [k,l] \in \mathcal{E} \\ 0 & \text{otherwise,} \end{cases} \\ &= -L_{kl}, \end{aligned}$$

and thus the proposition follows. □

Given this proposition, the objective function of $\langle MC\rangle$ can be written as

$$L \bullet X = -\sum_{[i,j]\in\mathcal{E}} L_{ij}v_{ij}v_{ij}^T \bullet X = -\sum_{[i,j]\in\mathcal{E}} L_{ij}v_{ij}^T X v_{ij} = -\sum_{[i,j]\in\mathcal{E}} L_{ij}z_{ij}.$$

Thus, we can remove X from the objective function.

With X removed from the problem, we obtain the relaxation problem:

$$\langle SOCP2 - MC\rangle \begin{cases} \text{maximize} & -\sum_{[i,j]\in\mathcal{E}} L_{ij}z_{ij} \\ \text{subject to} & x_i^2 \le 1, \quad i = 1, \ldots, n \\ & (x_i + x_j)^2 \le s_{ij}, \quad [i,j] \in \mathcal{E}, \\ & (x_i - x_j)^2 \le z_{ij}, \quad [i,j] \in \mathcal{E}, \\ & s_{ij} + z_{ij} = 4, \quad [i,j] \in \mathcal{E}. \end{cases}$$

Because $\langle SOCP2-MC\rangle$ is a convex quadratic program, the conversion of this to SOCP is straightforward by using the technique described in Section 2.

Notice that the number of variables in $\langle SOCP2 - MC\rangle$ depends on the graph structure. Because the number is $O(|\mathcal{E}|)$, if the graph is sparse, then the size of $\langle SOCP2-MC\rangle$ is relatively small. On the other hand, if the graph is dense, there will be $O(n^2)$ variables and it will be difficult to solve $\langle SOCP2-MC\rangle$. In that case, we should consider eliminating several inequalities to fit our purpose.

3.3. **The triangle inequalities.** Let us consider $\langle MC\rangle$ as $\langle QP\rangle$. Then it is true that

$$X_{ij} + X_{jk} + X_{ik} \ge -1, \tag{23}$$
$$X_{ij} - X_{jk} - X_{ik} \ge -1, \tag{24}$$
$$-X_{ij} - X_{jk} + X_{ik} \ge -1, \tag{25}$$
$$-X_{ij} + X_{jk} - X_{ik} \ge -1, \tag{26}$$

as at least two of nodes i, j, k should be contained in the same set. These inequalities are called 'triangle inequalities' and play an important role in obtaining better bounds in the SDP relaxation and the lift-and-project method.

In Kim and Kojima's SOCP relaxation, it is difficult to utilize these inequalities, because their method does not use X. Our SOCP relaxation also does not use X. However, we can make use of these kinds of inequality, as our problem inherits the graph structure of the original problem.

Proposition 2. *if* $[i,j],[j,k],[k,l] \in \mathcal{E}$, *then*

$$z_{ij} + z_{jk} + z_{ik} \leq 8 \tag{27}$$
$$z_{ij} + s_{jk} + s_{ki} \leq 8 \tag{28}$$
$$s_{ij} + s_{jk} + z_{ki} \leq 8 \tag{29}$$
$$s_{ij} + z_{jk} + s_{ki} \leq 8. \tag{30}$$

Proof. Because the diagonal elements of X are always 1,

$$\begin{aligned} z_{ij} + z_{jk} + z_{ik} &= v_{ij}^T X v_{ij} + v_{jk}^T X v_{jk} + v_{ik}^T X v_{ik} \\ &= 2(X_{ii} + X_{jj} + X_{kk}) - 2(X_{ij} + X_{jk} + X_{ik}) \\ &= 6 - 2(X_{ij} + X_{jk} + X_{ik}). \end{aligned}$$

From (23), it follows that

$$z_{ij} + z_{jk} + z_{ik} \leq 8.$$

The rest of the proposition can be proved similarly, and thus we omit the proof. □

4. Numerical experiments

We have implemented a branch-and-bound method for $\langle MC \rangle$ using the following four relaxation problems:

1. SDP: the SDP relaxation.
2. SOCP1: the SOCP relaxation proposed by Kim and Kojima.
3. SOCP2: the SOCP relaxation by $\langle SOCP2 - MC \rangle$.
4. SOCP3: SOCP2 with the triangle inequalities (27).

We adopted a depth-first search in the branch-and-bound method.

SDPA ([5]) was used to solve SDP. SOCP was solved by our own implementation of the primal-dual path-following algorithm. In both solvers, the HKM direction ([10, 12, 13]) was used and the Mehrotra-type predictor-corrector method was adopted. All computations were performed on an Intel Pentium-based computer (CPU: Intel Celeron 733 MHz, Memory: 512 MB, OS: VINE Linux 2.1, C and C++ compilers: egcs-2.91.66).

Our SOCP solver, which we implemented from scratch to exploit sparse data structures, is far inferior to existing commercial products. For example, the MOSEK solver ([1]) solves some problems in the DIMACS challenge ([18]) more than 50 times faster than ours. In addition, many modern solvers can deal with simple relations such as (21) very efficiently by elimination, but we did not implement such a technique.

However, we still took advantage of using our own code. In the branch-and-bound method, problems at one level of the branching tree will have the same problem structure. In the interior-point method, we should therefore solve the linear system having the same non-zero pattern to calculate the search direction. We reused our sparse data areas for such problems to save CPU time for symbolic Choleskey factorizations.

We generated the following two types of MAX-CUT problems by using `rudy`, a graph generator written by Giovanni Rinaldi (See [9]).

1. G_{wr}: a general random graph. $1 \leq w_{ij} \leq 50$.
2. G_{p2}: a union of two planar random graphs having the same set of vertices. The weight was always 1.

Each figure in the tables is an average of 10 trials, if not otherwise specified.

The edge density of a general graph is defined by $2|\mathcal{E}|/|V|(|V|-1)$, while the density of a planar graph (p-density) is defined by $|\mathcal{E}|/3(|V|-2)$. For G_{p2}, which is not a planar graph in general, we use the term 'p2-density' for the p-density of the original planar graphs. Notice that the number of edges of G_{p2} is between d and $2d$, where d is the number of edges in the original planar graphs.

4.1. **Comparison in quality of relaxed problems.** We check the quality of the solutions of the relaxed problems. The relative error, denoted by ϵ in the tables, is defined by

$$\epsilon := \frac{\theta_{\rm ubd} - \theta_{\rm opt}}{\theta_{\rm opt}},$$

where $\theta_{\rm ubd}$ and $\theta_{\rm opt}$ are the optimal values of the relaxed and original problems, respectively. The *tri* column shows the average number of triangle inequalities added to SOCP3.

TABLE 1. Relative error of relaxed problems: G_{wr} (density 10%)

	SDP		SOCP1		SOCP2		SOCP3		
$\|V\|$	time	ε	time	ε	time	ε	time	ε	tri.
30	0.075	0.019	0.038	0.450	0.038	0.061	0.043	0.042	1.6
40	0.154	0.029	0.104	0.445	0.158	0.114	0.262	0.072	7.1
50	0.286	0.032	0.237	0.420	1.097	0.155	2.110	0.088	16.0
60	0.454	0.037	0.454	0.390	4.171	0.182	7.829	0.111	26.7

TABLE 2. Relative error of relaxed problems: G_{p2} (p2-density 30%)

	SDP		SOCP1		SOCP2		SOCP3		
$\|V\|$	time	ε	time	ε	time	ε	time	ε	tri.
40	0.204	0.035	0.076	0.602	0.058	0.119	0.088	0.048	5.8
50	0.399	0.029	0.153	0.755	0.123	0.141	0.196	0.045	11.3
60	0.653	0.028	0.304	0.769	0.248	0.125	0.394	0.044	12.2
70	1.073	0.036	0.473	0.795	0.465	0.139	0.773	0.061	11.6
80	1.550	0.040	0.713	0.902	0.755	0.153	1.220	0.066	17.1

Table 1 shows that SOCP3 gives better bounds than SOCP2, as is theoretically assured. SOCP2 also gives much better bounds than SOCP1. On the other hand, SOCP1 consumes the least CPU time, while SOCP3 consumes the most. In this table, SDP always gives the best bounds, while consuming as much CPU time as SOCP1.

In G_{p2}, the ratio of the errors of SOCP1 to SOCP2 is larger than that in G_{wr}. This implies that our relaxation will be more efficient for nearly planar graphs. Furthermore, the errors of SOCP3 are less than half those of SOCP2. It seems that, because G_{p2} is close to a planar graph, we can choose more of the triangle inequalities that effectively bound the feasible region. In G_{p2}, SDP gives slightly better bounds than SOCP3, consuming slightly more CPU time.

4.2. **Results of branch-and-bound methods.** Here we show the results of the branch-and-bound method. Tables 3, 4, and 5 show the results. The *time* column shows the CPU time consumed and the *node* column shows the number of relaxed problems solved in the branch-and-bound method. In the cell marked *, only seven of the ten test problems could be solved in the predefined time.

TABLE 3. Branch-and-Bound: G_{wr} (density 10%)

	SDP		SOCP1		SOCP2		SOCP3	
$\|V\|$	nodes	time	nodes	time	nodes	time	nodes	time
30	37.90	1.50	2508.20	15.56	61.60	0.71	58.00	0.70
40	95.80	7.98	78895.00	747.13	362.20	10.55	314.80	10.31
50	286.40	35.24	* 2524384.80	37631.23	2714.40	252.68	1058.60	146.64
60	956.20	233.03	NA	NA	24999.00	7184.17	9756.20	4816.57

We see immediately from Table 3 that SOCP1 does not work efficiently in branch-and-bound methods for solving MAX-CUT problems. The number of nodes solved by SOCP1 is by far larger than those by

TABLE 4. Branch-and-Bound: G_{wr} (density 2%)

	SDP		SOCP2		SOCP3	
$\|V\|$	nodes	time	nodes	time	nodes	time
100	402.00	443.93	1600.20	58.09	1461.60	50.98
110	732.80	1396.10	2190.80	190.37	2101.40	185.39
120	802.30	1788.62	7393.10	669.27	7677.40	657.09

TABLE 5. Branch-and-Bound: G_{p2} (p2-density 30%)

	SDP		SOCP2		SOCP3	
$\|V\|$	nodes	time	nodes	time	nodes	time
40	32.20	4.32	66.00	0.81	43.50	0.62
50	37.80	9.92	127.00	2.61	53.20	1.57
60	98.70	38.64	326.60	7.70	409.60	6.50
70	336.20	188.89	860.00	26.79	855.20	23.28
80	367.80	346.87	2279.20	68.46	961.40	37.60
90	316.00	489.14	1716.80	103.17	926.20	60.91
100	1950.60	3601.76	8823.00	419.70	5956.80	274.33
130	NA	NA	NA	NA	29165.80	2317.70
150	NA	NA	NA	NA	61814.00	7820.23

the other methods, and SOCP1 consumes the most CPU time. The difference in performance between SOCP1 and the other methods is so large that we could not use SOCP1 in the following tables.

In Table 3, SOCP2 and SOCP3 consume approximately the same CPU time when the graph is small, but for larger graphs, SOCP3 consumes less time. SDP is far superior to the other methods in this case.

Comparing Tables 3 and 4, we notice that the edge density significantly affects the performance of SOCP2 and SOCP3; they perform better if the edge density is small. This is not surprising, because in SOCP2 and SOCP3, the problem size is proportional to the number of edges. On the other hand, it seems that SDP cannot deal with sparse graphs efficiently. As a result, both SOCP2 and SOCP3 outperform SDP in Table 4.

In Table 5, SOCP2 and SOCP3 also outperform SDP, and the performance gap becomes large compared to Table 3. SOCP3 is far superior to the others in terms of CPU time consumed. It seems that the use of the triangle inequalities is effective for nearly planar graphs, as we observed in the previous subsection.

In view of Table 2, SDP gives a better bound than SOCP3, while consuming a comparable amount of CPU time. Furthermore, Table 5 shows that SOCP3 uses more nodes than SDP. Nevertheless, the total time consumed by the branch-and-bound method using SOCP3 is much less than that using SDP. One reason for this may be as follows. In our branch-and-bound method, we choose the value-fixing node from nodes having the maximum number of edges. This implies that, as the branch-and-bound method goes down the branching tree, the child problems become more and more sparse. For our SOCP relaxation, sparser data means a smaller problem, which can be solved in shorter time. As a result, the branch-and-bound method speeds up as it goes down the tree. On the other hand, for SDP, it is difficult to make use of the sparsity of X efficiently.

5. Concluding Remarks

In this paper, we proposed a new relaxation scheme for MAX-CUT problems using SOCP. Numerical experiments show that our method is superior to Kim and Kojima's SOCP relaxation applied to MAX-CUT problems. Compared to the SDP relaxation, our method gives a better performance when solving MAX-CUT problems for sparse or structured graphs.

If we could incorporate the triangle inequality into SDP relaxation, we would obtain tighter bounds. However, in our trial, SDPA will not work with triangle inequalities, because the number of linear constraints is too large. It seems that the number of linear inequality constraints heavily affects the CPU time required by SDPA.

Proving a theoretical bound of our SOCP relaxation and investigating a connection to other relaxations will be the subjects of further research. In addition, checking the efficiency of our SOCP relaxation by extensive numerical experiments using more sophisticated SOCP solvers is another important issue.

Acknowledgments

The authors thank Dr. K. Fujisawa of Kyoto University for kindly teaching us how to use SDPA and sometimes rewriting his code in response to our requests. The second author thanks Dr. Y. Ishizuka of Sophia University for his supervision and helpful comments.

This research is supported in part by the Ministry of Education, Science, Sports, and Culture of Japan.

References

[1] Andersen, E. D., C. Roos, T. Terlaky (2000). On implementing a primal-dual interior-point method for conic quadratic optimization. Helsinki School of Economics and Buisiness Administration, Helsinki, Finland.

[2] Balas, E., S. Ceria, G. Cornuéjols (1993). A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming* 58, 295–323.

[3] Faybusovich, L. (1995). Jordan algebras, symmetric cones and interior point methods. Technical report, Department of Mathematics, University of Notre Dame, Notre Dame, IN, USA.

[4] Fujie, T., M. Kojima (1997). Semidefinite relaxation for nonconvex programs. *Journal of Global Optimization* 10, 367–380.

[5] Fujisawa, K., M. Kojima, K. Nakata (1998). SDPA (Semidefinite Programming Algorithm) User's Manual, B-308, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology.

[6] Goemans, M. X., D. P. Williamson (1995). Improved approximation algorithms for maximum cut and satisfiability programs using semidefinite programming. *Journal of Assoc. Comut. Math.* 42, 1115–1145.

[7] Grötschel, M., L. Lovász, A. Schrijver (1988). Geometric algorithms and combinatorial optimization. *Springer,* New York, NY, USA.

[8] Helmberg, C., F. Rendl (1998). Solving quadratic $(0, 1)$-problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82, 291–315.

[9] Helmberg, C., F. Rendl (2000). A spectral bundle method for semidefinite programming. *SIAM J. Optim.*, 10, 673–696.

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

[11] Kim, S., M. Kojima (2000) Second Order Cone Programming Relaxation of Nonconvex Quadratic Optimization Problems. Technical Report, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology.

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

[13] Monteiro, R.D.C. (1995). Primal-dual path-following algorithms for semidefinite programming. *SIAM J. Optim.* 7:663–678.

[14] Monteiro, R.D.C., and T. Tsuchiya (1999). Polynomial convergence of primal-dual algorithms for the second-order cone program based on the MZ-family of directions. Technical report, School of Industrial and Systems Engineering, Goergia Institute of Technology.

[15] Muramatsu, M. (2000). On a commutative class of search directions for linear programming over symmetric cones. Technical Report, CS-00-02, Department of Computer Science, the University of Electro-Communications, Tokyo, Japan. (to appear in *Journal of Optimization Theory and its Application.*)

[16] Nesterov, Yu.E. (1998). Semidefinite relaxation and nonconvex quadratic optimization. *Optimization Methods and Software 9,* 141–160.

[17] Nesterov, Y.E., M.J. Todd (1995). Primal-dual interior-point methods for self-scaled cones, Technical Report 1125, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York, 14853–3801.

[18] Pataki, G., S. Schmieta (1999). The DIMACS library of semidefinite-quadratic-linear programs. Technical Report, Computational Optimization Research Center, Columbia University, NY, USA.

[19] Poljak, S., F. Rendl, H. Wolkowicz (1995). A recipe for semidefinite relaxation for $(0, 1)$-quadratic programming. *Journal of Global Optimization* 7, 51–73.

[20] Sherali, H.D., C.H. Tuncbilek (1992). A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique. *Journal of Global Optimization* 2, 101–112.

[21] Schmieta, S.H., F. Alizadeh (1998). Associative algebras, symmetric cones and polynomial time interior point algorithms. *Rutcor Research Report*, Rutgers University, NJ, USA.

[22] Sturm, J.F. (1999) Using SEDUMI 1.0x, A MATLAB toolbox for optimization over symmetric cones. Technical report, Dept. of Quantitative Economics, Maastricht University, Maastricht, The Netherlands.

[23] Tsuchiya, T. (1997) A polynomial primal-dual path-following algorithm for second-order cone programming. *Research Memorandum No. 649*, The Institute of Statistical Mathematics, Tokyo, Japan.

[24] Tsuchiya, T. (1999) A convergence analysis of the scaling-invariant primal-dual path-following algorithms for second-order cone programming. *Optimization Methods and Software* 11 and 12, 141–182.

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