Journal of Computational Mathematics, Vol.23, No.6, 2005, 619–634.

# A FEASIBLE DIRECTION ALGORITHM WITHOUT LINE SEARCH FOR SOLVING MAX-BISECTION PROBLEMS *1)

Feng-min Xu Cheng-xian Xu Hong-gang Xue
(*Department of Mathematics of Faculty of Science, Xi'an Jiaotong University, Xi'an, 710049, China*)

**Abstract**

This paper concerns the solution of the NP-hard max-bisection problems. NCP functions are employed to convert max-bisection problems into continuous nonlinear programming problems. Solving the resulting continuous nonlinear programming problem generates a solution that gives an upper bound on the optimal value of the max-bisection problem. From the solution, the greedy strategy is used to generate a satisfactory approximate solution of the max-bisection problem. A feasible direction method without line searches is proposed to solve the resulting continuous nonlinear programming, and the convergence of the algorithm to KKT point of the resulting problem is proved. Numerical experiments and comparisons on well-known test problems, and on randomly generated test problems show that the proposed method is robust, and very efficient.

*Mathematics subject classification:* 90C27
*Key words*: Max-Bisection problem, Feasible direction algorithm, NCP function, Convergence.

## 1. Introduction

This paper concerns the solution of the max-bisection problem for a given undirected graph $G = (V, E)$ with $V = \{1, 2, \cdots, n\}$ the node set, $E$ the edge set, and $n$ is even. Let $W = (w_{ij})_{n\times n}$ be the symmetric weight matrix with $w_{ij} > 0$ if $(i, j) \in E$ and $w_{ij} = 0$ if $(i, j) \notin E$. The max-bisection problem is to partition the node set $V$ into two subsets $S$ and $V\backslash S$ having equal cardinality such that the sum $\sum_{i\in S,\ j\in V\backslash S} w_{ij}$ is maximized. The problem can be formulated by assigning each node a binary variable $x_j$

$$(MB): \begin{cases} MB(S) = \text{Max} \ \ \frac{1}{4}\sum\limits_{i,j} w_{ij}(1 - x_i x_j) \\ s.t. \ \ e^T x = 0, \\ \ \ x_j^2 = 1, \ \ j = 1, \cdots, n, \end{cases}$$

where $e \in R^n$ is the column vector of all ones. The constraint $x_j^2 = 1$ implies that $x_j$ takes either 1 or $-1$, so that we will have either $S = \{j|x_j = 1\}$ or $S = \{j|x_j = -1\}$. The constraint $e^T x = 0$ ensures $|S| = |V \setminus S|$.

The max-bisection problem is NP-hard [1], and has wide applications in real world. Approximation algorithms are available, and polynomial time approximation schemes exist for the problem over dense graphs [3] and over planar graphs [4]. Frieze and Jerrum[6] extended Goemans-Willamson approach [5] to max-bisection problems, giving a randomized 0.651 approximation algorithm for the maximum weight bisection problem. Ye [7] improved the performance ratio of the algorithm to 0.6993 by combining the Frieze-Jerrum approach with some

* Received September 21, 2004; Final revised May 6, 2005
1) This work is supported by National Key Laboratory of Mechanical Systems and National Natural Key Product Foundations of China 10231060.

rotation argument that is applied to the optimal solution of the semi-definite relaxation of the problem. Halperin and Zwith [8] further improved the approximation ratio to 0.7016 by strengthening to SDP relaxation with the triangle inequalities. All these algorithms are based on the semi-definite relaxation of problem MB

$$(SDP): \begin{cases} SDP(S) = \text{Max} \;\; L \cdot X, \\ s.t. \;\; Diag(X) = e, \\ \;\; ee^T \cdot X = 0, \\ \;\; X \succeq 0, \end{cases}$$

here $L = \frac{1}{4}(\text{Diag}(We) - W)$, $X \in R^{n \times n}$ is a symmetric matrix, $L \cdot X = trace(LX)$ is the matrix inner product, and $X \succeq 0$ means $X$ positive semi-definite. It is clear that (SDP) is a relaxation of (MB), and since $X = xx^T$ is feasible for (SDP) for any feasible solution $x$ of (MB), we have $SDP(S) \geq MB(S)$.

In this paper, we will propose a continuous model for the solution of max-bisection problems, and a feasible direction algorithm without line search to solve the resulting continuous model. Unlike the available relaxation methods for max-bisection problems, NCP functions are employed to convert the max-bisection problem to a continuous nonlinear programming, and then the resulting nonlinear programming problem is solved using the feasible direction method without line search. The convergence property of the proposed algorithm is studied, and numerical experiments and comparisons on some well-known test problems and on some randomly generated problems are made to show the efficiency of the proposed algorithm on both the CPU times and solutions.

The rest of paper is organized as follows. In section 2 we convert the max-bisection into a continuous nonlinear programming problem by using NCP functions. The relationship between the solutions of the max-bisection problem and the resulting nonlinear programming problem is analyzed. The feasible direction method without line searches are presented in section 3. The convergence of the algorithm to KKT point of the resulting nonlinear programming is proved. Numerical results and comparisons are reported in section 4, and it is observed that the algorithm is effective and efficient on both the CPU times and the solutions. Section 5 gives the conclusions.

## 2. The Continuous Model of Max-Bisection Problem

In this section we formulate the max-bisection problem into a continuous nonlinear programming by using NCP functions, and analyze the relationship between the solutions of the max-bisection and the resulting continuous nonlinear programming.

The max-bisection problem can be rearranged as

$$(MB): \begin{cases} MB(S) = \text{Max} \quad x^T LX \\ s.t. \;\; e^T x = 0, \\ \;\; x_j^2 = 1, \;\; j = 1, \cdots, n, \end{cases}$$

where $L = \frac{1}{4}(\text{Diag}(We) - W)$. If $w_{ij} \geq 0$ for all $i,\ j$, then $L$ is a Laplace matrix, and hence, positive semi-definite ($L \succeq 0$) [9]. Without loss of generality, we will assume, in the rest of the paper, that $L$ is positive definite and $L_{ii} > 0, i = 1, \cdots, n.$ because of no any effects on optimal solutions.

Adding constraints $-1 \leq x_j \leq 1, \quad j = 1, \cdots, n$ to problem $(MB)$ gives no effects on its

solutions. The constraints $x_j^2 = 1$ and $-1 \le x_j \le 1, \quad j = 1, \cdots, n$ can be expressed as

$$\begin{cases} (1+x_j)(1-x_j) = 0, \\ (1+x_j) \ge 0, \\ (1-x_j) \ge 0, \end{cases} \qquad 1 \le j \le n, \tag{2.1}$$

The complementary conditions (2.1) can be replaced by an equality constraint using NCP functions to get a continuous nonlinear optimization problem with feasible points restricted in $[-1, 1]^n$. Two commonly used NCP functions are (see [10])

$$\Phi_F(a,b) = \sqrt{a^2+b^2} - a - b = 0 \Leftrightarrow ab = 0, a \ge 0, b \ge 0,$$
$$\Phi_M(a,b) = \mathrm{Min}\{a,b\} = 0 \Leftrightarrow ab = 0, a \ge 0, b \ge 0.$$

With these two NCP functions, then problem $(MB)$ can equivalently be described by the following two nonlinear programming problems

$$(NP1): \begin{cases} V^* = \mathrm{Max} \;\; x^T L x \\ s.t. \;\; \phi_F(1-x_j, 1+x_j) = 0, j = 1, \cdots, n, \\ \quad e^T x = 0, \end{cases}$$

or

$$(NP2): \begin{cases} \mathrm{Max} \;\; x^T L x \\ s.t. \;\; \phi_M(1-x_j, 1+x_j) = 0, j = 1, \cdots, n, \\ \quad e^T x = 0, \end{cases}$$

Since these two nonlinear programming are equivalent to problem $(MB)$, the solution of the max-bisection problem can be obtained by finding the global solution of one of these two problems (if solvable). Since the problem is to maximize a convex function over a compact feasible region, the optimal solution exists, but may not be unique.

The rest of the paper focus on the solution of problem $(NP1)$(We will discuss the solution of problem (NP2) in another paper, since the properties of function $\Phi_M(a,b)$ is quite different from the properties of function $\Phi_F(a,b)$). At first, let us consider the following nonlinear programming

$$(NP3): \begin{cases} V^{**} = \mathrm{Max} \;\; x^T L x \\ s.t. \;\; \phi_F(1-x_j, 1+x_j) \le 0, j = 1, \cdots, n, \\ \quad e^T x = 0, \end{cases}$$

Let

$$F_1 = \{x \mid e^T x = 0, \phi_F(1-x_j, 1+x_j) = 0, j = 1, \cdots, n.\}$$
$$F_3 = \{x \mid e^T x = 0, \phi_F(1-x_j, 1+x_j) \le 0, j = 1, \cdots, n.\}$$

be feasible regions of problem $(NP1)$ and $(NP3)$, respectively. The following theorem gives the relationship between the solutions of problems $(NP1)$ and $(NP3)$.

**Theorem 2.1.** *Let $x$ be an optimal solution of $(NP3)$, then $x$ is an optimal solution of problem $(NP1)$.*

*Proof.* At first, we prove that the optimal solution of $(NP3)$ is a feasible point of problem $(NP1)$. Suppose $x$ is the optimal solution of $(NP3)$, but not feasible to problem $(NP1)$. Since $e^T x = 0$, without loss of generality, it is assumed that there exists indices $s, t$ such that

$$x_s + x_t = 0, \quad |x_s| < 1, \quad |x_t| < 1.$$

Define

$$y_i = \begin{cases} x_i & i \neq s,t, \\ sign(x_i) & i = s,t, \end{cases}$$

from the definition of $y$, $y \in F_3$ holds, and we have

$$\begin{aligned} y^T Ly - x^T Lx &= 2(\sum_{j=1}^{n} y_s L_{sj} y_j + \sum_{j=1}^{n} y_t L_{tj} y_j \\ &\quad - \sum_{j=1}^{n} x_s L_{sj} x_j - \sum_{j=1}^{n} x_t L_{tj} x_j) \\ &\quad -y_s^2 L_{ss} - y_t^2 L_{tt} + x_s^2 L_{ss} + x_t^2 L_{tt} - 2y_s L_{st} y_t + 2x_s L_{st} x_t. \end{aligned} \tag{2.2}$$

Since $x$ is the optimal solution of $(NP3)$, from the KKT condition to the optimal solution of problem $(NP3)$ and $|x_s| < 1,\ |x_t| < 1$ we obtain $\sum_{j=1}^{n} L_{tj} x_j = 0$ and $\sum_{j=1}^{n} L_{sj} x_j = 0$. Using these two equalities and the definition of vector $y$, we can have

$$\begin{aligned} &2(\sum_{j=1}^{n} y_s L_{sj} y_j + \sum_{j=1}^{n} y_t L_{tj} y_j - \sum_{j=1}^{n} x_s L_{sj} x_j - \sum_{j=1}^{n} x_t L_{tj} x_j) \\ &= 2(\sum_{j=1}^{n} y_s L_{sj} x_j + \sum_{j=1}^{n} y_t L_{tj} x_j) + y_s L_{ss}(y_s - x_s) \\ &\quad + y_s L_{st}(y_t - x_t) + y_t L_{tt}(y_t - x_t) + y_t L_{st}(y_s - x_s) \\ &= 2(y_s L_{ss}(y_s - x_s) + y_s L_{st}(y_t - x_t) \\ &\quad + y_t L_{tt}(y_t - x_t) + y_t L_{st}(y_s - x_s)), \end{aligned} \tag{2.3}$$

Combining inequalities (2.2) and (2.3) generates

$$\begin{aligned} y^T Ly - x^T Lx &= 2(y_s L_{ss}(y_s - x_s) + y_t L_{tt}(y_t - x_t)) \\ &\quad -y_s^2 L_{ss} - y_t^2 L_{tt} + x_s^2 L_{ss} + x_t^2 L_{tt} \\ &\quad +2(y_s L_{st}(y_t - x_t) + y_t L_{st}(y_s - x_s)) \\ &\quad -2y_s L_{st} y_t + 2x_s L_{st} x_t. \end{aligned} \tag{2.4}$$

Rearranging the first part on the right of (2.4) gives

$$\begin{aligned} &2(y_s L_{ss}(y_s - x_s) + y_t L_{tt}(y_t - x_t)) - y_s^2 L_{ss} - y_t^2 L_{tt} + x_s^2 L_{ss} + x_t^2 L_{tt} \\ &= y_s^2 L_{ss} - 2y_s L_{ss} x_s + x_s^2 L_{ss} + y_t^2 L_{tt} - 2y_t L_{tt} x_t + x_t^2 L_{tt} \\ &= (y_s - x_s)^2 L_{ss} + (y_t - x_t)^2 L_{tt} > 0, \end{aligned} \tag{2.5}$$

since $L_{ii} > 0$ for all $i = 1, 2, \cdots, n$. For the remaining part on the right of (2.4) we have

$$\begin{aligned} &2(y_s L_{st}(y_t - x_t) + y_t L_{st}(y_s - x_s)) - 2y_s L_{st} y_t + 2x_s L_{st} x_t \\ &= 2(y_s(y_t - x_t)) L_{st} + 2(x_s(x_t - y_t)) L_{st} > 0, \end{aligned} \tag{2.6}$$

since $L_{ij} \leq 0$ when $i \neq j$.

Inequalities (2.5) and (2.6) give the result $y^T Ly > x^T Lx$ which contradicts the fact that $x$ is the optimal solution of $(NP3)$. The contradiction proves that $x$ is feasible to problem $(NP1)$ .

The optimality of $x$ to problem $(NP1)$ comes from the fact that $x \in F_1$, and $F_1 \subset F_3$. This completes the proof. ♯

The algorithm for the solution of max-bisection problems, presented in the next section,

solves the following problem

$$(NP4): \begin{cases} \text{Max} \;\; x^T L x \\ s.t. \;\; \phi_F(1-x_j, 1+x_j) \le 0, j = 1, \cdots, n, \\ \quad e^T x = 0, \\ \quad \|x\|_2 = 1, \end{cases}$$

and generates a solution of problem $(NP1)$ from the solution of problem $(NP4)$. In order to understand the relationship between the solutions of problems $(NP3)$ and $(NP4)$, we consider the following auxiliary problem

$$(NP5): \begin{cases} \text{Max} \;\; x^T L x \\ s.t. \;\; \phi_F(\sqrt{n}-x_j, \sqrt{n}+x_j) \le 0, j = 1, \cdots, n, \\ \quad e^T x = 0, \\ \quad \|x\|_2 \le \sqrt{n}, \end{cases}$$

Define the feasible sets for both the problems

$$F_4 = \{x \mid e^T x = 0, \|x\|_2 = 1, \phi_F(1-x_j, 1+x_j) \le 0, j = 1, \cdots, n.\}$$

$$F_5 = \{x \mid e^T x = 0, \|x\|_2 \le \sqrt{n}, \phi_F(\sqrt{n}-x_j, \sqrt{n}+x_j) \le 0, j = 1, \cdots, n.\}.$$

The next theorem gives the relationship between the optimal solutions of problems $(NP4)$ and $(NP5)$.

**Theorem 2.2.** *Suppose $x^*$ is the optimal solution of $(NP4)$, then $\sqrt{n}x^*$ is the optimal solution of $(NP5)$.*

*Proof.* For any $\overline{x} \in F_5$, we have $\frac{\overline{x}}{\sqrt{n}} \in F_4$. Then

$$f(\sqrt{n}x^*) = nf(x^*) \ge nf(\frac{\overline{x}}{\sqrt{n}}) = f(\overline{x}).$$

That is, $\sqrt{n}x^*$ is the optimal solution of $(NP5)$.

Theorem 2.2 implies that the optimal solution of $(NP5)$ can directly be achieved from the optimal solution of $(NP4)$. Let $x^*$ be the optimal solution of problem $(NP5)$. Since $F_3 \subset F_5$, $f(x^*) \ge f(x)$ holds for any $x \in F_3$, that is, ${x^*}^T L x^*$ provides an upper bound on the optimal value of $(NP3)$. An approximate solution of $(NP3)$, and hence of the max-bisection problem is then generated from the solution of problem $(NP4)$ using the following way. Let $y = sign(x^*)$, and define $S = \{i | y_i = 1\}$. If $|S| = \frac{n}{2}$, then $y$ is the desired approximate solution of problem $(NP3)$. If $|S| \ne \frac{n}{2}$, a greedy algorithm is used to update $y$ such that $|S| = \frac{n}{2}$ (see Frieze and Jerrum [6]). Even though this feasible solution can not be guaranteed to be optimal to problem $(NP3)$, hence the max-bisection problem, it is feasible to problem $(NP3)$ with satisfactory objective function value. Numerical results given in section 4 show the conclusion.

Finally, it needs to point out that the two NCP functions are not differentiable at points $a = 0, b = 0$ and $a = b$. However, these points are not contained in the feasible region $F_4$ of problem $(NP4)$. When the feasible direction method presented in next section is used to get a solution of problem $(NP4)$, all the iterates are feasible and the gradients for all functions including objective and constraints can be evaluated at all iterates. In fact, only the gradients of the objective function are evaluated at all iterates in the proposed algorithm.

## 3. The Feasible Direction Algorithm Without Linear Search

In this section, the feasible direction algorithm without line search is presented for the solution of problem $(NP4)$. The algorithm employs no line search and no calculation on matrices,

and thus greatly reduces the calculation expenses. It is shown that the algorithm is convergent to KKT points of problem $(NP4)$. Before we derive the algorithm, some basic properties of problem $(NP4)$ are discussed based on following lemma.

**Lemma 3.1** *The NCP function $\phi_F(1-x_j, 1+x_j)$ for $j = 1, 2, \cdots, n$ is strictly convex for all $x_j \in R$.*

*Proof.* Since

$$\phi_F(1-x_j, 1+x_j) = \sqrt{2x_j^2+2} - 2,$$

the first-order derivative and the second-order derivative of the function $\phi_F(1-x_j, 1+x_j),\ j = 1, 2, \cdots, n$ are given by

$$\phi_F^{'}(1-x_j, 1+x_j) = \frac{2x_j}{\sqrt{2x_j^2+2}},$$

$$\phi_F^{''}(1-x_j, 1+x_j) = \frac{4}{(2x_j^2+2)\sqrt{2x_j^2+2}} > 0.$$

Since the second-order derivative of $\phi_F(1-x_j, 1+x_j),\ (j = 1, 2, \cdots, n)$ is positive for all $x_j \in R$, $\phi_F(1-x_j, 1+x_j)$ is strictly convex for $j = 1, 2, \cdots, n$.

The conclusion of the lemma implies that the feasible region $F_4$ of problem $(NP4)$ is convex.

Let $x^k$ be a feasible point of problem $(NP4)$, the algorithm simply generates the next iterative point $x^{k+1}$ using the iteration

$$x^{k+1} = \frac{g^k}{\|g^k\|_2}, \tag{3.1}$$

where $g^k = 2Lx^k$ is the gradient of objective function $f(x)$ at point $x^k$. The next lemma shows that $x^{k+1} \in F_4$.

**Lemma 3.2** *Let $x^k$ be a feasible point of problem $(NP4)$, then $x^{k+1} \in F_4$.*

**Proof.** The definition of $x^{k+1}$ gives $\|x^{k+1}\| \leq 1$, and hence

$$\phi_F(1-x_j^{k+1}, 1+x_j^{k+1}) \leq 0, j = 1, \cdots, n$$

are satisfied. On the other hand, using the fact $Le = 0$ we have

$$e^T x^{k+1} = \frac{e^T g^k}{\|g^k\|_2} = \frac{2e^T Lx^k}{\|g^k\|_2} = \frac{2x^k Le}{\|g^k\|_2} = 0.$$

Thus, $x^{k+1} \in F_4$ holds.

The following lemma provides the first-order necessary condition for optimal solutions of problem $(NP4)$.

**Lemma 3.3.** *Suppose $x^*$ is an optimal solution of problem $(NP4)$, then $x^*$ is an eigenvector of the matrix $L$ that satisfies the constraints of problem $(NP4)$, that is, there exists an eigenvalue $\lambda_2$, such $Lx^* = \lambda_2 x^*$.*

**Proof.** Suppose $x^*$ is an optimal solution of problem $(NP4)$. It follows from the last two constraints in problem $(NP4)$ that the first constraint of the problem is inactive at the solution $x^*$ for all $j = 1, 2, \cdots, n$. Then applying the KKT condition to problem $(NP4)$, there exist Lagrange multipliers $\lambda_1,\ \lambda_2$ such that

$$g^* - \lambda_1 e - 2\lambda_2 x^* = 0$$

holds at $x^*$. Pre-multiplying the equation using vector $e$, and using the fact $g^* = 2Lx^*$, $Le = 0,\ e^T x^* = 0$, we obtain $\lambda_1 = 0$. Then

$$g^* - 2\lambda_2 x^* = 0 \Leftrightarrow Lx^* - \lambda_2 x^* = 0.$$

This indicates $x^*$ is an eigenvector of the matrix $L$. The proof is completed.

Lemma 3.3 shows that optimal solutions of problem $(NP4)$ can be found from the eigenvectors of the matrix $L$ that satisfy the constraints of problem $(NP4)$. The algorithm proposed in this section converges to an eigenvector of the matrix $L$ satisfying constraints in $(NP4)$.

Define $d^k = x^{k+1} - x^k$ as a search direction, the next two lemmas show that if $d^k = 0$, then $x^k$ is a KKT point of problem $(NP4)$, and if $d^k \neq 0$ , then $d^k$ is a feasible ascent direction of problem $(NP4)$ at point $x^k$.

**Lemma 3.4.** *If $d^k = 0$, then $x^k$ is an eigenvector of the matrix $L$ that satisfies the constraints of problem $(NP4)$, that is, $x^k$ is a KKT point of problem $(NP4)$.*

*Proof.* From the definition of $d^k$, we have

$$d^k = \frac{g^k}{\|g^k\|_2} - x^k = 0,$$

that is

$$Lx^k - \frac{\|g^k\|_2}{2} x^k = 0.$$

This shows that $x^k$ is an eigenvector of the matrix $L$ satisfying constraints of problem $(NP4)$. This completes the proof of the lemma.

**Lemma 3.5.** *Suppose $d^k \neq 0$, then $d^k$ is a feasible ascent direction of problem $(NP4)$ at $x^k$.*

*Proof.* The feasibility of the direction $d^k$ comes from the feasibility of points $x^{k+1}, x^k$ and the convexity of the feasible region $F_4$.

Since $\|x^k\|_2 = 1$, we have

$$\begin{aligned}&(\nabla f(x^k))^T d^k = (g^k)^T (x^{k+1} - x^k)\\ &= \|g^k\|_2 - (g^k)^T x^k \geq \|g^k\|_2 (1 - \|x^k\|_2) \geq 0,\end{aligned}$$

If $(\nabla f(x^k))^T d^k > 0$, then $d^k$ is an ascent direction. If $(\nabla f(x^k))^T d^k = 0$, then

$$f(x^k + \alpha d^k) = f(x^k) + \alpha(\nabla f(x^k))^T d^k + \alpha^2 (d^k)^T L d^k,$$

and the positive definiteness of the matrix $L$ also shows that $d^k$ is ascent. The proof is completed.

Lemma 3.5 implies that $\alpha = 1$ is the best choice for the step length in the direction $d^k$. It is the reason why we adopt the above iterative format without line searches. No line search in iterations greatly reduces the computational cost, and increase the speed of the algorithm to achieve the solution.

The following theorem gives the convergence of the algorithm to KKT points of problem $(NP4)$.

**Theorem 3.6.** *Suppose $d^k \to 0$. Then any accumulation point $x^*$ of the sequence $\{x^k\}$ is an eigenvector of the matrix $L$ that satisfies the constraints of problem $(NP4)$, that is, $x^*$ is a KKT point of $(NP4)$.*

*Proof.* Let $x^*$ be an accumulation point of the sequence $\{x^k\}$. Without loss of generality, assume that $x^k \to x^*$. It follows from the definition of $d^k$, and the continuity of $g(x)$, we have

$$\lim_{k\to\infty} d^k = \lim_{k\to\infty} \left(\frac{g^k}{\|g^k\|_2} - x^k\right) = \frac{g^*}{\|g^*\|_2} - x^* = 0.$$

That is,

$$Lx^* - \frac{\|g^*\|_2}{2}x^* = 0.$$

This shows that $x^*$ is an eigenvector of the matrix $L$ satisfying constraints of $(NP4)$, and the proof is completed.

The rest of this section is devoted to the proof of the convergence of the infinite sequence $\{d^k\}$, generated by the proposed algorithm, to zero vector.

**Lemma 3.7**[11]. *Suppose $A \succ 0$ and $B \succ 0$, then*

$$\lambda_{min}(A)\lambda_{max}(B) \leq A \cdot B \leq n\lambda_{max}(A)\lambda_{max}(B)$$

.

**Lemma 3.8.** *Let $d^k \neq 0$, then the following inequalities hold*

$$\lambda_{min}(L)\|d^k\|_2^2 \leq f(x^{k+1}) - f(x^k) \leq \|g^k\|_2\|d^k\|_2 + \lambda_{max}(L)\|d^k\|_2^2.$$

*Proof.* Since

$$f(x^{k+1}) - f(x^k) = (g^k)^T d^k + (d^k)^T L d^k. \tag{3.2}$$

$$(g^k)^T d^k \leq \|g^k\|_2\|d^k\|_2, \tag{3.3}$$

From Lemma 3.7 and inequality (3.3), we have

$$f(x^{k+1}) - f(x^k) \leq \|g^k\|_2\|d^k\|_2 + \lambda_{max}(L)\|d^k\|_2^2. \tag{3.4}$$

Furthermore,

$$(g^k)^T d^k \geq 0. \tag{3.5}$$

and Lemma 3.7 gives

$$f(x^{k+1}) - f(x^k) \geq \lambda_{min}(L)\|d^k\|_2^2. \tag{3.6}$$

Inequalities (3.4) and (3.6) give the conclusion of the Lemma.

**Theorem 3.9.** *If $d^k \neq 0$ for any $k > 0$, then $\|d^k\|_2 \to 0$.*

*Proof.* From Lemma 3.8, for any $m > 0$ we have

$$\begin{aligned}
\sum_{i=0}^{m} \|d^k\|_2^2 &\leq \tfrac{1}{\lambda_{min}(L)} \sum_{i=0}^{m} (f(x^{k+1}) - f(x^k)) \\
&= \tfrac{1}{\lambda_{min}(L)}[f(x^m) - f(x^0)] \\
&\leq \tfrac{1}{\lambda_{min}(L)}(x^*)^T L x^* \\
&\leq \tfrac{\lambda_{max}(L)}{\lambda_{min}(L)}\|x^*\|_2^2 \\
&\leq \tfrac{\lambda_{max}(L)}{\lambda_{min}(L)}.
\end{aligned}$$

That is, $\sum\limits_{i=0}^{+\infty} \|d^k\|_2^2$ is convergent, and hence $\|d^k\|_2 \to 0$ holds.

When the algorithm is implemented to solve problem $(NP4)$, the condition $\|d^k\| \leq \epsilon$ or $f(x^{k+1}) - f(x^k) \leq \epsilon$ is used to terminate the iteration.

## 4. Numerical Experiments

In this section we report numerical results and comparisons to show effectiveness and efficiency of the proposed feasible direction method without line searches. The algorithm is

programmed in Matlab 6.0. All the constraints $\phi_F(1-x_j, 1+x_j) \leq 0, j = 1, \cdots, n$ in problem $(NP4)$ are inactive at all iterates, but they are necessary to restrict the feasible region. The value $\epsilon = 0.001$ is used in the termination conditions $\|d^k\| \leq \epsilon$ or $f(x^{k+1}) - f(x^k) \leq \epsilon$. Observations on experiments found that if $\epsilon$ is too small, more iterations are required without any improvement on the final solution and it will cost some computational time. If $\epsilon$ is too large, iterations will be terminated when far away from a satisfactory solution, and further improvement on the final solution can be obtained by increasing the value of $\epsilon$. It is found that the value $\epsilon = 0.001$ is worth recommending.

Let $x$ be the optimal solution of $(NP4)$, achieved by the proposed algorithm. Define $y = sign(x)$ and $S = \{i|y_i = 1\}$. If $|S| = \frac{n}{2}$, then accept $y$ as an approximate solution of problem $(NP3)$. If $|S| \neq \frac{n}{2}$, a greedy strategy is used to update $y$. Assume $|S| \geq \frac{n}{2}$. Let $\xi(i) = \sum_{j \in V \setminus S} w_{ij}$ for each $i \in S$, and rearrange the set $S = \{i_1, i_2, \cdots, i_{|S|}\}$, such that $\xi(i_1) \geq \xi(i_2) \geq \cdots, \xi(i_{|S|})$. Then assign $S = \{i_1, i_2, \cdots, i_{|\frac{n}{2}|}\}$, that is, $|S| = \frac{n}{2}$. Then set the solution $y$ as $y_i = 1,\ i \in S$ and $y_i = -1, i \notin S$.

The numerical experiments and comparisons are made on some test max-bisections problems available in literatures, and on some randomly generated test problems. The results of the proposed algorithm are presented and compared with those generated using Y. Ye's approximation method that also generates an approximate solution to max-bisection problems with performance ratio 0.699. The software package SDPPack[15] is also employed to solve the SDP relaxation of max-bisection problems, and offers an upper bound for the the optimal value of the problem.

The first set of experiments are made on 6 max-bisection problems existing in literatures. The details of these test problems are given in appendix A. Initial points $x^0$ for these problems are randomly generated such that constraints $e^T x^0 = 0$, and $\phi_F(1 - x_j^0, 1 + x_j^0) \leq 0$ J $= 1, \cdots, n$ are satisfied. Then normalizing $x^0 = x^0/\|x^0\|_2$, gives an initial feasible point $\|x^0\|_2 = 1$.

The first experiments that we made on these test problems check the effects of different initial points on the algorithm. 10 different initial points are generated for each test problem to have the experiments. It is observed that effects on the solution procedure given by different initial points are not obvious, that is, the same solution is obtained by the algorithm from different initial points, and the difference between CPU times are very small. It seems that the main reason to have such a small effect on different initial points is that the algorithm employs no any line searches and no any computations on matrices in determining search directions. Table 1 gives the results for **Problem 5** with 4 different initial points where $x^0$, $f^*$, Iter and $CPU$ denote the initial point, the optimal value of the max-bisection, the number of iterations to reach the solution, and the CPU time.

Table 1: Numerical Results for Problem 5

| $x^0$ | $f^*$ | Iter | $CPU$ |
|---|---|---|---|
| (0.1;0.1;-0.2;-0.2;-0.2;0.4) | 7 | 33 | 0.2 |
| (0.3;-0.4;0.1;0.6;-0.2;-0.4) | 7 | 41 | 0.22 |
| (-0.2;-0.2;0.3;0.3;0.1;-0.3) | 7 | 36 | 0.21 |
| (-0.3;0.3;0.4;0.5;-0.5;-0.4) | 7 | 41 | 0.22 |

Table 2 presents the comparison of the results obtained by the algorithm proposed in this paper, Y. Ye's approximate algorithm and the SDPPack, where $UB$ means the upper bound given by $(SDP)$ relaxation in SDPPack, $f^*$ and $CPU$ mean the function value and CPU time of the proposed algorithm (Continue) and the 0.699 approximation algorithm. $\triangle = |\ |S| - |V \setminus S|\ |$, calculated at the solution $x^*$ of problem $(NP4)$, and $\rho$ in column 8 is the ratio of the function

values of $(NP3)$ at $y$ and of $(NP5)$ at $\sqrt{n}x^*$, that is, $\rho = \frac{f^*}{f(\sqrt{n}x^*)}$ with $f^* = f(y)$. It can be observed from the table that both the algorithms generate the same solutions for these 6 test problems, optimal for problems 2-6 and an approximate solution for problem 1, but the algorithm proposed in this paper uses less CPU times than those used by 0.699 approximation algorithm. For problems 3-6, we have $\triangle = 0$. This indicates that the algorithm directly generates the optimal solutions for these problem and the greedy strategy is not used. $\triangle = 2$ for **Problem 1** to $\triangle = 1$ for **Problem 2** indicates the greedy strategy is used for Problem 1 and Problem 2 to change either two or one node to get the desired bisection.

Table 2: Numerical Results

| Problem | UB | Continue | | 0.699 | | $\triangle$ | $\rho$ |
|---|---|---|---|---|---|---|---|
| | | $f^*$ | CPU | $f^*$ | CPU | | |
| Problem1 | 38.7 | 37 | 0.5 | 37 | 2.6 | 2 | 0.657 |
| Problem2 | 37.9 | 37 | 0.63 | 37 | 2.9 | 1 | 0.738 |
| Problem3 | 19 | 19 | 2 | 19 | 3.6 | 0 | 0.957 |
| problem4 | 51 | 51 | 0.6 | 51 | 2.2 | 0 | 0.862 |
| Problem5 | 7.36 | 7 | 0.2 | 7 | 2.1 | 0 | 0.802 |
| Problem6 | 41 | 38 | 0.7 | 38 | 2.2 | 0 | 0.82 |

The next set of experiments are made to observe the iteration progress of the proposed algorithm on **Problem 2** and **Problem 3**. This is because the algorithm uses more CPU time to find the solution of problem 3. It was observed that the algorithm mad 80 iterations to get the optimal solution $x^*$ of problem $(NP4)$. The converted solution $y$ is optimal to problem 3, and hence the original max-section problem. Table 3 presents part of function values $f(x^k)$ and values of $\|d^k\|$ in iterations. From the table it can be found that the objective function values are monotonically increased, and $\|d^k\|$ converges to zero. The detailed information about the progress of the algorithm on these two problems are given in Figures 1, 2, 3 and 4, respectively.

Table 3: Numerical Results for Problem 3

| The ith iteration | $d^k$ | $f(x^k)$ |
|---|---|---|
| 1 | 0.1732 | 0.8515 |
| 5 | 0.0241 | 0.9033 |
| 10 | 0.0189 | 0.9071 |
| 15 | 0.0205 | 0.9113 |
| 20 | 0.0241 | 0.9161 |
| 30 | 0.0335 | 0.9324 |
| 40 | 0.0367 | 0.9573 |
| 50 | 0.0284 | 0.9780 |
| 60 | 0.0181 | 0.9873 |
| 70 | 0.0109 | 0.9915 |
| 80 | 0.0069 | 0.9930 |

Another set of experiments are made on a group of randomly generated test problems. The dimension of these test problems is from 10 to 200. Since the memory restrictions in PC, and CPU time for 0.699 approximation algorithm, we didn't carry out experiments on problems

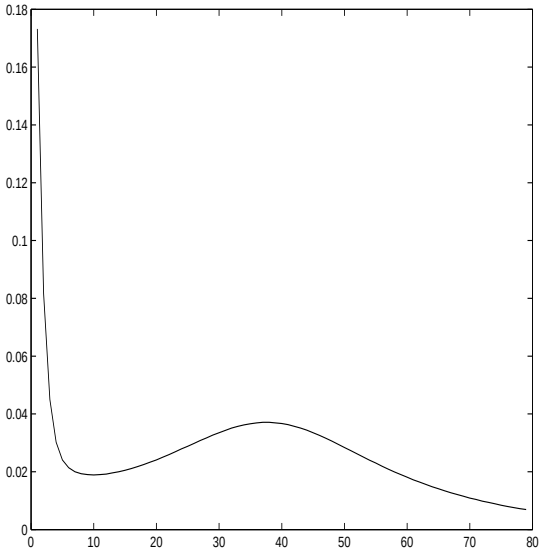


Figure 1 The convergence curve of $\|d^k\|$ in Problem 3

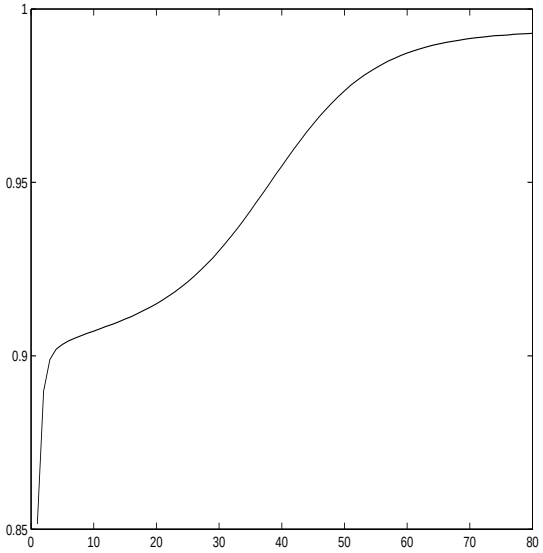


Figure 2 The convergence curve of $f(x^k)$ in Problem 3

with dimension larger than 300. The procedure to generate a random graph is as follows. Let $p \in (0,1)$ be any given number that is used to control the density of edges in a graph, and $n$ be given. For each pair $(i,j),\ i \neq j$, a random number in $(0,1)$ was generated. If the number is less than or equal to $p$, then there is an edge between nodes $i$ and $j$, and set $w_{ij} = 1$; Otherwise, there is no edge between nodes $i$ and $j$, and $w_{ij} = 0$. All of the test problems and numerical results are listed in Table 4, where $n$ is the dimension of problems, the notations have the same meaning as those in Table 3.

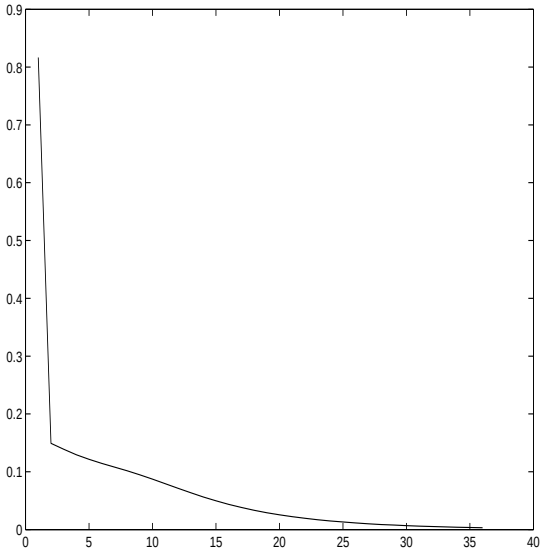


Figure 3 The convergence curve of $\|d^k\|$ in Problem 2

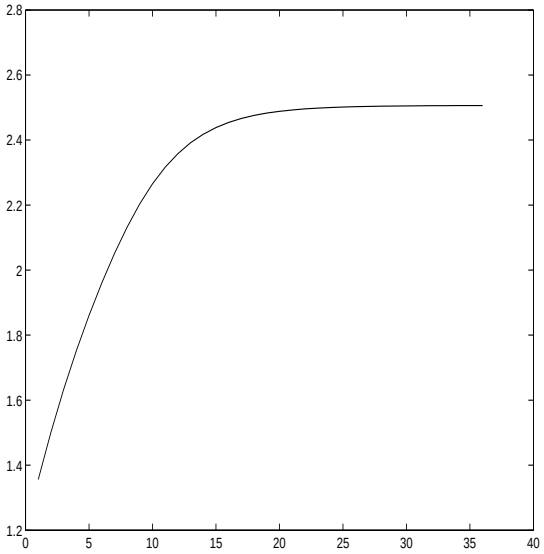


Figure 4 The convergence curve of $f(x^k)$ in Problem 2

The test problems in final group are randomly generated using a similar way as the problem in previous group. The difference is the value of weight $w_{ij}$ on an edge. If an edge exists, its weight is set to a random integer number in $[1, 100]$. All of the test problems and numerical results are listed in Table 5.

It can be observed from these two tables that the proposed algorithm provides better solutions with much less CPU times than 0.699 approximation algorithm does. The experiments and comparisons on either the existing test problems or the randomly generated test problems

Table 4: Numerical Results for $p = 0.3$ and $p = 0.6$

| | | | Continue | | | 0.699 | |
|---|---|---|---|---|---|---|---|
| p | n | UB | $f^*$ | CPU | $\triangle$ | $f^*$ | CPU |
| 0.3 | 10 | 2 | 2 | 0.01 | 1 | 2 | 0.08 |
| 0.6 | 10 | 13.5 | 13 | 0.01 | 0 | 13 | 0.09 |
| 0.3 | 60 | 122 | 115 | 0.04 | 2 | 114 | 3.59 |
| 0.6 | 60 | 395 | 378 | 0.2 | 1 | 374 | 2.9 |
| 0.3 | 100 | 331 | 310 | 0.21 | 3 | 307 | 15.62 |
| 0.6 | 100 | 1078 | 1043 | 0.8 | 3 | 1039 | 18.8 |
| 0.3 | 200 | 1237 | 1145 | 1.2 | 2 | 1048 | 262 |
| 0.6 | 200 | 4151 | 3996 | 1.5 | 1 | 3938 | 262 |

Table 5: Numerical Results for $p = 0.3$ and $p = 0.6$

| | | | Continue | | | 0.699 | |
|---|---|---|---|---|---|---|---|
| p | n | UB | $f^*$ | CPU | $\triangle$ | $f^*$ | CPU |
| 0.3 | 10 | 505 | 505 | 0.08 | 1 | 505 | 0.2 |
| 0.6 | 20 | 3502 | 3442 | 0.06 | 0 | 3442 | 0.28 |
| 0.3 | 60 | 16988 | 16370 | 1.4 | 3 | 16242 | 3.17 |
| 0.6 | 60 | 30460 | 29744 | 1.2 | 2 | 29612 | 6.8 |
| 0.3 | 100 | 44582 | 42957 | 2.1 | 2 | 42689 | 24.966 |
| 0.6 | 100 | 83992 | 81910 | 2.4 | 1 | 81779 | 35.5 |
| 0.3 | 200 | 17106 | 16910 | 3 | 3 | 158030 | 311.8 |
| 0.6 | 200 | 325349 | 317592 | 3.5 | 2 | 303764 | 299 |

show the effectiveness and efficiency of the proposed algorithm for the solution of max-bisection problems. Further researches will be made on the algorithm to reveal more theoretical properties and characters of the algorithm.

## 5. Conclusion

A feasible direction algorithm without line searches are proposed for the solution of the NP-hard max-bisection problems. NCP functions are employed to convert max-bisection problems into continuous nonlinear programming problems. Solving the resulting continuous nonlinear programming problem generates a solution that gives an upper bound on the optimal value of the max-bisection problem. From the solution, the greedy strategy is used to generate a satisfactory approximate solution (generally optimal) of the max-bisection problem. A feasible direction method without line searches is proposed to solve the resulting continuous nonlinear programming, and the convergence of the algorithm to KKT point of the resulting problem is proved. Since no line searches and no computations on matrices are required in the iteration of the algorithm, the CPU time of the algorithm to reach the termination is less. Numerical experiments and comparisons on well-known test problems existing in literatures, and on randomly generated test problems show that the proposed method is robust, and very efficient.

## Appendix A: Test problems in the first group

• **Problem 1** A graph from Rendle and Wolkowicz[12].

Table 6: Edge set from Rendle and Wolkowicz[12]

| node | Connections to |
|---|---|
| 1 | 7,12,13,14,15,16,17 |
| 2 | 12,17,18,20 |
| 3 | 5,11,13,14,18,19,20 |
| 4 | 6,9 |
| 5 | 7,9,10,12,16,19 |
| 6 | 16,18,20 |
| 7 | 8,9,11,16 |
| 8 | 15,18 |
| 9 | 11,15,19 |
| 11 | 14,17,18,20 |
| 12 | 14 |
| 13 | 18,20 |
| 14 | 16,18,20 |
| 16 | 18 |
| 17 | 18 |
| 18 | 20 |

• **Problem 2** A graph from Cullum et al.[13].

Table 7: Edge set from Cullum et al.[13]

| node | Connections to |
|---|---|
| 1 | 4,5,6,7,20 |
| 2 | 4,10,12,15,17 |
| 3 | 4,5,10,13,16,19 |
| 4 | 7,8,11,12 |
| 5 | 10,11,12 |
| 6 | 13,14,18 |
| 8 | 9,16,17 |
| 9 | 11,12,14 |
| 10 | 19,20 |
| 11 | 14,15,19,20 |
| 12 | 15,18,19 |
| 13 | 15,19 |
| 15 | 18 |
| 16 | 17 |
| 18 | 20 |

• **Problem 3** A path containing 20 nodes and 19 edges.
• **Problem 4** Two complement graphs of 10 nodes each connected one edge.
• **Problem 5** A six-star graph.
• **Problem 6** A graph from Barnes and Hoffman[14]

Table 8: A six-star graph

| node | Connections to |
|---|---|
| 1 | 2,5,6 |
| 2 | 3,6 |
| 3 | 4,6 |
| 4 | 5,6 |
| 5 | 6 |

Table 9: Edge set from Barnes and Hoffman[14]

| node | Connections to |
|---|---|
| 1 | 2,3,4,7,8,17 |
| 2 | 3,10,14,15,16 |
| 3 | 8,12,16 |
| 4 | 7,9,11,17 |
| 5 | 6,9,11,15,16,20 |
| 6 | 7 |
| 7 | 9,15,16 |
| 8 | 10,12,14,16,18 |
| 9 | 12,20 |
| 10 | 12,14,16,19 |
| 11 | 18,19,20 |
| 12 | 13,15 |
| 13 | 14,16,18,19 |
| 14 | 16,18,19 |
| 15 | 16,17,19 |
| 17 | 18 |

## References

[1] K. G. Murty and S. N. Kabadi, Some NP-complete problems in quadratic and nonlinear programming, *Mathematics programming*, **39** (1987), 117-129 .

[2] J. Hastad, Some optimal inapproachability results. In proceedings of the 29th Annual ACM symposium on the theory of computing, ACM, New York, (1997), 1-10.

[3] S. Arora, D. Karger and M. Karpinski, Polynomial time approximation schemes for dense instance of NP-hard problems. In proceedings of the 27th Annual ACM symposium on the theory of computing, ACM, New York, (1995), 284-293.

[4] K. Jansen,M. Karpinski and A. Lingas, A polynomial time approximation scheme for Max-Bisection on planer graphs. Electronic Colloquium on Computational Complexity, Report TR00-064, 2000.

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

[6] A. Frieze and M. Jerrum, Improved Approximation Algorithms for Max k-cut and Max Bisection, Proc.4th IPCO Conference(1995), 1-13.

[7] Y. Ye, A 0.699-approximation algorithms for Max-bisection, mathematical programming, 90(2001), 101-111.

[8] E. Halperin and U. Zwick, A unified framework for obtaining improved approximation algorithms for maximum graph bisection problems, *Random Struct Algor*, *20* (2002), 382-402.

[9] S. Poljak and F. Rendle, Solving the Max-cut Problem using Eigenvalues. *Discrete Applied Mathematics*, **62** (1994), 249-278.

[10] A. Fischer, A special Newton-type Optimization Method. *Optimization*, **24** (1992), 269-284.

[11] H. Liu, X. Wang, S. Liu, Feasible direction algorithm for solving SDP realxation of the quadratic -1,1 programming, *Optimization Methods and Software*, **19** (2004), 125-136.

[12] F. Rendle and H. Wolkowicz, A projection technique for partitioning the nodes of a graph, Technical Report 20, University of Waterloo, 1990.

[13] J. Cullum, W. E. Donath and P. Wolfe, The minimization of certain nondifferentiable sums of eigenvalues of symmetric matrices, *Mathematical Programming Study*, **3** (1975), 35-55.

[14] E. R. Barnes and A. J. Hoffman, Partitioning, Spectra and linear programming, Academic Press, U.S.A, 1984.

[15] F. Alizadeh, J. -P. Haeberly, M. V. Nayakkankuppam, M. L. Overton and S. Schmieta, user s guide -version 0.9Beta, Technical Report TR1997-737, Courant Institute of Mathematical Science, NYU, New York, NY, June 1997.