Journal of Computational Mathematics, Vol. 24, No. 6, 2006, 749–760.

AN EFFECTIVE CONTINUOUS ALGORITHM FOR APPROXIMATE SOLUTIONS OF LARGE SCALE MAX-CUT PROBLEMS [*1)]

Cheng-xian Xu, Xiao-liang He and Feng-min Xu
(*Department of Mathematics, Xi'an Jiaotong University, Xi'an 710049, China*)

Abstract

An effective continuous algorithm is proposed to find approximate solutions of NP-hard max-cut problems. The algorithm relaxes the max-cut problem into a continuous nonlinear programming problem by replacing n discrete constraints in the original problem with one single continuous constraint. A feasible direction method is designed to solve the resulting nonlinear programming problem. The method employs only the gradient evaluations of the objective function, and no any matrix calculations and no line searches are required. This greatly reduces the calculation cost of the method, and is suitable for the solution of large size max-cut problems. The convergence properties of the proposed method to KKT points of the nonlinear programming are analyzed. If the solution obtained by the proposed method is a global solution of the nonlinear programming problem, the solution will provide an upper bound on the max-cut value. Then an approximate solution to the max-cut problem is generated from the solution of the nonlinear programming and provides a lower bound on the max-cut value. Numerical experiments and comparisons on some max-cut test problems (small and large size) show that the proposed algorithm is efficient to get the exact solutions for all small test problems and well satisfied solutions for most of the large size test problems with less calculation costs.

Mathematics subject classification: 90C27.
Key words: Max-cut problems, Algorithm, Feasible direction method, Laplacian matrix, Eigenvectors.

1. Introduction

The max-cut problem is to partition the vertex set of an undirected graph, denoted by $G(V, E)$, into two parts in order to maximize the sum of the weights on the edges between these two parts, where V with $|V| = n$ is the set of n vertices and E the edge set of the graph. This problem has long been known to be NP-hard, and it is solvable in polynomial time only for some special classes of graphs [10]. Because of its theoretical and practical importance, and because efficient algorithms for NP-hard combinatorial optimization problems are unlikely to exist, many approximate algorithms (see [11],[15],[21],[23]) have been proposed to solve max-cut problems at an approximation factor ρ, that is, to find a *cut* $(S, \bar{S})$ such that $w(S, \bar{S}) \geq \rho w^*$, where S and $\bar{S} = V \backslash S$ denote the cut, $w(S, \bar{S})$ is the value of the *cut* $(S, \bar{S})$, w^* is the max-cut value, and ρ is generally called the performance guarantee of an algorithm. Among these approximate algorithms, the most famous is the randomization algorithm with performance guarantee $\rho = 0.87856$ proposed by Goemans and Williamson [9]. The algorithm relaxes each binary variable in $\{-1, 1\}$ to a unit vector in space R^n to form a semi-definite programming

[*] Received November 30, 2005.
[1)] This work is supported by National Natural Science Foundation of China at 10231060.

problem, hence increasing the problem dimension from n to $n \times n$, and the resulting SDP problem is then solved using any existing semi-definite programming algorithms, for example, interior algorithms. Then an approximate solution to the max-cut problem is generated from the optimal solution of the relaxed SDP problem using a randomization algorithm. Although extremely interesting because Goemans and Williamson's algorithm has the best worst case performance guarantee, it is of complex design and its computation time may prohibitive from large scale max-cut problems [7]. For solving large scale max-cut problems, some nonlinear programming methods are proposed in [6],[7],[12],[18]. The strengthened semi-definite programming relaxation [4] and the rank two relaxation [7] of max-cut problems are modifications of Goemans and Williamson's work. The algorithm in [22] generates an approximate solution to the max-cut problem by minimizing the largest eigenvalue of the matrix that is the sum of the Laplacian matrix of the graph and a variable diagonal matrix. Since the algorithm calculates the largest eigenvalues of a sequence of given matrices satisfying the constraints and the objective function in minimization is not differentiable everywhere, it is of complex design and not applicable for the solution of large scale max-cut problems.

In this paper, we present an effective continuous algorithm for approximate solutions of large scale max-cut problems. The algorithm relaxes the max-cut problem into a continuous nonlinear programming problem that finds the largest eigenvalue of the Laplacian matrix of the underlying graph by maximizing a convex quadratic function subject to a single constraint. The constraint restricts the length of the variable vectors. An efficient feasible direction method is used to perform the maximization of the resulting nonlinear programming problem. The method only employs the gradient evaluations of the objective function and no any matrix calculations and no line searches are required. This greatly reduces the calculation cost in the implementation of the algorithm and increases the efficiency, and makes the algorithm applicable to large scale max-cut problems. The convergence of the feasible direction method to KKT points of the nonlinear programming is proved. If the solution obtained by the feasible direction method is a global solution of the resulting nonlinear programming, the solution provides an upper bound on the optimal value of the max-cut. A feasible solution to the max-cut problem can then be generated from the solution of the nonlinear programming, and provides a lower bound for the max-cut value. Numerical experiments and comparisons on some well-known max-cut test problems (small size) and on some large size problems that are randomly generated by the procedure **rudy** are made to show the efficiency of the proposed method on both the computation time and resulting solutions.

Let $w_{ij} = w_{ji}$ be the weight on edge $e_{ij} \in E$ of a graph $G(V, E)$, where $w_{ij} = 0$ if there is no edge connecting vertices V_i and V_j. Using the Laplacian matrix of the graph $L = \frac{1}{4}(\mathrm{Diag}(We) - W) = (L_{ij})_{n\times n}$ with weight matrix $W = (w_{ij})_{n\times n}$, the max-cut problem can be expressed as

$$(\mathrm{MC}) : \begin{cases} \mathrm{Max} & x^T L x \\ s.t. & x_i^2 = 1, \quad i = 1, \cdots, n, \end{cases}$$

where

$$L_{ij} = \begin{cases} -w_{ij}, & i \neq j, \\ \sum_{k\neq i}^{n} w_{ik}, & i = j, \end{cases}$$

The Laplacian matrix L is positive semi-definite. The constraints in (MC) restrict each variable taking values either 1 or -1, and hence it is a combinatorial optimization problem. Goemans and Williamson in [9] relaxe the problem to formulate a semi-definite programming problem by

replacing each binary variable x_i with one unit vector $v_i \in R^n$ and the scalar product $x_i x_j$ by inner product $v_i^T v_j$

$$(\text{SDP}) \quad \begin{cases} \text{Max} & L \bullet V \\ \text{s.t.} & diag(V) = e \\ & V \succeq 0, \end{cases}$$

where $L \bullet V = \sum_{i,j=1}^{n} L_{ij} V_{ij}$, $V = [v_1, \cdots, v_n]$ and $V \succeq 0$ means V is positive semi-definite, e is the vector of all ones. The rank-two algorithm in [7] relaxes the max cut problem to form an unconstrained optimization problem by replacing each binary variable x_i with one unit vector in space R^2 and then using polar coordinates

$$(\text{PMC}) \quad \text{Min} \;\; f(\theta) = \frac{1}{2} W \bullet cos(T(\theta)), \;\; \forall\, \theta \in R^n,$$

where $\theta = (\theta_1, \cdots, \theta_n)^T$, $\theta_i \in [0, 2\pi]$, $i = 1, 2, \cdots, n$, $T(\theta)$ is a sckew-symmetric matrix with entries

$$T_{ij} = \theta_i - \theta_j, \;\; \forall\, i, j = 1, 2, \cdots, n$$

Poljak and Rendle in [22] show that

$$(\text{EMC}) \;\; \varphi^* = \text{Min}\, \{\varphi(u) = n\lambda_{max}(L + diag(u)) | e^T u = 0\}$$

provides an upper bound on the max-cut value w^* and proposes an algorithm to find the eigenvalue bound by minimizing the function $\varphi(u)$ for all $u \in \{u | e^T u = 0\}$, where $\lambda_{max}(M)$ denotes the largest eigenvalue of the symmetric matrix M, $diag(u)$ is the diagonal matrix with $u_i, i = 1, 2, \cdots, n$ as diagonal entries. The algorithm needs to calculate the largest eigenvalue $\lambda_{max}(L + diag(u))$ of the matrix $L + diag(u)$ and the corresponding eigenvector x for any given vector u satisfying $e^T u = 0$ and then minimizing the function $\varphi(u)$. Since the function $\varphi(u)$ is not differentiable everywhere, a subgradient method is employed to implement the minimization.

The rest of the paper is organized as follows. The continuous relaxation of the max-cut problem is presented in section 2. The relaxation is obtained by replacing the n constraints in problem (MC) using one single continuous constraint. The resulting nonlinear programming problem is equivalent to find the largest eigenvalue and the corresponding eigenvector of the Laplacian matrix L. The feasible direction method for the solution of the relaxed continuous nonlinear programming problem is described in Section 3. The method is motivated from the characteristic of the optimal solution of the nonlinear programming. The convergence properties of the feasible direction method are then analyzed. A neighborhood search strategy is proposed in section 4. This strategy is designed to improve the approximate solution generated from the solution of the nonlinear programming problem. Numerical experiments and comparisons with the GW and rank two algorithms on some standard max-cut test problems and on some randomly generated large scale max-cut problems are reported in Section 5. Conclusions and extensions are given in section 6.

2. The Relaxed Continuous Model

In this section we relax the max-cut problem (MC) into a continuous nonlinear programming problem. It can be understood that optimal solutions of problem (MC) will not change if we

replace the positive semi-definite matrix L by a positive definite matrix $L+\sigma I$ where $\sigma > 0$ is a constant. Therefore, in the following we will assume that the matrix L is positive definite, and hence the gradient $g = 2Lx$ of the objective function in (MC) will not be zero for any $x \neq 0$.

Let $F_1 = \{x | x_i^2 = 1, i = 1, 2, \cdots, n\}$ be the feasible set of problem (MC). It is clear that any point in set F_1 satisfies the single constraint $\|x\|^2 = n$. Therefore, the continuous model of the max-cut problem (MC) can be obtained by relaxing the constraints on the binary variables $x_i, i = 1, 2, \cdots, n$ using the single continuous constraint, that is, the continuous model has the form

$$(\text{NLP}) : \begin{cases} \text{Max} & f(x) = x^T L x \\ s.t. & \|x\|^2 = n. \end{cases}$$

The feasible set F_1 of problem (MC) is a subset of the feasible region $F_2 = \{x | \|x\|^2 = n\}$ in problem (NLP), and hence if $x^{(1)}$ is a global solution of problem (NLP), then the function value $f(x^{(1)})$ gives an upper bound on the max-cut value, that is,

$$w^* \leq f(x^{(1)}) = (x^{(1)})^T L x^{(1)},$$

where w^* denotes the max-cut value of the graph $G(V, E)$. It follows from the KKT condition of the problem (NLP) that there exists a Lagrangian multiplier λ^* at the solution $x^{(1)}$ such that

$$Lx^{(1)} = \lambda^* x^{(1)}$$

holds. That is, λ^* is an eigenvalue of the matrix L and $x^{(1)}$ is the corresponding eigenvector. This implies that eigenvectors of the matrix L are KKT points of problem (NLP). If $x^{(1)}$ is a global solution of problem (NLP), then λ^* is the largest eigenvalue $\lambda_{max}(L)$ of the matrix L and we have

$$w^* \leq (x^{(1)})^T L x^{(1)} = n\lambda_{max}(L).$$

The feasible direction method described in the next section is used to find the largest eigenvalue and the corresponding eigenvector of the matrix L that satisfies the constraint in problem (NLP). It can be observed that problem (NLP) is a special case of the problem (EMC) with $u = 0$. Most of the existing algorithms (see [9],[7],[22]) attempt to find an upper bound of a given max-cut problem and then generate an approximate solution to the max-cut from the upper bound. The problem (NLP) will generate upper bounds not so good as that given by the problem (EMC), but it greatly reduces the calculation cost to find an approximate solution of the max-cut problem, and very efficient method can be designed to generate the solution of problem (NLP).

Let $x^{(1)}$ be the solution of problem (NLP). In general the entries of the solution $x^{(1)}$ will not exactly equal to 1 or -1, that is, $x^{(1)}$ is not feasible to problem (MC). The feasible point of the max-cut problem that is closest to $x^{(1)}$ will be selected as an approximation to the max-cut solution. It is the point $\hat{x}^{(1)} = sign(x^{(1)})$ with $\hat{x}_i^{(1)} = sign(x_i^{(1)}), i = 1, 2, \cdots, n$.

3. The Feasible Direction Method for the Solution of Problem (NLP)

In this section, a feasible direction method without line searches is presented for the solution of problem (NLP). The method employs only gradient evaluations of the objective function in problem (NLP), and no calculations on any matrices and no line searches, thus greatly reduces the calculation costs and increases the efficiency of the method. In the following it will be

assumed that there exists no any isolated vertex or subgraph in a given graph $G(N, E)$, that is, the given undirected graph is connected.

Analysis in the previous section shows that optimal solutions of problem (NLP) can be found from the eigenvectors of the matrix L that satisfy the constraint in problem (NLP). This motivates the following iteration

$$x^{(k+1)} = \sqrt{n}\frac{g^{(k)}}{\|g^{(k)}\|} = \sqrt{n}\frac{Lx^{(k)}}{\|Lx^{(k)}\|},\ k = 1, 2, \cdots \tag{3.1}$$

to get an optimal solution of the equality constrained nonlinear programming problem (NLP). The iteration (3.1) 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 iterate;

(2) The point $x^{(k+1)}$ is feasible to problem (NLP), that is, $\|x^{(k+1)}\|^2 = n$.

(3) If the sequence $\{x^{(k)}\}$ converges to x^*, then x^* is feasible to problem (NLP) , and we have

$$x^* = \lim_{k\to\infty} x^{(k+1)} = \lim_{k\to\infty} \sqrt{n}\frac{Lx^{(k)}}{\|Lx^{(k)}\|} = \sqrt{n}\frac{Lx^*}{\|Lx^*\|}.$$

This indicates that x^* is an eigenvector of the matrix L that satisfies the constraint in problem (NLP).

Define

$$d^{(k)} = \sqrt{n}\frac{g^{(k)}}{\|g^{(k)}\|} - x^{(k)}$$

as a search direction. Then the iteration (3.1) can be written as

$$x^{(k+1)} = x^{(k)} + d^{(k)}. \tag{3.2}$$

The following lemmas show that if $d^k = 0$, then x^k is a KKT point of problem (NLP), hence an eigenvector of the Laplacian matrix L, and if $d^k \neq 0$, then $x^k + d^k$ is feasible to problem (NLP) and increases the function value.

Lemma 3.1. *If $d^{(k)} = 0$, then $x^{(k)}$ is an eigenvector of the matrix L that satisfies the constraint in problem* (NLP)*, that is, $x^{(k)}$ is a KKT point of* (NLP)*.*

Proof. It is clear that $x^{(k)}$ satisfies the constraint in problem (NLP). Since $d^{(k)} = \sqrt{n}\frac{g^{(k)}}{\|g^{(k)}\|} - x^{(k)} = 0$ and $\|g^{(k)}\| = \|2Lx^{(k)}\| \neq 0$, we have $Lx^{(k)} - \frac{\|g^{(k)}\|}{2\sqrt{n}}x^{(k)} = 0$. Thus, $x^{(k)}$ is an eigenvector of the matrix L and satisfies the constraint in problem (NLP). This completes the proof of the lemma.

Lemma 3.2. *Suppose $d^{(k)} \neq 0$, then $x^{(k+1)} = x^{(k)} + d^{(k)}$ is feasible to problem* (NLP)*, and*

$$\|g^{(k)}\|\|d^{(k)}\| + \lambda_1\|d^{(k)}\|^2 \geq f(x^{(k+1)}) - f(x^{(k)}) \geq \lambda_n\|d^{(k)}\|^2 > 0,$$

where λ_n and λ_1 denote the smallest and the largest eigenvalues of the matrix L.

Proof. The feasibility of the iterate $x^{(k+1)}$ directly comes from the definition (3.1). Using the fact that $\|x^{(k)}\| = \sqrt{n}$, we have

$$(g^{(k)})^T d^{(k)} = (g^{(k)})^T(\sqrt{n}\tfrac{g^{(k)}}{\|g^{(k)}\|} - x^{(k)}) = \|g^{(k)}\|\|\ x^{(k)}\| - (g^{(k)})^T x^{(k)} \geq 0.$$

Since

$$f(x^{(k+1)}) = f(x^{(k)} + d^{(k)}) = f(x^{(k)}) + (g^{(k)})^T d^{(k)} + (d^{(k)})^T L d^{(k)},$$

the conclusion comes from the positive definiteness of the matrix L.

Lemma 3.2 indicates that if $d^{(k)} \neq 0$ for all $k = 1, 2, \cdots$, then $\{f(x^{(k)})\}$ is a monotonically increasing sequence, that is, the iteration (3.1) generates iterates with monotonically increasing objective function values.

The following theorem gives the convergence of the feasible direction method to eigenvalues of the matrix L that satisfies constraint in problem (NLP).

Theorem 3.3. *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 constraint of problem* (NLP)*, that is, x^* is a KKT point of* (NLP).

Proof. $\|d^{(k)}\| \to 0$ imply $\|x^{(k+1)} - x^{(k)}\| \to 0$. Let x^* be an accumulation point of the sequence $\{x^{(k)}\}$. Without loss of generality, we assume that $x^{(k)} \to x^*$, then x^* is feasible to problem (NLP). It follows from the definition of $d^{(k)}$, and the continuity of the gradient $g(x) = 2Lx$, we have

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

That is,

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

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

Theorem 3.4. *If $d^{(k)} \neq 0$ for all $k > 0$, then $\|d^{(k)}\| \to 0$.*

Prof. From Lemma 3.2 and monotonically increasing of the sequence $\{f(x^{(k)})\}$, for any $K > 0$ we have

$$\begin{aligned}
\sum_{k=0}^{K} \|d^{(k)}\|^2 &\leq \frac{1}{\lambda_n}\sum_{k=0}^{K}(f(x^{(k+1)}) - f(x^{(k)})) \\
&= \frac{1}{\lambda_n}[f(x^{(K)}) - f(x^{(0)})] \\
&\leq \frac{1}{\lambda_n}(x^{(K)})^T L x^{(K)} \\
&\leq \frac{\lambda_1}{\lambda_n}\|x^{(K)}\|^2 \\
&\leq \frac{\lambda_1}{\lambda_n} n,
\end{aligned}$$

where we use the fact $\|x^{(k)}\| = \sqrt{n}$. This shows that $\sum_{i=0}^{+\infty} \|d^{(k)}\|^2$ is convergent, and hence $\|d^{(k)}\| \to 0$ holds.

Based on the conclusion of Theorem 3.4, the condition $\|d^{(k)}\| \leq \epsilon$ and/or $f(x^{(k+1)}) - f(x^{(k)}) \leq \epsilon$ will be used to terminate the iteration in the implementation of the method.

Since the objective function in problem (NLP) is convex, there is no guarantee that the solution generated from the feasible direction method is a global solution (largest eigenvalue) of problem (NLP). A local search strategy (called neighborhood search) will be employed after a solution is obtained with the attempt to improve the solution. Global strategies such as branch and bound method will be further studied to generate a global solution. However, numerical experiments in section 5 show that the proposed algorithm generates the global max-cut for all small size test problems and satisfaction approximate solutions for large size test problems.

4. Neighborhood Search

In this section, we will omit the superscript $^{(k)}$ in all the feasible point $x^{(k)}$ for convenience. After a feasible point $\hat{x}$ to the max-cut problem (MC) is obtained from the solution x of problem (NLP), the neighborhood search strategy is employed to find weather $\hat{x}$ is a local maximum or not. A $cut(S, \bar{S})$ with $\bar{S} = V \backslash S$ is said to be a local maximum in a neighborhood when

$$w(S, \bar{S}) \geq max\{w(S \bigcup \{i\}, \bar{S} \backslash \{i\}), w(S \backslash \{i\}, \bar{S} \bigcup \{i\}), i = 1, 2, \cdots, n\}.$$

The neighborhood of the point $\hat{x}$ consists of n points, and each point is obtained from $\hat{x}$ by changing its one variable value from $\hat{x}_i$ to $-\hat{x}_i$ and $i = 1, 2, \cdots, n$. Let $\hat{x}(j)$ be the neighbor point that is obtained by changing $\hat{x}_j$ to $-\hat{x}_j$. Then the change of the function value from point $\hat{x}$ to point $\hat{x}(j)$ is

$$\begin{aligned} \varpi(j) &= \hat{x}(j)^T L \hat{x}(j) - \hat{x}^T L \hat{x} \\ &= \sum_{i \neq j, k \neq j}^{n} \hat{x}_i L_{ik} \hat{x}_k - 2 \sum_{i=1, i \neq j}^{n} \hat{x}_i L_{ij} \hat{x}_j - \sum_{i=1, k=1}^{n} \hat{x}_i L_{ik} \hat{x}_k \\ &= -4 \sum_{i=1, i \neq j}^{n} \hat{x}_i L_{ij} \hat{x}_j. \end{aligned}$$

If $\varpi(j) > 0$, it means changing the value of the variable from $\hat{x}_j$ to $-\hat{x}_j$ will increase the value of the function, and $\hat{x}(j)$ is a point better than $\hat{x}$ is. If $\varpi(j) \leq 0$ holds for all $j = 1, 2, \cdots, n$, then $\hat{x}$ is the local maximizer and will be accepted as an approximation to the max-cut.

Let $\varpi(k) > 0$ and $\hat{x}(k)$ is the best solution obtained at some stage of the neighborhood search. In order to continue the neighborhood search, we need to update the values of $\varpi(j)$ for all $j = 1, 2, \cdots, n$. This can be done by

$$\varpi(j) \Leftarrow \begin{cases} \varpi(j) + \varpi(k) - 8\hat{x}_j L_{jk} \hat{x}_k, & j \neq k, \\ -\varpi(j), & j = k, \end{cases} \tag{4.1}$$

With the updated values of $\varpi(j)$, $j = 1, 2, \cdots, n$, the neighborhood search can be continued at the new point. The following is the neighborhood search procedure.

Neighborhood Search Method

Step 1: From given $\hat{x}$, set $\underline{f} = f(\hat{x})$, and calculate $\varpi(j)$ for $j = 1, 2, \cdots, n$;

Step 2: If $\varpi(j) \leq 0$ for all $j = 1, 2, \cdots, n$, then $\bar{x} = \hat{x}$ and stop; Otherwise select

$$k = argmax\{\varpi(j) | j = 1, 2, \cdots, n\};$$

Step 3: Set $\underline{f} = \underline{f} + \varpi(k)$, and update $\varpi(j)$, $j = 1, 2, \cdots, n$ as in (4.1), and then go to Step 2;

Numerical experiments show that the local search is generally stopped at Step 2 of the first round on most of the test problems, that is, the approximate feasible point $\hat{x}^{(k)}$ obtained by the proposed feasible direction method are the local maximums.

5. Implementation

In this section we present the implementation of the algorithm. The algorithm is programmed in Matlab 6.0, and experiments are implemented in PC Ienovo 3422 (Pentium 256M

DDR). Since the algorithm attempts to find an approximate solution of the underlying graph and experiments show that the value of ϵ in termination is not crucial, and hence the value $\epsilon = 0.0001$ is used in the conditions

$$\|d^k\| \leq \epsilon \qquad \text{or} \qquad f(x^{k+1}) - f(x^k) \leq \epsilon,$$

that are used to terminate the iteration of the feasible direction method. The value $\sigma = 10$ is used to make sure $L + \sigma I$ is positive definite, but the objective function value at each iteration point is still calculated as $(x^k)^T L x^k$ in the implementation. The initial points for all test problems are randomly generated by $x^0 = sign(rand(n, 1))$ which satisfies $\|x^0\| = \sqrt{n}$, and is feasible to problem (NLP).

Test problems: The first set of test problems consists of five small size max-cut test problems that are widely used in literature [3]. These are:

1. C5: an unweighted graph of 5-cycle with unit edge weights.
2. K5: a complete unweighted graph of 5 nodes with unit weights on all edges.
3. KA5: a complete graph of 5 nodes with weights given by the weight matrix

$$W(G_3) = \begin{bmatrix} 0 & 1.52 & 1.52 & 1.52 & 0.16 \\ 1.52 & 0 & 1.60 & 1.60 & 1.52 \\ 1.52 & 1.60 & 0 & 1.60 & 1.52 \\ 1.52 & 1.60 & 1.60 & 0 & 1.52 \\ 0.16 & 1.52 & 1.52 & 1.52 & 0 \end{bmatrix}.$$

4. AW_9^2: the antiweb unweighted graph of 9 nodes with unit edge weights (see [1]).
5. BG: a graph of 12 nodes with weight matrix given by

$$W(G_5) = \begin{bmatrix} 0 & 2 & 2 & 0 & 2 & 4 & 0 & 2 & 2 & 2 & 2 & 2 \\ 2 & 0 & 0 & 2 & 4 & 2 & 2 & 0 & 2 & 2 & 2 & 2 \\ 2 & 0 & 0 & 0 & 4 & 4 & 2 & 4 & 4 & 0 & 2 & 2 \\ 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 4 & 2 & 2 \\ 2 & 4 & 4 & 0 & 0 & 2 & 2 & 2 & 2 & 4 & 2 & 4 \\ 4 & 2 & 4 & 2 & 2 & 0 & 2 & 0 & 2 & 2 & 2 & 2 \\ 0 & 2 & 2 & 0 & 2 & 2 & 0 & 0 & 0 & 4 & 2 & 2 \\ 2 & 0 & 4 & 0 & 2 & 0 & 0 & 0 & 0 & 4 & 4 & 2 \\ 2 & 2 & 4 & 2 & 2 & 2 & 0 & 0 & 0 & 2 & 2 & 2 \\ 2 & 2 & 0 & 4 & 4 & 2 & 4 & 4 & 2 & 0 & 4 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 4 & 2 & 4 & 0 & 4 \\ 2 & 2 & 2 & 2 & 4 & 2 & 2 & 2 & 2 & 2 & 4 & 0 \end{bmatrix}.$$

The second set of tests contains 15 randomly generated large size test problems with nodes from 800 to 3000 and positive weights on all edges. These graphs are generated by the procedure **rudy**, a machine independent graph generator written by G. Rinaldi (see [14]). Table 5.1 contains the information of these graphs, where Range of Weight indicates the value on each edge of a given graph. 1 means it is an unweighted graph, that is, the weight on each edge is 1, and (-1,1) means the weight on each edge is randomly generated in the range (-1,1).

Table 5.1 Details about test graphs in second group

problem	No. of Nodes	Density of Edges	Type	Range of Weight
G11	800(100× 8)	1600 Edges	Toroidal-grid-2D	(-1,1)
G12	800(50 × 16)	1600 Edges	Toroidal-grid-2D	(-1,1)
G13	800(25 × 32)	1600 Edges	Toroidal-grid-2D	(-1,1)
G14	800	4694 Edges	Unweighted Planar	1
G15	800	4661 Edges	Unweighted Planar	1
G22	2000	19990 Edges	Unweighted	1
G23	2000	19990 Edges	Unweighted	1
G24	2000	19990 Edges	Unweighted	1
G32	2000(100 × 20)	4000 Edges	Toroidal-grid-2D	(-1,1)
G33	2000(80 × 25)	4000 Edges	Toroidal-grid-2D	(-1,1)
G34	2000(50 × 40)	4000 Edges	Toroidal-grid-2D	(-1,1)
G38	2000	99%	Unweighted Planar	1
G44	1000	2%	Unweighted	1
G50	3000(25 × 120)	6000 Edges	Toroidal-grid-2D	1
G52	1000	100%	Unweighted Planar	1

Results Table 5.2 gives the numerical results of the algorithm on the test problems in the first set. We do not present the comparison of the proposed algorithm with the other existing algorithms since most of the existing algorithms such as GW

Table 5.2 Computational Results for the first set

Problem	μ^*	f^*	CPU (Sec)
C5	4	4	0.01
K5	6	6	0.001
KA5	9.28	9.28	0.001
AW_9^2	12	12	0.01
BG	88	88	0.001

randomization algorithm [9], rank-two relaxation algorithm [7] and the eigenvalue upper bound algorithm [22] can find the global solution and the calculation times are not serious for all these small test problems. In the table the first column gives the problem names, μ^* gives the max-cut values of these test problems, f^* the objective values of these test problems generated by the proposed algorithm in this paper, CPU the CPU time of the algorithm implementation to achieve the value f^*. It can be observed from table 5.2 that the proposed continuous algorithm finds the global solution for all these five test problems with a little calculation costs. It has been observed in the experiments that the sequence of objective function values $\{f^{(k)}\}$ monotonically increases and converges to f^*, and the sequence $\|d^{(k)}\|$ converges to zero very quickly.

Table 5.3 gives the results and comparisons with the GW randomization algorithm [9] and the rank-two algorithm [7] on 15 large size test problems in the second set. The results of the GW and rank-two algorithms are reported in [7] while the results with star $*$ in WG-cut column are reported in [2] since these results are not available in [7]. The GW randomized approximation algorithm uses the dual-scaling interior point algorithm and an iterative linear equation solver in a code DSDP [8]. It is currently one of the fastest interior-point codes for solving SDP problems. The solution of the SDP problem gives an upper bound to the optimal value of the test problem, denoted by SUB in the table. Then the randomization procedure generates an approximate solution to the max-cut from the resulting solution of the SDP problem, which

generates a lower bound on the optimal value of the max-cut. The rank-two algorithm uses a subgradient method to minimize the resulting nonlinear programming problem (PMC), and then a Procedure-Cut is used to generate an approximate solution from the resulting solution of nonlinear programming. In the table, the columns headed with GW-Cut, RT-Cut and EC-Cut present the approximate values to the max-cut generated by the GW algorithm, Rank-two algorithm and the effective continuous algorithm, respectively. The columns headed with CPU are implementation times used by these three algorithms to achieve the approximate solutions. It is important to note that the timing for GW algorithm and the rank-two algorithm were obtained on a HP 9000/785/C3600 machine with a 367 MHZ processor and a SGI Original2000 machine, respectively, while our results in timing are obtained on a personal computer Ienovo 3422. It can be understood that the PC Ienovo 3422 is not comparable with both the machines, and we list the timing for these three algorithms to help understanding the efficiency of the effective continuous algorithm. The following observations can be made based on the results in the table.

Table 5.3 Comparisons on large size test problems

Problem	Size	SUB	GW		RT		EC	
			GW-Cut	CPU	RT-Cut	CPU	EC-Cut	CPU
G11	800	629	542	16.6	524	0.06	542	0.33
G12	800	624	540	17.7	512	0.06	532	0.34
G13	800	647	564	18.2	536	0.06	554	0.48
G14	800	3192	2922	35.2	3016	0.09	2941	1.33
G15	800	3172	2938	32.1	3011	0.09	2944	0.8
G22	2000	14136	12960	4123.3	13148	0.36	13148	1.74
G23	2000	14146	13006	3233.5	13197	0.37	13148	4.8230
G24	2000	14141	12933	3250.7	13195	0.30	13236	2.45
G32	2000	1568	1338	142.6	1306	0.18	1338	5.43
G33	2000	1544	1330	132.5	1290	0.14	1325	2.54
G34	2000	1547	1334	156.7	1276	0.12	1292	2.34
G38	2000	8015	7037*				7341	1.843
G44	1000	7028	6170*				6423	1.332
G50	3000	5988	5880	264.6	5748	0.17	5803	12.76
G52	1000	4009	3520*				3698	0.49

(1) The GW method is not effective for solving large size max-cut problems, especially for problems with a large number of edges, since it increases the dimension of a problem from n to n^2, while both the rank-two and effective continuous algorithms remain the problem dimension in n.

(2) Although, the timing of both the rank-two algorithm and the effective continuous algorithm are not comparative because of large difference between machines implementing the calculation, the results of the effective continuous algorithm indicate the efficiency of the algorithm for the solution of large scale max-cut problems.

(3) The GW algorithm provides the currently best conclusion on its performance guarantee in theory, but the approximate solutions obtained by the effective continuous algorithm and the rank-two algorithm are better than the solutions generated by the GW algorithm in most cases. The reason is that the upper bound obtained from the GW algorithm is closer to the value of the max-cut than the upper bounds obtained in both the rank-two and effective continuous algorithms. But better upper bounds do not imply better solutions.

(4) It is possible for both the rank-two and effective continuous algorithms to improve the

approximate solutions for large scale max-cut problems by running the algorithm several times starting from different initial points in a code, that is, to get better solutions by increasing some calculation cost since the time to running the algorithm once takes less times, while this procedure is not possible for GW algorithm because of its calculation cost of each time.

6. Conclusions and Extensions

An effective continuous algorithm is proposed to find approximate solutions of NP-hard max-cut problems. The algorithm relaxes a max-cut problem into a continuous nonlinear programming problem by using one single continuous constraint to replace the n discrete constraints in the original problem. The nonlinear programming problem is to find the largest eigenvalue of the Laplacian matrix of the underlying graph. Then an approximate solution to the max-cut problem is generated from the eigenvector corresponding to the largest eigenvalue. An feasible direction method is designed to solve the resulting nonlinear programming problem. The method employs only the gradient evaluations of the objective function in the nonlinear programming problem, and no any matrix calculations and no line searches are required. This greatly reduces the calculation cost of the method to achieve the solution, and is suitable to the solution of large size max-cut problems. The convergence properties of the proposed method to a KKT point of the nonlinear programming are analyzed. If the solution obtained by the proposed algorithm is a global solution of the nonlinear programming problem, the solution will provide an upper bound on the max-cut value. The approximate solution generated from the solution of the nonlinear programming provides a lower bound on the max-cut value. Numerical experiments and comparisons with Goemans and Williamson's randomization algorithm and rank-2 algorithm on some max-cut test problems (small size and large size) are performed, and the results show that the proposed algorithm is efficient to get the exact solutions for all small test problems and well satisfied solutions for most of the large size test problems with less calculation costs. This shows that the proposed algorithm is suitable for the solution of large size max-cut problems.

Further researches on the proposed algorithm are required to refine the algorithm in theory and in the implementation. Though the numerical results of the algorithm are good enough, theoretical works are required to show the performance guarantee. We hope an acceptable performance guarantee can be proved. Furthermore, since the solution generated may not be a global solution of the max-cut, strategies need to be create to ensure a global solution. These include branch-and-bound approaches and running the algorithm many times starting from different initial points.

References

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

[2] Alperin H. and Nowak I., Lagrangian smoothing heuristics for max-cut, *Journal of Heuristics*, **11**:5-6 (2005), 447-463, 2005.

[3] Anjos, M., New convex relaxation for maximum cut and VLSI layout problems, A Thesis presented to the university of Waterloo. Waterloo, Ontario, Canada, 2001.

[4] Anjos, W. and Wolkowicz, H., A strengthened SDP relaxation via a second lifting for the Max Cut problem, Technical Report Research Report CORR 95-55, University of Waterloo, Waterloo, Onratio, 1999.

[5] Barahona F. and Anbil R., The volume algorithm: producing primal solutions with a subgradient method, *Mathematical Programming*, **87** (2000), 385-399.

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

[7] Burer, S., Monteiro, R.D.C., and Zhang, Y., Rank-two relaxation heuristics for MAX-CUT and other binary quadratic programs, *SIAM J. on Optimization*, **12** (2001), 503-521.

[8] Choi, C. and Ye, Y., Solving sparse semidefinite programs using the dual scaling algorithm with an iterative solver, Working paper, Department of Management Science, University of Iowa, IA, 2000.

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

[10] Hadlock F., Finding a maximum cut of a planar graph in polynomial time, *SIAM J. Comput.*, **4** (1975), 221-225.

[11] Haglin, D.J. and Venkatesan, S.M., Approximation and intractbility results for the maximum cut problem and its variants, *IEEE Trans Comput*, **40** (1991), 110-113.

[12] Han, Q., Ye, Y., and Zhang, J., An improved rounding method and semidefinite programming relaxation for graph partition, *Mathematical Programming*, **92** (2000), 509-535.

[13] Helmberg F. and Rendle F., Solving (0-1) quadratic problem by semi-definite programming and cutting plane, *Mathematical Programming*, **82** (1998), 291-315.

[14] Helmberg, C. and Rendle, F., A spectral bundle method for semidefinite programming, *SIAM J. Optimization,* **10** (2000), 673-695.

[15] Hofmeister, T. and Lefmann, H. A combinatorial design approach to MAXCUT, Proceedings of 13th Symposium on Theoretical Aspects of Computer Science, 1995.

[16] Huang, Z.H., Han, J., Xu, D. and Zhang, L. The non-interior continuation methods for solving the P_0-function nonlinear complementarity problem, *Science in China*, **44** (2001), 1107-1114.

[17] Huang, Z.H., Zhang, L. and Han, J., A hybrid smoothing-nonsmooth Newton-type algorithm yielding an exact solution of the P_0-LCP, *Journal of Computational Mathematics*, **22** (2004), 797-806.

[18] Liu, H., Wang, S. and Liu, S., Feasible direction algorithm for solving SDP relaxation of the quadratic $\{-1, 1\}$ programming, *Optimization Methods and Software*, **19** (2004), 125-136.

[19] Mitchell J E., Computational experience with an interior point cutting plane algorithm, *SIAM J. Optimization*, **10** (2000), 1212-1227.

[20] Mitchell J E., Restarting after branching in the SDP approach to max-cut and similar combinatorial optimization problems, *Journal of Combinatorial Optimization*, **5** (2001), 151-166.

[21] Poljak, S. and Turzik, D., A polynomial algorithm for constructing a large bipartite subgraph with an application to a satisfiability problem, *Can. J. Math.*, **34** (1982), 519-524.

[22] Poljak S and Rendle F., Solving the max-cut problem using eigenvalues, *Discret Applied Mathematics*, **62** (1995), 249-278.

[23] Vitanyi, P.M., How well can a graph be n-colored? *Discr. Math.*, **34** (1981), 69-80.

[24] Xu, D., Han, J., Huang, Z.H. and Zhang, L., Improved approximation algorithms for MAX n/2-DIRECTED-BISECTION and MAX n/2-DENSE-SUBGRAPH, *Journal of Global Optimization*, **27** (2003), 399-410.

[25] Xu, D., Ye, Y. and Zhang, J., Approximating the 2-catalog segmentation problem using semidefinite programming relaxations, *Optimization Methods and Software*, **18** (2003), 705-719.