Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 963563, 7 pages
http://dx.doi.org/10.1155/2013/963563

Research Article

A Rank-Two Feasible Direction Algorithm for the Binary Quadratic Programming

Xuewen Mu[1] and Yaling Zhang[1,2]

[1] *School of Mathematics and Statistics, Xidian University, Xi'an 710071, China*
[2] *School of Computer Science, Xi'an Science and Technology University, Xi'an 710054, China*

Correspondence should be addressed to Xuewen Mu; xdmuxuewen@hotmail.com

Received 16 March 2013; Accepted 3 October 2013

Academic Editor: Luigi Muglia

Based on the semidefinite programming relaxation of the binary quadratic programming, a rank-two feasible direction algorithm is presented. The proposed algorithm restricts the rank of matrix variable to be two in the semidefinite programming relaxation and yields a quadratic objective function with simple quadratic constraints. A feasible direction algorithm is used to solve the nonlinear programming. The convergent analysis and time complexity of the method is given. Coupled with randomized algorithm, a suboptimal solution is obtained for the binary quadratic programming. At last, we report some numerical examples to compare our algorithm with randomized algorithm based on the interior point method and the feasible direction algorithm on max-cut problem. Simulation results have shown that our method is faster than the other two methods.

1. Introduction

In this paper, we consider the following binary quadratic programming:

$$\begin{aligned} &\min \quad x^T Q x + 2r^T x \\ &\text{s.t.} \quad x_i^2 = 1, \quad \text{for } i = 1, \ldots, m, \end{aligned} \tag{1}$$

where Q is real $m \times m$ symmetric matrices and r is a real m-dimensional column vector.

The binary quadratic programming is a fundamental problem in optimization theory and practice. Some combinatorial optimization problems and engineering problems can be modeled as binary quadratic programming, such as VLSI design, statistical physics, max-cut problem [1], the optimal multiuser detection [2–5], image processing [6], and the design of FIR filters with discrete coefficients [7]. These problems are known to be *NP*-hard [1]. One typical approach to solve these problems is to construct lower bounds for approximating the optimal value. Now, the semidefinite programming (SDP) relaxation approach had been studied and proven to be quite powerful for finding approximate optimal solutions. Based on solving its semidefinite programming relaxation, Goemans and Williamson [8] developed a randomized algorithm for the max-cut problem, which provides an approximate solution guaranteed to be within a factor of 0.87856 of its optimal value. Interior point method is a powerful method for SDP with small and moderate scale. But the interior point method is limited to problems of moderate size, which cannot solve SDP with large scale efficiently [1]. So Goemans and Williamson's method based on the interior point method is not adapted to solve the large scale max-cut problems.

Some efficient nonlinear programming algorithms only based on gradient for solving the SDP relaxation of the max-cut problem have been developed. Homer and Peinado [9] proposed a parallel and distributed approximation algorithms for max-cut problem. In the algorithm, the author transformed the max-cut SDP relaxation into a constrained nonlinear programming problem in the new variable V for using the change of variables $X = VV^T$, $V \subset R^{n \times n}$, where X is the primal matrix variable of the SDP relaxation. Burer and Monteiro [10] proposed a projected gradient algorithm for solving the max-cut SDP relaxation

by using the change of variable $X = LL^T$, where L is a lower triangular matrix. The rank-two relaxation heuristics algorithm in [11] relaxed the max-cut problem to form an unconstrained optimization problem by replacing each binary variable with one unit vector in space R^2 and then using polar coordinates. In [12], the rank-two SDP relaxation model is proposed for maximal independent set problem. Based on the low-rank decomposition of the semidefinite matrix, Liu et al. [13] proposed a feasible direction method to solve a nonlinear programming model of binary quadratic programming.

In the paper, we propose a rank-two feasible direction method for the binary quadratic programming. we restrict the rank of matrix variable to be two in the semidefinite programming relaxation and obtain a quadratic objective function with simple quadratic constraints. A feasible direction method is used to solve the nonlinear programming. We also give the analysis of the convergence and the complexity of the method. The randomized algorithm is used to obtain the suboptimal solution of the binary quadratic programming. At last, we compare our method with the randomized algorithm based on the interior point method and the feasible direction method on max-cut problem. Simulation results show that our method costs less CPU time than the two methods.

2. The SDP Relaxation Method for Binary Quadratic Programming

In this section, we introduce the SDP relaxation of binary quadratic programming problem [1].

In problem (1), let $n = m + 1$, $z = [x^T, x_n]^T (x_n = 1)$, and $C_1 = \begin{pmatrix} Q & r \\ r^T & 0 \end{pmatrix}$; then problem (1) can be formulated as

$$\begin{aligned} \min \quad & z^T C_1 z \\ \text{s.t.} \quad & z \in \{-1, 1\}^n. \end{aligned} \tag{2}$$

It is well known that problem (2) is also NP-hard [1].

Let $C_2 = C_1 - (\lambda_{\max}(C_1) + 1) I_n$, where $\lambda_{\max}(C_1)$ denotes the largest eigenvalue of the matrix C_1 and I_n denotes the unit matrix; then C_2 is a negative definite matrix. Problem (2) is equivalent to the following problem below:

$$\begin{aligned} \min \quad & z^T C_2 z + \left(\lambda_{\max}\left(C_1\right) + 1\right) n \\ \text{s.t.} \quad & z \in \{-1, 1\}^n. \end{aligned} \tag{3}$$

Letting $Z = zz^T$ and ignoring the constant term, then problem (3) is equivalent to the following problem:

$$\begin{aligned} \min \quad & C_2 \bullet Z \\ \text{s.t.} \quad & z_{ii} = 1, \quad i = 1, 2, \ldots, n \\ & \operatorname{rank}(Z) = 1 \\ & Z \succeq 0, \end{aligned} \tag{4}$$

where $C_2 \bullet Z = \operatorname{tr}(C_2^T Z)$ and z_{ii} is the diagonal elements of matrix Z. In addition, $Z \succeq 0$ denotes that matrix Z is semidefinite. Ignoring the nonconvex "rank one" constraint, the SDP relaxation is given as follows [1]:

$$\begin{aligned} \min \quad & C_2 \bullet Z \\ \text{s.t.} \quad & z_{ii} = 1, \quad i = 1, 2, \ldots, n \\ & Z \succeq 0. \end{aligned} \tag{5}$$

Interior point methods have been proved to be quite efficient for small and moderate scale SDP. In the 0.878 randomized method by Goemans and Williamson [8], the author solved the SDP relaxation problem (5) by interior point methods. However, interior point methods are second-order method, so they are quite time and memory intensive and not adapted to large scale binary quadratic problems. The complexity of the primal-dual interior point method based on AHO search direction for the SDP relaxation (5) of the max-cut is $O(n^{4.5} \ln(1/\epsilon))$ [14, 15].

3. The Rank-Two SDP Relaxation for Binary Quadratic Programming

In [11], the rank-two SDP relaxation model is proposed for max-cut problem based on the polar direction. In [12], the rank-two SDP relaxation model is proposed for maximal independent set problem. In this section, we present a rank-two SDP relaxation based on the rank-two approximate matrix for binary quadratic programming.

In SDP relaxation problem (5), let $C = -C_2$; we have

$$\begin{aligned} \max \quad & C \bullet Z \\ \text{s.t.} \quad & z_{ii} = 1, \quad i = 1, 2, \ldots, n \\ & Z \succeq 0. \end{aligned} \tag{6}$$

Obviously, matrix C is positive definite.

Let $Z = xx^T + yy^T$, $x, y \in R^n$ [12]; then $C \bullet Z = x^T C x + y^T C y$ and $Z_{ii} = x_i^2 + y_i^2 = 1$. We obtain the rank-two SDP relaxation of binary quadratic programming as follows:

$$\begin{aligned} \max \quad & x^T C x + y^T C y \\ \text{s.t.} \quad & x_i^2 + y_i^2 = 1, \quad i = 1, 2, \ldots, n; \end{aligned} \tag{7}$$

where x_i and y_i are the elements of vector x and y. Obviously, matrix Z satisfies that $\operatorname{rank}(Z) = 2$, $Z \succeq 0$, so problem (7) is a rank-two SDP relaxation problem.

Problem (7) is also a nonlinear programming with quadratic objective function and constraints. Compared to the n^2 variables of SDP relaxation, the rank-two relaxation has only $2n$ variables, so this approach possesses scalability for solving large-scale binary quadratic programming problems.

Let

$$f(x, y) = x^T C x + y^T C y, \qquad h_i(x, y) = x_i^2 + y_i^2 - 1, \quad i = 1, 2, \ldots, n, \tag{8}$$

then the gradients of the function $f(x, y)$ and $h_i(x, y)$ are

$$\nabla f(x, y) = (2Cx, 2Cy), \qquad \nabla h_i(x, y) = \left(2e_i e_i^T x, 2e_i e_i^T y\right), \quad i = 1, 2, \ldots, n. \tag{9}$$

The KKT condition for problem (7) is given here. If the variable $(x, y) \in R^{n\times 2}$ in problem (7) satisfies the following condition:

$$\begin{aligned} \nabla f(x, y) &= \sum_{i=1}^{n} \mu_i \nabla h_i(x, y), \\ x_i^2 + y_i^2 &= 1, \quad i = 1, 2, \ldots, n, \\ \mu_i &\geq 0, \quad \exists i \text{ for } \mu_i > 0, \end{aligned} \tag{10}$$

then (x, y) is a KKT point for problem (7).

It is simple to obtain that

$$\begin{aligned} \nabla f(x, y) = \sum_{i=1}^{n} \mu_i \nabla h_i(x, y) &\Longleftrightarrow ((Cx)_i, (Cy)_i) \\ &= \mu_i (x_i, y_i), \quad i = 1, 2, \ldots, n. \end{aligned} \tag{11}$$

Then we have the equivalent KKT condition for problem (7) as follows:

$$\begin{aligned} ((Cx)_i, (Cy)_i) &= \mu_i (x_i, y_i), \quad i = 1, 2, \ldots, n \\ x_i^2 + y_i^2 &= 1, \quad i = 1, 2, \ldots, n \\ \mu_i &\geq 0, \quad \exists i \text{ for } \mu_i > 0. \end{aligned} \tag{12}$$

4. The Rank-Two Feasible Direction Algorithm for Binary Quadratic Programming

Feasible direction algorithm is an efficient algorithm for some special nonlinear programming problems. In [13], the feasible direction algorithm is applied to solve the low-rank nonlinear programming relaxation for binary quadratic programming problems. In [16], the feasible direction algorithm is applied to solve a rank one nonlinear programming relaxation for max-cut problem.

In this section, we extend the feasible direction algorithm to solve problem (7). The algorithm employs only gradient evaluations of the objective function in problem (7), and no calculations on any matrices and no line searches, thus greatly reduces the calculation costs and increases the efficiency of the algorithm.

In the rank-two feasible direction algorithm, we give the following iteration for problem (7):

$$\left(x_i^{k+1}, y_i^{k+1}\right) = \frac{\left((2Cx)_i^k, (2Cy)_i^k\right)}{\left(\left\|(2Cx)_i^k\right\|^2 + \left\|(2Cy)_i^k\right\|^2\right)^{0.5}}, \quad i = 1, 2, \ldots, n, \quad k = 0, 1, \ldots, \tag{13}$$

where (x_i^{k+1}, y_i^{k+1}) denotes the element pair of matrix variable (x^{k+1}, y^{k+1}).

The iteration (13) is very simple and has the following characteristics.

(1) No matrix calculations and no line searches are required, and only one gradient evaluation is needed to get the new iteration.

(2) The new iteration point (x^{k+1}, y^{k+1}) is feasible to problem (7).

(3) If the sequence (x^{k+1}, y^{k+1}) converges to (x^*, y^*), then (x^*, y^*) is feasible to problem (7).

Define direction $((d_x)^k, (d_y)^k)$ as follows:

$$\left((d_x)_i^k, (d_y)_i^k\right) = \left(x_i^{k+1}, y_i^{k+1}\right) - \left(x_i^k, y_i^k\right), \quad i = 1, 2, \ldots, n, \tag{14}$$

as a search direction, where $((d_x)_i^k, (d_y)_i^k)$ is the elements pair of $((d_x)^k, (d_y)^k)$. Then the iteration (13) can be written as

$$\left(x^{k+1}, y^{k+1}\right) = \left(x^k, y^k\right) + \left((d_x)^k, (d_y)^k\right). \tag{15}$$

The following lemmas show that if $((d_x)^k, (d_y)^k) = 0$, then (x^k, y^k) is a KKT point of problem (7), and if $((d_x)^k, (d_y)^k) \neq 0$, then $((d_x)^k, (d_y)^k)$ is a feasible increasing direction for problem (7).

Lemma 1. *If* $((d_x)^k, (d_y)^k) = 0$, *then* (x^k, y^k) *is a KKT point of* (7).

Proof. It is clear that (x^k, y^k) satisfies the constraint in problem (7). Since $((d_x)^k, (d_y)^k) = 0$, then

$$\left((Cx)_i^k, (Cy)_i^k\right) = \left(\left\|(Cx)_i^k\right\|^2 + \left\|(Cy)_i^k\right\|^2\right)^{0.5} \left(x_i^k, y_i^k\right), \quad i = 1, 2, \ldots, n. \tag{16}$$

Let $\mu_i^k = (\|(Cx)_i^k\|^2 + \|(Cy)_i^k\|^2)^{0.5}$; by the KKT condition (12), we have that (x^k, y^k) is a KKT point of (7). This completes the proof of the lemma. □

Lemma 2. *Suppose that* $((d_x)^k, (d_y)^k) \neq 0$; *then* $((d_x)^k, (d_y)^k)$ *is a feasible increasing direction for problem* (7), *and iteration point* (x^{k+1}, y^{k+1}) *is feasible to problem* (7).

Proof. The feasibility of the iteration point (x^{k+1}, y^{k+1}) directly comes from definition (13). Using the fact that $(x_i^k)^2 + (y_i^k)^2 = 1$, we have

$$\begin{aligned}
&\nabla^T f(x,y)\left((d_x)^k,(d_y)^k\right)\\
&= \nabla^T f(x,y)\left(\left(x^{k+1},y^{k+1}\right)-\left(x^k,y^k\right)\right)\\
&= \sum_{i=1}^{n}\left\{\left((2Cx)_i^k,(2Cy)_i^k\right)^T\frac{\left((2Cx)_i^k,(2Cy)_i^k\right)}{\left(\left\|(2Cx)_i^k\right\|^2+\left\|(2Cx)_i^k\right\|^2\right)^{0.5}}\right.\\
&\qquad\left.-\left((2Cx)_i^k,(2Cy)_i^k\right)^T\left(x_i^k,y_i^k\right)\right\}\\
&= \sum_{i=1}^{n}\left\{\left(\left\|(2Cx)_i^k\right\|^2+\left\|(2Cy)_i^k\right\|^2\right)^{0.5}\right.\\
&\qquad\left.-\left((2Cx)_i^k,(2Cy)_i^k\right)^T\left(x_i^k,y_i^k\right)\right\}\\
&\geq 0.
\end{aligned}\tag{17}$$

So direction $((d_x)^k,(d_y)^k)$ is a feasible increasing direction for problem (7). □

The convergence of the feasible direction method is concluded by the following lemmas.

Lemma 3. *Suppose that* $((d_x)^k,(d_y)^k) \to 0$*; then any accumulation point* (x^*, y^*) *is a KKT point of* (7).

Proof. Let (x^*, y^*) be an accumulation point of the sequence $\{(x^k, y^k)\}$; it is simple to obtain the result by Lemma 1. □

Lemma 4 (see [1]). *Let matrixes A and B be positive definite; then* $A \bullet B$ *is bounded by*

$$\lambda_{\min}(A)\operatorname{tr}(B) \leq A \bullet B \leq \lambda_{\max}(A)\operatorname{tr}(B), \tag{18}$$

where $\lambda_{\min}(A)$ *and* $\lambda_{\max}(A)$ *denote the smallest and the largest eigenvalues of the matrix A.*

Lemma 5. *If* $((d_x)^k,(d_y)^k) \neq 0$ *for all* $k > 0$*, then* $((d_x)^k,(d_y)^k) \to 0$.

Proof. Since

$$\begin{aligned}
&f\left(x^{k+1},y^{k+1}\right)\\
&\quad= f\left(\left(x^k,y^k\right)+\left((d_x)^k,(d_y)^k\right)\right)\\
&\quad= f\left(x^k,y^k\right)+\nabla^T f\left(x^k,y^k\right)\left((d_x)^k,(d_y)^k\right)\\
&\qquad+C\bullet\left(\left((d_x)^k,(d_y)^k\right)\left((d_x)^k,(d_y)^k\right)^T\right),
\end{aligned}\tag{19}$$

from Lemma 4, we have

$$\begin{aligned}
&f\left(x^{k+1},y^{k+1}\right)-f\left(x^k,y^k\right)\\
&\quad\geq C\bullet\left(\left((d_x)^k,(d_y)^k\right)\left((d_x)^k,(d_y)^k\right)^T\right)\\
&\quad\geq \lambda_{\min}(C)\operatorname{tr}\left(\left((d_x)^k,(d_y)^k\right)\left((d_x)^k,(d_y)^k\right)^T\right)\\
&\quad= \lambda_{\min}(C)\left\|\left((d_x)^k,(d_y)^k\right)\right\|_F^2,
\end{aligned}\tag{20}$$

where $\|A\|$ denotes the Frobenius norm matrix A.

From Lemma 2 and Lemma 4, for any $K > 0$, we have

$$\begin{aligned}
&\sum_{k=0}^{K-1}\left\|\left((d_x)^k,(d_y)^k\right)\right\|_F^2\\
&\quad\leq \frac{1}{\lambda_{\min}(C)}\sum_{k=0}^{K-1}\left(f\left(x^{k+1},y^{k+1}\right)-f\left(x^k,y^k\right)\right)\\
&\quad= \frac{1}{\lambda_{\min}(C)}\left(f\left(x^K,y^K\right)-f\left(x^0,y^0\right)\right)\\
&\quad\leq \frac{1}{\lambda_{\min}(C)}f\left(x^K,y^K\right)\\
&\quad= \frac{1}{\lambda_{\min}(C)}C\bullet\left(\left(x^K,y^K\right)\left(x^K,y^K\right)^T\right)\\
&\quad\leq \frac{\lambda_{\max}(C)}{\lambda_{\min}(C)}\left\|\left(x^K,y^K\right)\right\|_F^2\\
&\quad= \frac{n\lambda_{\max}(C)}{\lambda_{\min}(C)}.
\end{aligned}\tag{21}$$

This shows that $\sum_{k=1}^{K-1}\|((d_x)^k,(d_y)^k)\|_F^2$ is convergent, and hence $((d_x)^k,(d_y)^k) \to 0$ holds. □

In view of Lemmas 1 and 5, the termination criterion used in the rank-two feasible algorithm is $\|((d_x)^k,(d_y)^k)\|_F < \epsilon$, where ϵ is a prespecified constant.

Lemma 6. *For all initial point* (x^0, y^0) *and* $\epsilon > 0$*, the rank-two feasible direction algorithm terminates in* $[n\lambda_{\max}(C)/\epsilon^2\lambda_{\min}(C)]$ *iterations, where* $[n\lambda_{\max}(C)/\epsilon^2\lambda_{\min}(C)]$ *is an integer which does not exceed* $n\lambda_{\max}(C)/\epsilon^2\lambda_{\min}(C)$.

Proof. Based on Lemma 5, the number of iterations of the rank-two feasible direction algorithm is finite. Let K be the number of iterations; then

$$\sum_{k=0}^{K-1}\left\|\left((d_x)^k,(d_y)^k\right)\right\|_F^2 = K\epsilon^2 \leq \frac{n\lambda_{\max}(C)}{\lambda_{\min}(C)}. \tag{22}$$

So we obtain

$$K \leq \frac{n\lambda_{\max}(C)}{\lambda_{\min}(C)\epsilon^2}. \tag{23}$$

□

Now we conclude that $[n\lambda_{\max}(C)/\epsilon^2\lambda_{\min}(C)]$ is an upper bound on the number of iterations.

Since problem (7) is nonconvex, there is no guarantee that the solution generated from the feasible direction method is a global solution. However, numerical experiments in Section 5 show that the proposed algorithm always converges to the optimal solution set of problem (7).

Now, we derive the complexity of our algorithm.

The complexity of computation of the gradient is $O(n^2)$. Each norm of the gradient of the objective function can be computed in $O(n)$, so we conclude that the overall complexity to evaluate the next iteration point is $O(n^2)$. Together with Lemma 6, we get that the overall complexity of the rank-two feasible direction algorithm is $O((\lambda_{\max}(C)/\epsilon^2\lambda_{\min}(C))n^3)$. Here we can choose C satisfying $\lambda_{\max}(C)/\lambda_{\min}(C) < 2$; then the complexity does not exceed $O((2/\epsilon^2)n^3)$. We know that the complexity of the primal-dual interior point method based on AHO direction is $O(n^{4.5}\ln(1/\epsilon))$ [14, 15]. It is obvious that the complexity of the primal-dual method is higher than that of our algorithm, so our algorithm is faster than the interior-point method for the large-scale SDP relaxation of binary quadratic programming problems. In addition, the complexity of low rank feasible direction method is $O((2/\epsilon^2)n^{3.5})$ [13], which is higher than that of our method.

Let the KKT point of problem (7) be (x^*, y^*); then we can obtain the rank-two solution $Z^* = x^*(x^*)^T + y^*(y^*)^T$. Since the rank-two relaxation has the same form as Goemans and Williamson's relaxation [8], except that ours has variables in $R^{n\times 2}$ rather than $R^{n\times n}$, the same analysis as Goeman and Williamson, with minimal changes, can be applied. By the randomized cut generation scheme, the suboptimal solution of binary quadratic programming problem is obtained.

5. Numerical Results

In this section we present computational results by comparing our method with GW randomized algorithm [8] based on interior point method and low-rank feasible direction algorithm to find approximate solutions to the max-cut problem.

In interior point method, we solve the SDP relaxation by three SDP solvers, which include SDPpack software [17], SeDuMi [18], and DSDP [19]. SeDuMi is one of state-of-the-art SDP solvers. The code DSDP uses a dual-scaling interior-point algorithm and an iterative linear-equation solver. It is currently one of the fastest interior-point codes for solving SDP problems. Low-rank feasible direction algorithm is one of the efficient methods for the max-cut problem, and the algorithm is faster than the projected gradient algorithm [13]. The projected gradient algorithm [10] is faster than Homer and Peinado algorithm [9].

All the algorithms are run in the MATLAB 7.0 environment on an Inter Core2 D2.0 GHz personal computer with 2.0 GB of RAM.

5.1. Max-Cut Problem. The max-cut problem is one of the standard *NP*-complete problems defined on graphs [8]. 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 max-cut 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 max-cut problem can be formulated as the integer quadratic program as follows:

$$\begin{aligned} \max \quad & \frac{1}{2}\sum_{i<j} a_{ij}\left(1 - x_i x_j\right) \\ \text{s.t.} \quad & x_i \in \{-1, 1\}, \quad i = 1, \ldots, n. \end{aligned} \tag{24}$$

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 $C = (1/4)L$; the max-cut problem may be interpreted as a special case of the problem (1).

5.2. Numerical Results for the Random Graphs. The first set of test problems contains random graphs with two different edge densities 0.8 and 0.2, which denotes the dense random graphs and sparse random graphs, respectively. The weight on each edge is 1. We select problems in size from $n = 50$ to $n = 350$ for comparing the suboptimal value of max-cut problem and the CPU time of the four methods.

For the interior point method, we use the codes by two SDP solvers, which include SDPpack software [17] and SeDuMi [18]. In our algorithm, the iteration stops once $\|((d_x)^k, (d_y)^k)\|_F < \varepsilon$ is found. The result is shown in Table 1.

In Table 1, "SDPpack" stands for randomized algorithm based on interior point method solved by SDPpack software, "SeDuMi" for randomized algorithm based on interior point method solved by SeDuMi software, "FD" for feasible direction algorithm coupled with the randomized method, "R2FD" for our rank-two feasible direction algorithm coupled with the randomized method, "CPU" for the CPU time, "Values" for the suboptimal value of the max-cut problem based these methods, and "Density" for edge density of the random graphs.

The SDPpack and SeDuMi provide the currently best conclusion on its performance guarantee in theory. The results in Table 1 show that the approximate solutions obtained by R2FD are as good as those generated by SDPpack, SeDuMi, and FD. In addition, the CPU time of our method is less than that of SDPpack, SeDuMi, and FD. In particular, with the increase of the size of the max-cut problem, the ratios

Table 1: Comparison results for the random graphs.

Size	Density	SDPpack		SeDuMi		FD		R2FD	
		Values	CPU	Values	CPU	Values	CPU	Values	CPU
50	0.2	319	0.73	319	0.59	319	0.05	319	0.04
50	0.8	781	0.67	781	0.59	781	0.06	779	0.04
100	0.2	1185	9.27	1185	1.64	1185	0.21	1182	0.09
100	0.8	3074	10.00	3074	1.69	3075	0.28	3071	0.11
150	0.2	2521	54.46	2521	4.23	2524	0.92	2525	0.19
150	0.8	6811	41.99	6811	4.17	6813	1.15	6811	0.22
200	0.2	4404	121.33	4404	8.71	4407	2.19	4413	0.35
200	0.8	12046	152.46	12046	8.02	12051	2.98	12052	0.39
250	0.2	6729	301.64	6729	12.32	6739	4.64	6738	0.47
250	0.8	18703	336.20	18703	15.15	18701	6.87	18711	0.62
300	0.2	9521	618.06	9521	27.17	9527	10.83	9551	0.86
300	0.8	26501	687.51	26501	26.44	26513	13.85	26541	1.05
350	0.2	12934	1076.24	12934	39.13	12945	18.60	12964	1.33
350	0.8	36067	1345.38	36067	40.92	36064	25.31	36094	1.43

Table 2: Comparison results for G-set graphs.

Graph	Size	DSDP		SeDuMi		FD		R2FD	
		Values	CPU	Values	CPU	Values	CPU	Values	CPU
G01	800	11404	66	11389	590	11446	258	11469	4
G03	800	11403	28	11413	626	11431	72	11444	4
G13	800	552	22	552	542	554	384	540	3
G14	800	2979	45	2979	645	2986	336	2998	4
G15	800	2972	32	2974	728	2975	55	2974	4
G22	2000	12978	817	12979	11213	12995	5630	13146	35
G23	2000	12971	898	12976	12546	12969	6637	13074	34
G24	2000	12959	802	12960	10859	12993	5236	13062	37
G35	2000	7438	1387	7438	13236	7446	7397	7504	38
G36	2000	7421	1717	7422	14078	7427	7104	7480	32
G37	2000	7441	1390	7445	13569	7449	6891	7495	35
G43	1000	6504	91	6497	1192	6525	487	6528	8
G44	1000	6470	90	6479	1195	6507	496	6520	7
G53	1000	3731	70	3738	1135	3747	738	3753	7

of the CPU time between our methods to the three methods decrease quickly.

5.3. Numerical Results for the G-Set Graphs. The second set of test problems are from the so-called G-set graphs, which are randomly generated by the procedure rudy, a machine independent graph generator written by Rinaldi [20], Helmberg and Rendl [21], and Alperin and Nowak [22]. [20–22]. The test problems include 14 randomly generated large size test problems with nodes from 800 to 2000. Recently, Choi and Ye [19] reported computational results on a subset of G-set graphs that were solved as max-cut problems using their SDP code COPL-DSDP, or simply DSDP. The code DSDP uses a dual-scaling interior-point algorithm and an iterative linear-equation solver. The SDPpack software does not work when the size of the max-cut problems is larger than 350, so we give the results by the randomized method based on the dual-scaling algorithm solved by the DSDP software [19].

Table 2 gives the results of comparison among our R2FD method, the FD method, and the randomized method based on DSDP and SeDuMi on 14 large size test problems in the second set. In Table 2, "DSDP" presents the randomized method based on the dual-scaling algorithm by the DSDP software.

The results in Table 2 show that the approximate solutions by our method is nearly as good as those of the DSDP cuts. But our method which reaches solutions of the problems is at least 10 times faster than the FD method, 7 times faster than DSDP, and 100 times faster than SeDuMi. In particular, for G35, G36, and G37, the CPU time of our method is almost 40 times less than that of DSDP. Furthermore, We observe that R2FD took less than 5 minutes to return approximate solutions to all the 14 test problems, which required more

than 2 hours of computation by the DSDP, more than 11 hours of computation by the FD, and more than 22 hours of computation by the SeDuMi.

6. Conclusion

Because the interior-point method and feasible direction method increase the dimension of a problem from n to n^2 and rn (r is the function of n), so the two methods cost more CPU time than our method for solving large size binary quadratic programming problems, especially for problems with a large number of edges. The rank-two feasible direction method only increases the dimension of a problem from n to $2n$, so it is efficient for solving large size binary quadratic programming.

Acknowledgments

The work of Xuewen Mu is supported by the National Science Foundation for Young Scientists of China (Grant nos. 11101320 and 61201297) and the Fundamental Research Funds for the Central Universities (Grant no. K50511700007). The work of Yaling Zhang is supported by the Xian University of Science and Technology Cultivation Foundation in Shaan Xi Province of China (Program no. 2010032).

References

[1] C. Helmberg, *Semidefinite Programming for Combinatorial Optimization*, Konrad-Zuse-Zentrum fur Information-Stechnik, Berlin, Germany, 2000.

[2] F. Barahona, M. Groetschel, M. Juenger, and G. Reinelt, "An application of combinatorial optimization to statiscal optimization and circuit layout design," *Operations Research*, vol. 36, no. 3, pp. 493–513, 1988.

[3] P. H. Tan and L. K. Rasmussen, "The application of semidefinite programming for detection in CDMA," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 8, pp. 1442–1449, 2001.

[4] F. Hasegawa, J. Luo, K. Pattipati, and P. Willett, "Speed and accuracy comparation of techniques to solve a binary programming problem with application to syschronous CDMA," *IEEE Transaction on Communication*, vol. 52, pp. 2775–2780, 2004.

[5] X. Mu and Y. Zhang, "A new rank-two semidefinite programming relaxation method for multiuser detection problem," *Wireless Personal Communications*, vol. 65, pp. 223–233, 2012.

[6] J. Keuchel, C. Schnörr, C. Schellewald, and D. Cremers, "Binary partitioning, perceptual grouping, and restoration with semidefinite programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 11, pp. 1364–1379, 2003.

[7] S.-P. Wu, S. Boyd, and L. Vandenberghe, "FIR filter design via semidefinite programming and spectral factorization," in *Proceedings of the 35th IEEE Conference on Decision and Control*, pp. 271–276, Kobe, Japan, December 1996.

[8] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the Association for Computing Machinery*, vol. 42, no. 6, pp. 1115–1145, 1995.

[9] S. Homer and M. Peinado, "Design and performance of parallel and distributed approximation algorithms for maxcut," *Journal of Parallel and Distributed Computing*, vol. 46, no. 1, pp. 48–61, 1997.

[10] S. Burer and R. D. C. Monteiro, "A projected gradient algorithm for solving the maxcut SDP relaxation," *Optimization Methods and Software*, vol. 15, no. 3-4, pp. 175–200, 2001.

[11] S. Burer, R. D. C. Monteiro, and Y. Zhang, "Rank-two relaxation heuristics for max-cut and other binary quadratic programs," *SIAM Journal on Optimization*, vol. 12, no. 2, pp. 503–521, 2001.

[12] S. Burer, R. D. C. Monteiro, and Y. Zhang, "Maximum stable set formulations and heuristics based on continuous optimization," *Mathematical Programming A*, vol. 94, no. 1, pp. 137–166, 2002.

[13] H. Liu, X. Wang, and S. Liu, "Feasible direction algorithm for solving the SDP relaxations of quadratic $\{-1, 1\}$ programming problems," *Optimization Methods & Software*, vol. 19, no. 2, pp. 125–136, 2004.

[14] F. Alizadeh, P. A. Haeberly, and M. L. Overton, "Primal-dual interior-point methods for semidefinite programming: convergence rates, stability and numerical results," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 746–768, 1998.

[15] M. J. Todd, "Semidefinite optimization," *Acta Numerica*, vol. 10, pp. 515–560, 2001.

[16] C.-x. Xu, X.-l. He, and F.-m. Xu, "An effective continuous algorithm for approximate solutions of large scale max-cut problems," *Journal of Computational Mathematics*, vol. 24, no. 6, pp. 749–760, 2006.

[17] F. Alizadeh, J. P. Haeberly, M. V. Nayakkankuppam, M. L. Overton, and S. Schmieta, "SDPpack user's guide-version 0.9Beta," Tech. Rep. TR1997-737, Courant Institute of Mathematical Science, New York, NY, USA, June 1997.

[18] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11-12, no. 1–4, pp. 625–653, 1999.

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

[20] G. Rinaldi, "Rudy graph generator," http://www-user.tu-chemnitz.de/helmberg/rudy.tar.gz.

[21] C. Helmberg and F. Rendl, "A spectral bundle method for semidefinite programming," *SIAM Journal on Optimization*, vol. 10, no. 3, pp. 673–696, 2000.

[22] H. Alperin and I. Nowak, "Lagrangian smoothing heuristics for max-cut," *Journal of Heuristics*, vol. 11, no. 5-6, pp. 447–463, 2005.