Journal of the Operations Research Society of Japan
2003, Vol. 46, No. 2, 164-177

A NEW SECOND-ORDER CONE PROGRAMMING RELAXATION FOR MAX-CUT PROBLEMS

Masakazu Muramatsu
The University of Electro-Communications

Tsunehiro Suzuki
Sophia University

(Received Received: May 16, 2002; Revised: January 17, 2003)

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 gives better bounds than those based on the spectral decomposition proposed by Kim and Kojima [16], and that the efficiency of the branch-and-bound method using our relaxation is comparable to that using semidefinite relaxation in some cases.

Keywords: Combinatorial optimization, second-order cone programming, MAX-CUT, relaxation, branch-and-bound method

1. Introduction

In branch-and-bound methods for solving integer programming problems or nonconvex quadratic problems, the choice of relaxation problem significantly affects overall performance of the algorithm. It is now popular to use semidefinite programming (SDP) ([8, 10, 11, 21, 24, 30]) 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., [13]), 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, 25]) 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 ([7]), which also includes SDP and LP as special cases. Recently, primal-dual interior-point algorithms were developed for both SOCP ([19, 28, 29]) and symmetric cone programming ([20, 22, 26]). Several softwares have been implemented to solve SOCP (e.g., [1]) and symmetric cone programming (e.g., [27]). 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 [16] 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 [16] that their problems spent much less CPU time than SDP in practice.

Their way of constructing SOCP is based on the spectral decomposition of indefinite matrices. 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 [12] for example) of MAX-CUT problems.

The aim of this paper is to provide an efficient SOCP relaxation suitable for graph-based combinatorial optimization problems. We propose a new relaxation problem for the MAX-CUT problem using SOCP. Our relaxation uses the same framework as [16], but our strategy for choosing valid 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 problem always gives a better bound than that 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 overall 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.

There are several papers concerning exact solution of MAX-CUT problems for random graphs. Barahona, Jünger and Reinelt [3] solved sparse random MAX-CUT problems of up to 100 nodes using linear relaxation and the branch-and-cut method. For dense graphs which is more difficult to solve, Helmberg and Rendl ([12]) reported that it takes several days for their workstation to compute an optimal solution of dense 100-nodes problem using the branch-and-bound method with the SDP relaxation. It was also reported in [12] that SDP relaxation has some troubles in solving sparse or nearly planar graphs. It will be seen in Section 4 that the SDP relaxation outperforms our SOCP relaxation for dense graphs, while for sparse graphs our SOCP relaxation has better performance than the SDP relaxation.

After the first version of this paper was released, Barahona and Ladányi reported in [4] that their branch-and-cut algorithm solved several 100-nodes problems having edge density 30 % (see Section 4 for the definition of edge density) exactly. This may be one of the best results to date for intermediate (i.e., neither sparse nor dense) 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:

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

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

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

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:

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

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:

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

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 [16] 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

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

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

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

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:

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

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 [16] proposed a technique to reduce the number of variables, and demonstrated that with this technique, the SOCP relaxation could have overall performance as good as that of LP relaxation and SDP relaxation. We next explain 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, summing up (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:

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

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

The relaxation problem using this parameter-reducing technique is called the Kim and Kojima's SOCP relaxation in this paper.

A substantial advantage of the Kim and Kojima's SOCP relaxation 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 on 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 Relaxations

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

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

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 $x^TLx/4$. Therefore, we can write the MAX-CUT problem as

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

Because

$$x_j = 1 \text{ or } -1 \Leftrightarrow x_j^2 = 1 \Leftrightarrow x_j^2 \le 1 \text{ and } x_j^2 \ge 1 \Leftrightarrow x^Te_je_j^Tx \le 1 \text{ and } x^T(-e_je_j^T)x \le -1,$$

and

$$\text{maximize } x^TLx/4 \Leftrightarrow \text{ minimize } \theta \text{ subject to } -x^TLx/4 \le \theta,$$

$\langle MC\rangle$ belongs to the nonconvex quadratic problems introduced in Section 2.

Now we introduce the Kim and Kojima's SOCP relaxation of $\langle MC\rangle$. First, we convert $\langle MC\rangle$ into the following nonconvex quadratic problem:

$$\begin{cases} \text{minimize} & \theta \\ \text{subject to} & -x^TLx/4 - \theta \le 0 \\ & x^Te_je_j^Tx \le 1,\ j = 1, \ldots, n, \\ & x^T(-e_je_j^T)x \le -1,\ j = 1, \ldots, n. \end{cases}$$

Let

$$L = \sum_{j=1}^{n} \lambda_j q_j q_j^T$$

be the eigenvalue decomposition with

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

and put $L^+ = \sum_{j=1}^{l} \lambda_j q_j q_j^T$. As in Section 2, Kim and Kojima's SOCP relaxation of $\langle MC \rangle$ is as follows:

$$\langle SOCP1 - MC \rangle \left\{ \begin{array}{ll} \text{minimize} & \theta \\ \text{subject to} & -x^T L^+ x/4 + \sum_{j=l+1}^{n} \lambda_j z_j - \theta \leq 0 \\ & x^T q_j q_j^T x - z_j \leq 0,\ j = l+1, \ldots, n, \\ & x^T e_j e_j^T x \leq 1,\ j = 1, \ldots, n, \\ & z_j \leq \sqrt{n},\ j = l+1, \ldots, n. \end{array} \right.$$

Here, the bound for $z_j = q_j^T X q_j$ comes from the fact that X_{ij} is either $+1$ or -1, and $\|q_j\| = 1$.

3.2. An SOCP relaxation for MAX-CUT problem

We 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:

$$e_i e_i^T, \quad i = 1, \ldots, n, \tag{10}$$
$$u_{ij} u_{ij}^T, \quad [i,j] \in \mathcal{E}, \tag{11}$$
$$v_{ij} v_{ij}^T, \quad [i,j] \in \mathcal{E}. \tag{12}$$

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:

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

where the last equality is due to the definition of L. This proves the proposition. □

For $v \in \mathbb{R}^n$ and $X \in \mathcal{S}(n)$, it holds that $vv^T \bullet X = \sum_{i,j} v_i v_j X_{ij} = v^T X v$. Using this relation, we can rewrite the objective function of $\langle MC \rangle$ as

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

(Notice that v_{ij} is a vector.)

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

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

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

$$\begin{aligned} X_{ij}+X_{jk}+X_{ik} &\geq -1, &(23)\\ X_{ij}-X_{jk}-X_{ik} &\geq -1, &(24)\\ -X_{ij}-X_{jk}+X_{ik} &\geq -1, &(25)\\ -X_{ij}+X_{jk}-X_{ik} &\geq -1, &(26)\end{aligned}$$

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*

$$\begin{aligned} z_{ij}+z_{jk}+z_{ik} &\leq 8 &(27)\\ z_{ij}+s_{jk}+s_{ki} &\leq 8 &(28)\\ s_{ij}+s_{jk}+z_{ki} &\leq 8 &(29)\\ s_{ij}+z_{jk}+s_{ki} &\leq 8. &(30)\end{aligned}$$

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

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

From (23), it follows that

$$z_{ij}+z_{jk}+z_{ik}\leq 8. \quad (31)$$

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

3.4. A relationship to linear relaxation

Suppose that we put $x_i=0$ for all i. Then we can eliminate the variables x and s from $\langle SOCP2-MC\rangle$ to have

$$\begin{cases} \text{maximize} & -\sum_{[i,j]\in\mathcal{E}} L_{ij}z_{ij}\\ \text{subject to} & 0\leq z_{ij}\leq 4, \quad [i,j]\in\mathcal{E},\end{cases} \quad (32)$$

which is the trivial linear relaxation of $\langle MC\rangle$ (see [2, 3, 25]). In other words, if we do not fix any node, then x can be zero thus the feasible region of $\langle SOCP2-MC\rangle$ includes that of (32). This means that the bound given by $\langle SOCP2-MC\rangle$ is no better than that of (32). However, once we fix several nodes to $+1$ or -1 in the branch-and-bound method, $\langle SOCP2-MC\rangle$ is no longer equivalent with the linear relaxation. Notice also that we can fix at least one node to $+1$ without loss of generality, thus our relaxation is different from the linear relaxation even at the first step of the branch-and-bound method.

4. Numerical Experiments

In this section we numerically compare 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).

SDPA 5.01 ([9]), an implementation of primal-dual interior-point method, was used to solve SDP. SOCP was solved by our own implementation of the primal-dual interior-point method. In both solvers, the HKM direction ([14, 17, 18]) 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 preliminary and has a lot of room for improvement. In speed, it is not yet competitive with some commercial codes such as the MOSEK solver ([1]). There are two reasons why we use our own code.

One is that when we started this research, it was difficult to find a callable C library function for solving SOCP, which is indispensable for implementation of the branch-and-bound method where we have to solve many SOCP problems.

The other is that we could devise some techniques to improve the efficiency of the interior-point method using special structure of our SOCP relaxation. One of such techniques is as follows. Suppose that in the branch-and-bound method, we fix values of some nodes to $+1$ or -1. If the set of non-fixed nodes is $\tilde{V}$ and $\tilde{\mathcal{E}} = \{\,[i,j] \in \mathcal{E} \,|\, i \in \tilde{V} \text{ or } j \in \tilde{V}\,\}$, then the variables in $\langle SOCP2-MC\rangle$ are x_j, $j \in \tilde{V}$, s_{ij}, $[i,j] \in \tilde{\mathcal{E}}$, and z_{ij}, $[i,j] \in \tilde{\mathcal{E}}$. Observe that the locations of nonzero coefficients of those variables depend only on $\tilde{V}$ and $\tilde{\mathcal{E}}$. The nonzero pattern of the coefficient matrix of SOCP is identical regardless of the values $+1$ or -1 of the fixed nodes. As a result, when the problems have the same $\tilde{V}$ and $\tilde{\mathcal{E}}$, we solve the linear system having the same non-zero pattern to calculate the search direction of the interior-point method. We could reuse our sparse data areas between 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 [13]).

1. G_{wr}: a general random graph. $1 \le w_{ij} \le 50$.
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 stated.

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

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

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

where $\theta_{\rm ubd}$ and $\theta_{\rm opt}$ are the optimal values of the relaxed and original problems, respectively.

In Table 1, we see the relative errors of relaxed problems for G_{wr} with running time of the solvers. The *tri* column in SOCP3 shows the average number of triangle inequalities. We used all the possible triangle inequalities of the given graphs.

According to this table, SOCP3 gives better bounds than SOCP2, as is theoretically assured. Even SOCP2 gives much better bounds than SOCP1. On the other hand, SOCP1 used the least CPU time, while SOCP3 used the most. In this table, SDP always gives the best bounds, while consuming as much CPU time as SOCP1.

Table 2 shows that the ratio of the errors between SOCP1 and SOCP2 in G_{p2} 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

We have implemented a branch-and-bound method for $\langle MC \rangle$. In branching, we pick up a node and fix its value to $+1$ or -1. The node is chosen from the ones having the maximum number of edges. We use a depth-first search.

Tables 3, 4, and 5 are the results of the branch-and-bound method. The *time* column and *node* column show the CPU time spent and the number of relaxed problems solved in the branch-and-bound method, respectively. In the cell marked *, only seven of the ten test problems could be solved in the predefined time.

From Table 3 showing the results for G_{wr}, we see immediately that SOCP1 does not work efficiently in the branch-and-bound methods for solving MAX-CUT problems. Both the number of nodes and CPU time are very large in SOCP1; SOCP1 could not give effective upper bound of the optimal value. The difference in performance between SOCP1 and the other methods was so large that we did not use SOCP1 in the following experiments.

In Table 3, the results for G_{wr} of 10 % edge density, SOCP2 and SOCP3 spent approxi-

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

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

mately the same CPU time when the graph is small, but for larger graphs, SOCP3 uses 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. Both SOCP2 and SOCP3 outperform SDP in Table 4.

In Table 5 showing the results for G_{p2}, SOCP2 and SOCP3 also outperform SDP, and the performance gap becomes large compared to Table 4. SOCP3 is far superior to the others in terms of CPU time used. The use of the triangle inequalities seems very 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 of 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.

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

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, since it is hard for SDP relaxation to exploit sparsity, this kind of speed-up cannot be expected.

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 case, SDPA will not work with triangle inequalities. It seems that the number of linear inequality constraints heavily affects the CPU time required by SDPA.

There are several algorithms to solve the SDP relaxation of MAX-CUT problems other than the primal-dual interior-point methods. The dual-scaling method by Benson, Ye, and Zhang ([5]), the spectral-bundle method by Helmberg and Rendl ([13]), and nonlinear programming formulation by Burer and Monteiro ([6]) are such algorithms. Most of such algorithms are said to be more efficient for solving the SDP relaxation of MAX-CUT problems, mainly because by exploiting sparsity of the coefficient matrices. However, their interest seems to solve as large SDP problems as possible, and not to solve the MAX-CUT problem itself. The efficiency of their methods when used in the branch-and-bound method is unknown. Checking the efficiency of these algorithms in the branch-and-bound method and comparing them to the SOCP relaxation proposed in this paper is another topic of research.

Application of the proposed SOCP relaxation to other graph-based problems is obvious in some cases. For example, consider the MAX-DICUT problem, which is the same problem as the MAX-CUT except the two partitioned sets must have the same number of nodes. It is easy to see that this problem can be formulated as $\langle MC \rangle$ with an additional equality constraint $e^T x = 0$. Since SOCP can handle arbitrary linear equalities, it is straightforward to apply our SOCP relaxation to the MAX-DICUT problems.

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 would like to thank Dr. K. Fujisawa of Kyoto University for giving us a chance to use SDPA and sometimes rewriting his code in response to our requests. The authors would like to thank two anonymous referees for their valuable comments to improve the presentation of this paper. 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, Culture, Sports, Science and Technology of Japan.

References

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

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

[3] F. Barahona, M. Jünger and G. Reinelt: Experiments in quadratic 0-1 programming. *Mathematical Programming,* **44** (1989) 127–137.

[4] F. Barahona and L. Ladányi: Branch and cut based on the volume algorithm: Steiner trees in graphs and max-cut. *IBM report RC22221* (2001).

[5] S. J. Benson and Y. Ye: DSDP3: Dual scaling algorithm for general positive semidefinite programming. *Working Paper,* Dept. of Management Science, University of Iowa, Iowa City, Iowa, USA (2001).

[6] S. Burer and R. Monteiro: A projected gradient algorithm for solving the maxcut SDP relaxation. *Optimization Methods and Software,* **15** (2001) 175–200.

[7] L. Faybusovich: Euclidean Jordan algebras and interior-point algorithms. *Journal of Positivity,* **1** (1997) 331–357.

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

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

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

[11] M. Grötschel, L. Lovász and A. Schrijver: *Geometric Algorithms and Combinatorial Optimization* (Springer, New York, 1988).

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

[13] C. Helmberg and F. Rendl: A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization,* **10** (2001) 673–696.

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

[15] S. Homer and M. Peinado: Design and performance of parallel and distributed approximation algorithms for maxcut. *manuscript,* Dept. of Computer Science and Center for Computational Science, Boston University, Boston, MA, USA (1995).

[16] S. Kim and M. Kojima: Second order cone programming relaxation of nonconvex quadratic optimization problems. *Optimization Methods and Software,* **15** (2001) 201–224.

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

[18] R. D. C. Monteiro: Primal-dual path-following algorithms for semidefinite programming. *SIAM Journal on Optimization,* **7** (1995) 663–678.

[19] R. D. C. Monteiro and T. Tsuchiya: Polynomial convergence of primal-dual algorithms for the second-order cone program based on the MZ-family of directions. *Mathematical Programming,* **88** (1999) 61–83.

[20] M. Muramatsu: On a commutative class of search directions for linear programming over symmetric cones. *Journal of Optimization Theory and Application,* **112** (2002) 595–625.

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

[22] Y. E. Nesterov and M.J. Todd: Primal-dual interior-point methods for self-scaled cones. *SIAM Journal on Optimization,* **8** (1998) 324–364.

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

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

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

[26] S. H. Schmieta and F. Alizadeh: Associative algebras, symmetric cones and polynomial time interior point algorithms. *Mathematics of Operations Research,* **26** (2001) 543–564.

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

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

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

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

Masakazu Muramatsu
Department of Computer Science
The University of Electro-Communications
W4-305, 1-5-1 Chofugaoka
Chofu-shi, Tokyo, 182-8585, Japan
E-mail: muramatu@cs.uec.ac.jp