Bull. Korean Math. Soc. **42** (2005), No. 4, pp. 837–849

# A SUCCESSIVE QUADRATIC PROGRAMMING ALGORITHM FOR SDP RELAXATION OF THE BINARY QUADRATIC PROGRAMMING

XUEWEN MU, SANYANG LIU, AND YALING ZHANG

ABSTRACT. In this paper, we obtain a successive quadratic programming algorithm for solving the semidefinite programming (SDP) relaxation of the binary quadratic programming. Combining with a randomized method of Goemans and Williamson, it provides an efficient approximation for the binary quadratic programming. Furthermore, its convergence result is given. At last, We report some numerical examples to compare our method with the interior-point method on Maxcut problem.

## 1. Introduction

In this paper, we present a successive quadratic programming algorithm for solving the semidefinite programming (SDP) relaxation of the binary quadratic programming, which is a fundamental problem in optimization theory and practice. VLSI design, statistical physics, and combinatorial optimization are all sources of the binary quadratic programming ([1, 2, 3]), which can be naturally relaxed to SDP problem. The idea has been used by several authors. For example, Goemans and Williamson[4] developed a randomized algorithm for the maximum cut problem, based on solving its SDP relaxation, which provides an approximate solution guaranteed to be within a factor of 0.87856 of its optimal value whenever the associated edge weighs are nonnegative. Efficient algorithms for solving the SDP have been recently developed. One approach is with the use of interior-point methods ([5, 6]). Other nonlinear programming methods have also been proposed. For instance, Helmberg and Rendl[7] introduce the spectral bundle method which builds on the

Received April 9, 2004. Revised June 21, 2005.

2000 Mathematics Subject Classification: 90C22, 90C55.

Key words and phrases: binary quadratic programming, successive quadratic programming algorithm, semidefinite programming, randomized method.

framework of the proximal method of Kiwiel; The approach by Homer and Peinado[8] for using the change of variables $X = VV^T, V \subset \Re^{n\times n}$, where $X$ is the primal matrix variable of the maxcut SDP relaxation, is to transform the maxcut SDP relaxation into a constrained nonlinear programming problem in the new variable $V$. More recently, Samuel Burer and Renato D. C. Monteiro[9] propose a variant of Homer and Peinado's method based on the constrained nonlinear programming reformulation of the maxcut SDP relaxation obtained by using the change of variable $X = LL^T$, where $L$ is a lower triangular matrix. In this paper, we consider the binary quadratic programming and its corresponding reformulation of the SDP relaxation directly. A successive quadratic programming algorithm for solving SDP relaxation of the binary quadratic programming is provided by using the SDP relaxation and the change of variables $X = VV^T, V \subset \Re^{n\times n}$. Furthermore, its convergence result is given. The step-size in our algorithm is obtained by solving $n$ easy quadratic equations without using the linear search technique. The computational experience with our method indicates that it is substantially faster than the interior-point method.

The paper is organized as follows. In Section 2, we present the binary quadratic programming problem and its relaxations. In Section 3, the successive quadratic programming algorithm of the relaxation problem is obtained and its convergence result is given. Some numerical examples are offered in the last section.

### 1.1. Notation and terminology

In this paper, $\Re, \Re^n$, and $\Re^{n\times n}$ denote the space of real numbers, real $n$-dimensional column vectors, and real $n \times n$ matrices, respectively. By $S^n$ we denote the space of real $n \times n$ symmetric matrices, and we definite $S^n_+$ and $S^n_{++}$ to be the subsets of $S^n$ consisting of the positive semidefinite and positive definite matrices respectively. We write $A \succeq 0$ and $A \succ 0$ to indicate that $A \in S^n_+$ and $A \in S^n_{++}$, respectively. We let $\mathrm{tr}(A)$ denote the trace of a matrices $A \in \Re^{n\times n}$, we defined $A \bullet B = \langle A, B\rangle = \mathrm{tr}(A^T B)$, and the Frobenius norm of $A \in \Re^{n\times n}$ is defined to be $\| A \|_F = (A \bullet A)^{1/2}$.

We adopt the convention of denoting matrices by capital letters and matrices entries by lowercase letters with double subscripts. For example, a matrix $A \in \Re^{n\times n}$ has entries $a_{ij}$ for $i, j = 1, \ldots, n$, In addition, we denote the rows of a matrix by lowercase letters with single subscripts. For example, $A \in \Re^{n\times n}$ has rows $a_i$ for $i = 1, \ldots, n$. In this paper, we

will often find it necessary to compute the dot product of two row vectors $a_i$ and $b_j$ which arise as rows of the matrices $A$ and $B$. Instead of denoting this dot product as $a_i b_j^T$, we will denote it as $\langle a_i, b_j \rangle$.

## 2. Binary quadratic programming and its relaxations

In this section, we consider the following binary quadratic programming and describe some of its relaxations.

$$(1) \qquad \begin{array}{ll} \max & x^T C x \\ \text{s.t.} & x_i^2 = 1 \ \text{ for } i = 1, \ldots, n \end{array}$$

Without loss of generality, we assume that $C$ is a positive definite matrix, because of the equivalence between $\mathrm{Max}_{x \in \{1,-1\}^n} x^T C x$ and

$$\mathrm{Max}_{x \in \{1,-1\}^n} x^T C x + \sum_{i=1}^{n} y_i (x_i^2 - 1)$$

for all $y \in \Re$. In mathematical term, it means that $C \succ 0$.

Let $X = xx^T$, the above problem equals the following problem,

$$(2) \qquad \begin{array}{ll} \max & C \bullet X \\ \text{s.t.} & (e_i e_i^T) \bullet X = 1 \ \text{ for } i = 1, \ldots, n, \\ & \mathrm{rank}(X) = 1 \\ & X \succeq 0 \end{array}$$

where $e_i$ is the unit vector whose $i$-th component is 1 and others are all 0.

Because rank one constraint is nonconvex, dropping rank one constraint yields a semidefinite programming relaxation of (1) as follows,

$$(3) \qquad \begin{array}{ll} \max & C \bullet X \\ \text{s.t.} & (e_i e_i^T) \bullet X = 1 \ \text{ for } i = 1, \ldots, n. \\ & X \succeq 0 \end{array}$$

Now, we consider the following problem,

$$(4) \qquad \begin{array}{ll} \max & C \bullet X \\ \text{s.t.} & (e_i e_i^T) \bullet X \leq 1 \ \text{ for } i = 1, \ldots, n. \\ & X \succeq 0 \end{array}$$

Obviously, any feasible solution to the problem (3) is a feasible solution to the problem (4), say, the problem (4) is a relaxation of the problem (3).

Conversely, we assumed that $X$ is an optimal solution to the problem (4) and let $Y$ be a matrix which is satisfied that $y_{ii} = 1$, for $i = 1, \ldots, n$

and $y_{ij} = x_{ij}, i \neq j$, for $i, j = 1, \ldots, n$. By using the $C \succeq 0$ and the fact that $x_{ii} \leq 1$ for $i = 1, \ldots, n$, we have $C \bullet X \leqslant C \bullet Y$. Along with that $Y$ is a feasible solution of the problem (3), the optimal values of the problem (3) and (4) coincide. This shows the problem (3) and (4) are equivalent.

We now present the nonlinear programming reformulation of the problem (4) which is the basis of our algorithm for finding an approximate solution of the binary quadratic programming. For every $X \in S_+^n$, there exists a matrix $V \in \Re^{n \times n}$ such that $X = VV^T$. Thus the problem can be stated as the following one,

$$\begin{array}{lll} & \max & C \bullet (VV^T) \\ (5) & \text{s.t.} & (e_i e_i^T) \bullet (VV^T) \leq 1 \ \text{ for } i = 1, \ldots, n. \\ & & V \in \Re^{n \times n} \end{array}$$

Notice that we have replaced the requirement $X \succeq 0$ with $X = VV^T, V \in \Re^{n \times n}$. So the objective function of the problem (5) is nonconvex, but the feasible set of the problem (5) is convex.

## 3. The successive quadratic programming algorithm to the relaxation problem

In this section, we develop and discuss the successive quadratic programming algorithm to solve the problem (5). Before giving the basic steps of the algorithm, we definite some functions as follows.

$$f : \Re^{n \times n} \mapsto \Re, f(V) = C \bullet (VV^T),$$

$$g_i : \Re^{n \times n} \mapsto \Re, g_i(V) = e_i e_i^T \bullet (VV^T) - 1, \ i = 1, \ldots, n.$$

Obviously,the gradient of function $f(V)$ at a point $V$ is $G = 2CV$,the gradient of function $g_i(V)$ at a point $V$ is $H_i = 2(e_i e_i^T)V$, $i = 1, \ldots, n$.

Given a matrix $V^k$ feasible for the problem (5), the feasible ascent direction $D^k$ of the function $f(V)$ at a point $V^k$ will be obtained by solving the following quadratic programming.

$$\begin{array}{lll} & \max & t - \frac{u}{2} D \bullet D \\ (6) & \text{s.t.} & -G^k \bullet D + t \leq 0 \\ & & (e_i e_i^T) \bullet (V^k (V^k)^T) - 1 + 2(e_i e_i^T) V^k \bullet D + t \leq 0 \end{array}$$

for $i = 1, \ldots, n$, where $u$ is a constant which is less than 1. $(t, D) \in \Re \times \Re^{n \times n}$.

PROPOSITION 3.1. *Given* $(t^k, D^k) \in \Re \times \Re^{n\times n}$ *is a optimal solution for problem* (6), *and if* $D^k \neq 0$, *then* $D^k$ *is the feasible ascent direction* $D^k$ *of the function* $f(V)$ *at a point* $V^k$.

*Proof.* Since $t = 0$, $D = 0$ is a feasible solution for problem (6), $(t^k, D^k)$ is a optimal solution for problem (6), and $D^k \neq 0$, so we have $t^k - \frac{u}{2} D^k \bullet D^k \geq 0$, that is to say $t^k \geq \frac{u}{2} D^k \bullet D^k > 0$. Furthermore, since $-G^k \bullet D^k + t^k \leq 0$, we have $G^k \bullet D^k \geq t^k > 0$. From the assumption $D^k \neq 0$, we obtain that $D^k$ is the ascent direction of function $f(V)$ at a point $V^k$. Next, we will prove $D^k$ is the feasible direction. Let $\delta_i > 0$, $i = 1, \ldots, n$, and we have

$$\begin{aligned} g_i(V^k + \delta_i D^k) &= (e_i e_i^T) \bullet (V^k + \delta_i D^k)(V^k + \delta_i D^k)^T - 1 \\ &= (e_i e_i^T) \bullet V^k (V^k)^T - 1 + 2\delta_i (e_i e_i^T) V^k \bullet D^k \\ &\quad + \delta_i^2 (e_i e_i^T) \bullet D^k (D^k)^T. \end{aligned}$$

If $(e_i e_i^T) V^k \bullet D^k \geq 0$, let $\delta_i < 1$, and satisfy $\delta_i^2 (e_i e_i^T) \bullet D^k (D^k)^T < t^k$, we obtain

$$g_i(V^k + \delta_i D^k) < (e_i e_i^T) \bullet (V^k (V^k)^T) - 1 + 2(e_i e_i^T) V^k \bullet D + t \leq 0.$$

If $(e_i e_i^T) V^k \bullet D^k < 0$, let $\delta_i$ satisfy

$$2\delta_i (e_i e_i^T) V^k \bullet D^k + \delta_i^2 (e_i e_i^T) \bullet D^k (D^k)^T < 0,$$

we obtain

$$0 < \delta_i \leq \frac{-2(e_i e_i^T) V^k \bullet D^k}{(e_i e_i^T) \bullet D^k (D^k)^T}.$$

We have $g_i(V^k + \delta_i D^k) \leq 0$.

Based on the analysis, we select $\delta = \min\{\delta_i, i = 1, \ldots, n\}$, which must satisfy

$$g_i(V^k + \delta D^k) \leq 0.$$

Furthermore, since the feasible region of problem (5) is convex, so $D^k$ is the feasible ascent direction $D^k$ of the function $f(V)$ at a point $V^k$.

It is difficult to solve problem (6) directly because of its large scale. Now, we consider the dual problem of problem(6)

$$
\begin{aligned}
\max \quad & \frac{1}{2u}\left\|-\lambda_1 G^k+\sum_{i=2}^{n+1}\lambda_i(2e_{i-1}e_{i-1}^T)V^k\right\|_F^2 \\
& +\sum_{i=2}^{n+1}\lambda_i((e_{i-1}e_{i-1}^T)\bullet V^k(V^k)^T-1) \\
\text{s.t.} \quad & \sum_{i=1}^{n+1}\lambda_i=1 \\
& \lambda_i\geq 0 \ \text{ for } i=1,\ldots,n+1.
\end{aligned} \tag{7}
$$

We can obtained the optimal solution of problem (6) by solving problem (7), which is easier than problem (6). Based on the reference [10], we have the proposition as follows.

PROPOSITION 3.2.

1) *Problem* (6) *has optimal solution* $(t^k, D^k)\in \Re\times\Re^{n\times n}$.
2) *Problem* (6) *has optimal solution* $(t^k, D^k)\in \Re\times\Re^{n\times n}$ *if and only if there is a matrix* $P^k$ *and the multipliers* $\lambda_i^k$, *for* $i=1,\ldots,n+1$, *which satisfy the following conditions.*
   a) $\sum_{i=1}^{n+1}\lambda_i^k=1$ *and* $\lambda_i^k\geq 0$ *for* $i=1,\ldots,n+1$.
   b) $\lambda_1^k(-G^k\bullet D^k+t^k)=0$.
   c) $\lambda_{i+1}^k((e_ie_i^T)\bullet V^k(V^k)^T-1+2(e_ie_i^T)V^k\bullet D^k+t^k)=0$, *for* $i=1,\ldots,n$.
   d) $P^k=-\lambda_1^kG^k+\sum_{i=2}^{n+1}\lambda_i^k(2e_{i-1}e_{i-1}^T)V^k$.
   e) $D^k=-\frac{1}{u}P^k$.
   f) $t^k=\frac{1}{u}\|P^k\|_F^2+\sum_{i=2}^{n+1}\lambda_i^k((e_{i-1}e_{i-1}^T)\bullet V^k(V^k)^T-1)$.
   g) $G^k\bullet D^k+t^k\leq 0$.
   h) $(e_ie_i^T)\bullet V^k(V^k)^T-1+2(e_ie_i^T)V^k\bullet D^k+t^k\leq 0$ *for* $i=1,\ldots,n$.
3) *the multiplier* $\lambda_i^k$ for $i=1,\ldots,n+1$ *satisfy the conditions a),...,h) if and only if they are the optimal solution of problem* (7).

*Proof.* See Reference [10].

By Proposition 3.2, we obtain the feasible ascent direction $D^k$ of the function $f(V)$ at a point $V^k$. Now, we will give a simple method to obtain an appropriate step-size.

From Proposition 3.1, we know that if $D^k\neq 0$, there is a step-size $\delta\geq 0$, which satisfy $g_i(V^k+\delta D^k)\leq 0$ for $i=1,\ldots,n+1$.

Given $\delta_i \geq 0$, by solving the equations $g_i(V^k + \delta D^k) = 0$ for $i = 1, \ldots, n+1$. we obtain

$$\delta_i = \frac{-(e_i e_i^T) V^k \bullet D^k}{(e_i e_i^T) \bullet D^k (D^k)^T} + \frac{\sqrt{((e_i e_i^T) V^k \bullet D^k)^2 - ((e_i e_i^T) \bullet V^k (V^k)^T - 1)((e_i e_i^T) \bullet D^k (D^k)^T)}}{(e_i e_i^T) \bullet D^k (D^k)^T}$$

We choose the step-size

$$\delta = \min \{\delta_i, i = 1, \ldots, n\} \tag{8}$$

PROPOSITION 3.3. *Given $G^k \neq 0$,the step-size from* (8) *is an appropriate step-size.*

*Proof.* Given scalar $h > 0$, We define $\varphi : \Re \mapsto \Re$ by $\varphi(h) = f(V^k + hD^k)$, and we have

$$\varphi(h) = f(V^k + hD^k) = f(V^k) + h\mathrm{tr}(G^k D^k) + h^2 C \bullet D^k (D^k)^T,$$

$$\varphi'(h) = \mathrm{tr}(G^k D^k) + 2hC \bullet G^k (G^k)^T.$$

Based on the Proposition 3.1 and $D^k \neq 0$, we have $\mathrm{tr}(G^k D^k) > 0$. Furthermore, since $C \succ 0$, so $C \bullet G^k (G^k)^T \geq 0$, Thus, when $h > 0, \varphi'(h) \geq 0$, then $\varphi(h)$ is a monotone increasing function. Directly following from (8), the step-size is an appropriate search step-size.

We are now ready to write the successive quadratic programming algorithm.

THE SUCCESSIVE QUADRATIC PROGRAMMING ALGORITHM. *Let $V^0$ be a feasible solution of the problem* (5), *which is satisfied that $(e_i e_i^T) \bullet (V^0 (V^0)^T) = 1$ for $i = 1, \ldots, n$. And a prespecified constant $\varepsilon > 0$. Let $u = 0.5/n$.*

(1) *Compute the gradient $G^k = 2CV^k$ for the function $f$ at the point $V^k$.*
(2) *Compute the feasible ascent direction $D^k$ by solving problem* (8). *By the proposition 3.2, we obtain $D^k$.*
(3) *Compute $\|D^k\|_F$.*
(4) *If $\|D^k\|_F < \varepsilon$, then stop; otherwise, go to 5.*
(5) *compute $\delta$ by formula* (8). *Let $V^{k+1} = V^k + \delta D^k$, $k = k + 1$. go to 1.*

*Now, we will prove the convergence of the above algorithm.*

PROPOSITION 3.4. *If $D^k = 0$, there are some multipliers $\mu_i^k$, which satisfy*

$$2CV^k = 2\sum_{i=1}^{n} \mu_i^k (2e_i e_i^T)V^k$$

$$\mu_i^k((e_i e_i^T) \bullet V^k(V^k)^T - 1) = 0, \text{ and } \mu_i^k \geq 0 \text{ for } i = 1, \ldots, n.$$

*That is to say the matrix $V^k$ is the $KKT$ point of the problem* (5).

*Proof.* When $D^k = 0$, in assertion with the conclusion $g$) of the Proposition 3.2, we have $t^k \leq G^k \bullet D^k = 0$, But by Proposition 3.1, we obtain $t^k \geq 0$, So $t^k = 0$. From the conclusions $e$), $d$) of the proposition 3.2, we have

$$-uD^k = P^k = -\lambda_1^k G^k + \sum_{i=2}^{n+1} \lambda_i^k (2e_{i-1}e_{i-1}^T)V^k = 0. \tag{9}$$

If $\lambda_1^k \neq 0$, Let $\mu_i^k = \frac{\lambda_{i+1}^k}{\lambda_1^k}$ for $i = 1, \ldots, n$. We have

$$2CV^k = 2\sum_{i=1}^{n} \mu_i^k (2e_i e_i^T)V^k.$$

Furthermore, by the conclusions $c$) of Proposition 3.2, we have

$$\mu_i^k((e_i e_i^T) \bullet V^k(V^k)^T - 1) = 0, \text{ and } \mu_i^k \geq 0 \text{ for } i = 1, \ldots, n.$$

If $\lambda_1^k = 0$, Following from (9), we have $\sum_{i=2}^{n+1} \lambda_i^k (2e_{i-1}e_{i-1}^T)V^k = 0$. Thus

$$\sum_{i=2}^{n+1} \lambda_i^k ((2e_{i-1}e_{i-1}^T)V^k \bullet V^k) = 0.$$

Based on the above formula and conclusion $f$) of Proposition 3.2, we have

$$\begin{aligned} t^k &= \frac{1}{u}\|P^k\|_F^2 + \sum_{i=2}^{n+1} \lambda_i^k ((2e_{i-1}e_{i-1}^T) \bullet V^k(V^k)^T - 1) \\ &= \sum_{i=2}^{n+1} \lambda_i^k ((2e_{i-1}e_{i-1}^T) \bullet V^k(V^k)^T - 1) \\ &= \sum_{i=2}^{n+1} \lambda_i^k (2e_{i-1}e_{i-1}^T) \bullet V^k(V^k)^T - \sum_{i=2}^{n+1} \lambda_i^k = -\sum_{i=2}^{n+1} \lambda_i^k = -\sum_{i=1}^{n+1} \lambda_i^k. \end{aligned}$$

Following from the conclusion $a$) of proposition 3.2, we have $t^k = -\sum_{i=1}^{n+1} \lambda_i^k = -1$, which contradicts $t_k = 0$. So $\lambda_1^k \neq 0$.

Based on the Proposition 3.4, the algorithm will terminated at the KKT point of problem (5) if the termination criterion is $\|D^k\|_F < \varepsilon$. Where, $\varepsilon$ is a constant which is enough small.

Not every KKT point of the problem (5) is a global solution. The following proposition give sufficient conditions for a KKT point of the problem to be a global solution.

PROPOSITION 3.5. *Assume that $V^k$ is a KKT point of problem* (5) *and Let*

$$\mu_i^k \geq 0 \text{ for } i = 1, \ldots, n;\ S^k = \sum_{i=1}^{n} \mu_i^k (e_i e_i^T) - C.$$

*If $S^k \succeq 0$, then $V^k (V^k)^T$ is a global solution to the problem* (4).

*Proof.* Since $V^k$ is a KKT point, together with Proposition 3.4, we have

$$G^k = 2CV^k = \sum_{i=1}^{n} \mu_i^k (e_i e_i^T) V^k,\ \mu_i^k \geq 0 \text{ for } i = 1, \ldots, n$$

Thus

$$\langle CV^k, V^k \rangle = \left\langle \sum_{i=1}^{n} \mu_i^k (e_i e_i^T) V^k, V^k \right\rangle = \sum_{i=1}^{n} \mu_i^k.$$

That is $\langle C, V^k (V^k)^T \rangle = \sum_{i=1}^{n} \mu_i^k$. If $S^k \succeq 0$, let $X$ be feasible for (4), then $\langle S^k, X \rangle \geq 0$, that is $\sum_{i=1}^{n} \mu_i^k \geq \langle C, X \rangle$. Therefore, $V^k (V^k)^T$ is a global solution of the problem (4).

## 4. Numerical results

In this section we present computational results by comparing our method with earlier method to find approximate solutions to the max-cut problem based on solving its SDP relaxation. As stated in the introduction, the purpose of the results presented here are to show that our successive quadratic programming algorithm is considerably faster than interior-point method.

### 4.1. Maxcut problem

The maxcut problem is one of the standard $NP$-complete problems defined on graphs[11]. Let $G = (V, E)$ denote an edge-weighted undirected graph without loops or multiple edges. We use $V = \{1, \ldots, n\}$, $ij$ for an edge with endpoints $i$ and $j$, and $a_{ij}$ for the weight of an edge $ij \in E$. For $S \subseteq V$ the cut $\delta(S)$ is the set of edges $ij \in E$ that have one endpoint in $S$ and the other endpoint in $V \setminus S$. The maxcut problem asks for the cut maximizing the sum of the weights of its edges. Here, we only work with the complete graph $K_n$. In order to model a graph in this setting, define $a_{ij} = 0$ for $ij \notin E$. $A = (a_{ij}) \in S^n$ is referred to as the weighted adjacency matrix of the graph. An algebraic formulation can be obtained by introducing cut vectors $x \in \{-1, 1\}^n$ with $x_i = 1$ for $i \in S$ and $x_i = -1$ for $i \in V \setminus S$.

The maxcut problem can be formulated as the integer quadratic program.

$$\begin{aligned} \max \quad & \tfrac{1}{2} \sum_{i<j} a_{ij}(1 - x_i x_j) \\ s.t. \quad & x_i \in \{-1, 1\}, \ i = 1, \ldots, n. \end{aligned}$$

The matrix $L(G) = \mathrm{Diag}(Ae) - A$ is called the Laplace matrix of the graph $G$, where $e$ is the unit vector whose every component is 1. And $\mathrm{Diag}(Ae)$ is the diagonal matrix whose diagonal elements are $Ae$. Let $C = \frac{1}{4}L$, the maxcut problem may be interpreted as a special case of the problem (1).

### 4.2. Numerical examples

We report the numerical example in this section. In the numerical example, we compare the computational results between our method and interior point method[5]. As stated before, the purpose of the results presented here is to show that our algorithm is substantially faster than interior point method. All the algorithms are run in the MATLAB 6.1 environment on a AMD AthlonXP1600+ personal computer with 128Mb of Ram. In all the test problems, we choose the initial iterate $L^0$ to be $n \times n$ identity matrix, $u$ to be $0.5/n$. In interior-point method, we solve the SDP relaxation by using SDPpack software[12].

We adopt the randomized cut generation scheme of Goemans and Williamson[4]. The iteration number of this algorithm is $n$ which is the scale of max-cut problem considered.

Here all the tested problems are random graphs with two different edge density 0.7 and 0.3. which denote the dense random graphs and sparse random graphs respectively. We select problems in size from $n =$

50 to $n = 300$ to compare the suboptimal value of maxcut problem,and the total time of the three methods. In our algorithm, the iteration stops once $\|D^k\|_F < \varepsilon$ is found. The result is shown in Table 1.

In Table 1, we use "SDP" presents for interior point algorithm based on semidefinite programming, "SQA" for our successive quadratic programming algorithm, "time" for the total time of two methods, "value-f" for the suboptimal value of the maxcut problem based these methods, "density" for edge density of the random graphs.

The Table 1 shows our method can generally reach solutions of the problems much faster than the interior-point method whether to the dense random graphs or the sparse random graphs.

Table.1 Comparison of the two method

| Size of graph | density | algo | value-f | time |
|---|---|---|---|---|
| 50 | 0.7 | SQA | 381.316300729441 | 0.1720 |
| | | SDP | 381.222818884658 | 1.2970 |
| 50 | 0.3 | SQA | 384.343853121280 | 0.4220 |
| | | SDP | 384.343853121280 | 3.2190 |
| 100 | 0.7 | SQA | 1385.94015100020 | 1.4530 |
| | | SDP | 1388.66226246641 | 13.7190 |
| 100 | 0.3 | SQA | 7179.35285955398 | 10.0000 |
| | | SDP | 7184.64381700424 | 149.2200 |
| 150 | 0.7 | SQA | 3027.47658904186 | 3.5470 |
| | | SDP | 3030.60112063104 | 75.6090 |
| 150 | 0.3 | SQA | 1527.69134585512 | 3.7500 |
| | | SDP | 1528.54917463768 | 82.5470 |
| 200 | 0.7 | SQA | 5323.36289231888 | 18.3750 |
| | | SDP | 5318.11148873339 | 184.5470 |
| 200 | 0.3 | SQA | 2676.47956117645 | 15.1250 |
| | | SDP | 2672.86837035170 | 200.8750 |
| 250 | 0.7 | SQA | 8156.16895709477 | 45.4850 |
| | | SDP | 8162.65641905066 | 440.6560 |
| 250 | 0.3 | SQA | 4129.85944356911 | 14.8750 |
| | | SDP | 4135.51356364934 | 480.8120 |
| 300 | 0.7 | SQA | 11686.0672820264 | 100.1720 |
| | | SDP | 11697.2745314466 | 2116.0620 |
| 300 | 0.3 | SQA | 5884.30709295320 | 80.6100 |
| | | SDP | 5889.15827541389 | 1892.8590 |

## 5. Conclusion

In this paper, we have proposed a successive quadratic programming algorithm for solving SDP relaxation of the binary quadratic programming. In the algorithm, we give a simple method for selecting the step-size. Using the randomized cut procedure of Goeman and Williamson, it can give a sub-optimal of max-cut problem. It is able to obtain a moderately accurate solution more quickly than interior point method. This paper has demostrated the single case of max-cut SDP relaxation, but we believe that the same results are apt to hold elsewhere.

## References

[1] F. Barahon and M. Grotschel, *An application of combinatorial optimization to statiscal optimization and circuit layout design,* Oper. Res. **36** (1998), no. 3, 493–513.

[2] K. C. Chang and D. H., *Eifficient algorithms for layer assignment problems,* IEEE Trans. on Computer Aided Design. CAD-**6** (1987), no. 1, 67–78.

[3] R. Chen, *A graph theoretic via minimization algorithm for teo layer printed circuit boards,* IEEE Trans. Circuits Systems CAS **30** (1983), no. 5, 284–299.

[4] M. X. Goeman and D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfiably problem using semidefinite programming,* J. ACM **42** (1995), 1115–1145.

[5] S. Benson, Y. Ye, and X. Zhang, *Solving large scale sprase semidefinite programming for combinatorial optimization,* SIAM J. Optim. **10** (2000), no. 2, 443–461.

[6] C. Helberg and F. Rendel, *An interior point method for semidefinite programming,* SIAM J. Optim. **10** (1990), 842–861.

[7] C. Helberg and F. Rendl, *A spectral bundle method for semidefinite programing,* SIAM J. Optim. **10** (2000), 673–696.

[8] M. Peinadoo and S. Homer, *Design and performance of parallel and distributed approximation algorithms for max-cut,* Journal of Parallel and Distributed Computing. **9** (1998), 141–160.

[9] S. Burer and R. D. C. Monteriro, *A projected gradient algorithm for solving the max-cut relaxation,* Optim. Methods Softw. **15** (2001), 175–200.

[10] M. Marko, Makeka, and Pekka Neittaanmaki, *Nonsmooth optimization: analysis and algorithms with application to optimization control,* World scientific, Singapore, 1992.

[11] C. Helmberg, *Semidefinite programming for combinatorial optimization,* Konrad-Zuse-Zentrum fur informationstechnik Berlin, Germany, 2000.

[12] M. V. Nayakakuppam, M. L. Overton, and S. Schemita, *SDPpack user's guide-version 0.9 Beta,* Technical Report Yr 1997-737, Courtant Institute of Mathematical Science, NYU, New York, NY, 1997.

Xuewen Mu, Sanyang Liu, Department of Applied Mathematics, Xidian University, Xi'an 710071, China
*E-mail*: xdmuxuewen@hotmail.com
liusanyang@263.net

Yaling Zhang, Department of Computer Science and Technology, Xi'an University of Science and Technology, Xi'an 710054, China
*E-mail*: zyldella@126.com