

Project Report
SF2827 Topics in Optimization
A Randomized Algorithm for MAX CUT

Tove Odland
840206-0423
odland@kth.se

March 31, 2009

Abstract

This report constitutes a close review of Williamson and Goemans' randomized approximation algorithm for MAX CUT presented in the article *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, *J. ACM*, 1995. For MAX CUT they prove a performance guarantee of 0.87856. Prior to the appearance of their algorithm the best known performance guarantee was $\frac{1}{2}$. This report has also been greatly inspired by Laurent and Rendl's review *Semidefinite Programming and Integer Programming*, *Discrete Optimization*, 2005.

Contents

1	Introduction to the MAX CUT Problem	3
1.1	Graph Theory	3
1.2	Complexity	3
1.3	Earlier Methods to solve MAX-CUT	4
1.4	Semidefinite Programming	4
2	A Randomized Algorithm for MAX-CUT	5
2.1	The Algorithm	5
2.2	Analysis of the Algorithm	6
2.3	Semidefinite Relaxation	10
2.4	Eigenvalue Bound	11
3	Further Analysis of the Randomized Algorithm	12
3.1	Large MAX CUT	12
3.2	Negative Edge Weights	14
4	Improvements to the Randomized Algorithm	16
4.1	The FKL-algorithm	16
4.2	A Deterministic Result	18
5	Summary	18

1 Introduction to the MAX CUT Problem

1.1 Graph Theory

Given an undirected graph $G = (V, E)$, where the edges $(i, j) \in E$ have weights w_{ij} , the MAX CUT problem is defined as finding the set of vertices $S \subseteq V$ such that the weight of the *cut* of $(S, \bar{S})$ is maximized. The cut is defined as the edges with one endpoint in S and the other in $\bar{S}$. The *weight of the cut* can be written as

$$w(S, \bar{S}) = \sum_{i \in S, j \notin S} w_{ij}.$$

We have $w_{ij} = w_{ji}$ and for $(i, j) \notin E$ we let $w_{ij} = 0$. The maximum weight of a cut in a graph G is also known as *the maximum cut* of G .

1.2 Complexity

The MAX CUT problem is one of Karp's original NP-complete problems presented in [10]. In addition, it has been shown in [3] that the unweighted MAX CUT problem, $w_{ij} = 1$ for $(i, j) \in E$, is also NP-complete.

The complexity class NP contains decision problems which can be solved by a nondeterministic machine in polynomial time. The complexity class P is the set of decision problems which can be solved by a deterministic machine in polynomial time. Hence all problems in P are also in NP, the converse has not been proven.

A set of problems X is *hard* for a set of problems Y if every problem in Y can "easily" be transformed into a problem in X that produces the same solution. Further, a set of problems X is *complete* for a set of problems Y if it is hard for Y and also a subset of Y .

Hence the set of NP-complete problems contain the most difficult problems in NP; the ones that are most likely not to be in P. If any NP-complete problem is found to be in P, then $\text{NP}=\text{P}$. As most NP-complete problems MAX CUT can be solved in polynomial time for some special classes of graphs, e.g., if the graph is planar, see [13].

Since it is unlikely that there exist efficient algorithms to solve NP-complete problems, the common approach is to use an ρ -*approximation algorithm*, if such an algorithm exists. Then ρ is the *performance guarantee*. For MAX CUT a ρ -approximation algorithm returns, in polynomial time, a cut that is at least ρ times the optimal value. The term ρ -approximation algorithm will also be used for the randomized polynomial time algorithm that returns solutions whose *expected value* is at least ρ times the optimal one.

This paper will be devoted to the randomized algorithm using semidefinite programming introduced by Williamson and Goemans in [4]. Their algorithm always delivers a solution of expected value at least 0.87856 times the optimal value.

It has been shown by Håstad in [7] that for $\rho > 0.94117$ there is no ρ -approximation algorithm for MAX CUT unless $\text{P}=\text{NP}$.

1.3 Earlier Methods to solve MAX-CUT

Prior to [4] the best known approximation algorithm for the MAX CUT problem had a performance guarantee of $\frac{1}{2}$, see e.g. [15]. That algorithm consists of iterating through the vertices and deciding whether to assign vertex i to S or not based on which placement maximizes the cut of vertices 1 to i . This algorithm is according to [4] essentially equivalent to flipping an unbiased coin for each vertex to decide whether to assign it to S or not.

For the *unweighted* MAX CUT problem there are a number of algorithms with performance guarantee $\frac{1}{2} + \phi$, where ϕ is a function of $m = |V|$, $n = |E|$ or Δ^1 . However, no progress had been made in improving the constant $\frac{1}{2}$ until the publication of [4].

1.4 Semidefinite Programming

The randomized algorithm in [4] uses semidefinite programming.

Semidefinite programming (SDP) is a special case of convex programming in which one minimizes a linear objective function subject to the constraint that an affine combination of matrices is positive semidefinite. Using the notation of Laurent and Rendl in [11] the following is the *primal-dual pair* of semidefinite programs

$$\begin{aligned} \max \quad & Tr(CX) \\ \text{s.t.} \quad & Tr(A_i X) = b_i, \quad \forall i = 1, \dots, m, \\ & X \succeq 0, \end{aligned} \tag{1}$$

$$\begin{aligned} \min \quad & b^T y \\ \text{s.t.} \quad & \sum y_i A_i - C - S = 0, \\ & S \succeq 0. \end{aligned} \tag{2}$$

When the matrix argument is diagonal, problems (1) and (2) are reduced to the primal-dual pair of standard LP-programs.

The set of positive semidefinite matrices constitutes a *convex cone*. A convex cone is a subset of a vector space that is closed under convex combinations. Therefore SDP is also said to be a special case of *linear programming over cones* or *cone-LP*.

An important characteristic of SDP that separates it from LP is that strong duality does not always hold. That is, even if there exist X^* optimal to (1) and (y^*, S^*) optimal to (2), then it might be the case that $Tr(CX^*) < b^T y^*$, i.e. the duality gap is non-zero.

A sufficient condition for strong duality to hold for semidefinite programs is the existence of strictly feasible interior points for both the primal and the dual problems. Weak duality holds by construction, see e.g. [8] and [11].

There are some methods for solving SDP, e.g. the *ellipsoid method* and various *interior point methods*. Generally, interior methods are faster, but using either it can be shown that SDP-problems can be solved in polynomial time given any fixed predetermined precision. In effect, given any $\epsilon > 0$ the running

¹The *maximum degree* of a graph, $\Delta(G)$, is $\max_{i \in V} \{deg(i)\}$, where $deg(i)$ is the *degree* of a vertex i , i.e. the number of edges in contact with vertex i .

time dependence on ϵ is polynomial in $\log(1/\epsilon)$ and the size of the input. In order to terminate in polynomial time these algorithms, according to [4], assume some requirements on the feasible region and the size of the optimal solution.

Interior point methods are based on solving the perturbed necessary and sufficient optimality conditions for (1) and (2) following a *central path*. Hence, interior methods are also referred to as *path-following methods* and they are generally described in e.g. [11]. In [11] there is also a discussion on different choices of step-length reduction and symmetrization of the equation $SX - \mu I = 0$.

2 A Randomized Algorithm for MAX-CUT

In this section we will look at the algorithm presented in [4], which is based on rounding a solution to a nonlinear relaxation of the MAX CUT problem. This relaxation can either be seen as a semidefinite program or as an eigenvalue minimization problem. Williamson and Goemans were the first to use semidefinite programming in design of approximation algorithms.

2.1 The Algorithm

The MAX CUT problem can be written as a optimization problem. Let $V = \{1, \dots, n\}$ be the vertex set of the graph G . The weight of the maximum cut is given by the integer quadratic program denoted by (Q):

$$(Q) \quad \max \quad \frac{1}{2} \sum_{(i,j) \in E} w_{ij}(1 - x_i x_j) \\ s.t. \quad x_i \in \{\pm 1\} \quad \forall i \in V.$$

We assume that the edge weights w_{ij} are nonnegative. The case when w_{ij} are allowed to be negative will be treated in Section 3.2. The program (Q) corresponds to dividing the nodes into two sets, S and $\bar{S}$, such that $S = \{i | x_i = 1\}$ and $\bar{S} = \{i | x_i = -1\}$. For an edge (i, j) where $i \in S$ and $j \in \bar{S}$, $x_i x_j = -1$ and $1 - x_i x_j = 2$, i.e. as the edge (i, j) is part of the cut, the term w_{ij} contributes to the sum in (Q). On the other hand, if i and j both belong to either S or $\bar{S}$, $x_i x_j = 1$ and $1 - x_i x_j = 0$, i.e. as the edge (i, j) is not part of the cut the contribution to the sum in (Q) will be zero.

The optimal value of (Q) is, as mentioned before, denoted by Z_{mc}^* . Since solving (Q) is an NP-complete problem one has to consider relaxations. The optimal solution to any relaxation of (Q) is an upper bound of Z_{mc}^* .

One can view (Q) as restricting x_i to be a one-dimensional vector of unit norm. The choice of Williamson and Goemans in [4] was to allow x_i to be multidimensional vector v_i of unit Euclidean length. The linear space spanned by the vectors $v_1, \dots, v_n$ has dimension at most n , i.e. the vectors can be assumed to belong to the n -dimensional unit sphere S_n . The following program (P) is the relaxation of (Q) used in [4]:

$$(P) \quad \max \quad \frac{1}{2} \sum_{(i,j) \in E} w_{ij}(1 - v_i^T v_j) \\ s.t. \quad v_i \in S_n \quad \forall i \in V.$$

Let Z_P^* denote the optimal value of (P) . In section 2.3 it will be shown that (P) is equivalent to a semidefinite program that we can solve using some semidefinite programming method.

The following is the randomized algorithm by Williamson and Goemans:

1. Solve (P) and obtain the optimal vectors $v_1, \dots, v_n$.
2. Generate a random vector r , uniformly distributed over S_n .
3. Let $S = \{i | v_i^T r \geq 0\}$.

The second and third step is equivalent to choosing a random hyperplane through S_n and checking on which side of it each v_i lies by observing the sign of the dot-product between v_i and r . The vector r can be seen as the normal to the random hyperplane passing through S_n . According to [4] this randomized step can be used since (P) is independent of the coordinate system; applying any orthonormal transformation to a set of vectors will give a solution with the same optimal value.

2.2 Analysis of the Algorithm

Let $v_1, \dots, v_n$ be any vectors in S_n and $w(S, \bar{S})$ denote the weight of the cut produced using the randomized algorithm in section 2.1, then $E[w(S, \bar{S})]$, the expectation of that weight, is given by

$$E[w(S, \bar{S})] = \sum_{(i,j) \in E} w_{ij} \Pr \left(\text{sgn}(v_i^T r) \neq \text{sgn}(v_j^T r) \right) \quad (3)$$

where $\Pr(A)$ is the *probability* of the event A and

$$\text{sgn}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}.$$

Clearly, $\Pr \left(\text{sgn}(v_i^T r) \neq \text{sgn}(v_j^T r) \right)$ is the probability that an edge (i, j) belongs to the cut.

For our analysis we need following lemma:

Lemma 1.

$$\Pr \left(\text{sgn}(v_i^T r) \neq \text{sgn}(v_j^T r) \right) = \frac{1}{\pi} \cdot \arccos(v_i^T v_j)$$

Proof. Let θ denote the angle between the two vectors v_i and v_j , i.e.

$$\theta = \arccos(v_i^T v_j).$$

Note that by symmetry

$$\Pr \left(\text{sgn}(v_i^T r) \neq \text{sgn}(v_j^T r) \right) = 2 \cdot \Pr \left(v_i^T r \geq 0, v_j^T r \leq 0 \right).$$

The intersection of the set $\{r \mid v_i^T r \geq 0, v_j^T r \leq 0\}$ and the sphere S_n is a spherical digon of angle θ . The probability of the set over the whole sphere is therefor $\theta/2\pi$, i.e.

$$\Pr \left(v_i^T r \geq 0, v_j^T r \leq 0 \right) = \frac{\theta}{2\pi}$$

which gives

$$\Pr \left(\text{sgn}(v_i^T r) \neq \text{sgn}(v_j^T r) \right) = \frac{\theta}{\pi} = \frac{1}{\pi} \cdot \arccos(v_i^T v_j)$$

which proves the lemma. $\square$

With Lemma 1 and (3) we get the important result

$$E[w(S, \bar{S})] = \frac{1}{\pi} \sum_{(i,j) \in E} w_{ij} \arccos(v_i^T v_j). \quad (4)$$

We can rewrite (4) as

$$E[w(S, \bar{S})] = \frac{1}{\pi} \sum_{(i,j) \in E} w_{ij} \arccos(v_i^T v_j) \frac{1 - v_i^T v_j}{2} \frac{2}{1 - v_i^T v_j}.$$

Note that we not to worry about dividing by zero since if $v_i^T v_j = 1$, then $\arccos(v_i^T v_j) = \arccos(1) = 0$.

Now let α denote

$$\alpha = \min_{0 \leq \theta \leq \pi} g(\theta),$$

where

$$g(\theta) = \frac{2}{\pi} \frac{\theta}{1 - \cos(\theta)}.$$

We obtain α for some $\theta_0 \in [0, \pi]$ and it follows that

$$g(\theta) \geq g(\theta_0) = \alpha, \quad \forall \theta \in [0, \pi].$$

As $\theta = \arccos(v_i^T v_j)$ we get

$$\frac{1}{2} \sum_{(i,j) \in E} \frac{2}{\pi} \frac{\arccos(v_i^T v_j)}{1 - v_i^T v_j} w_{ij} (1 - v_i^T v_j) \geq \alpha \cdot \frac{1}{2} \sum_{(i,j) \in E} w_{ij} (1 - v_i^T v_j), \quad (5)$$

and we then have the following theorem.

Theorem 1.

$$E[w(S, \bar{S})] \geq \alpha \cdot \frac{1}{2} \sum_{(i,j) \in E} w_{ij} (1 - v_i^T v_j)$$

Proof. Follows by (4), (5) and the non-negativity of w_{ij} . $\square$

Hence if we let $v_1, \dots, v_n$ be an optimal solution to (P) then

$$E[w(S, \bar{S})] \geq \alpha \cdot Z_P^* \geq \alpha \cdot Z_{mc}^*.$$

The last inequality follows from the fact that (P) is a relaxation of (Q) . Theorem 1 shows that the randomized algorithm has a performance guarantee that is at least α for the MAX CUT problem. However, this result requires that one can solve (P) exactly, which is not possible. It will be shown in section 2.3 that the algorithm is in fact a randomized $(\alpha - \epsilon)$ -approximation algorithm, for some predetermined precision ϵ .

Finally, the bound on α is given by the following lemma.

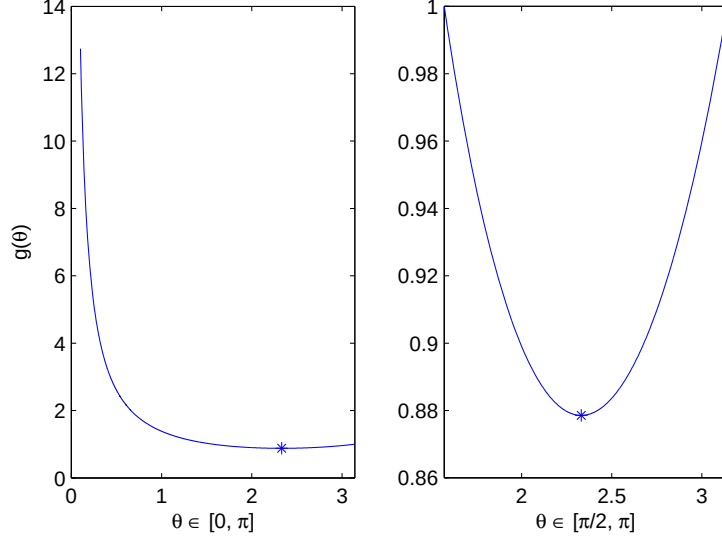


Figure 1: The plot on the left shows $g(\theta)$ over the whole interval $[0, \pi]$, while the plot on the right shows $g(\theta)$ over the interval $[\pi/2, \pi]$. The star in both plots indicates the point (θ_0, α) .

Lemma 2.

$$\alpha > 0.87856.$$

Proof. We have, as stated above, that for some $\theta_0 \in [0, \pi]$

$$\alpha = \frac{2}{\pi} \frac{\theta_0}{1 - \cos(\theta_0)}.$$

Solving for θ_0 we have

$$g(\theta) = \frac{2}{\pi} \frac{\theta}{1 - \cos(\theta)},$$

then

$$g'(\theta) = \frac{2}{\pi(1 - \cos(\theta))} \left(1 - \frac{\theta \sin(\theta)}{1 - \cos(\theta)} \right).$$

Set

$$g'(\theta)|_{\theta=\theta_0} = 0,$$

and solve for θ_0 . That is, find the nonzero root of $1 = \cos(\theta) + \theta \sin(\theta)$ which gives

$$\theta_0 = 2.331122..$$

It now remains to prove a rational bound for α , i.e. a finite representation. Observe in Figure 1 that

$$g(\theta) = \frac{2}{\pi} \frac{\theta}{1 - \cos(\theta)} \geq 1, \quad \forall \theta \in [0, \pi/2],$$

and that $\theta_0 \in [\pi/2, \pi]$ and the function $f(\theta) = 1 - \cos(\theta)$ is concave for $\theta \in [\pi/2, \pi]$, then for any $\tilde{\theta}_0$

$$f(\theta) - f(\tilde{\theta}_0) \leq (\theta - \tilde{\theta}_0)f'(\theta)$$

which gives

$$\begin{aligned} 1 - \cos(\theta) &\leq 1 - \cos(\tilde{\theta}_0) + (\theta - \tilde{\theta}_0)\sin(\tilde{\theta}_0) = \\ &= \theta\sin(\tilde{\theta}_0) + (1 - \cos(\tilde{\theta}_0) - \tilde{\theta}_0\sin(\tilde{\theta}_0)) \end{aligned}$$

If one lets $\tilde{\theta}_0 = 2.331122$, then we get

$$\begin{aligned} 1 - \cos(\tilde{\theta}_0) - \tilde{\theta}_0\sin(\tilde{\theta}_0) &< 0 \\ \Rightarrow 1 - \cos(\theta) &< \theta\sin(\tilde{\theta}_0), \end{aligned} \tag{6}$$

This implies that for $\theta_0 = 2.331122..$

$$\begin{aligned} \alpha &= \frac{2}{\pi} \frac{\theta_0}{1 - \cos(\theta_0)} > \left\{ (6) \right\} > \frac{2}{\pi} \frac{\theta_0}{\theta_0\sin(\tilde{\theta}_0)} = \\ &= \frac{2}{\pi\sin(\tilde{\theta}_0)} = 0.878566.. > 0.87856, \end{aligned}$$

which formally gives the bound on α . □

The above analysis gives rise to a new formulation (R) of the MAX CUT problem:

$$\begin{aligned} (R) \quad & \max \sum_{(i,j) \in E} w_{ij} \frac{\arccos(v_i^T v_j)}{\pi} \\ & s.t. \quad v_i \in S_n \quad \forall i \in V. \end{aligned}$$

Let the optimal value of (R) be denoted by Z_R^* . We have the following theorem.

Theorem 2. $Z_R^* = Z_{mc}^*$.

Proof. First show that $Z_R^* \geq Z_{mc}^*$. We have that (R) is a relaxation of (Q) and the objective function of (R) reduces to the objective function of (Q) in the case of v_i lying in one-dimensional space. Then it follows that $Z_R^* \geq Z_{mc}^*$.

Next show $Z_R^* \leq Z_{mc}^*$. Let $v_1, \dots, v_n$ denote an optimal solution to (R). From (4) we have that the randomized algorithm gives a cut of expected value exactly Z_R^* . There must exist a cut of value greater or equal to Z_R^* . It follows that $Z_R^* \leq Z_{mc}^*$. □

2.3 Semidefinite Relaxation

In this section it will be shown that (P) is equivalent to a semidefinite program.

Given a symmetric positive semidefinite $n \times n$ matrix X , one can find B , $m \times n$ full row-rank such that $X = B^T B$, in $O(n^3)$ time using incomplete Cholesky decomposition see e.g. [5]. Let the i th column of B be v_i for any unit vectors $v_1, \dots, v_n \in S_m$. Then we have $x_{ii} = 1$, $x_{ij} = v_i^T v_j$ and X is known as the Gram matrix of $v_1, \dots, v_n$.

This gives us that (P) is equal to the following semidefinite problem denoted by (SD) :

$$(SD) \quad \begin{aligned} \max \quad & \frac{1}{2} \sum_{(i,j) \in E} w_{ij}(1 - x_{ij}) \\ \text{s.t.} \quad & x_{ii} = 1 \quad \forall i \in V \\ & X = (x_{ij}) \succeq 0. \end{aligned}$$

Since (P) and (SD) are equivalent we have that $Z_P^* = Z_{SD}^*$.

Once we have solved (SD) we can use incomplete Cholesky decomposition to obtain $v_1, \dots, v_n \in S_m$ for some $m \leq n$.

We cannot solve (SD) exactly in polynomial time, however using a semidefinite programming algorithm we can, for some predetermined precision $\epsilon > 0$, obtain a set of vectors $v_1, \dots, v_n$ of value greater than $Z_P^* - \epsilon$ in time polynomial in the size of the input and $\log(1/\epsilon)$, see e.g. [11].

For these almost optimal vectors the randomized algorithm will produce a cut of expected value greater or equal to $\alpha \cdot (Z_P^* - \epsilon) \geq \alpha \cdot (Z_{mc}^* - \epsilon) \geq (\alpha - \epsilon) \cdot Z_{mc}^*$. The last inequality holds under the assumption that the weights are integral, i.e. $Z_{mc}^* \geq 1$, and knowing that $\alpha < 1$. Hence, we have an $(\alpha - \epsilon)$ -approximation algorithm for the MAX CUT problem for some predetermined precision ϵ .²

In [4] it is stated that the performance guarantee of the randomized algorithm is at least α . Some years later it was proven by Karloff in [9] that the performance guarantee is actually equal to α .

However, the algorithm usually behaves much better than this worst case bound, see e.g. the Results section of [4].

The next corollary follows from Theorem 1.

Corollary 1.

$$\frac{Z_{mc}^*}{Z_p^*} \geq \alpha.$$

Proof. As we know that $E[w(S, \bar{S})] \leq Z_{mc}^*$ we get

$$\frac{Z_{mc}^*}{Z_p^*} \geq \frac{E[w(S, \bar{S})]}{Z_p^*} \geq \alpha,$$

where the last inequality follows from Theorem 1. □

The quantity Z_{mc}^*/Z_p^* is known as the *integrality ratio*.

²If the weights are not integral, but still nonnegative an appropriate scaling will give us that we have $(\alpha - \epsilon_0)$ -approximation algorithm for the MAX CUT problem, for some $\epsilon_0 > 0$, for some predetermined $\epsilon_0(\epsilon)$.

2.4 Eigenvalue Bound

We define the Laplacian matrix $L = (l_{ij})$, where

$$l_{ij} = \begin{cases} -w_{ij} & i \neq j \\ \sum_{k=1}^n w_{ik} & i = j \end{cases}.$$

Let $\lambda_{max}(A)$ denote the maximum eigenvalue of the matrix A .

There is an eigenvalue upper bound on Z_{mc}^* , this was shown by Delorme and Poljak in [1]. We need the following lemma from [1].

Lemma 3. *Let $u \in \mathbb{R}^n$ such that $u_1 + \dots + u_n = 0$. Then*

$$\frac{n}{4} \lambda_{max}(L + \text{diag}(u))$$

is an upper bound for Z_{mc}^ .*

Proof. Let x be an optimal solution to (Q) , i.e. $x = \{\pm 1\}^n$, and note that $x^T L x = 4 \cdot Z_{mc}^*$. We can use the *Rayleigh principle*

$$\lambda_{max}(A) = \max_{\|x\|=1} x^T A x,$$

to obtain

$$\begin{aligned} \lambda_{max}(L + \text{diag}(u)) &\geq \frac{x^T (L + \text{diag}(u)) x}{x^T x} = \frac{x^T L x + \sum_{i=1}^n x_i^2 u_i}{x^T x} = \\ &= \frac{4Z_{mc}^* + \sum_{i=1}^n 1 \cdot u_i}{n} = \frac{4Z_{mc}^*}{n}. \end{aligned}$$

We get

$$\frac{n}{4} \lambda_{max}(L + \text{diag}(u)) \geq Z_{mc}^*$$

which proves the lemma. $\square$

To get the bound proposed in [1] let $g(u) = (n/4) \lambda_{max}(L + \text{diag}(u))$ and take the infimum of $g(u)$ over all vectors u such that $u_1 + \dots + u_n = 0$. That is, the eigenvalue upper bound for MAX CUT is given by the eigenvalue minimization program (EIG) :

$$\begin{aligned} (EIG) \quad & \text{Inf } g(u) \\ & \text{s.t. } u_1 + \dots + u_n = 0. \end{aligned}$$

It is known that eigenvalue minimization problems can be written as semi-definite programs and the fact that (EIG) is equivalent to (SD) was shown in [14].

3 Further Analysis of the Randomized Algorithm

3.1 Large MAX CUT

The algorithm in [4] can be refined for graphs having very large cuts. That is, the value of the relaxation (P) is a large fraction of the total weight of the graph. Define the total weight of the cut as

$$W_{tot} = \sum_{(i,j) \in E} w_{ij}.$$

Then define

$$h(t) = \frac{\arccos(1 - 2t)}{\pi}$$

and let γ be such that

$$\gamma = \operatorname{argmin}_{t \in (0,1]} \frac{h(t)}{t}.$$

We get that

$$\gamma = \frac{1 - \cos(\theta_0)}{2} \approx 0.84458$$

where θ_0 is as before the angle that gives α . Hence, it follows that $\frac{h(\gamma)}{\gamma} = \alpha$ and $\frac{h(t)}{t} \geq \alpha, \forall t \in (0, 1]$, which can also be seen in Figure 2.

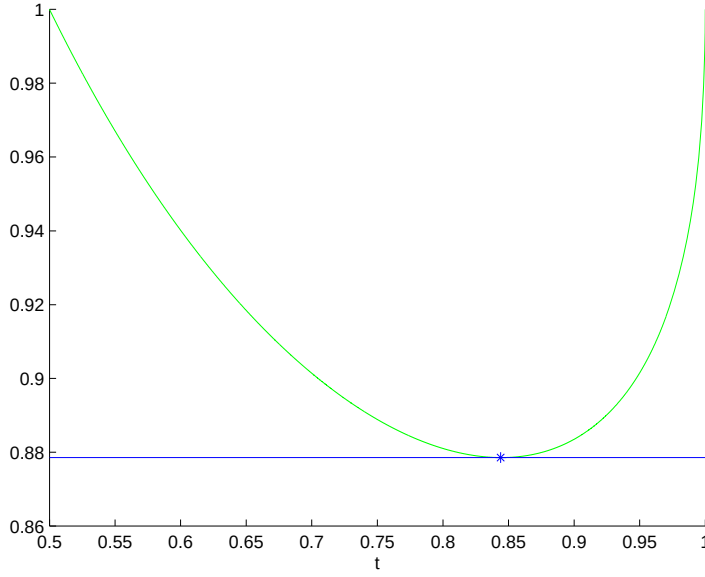


Figure 2: The green part represents $h(t)/t$ and the blue vertical line represents α . It is clear that $h(t)/t \geq \alpha$ for $t \in [0, 1]$.

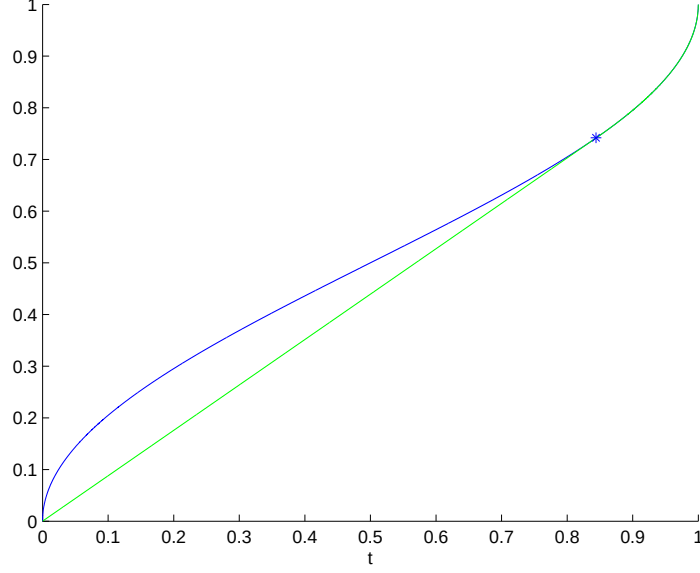


Figure 3: The green part represents $\tilde{h}(t)$, the blue part represents $h(t)$ and the star the point where $h(t)/t = \alpha$. It is clear that $\tilde{h}(t) \leq h(t)$ for $t \in [0, 1]$.

Theorem 3. *Let*

$$A = \frac{1}{W_{tot}} \sum_{(i,j) \in E} w_{ij} \frac{1 - v_i^T v_j}{2}$$

if $A \geq \gamma$, then

$$E[w(S, \bar{S})] \geq \frac{h(A)}{A} \sum_{(i,j) \in E} w_{ij} \frac{1 - v_i^T v_j}{2}.$$

This implies that we have performance guarantee of $h(A)/A - \epsilon$ when $A \geq \gamma$. The proof of Theorem 3 follows the one given in [11].

Proof. Let

$$\tilde{\alpha}(t) = \frac{h(t)}{t}, \quad t \in [\gamma, 1],$$

$$\tilde{\alpha}(t) = \alpha, \quad t \in [0, \gamma],$$

and

$$\tilde{h}(t) = \tilde{\alpha}(t) \cdot t, \quad t \in [0, 1].$$

We have that $\tilde{h}(t)$ is convex on $[0, 1]$ and $\tilde{h}(t) \geq h(t)$ see Figure 3.

Hence if we let

$$y_{ij} = \frac{1 - v_i^T v_j}{2},$$

then

$$\frac{E[w(S, \bar{S})]}{W_{tot}} = \sum_{(i,j) \in E} \frac{w_{ij}}{W_{tot}} \frac{\arccos(v_i^T v_j)}{\pi} = \sum_{(i,j) \in E} \frac{w_{ij}}{W_{tot}} \frac{\arccos(1 - 2y_{ij})}{\pi} =$$

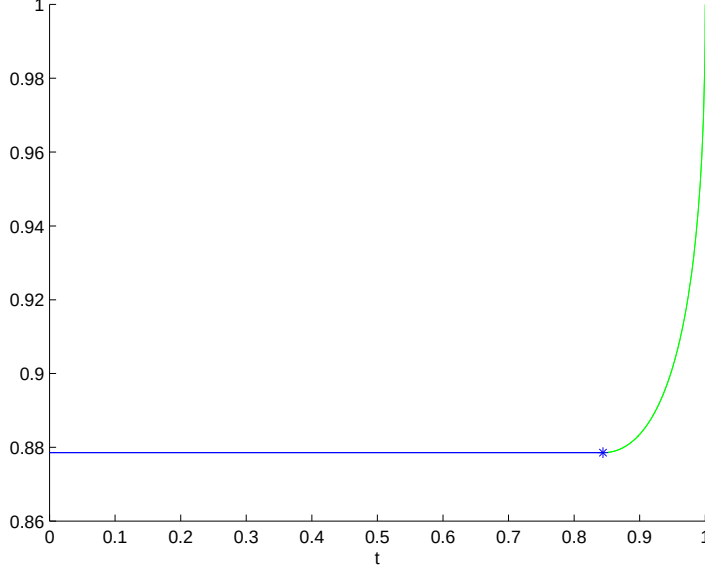


Figure 4: This graph shows the behavior of $\tilde{\alpha}(t)$ for $t \in [0, 1]$. The vertical blue line represents $\tilde{\alpha}(t)$ for $t \in [0, \gamma]$, the green part represents $\tilde{\alpha}(t)$ for $t \in [\gamma, 1]$ and the star indicates the point (γ, α) .

$$\begin{aligned}
&= \sum_{(i,j) \in E} \frac{w_{ij}}{W_{tot}} h(y_{ij}) \geq \left\{ \tilde{h}(t) \geq h(t) \right\} \geq \sum_{(i,j) \in E} \frac{w_{ij}}{W_{tot}} \tilde{h}(y_{ij}) \geq \left\{ \tilde{h}(t) \text{ convex} \right\} \geq \\
&\geq \tilde{h} \left(\sum_{(i,j) \in E} \frac{w_{ij}}{W_{tot}} y_{ij} \right) = \tilde{h} \left(\frac{1}{W_{tot}} \sum_{(i,j) \in E} w_{ij} \frac{1 - v_i^T v_j}{2} \right) = \tilde{h}(A) = \tilde{\alpha}(A) \cdot A
\end{aligned}$$

It follows that

$$\begin{aligned}
E[w(S, \bar{S})] &\geq W_{tot} \cdot \tilde{\alpha}(A) \cdot A = \tilde{\alpha}(A) \sum_{(i,j) \in E} w_{ij} \frac{1 - v_i^T v_j}{2} = \\
&= \begin{cases} \alpha \sum_{(i,j) \in E} w_{ij} \frac{1 - v_i^T v_j}{2} & A \in [0, \gamma] \\ \frac{h(A)}{A} \sum_{(i,j) \in E} w_{ij} \frac{1 - v_i^T v_j}{2} & A \in [\gamma, 1] \end{cases}.
\end{aligned}$$

□

Hence, we have performance guarantee for the algorithm that is at least $\tilde{\alpha}(A)$ which is greater than α for $A \geq \gamma$, see Figure 4.

3.2 Negative Edge Weights

Previously we have assumed that the edge weights w_{ij} are nonnegative. Now we will extend the analysis to the case when w_{ij} can be negative. This is the case in many practical problems. We need to redefine the performance guarantee since the optimal value can be negative.

Define W_- as

$$W_- = \sum_{(i,j) \in E} w_{ij}^-$$

where $w_{ij}^- = \min\{0, w_{ij}\}$. W_- is the “sum” of all edge weights that are negative. Then the following theorem is the extension for negative edge weights of Theorem 1.

Theorem 4. *If*

$$\{E[w(S, \bar{S})] - W_-\} \geq \alpha \cdot \left\{ \sum_{(i,j) \in E} w_{ij} \frac{1 - v_i^T v_j}{2} - W_- \right\}$$

then α is the performance guarantee for the randomized algorithm.

Proof. Note that

$$-W_- = \sum_{w_{ij} < 0} |w_{ij}|.$$

Then we can rewrite the left-hand side above as

$$\sum_{w_{ij} > 0} w_{ij} \frac{\arccos(v_i^T v_j)}{\pi} + \sum_{w_{ij} < 0} |w_{ij}| \left(1 - \frac{\arccos(v_i^T v_j)}{\pi} \right). \quad (7)$$

In a similar manner we can rewrite as $\left\{ \sum_{(i,j) \in E} w_{ij} \frac{1 - v_i^T v_j}{2} - W_- \right\}$ as

$$\begin{aligned} & \sum_{w_{ij} > 0} w_{ij} \frac{1 - v_i^T v_j}{2} + \sum_{w_{ij} < 0} |w_{ij}| \left(1 - \frac{1 - v_i^T v_j}{2} \right) = \\ & = \sum_{w_{ij} > 0} w_{ij} \frac{1 - v_i^T v_j}{2} + \sum_{w_{ij} < 0} |w_{ij}| \left(\frac{1 + v_i^T v_j}{2} \right). \end{aligned} \quad (8)$$

For the first term in (7) we have from before that

$$\frac{\arccos(v_i^T v_j)}{\pi} \frac{2}{1 - v_i^T v_j} = \frac{\theta}{\pi} \frac{2}{1 - \cos(\theta)} \geq \alpha,$$

which gives

$$\frac{\arccos(v_i^T v_j)}{\pi} \geq \alpha \cdot \frac{1 - v_i^T v_j}{2}.$$

For the second term in (7) note that

$$\left(1 - \frac{\arccos(v_i^T v_j)}{\pi} \right) = \frac{\pi - \arccos(v_i^T v_j)}{\pi} = \frac{\arccos(-v_i^T v_j)}{\pi},$$

which gives

$$\frac{\arccos(-v_i^T v_j)}{\pi} \geq \alpha \cdot \frac{1 + v_i^T v_j}{2}.$$

Hence we have that (7) $\geq \alpha$ (8), i.e.

$$\sum_{w_{ij} > 0} w_{ij} \frac{\arccos(v_i^T v_j)}{\pi} + \sum_{w_{ij} < 0} |w_{ij}| \left(1 - \frac{\arccos(v_i^T v_j)}{\pi} \right) \geq$$

$$\geq \alpha \left\{ \cdot \sum_{w_{ij} > 0} w_{ij} \frac{1 - v_i^T v_j}{2} + \sum_{w_{ij} < 0} |w_{ij}| \left(\frac{1 + v_i^T v_j}{2} \right) \right\}$$

which proves the theorem. $\square$

Note that if all weights are negative, the maximum cut is the empty set.

4 Improvements to the Randomized Algorithm

When Williamson and Goemans presented their randomized approximation algorithm in 1995 it was the first real progress to be made approximating MAX CUT in almost twenty years. There was a lot of excitement regarding the use of semidefinite programming in approximation algorithm design. Given the straightforwardness of their algorithm it was believed that a stronger result would soon follow from improvements on the original algorithm. But still today no one has been able to prove a better performance guarantee than α for the general MAX CUT problem with nonnegative edge weights.

4.1 The FKL-algorithm

There are, however, many examples of stronger results for special cases of MAX CUT. Inspired by [11] we will look briefly at an algorithm by Feige, Karpinski and Langberg [2] which gives a better performance guarantee for unweighted graphs with a bounded maximum degree Δ . For the case when $\Delta = 3$ they have some especially nice results which follow.

First we define that a vertex $i \in S$ as *misplaced* if it has more neighbors in S than in $\bar{S}$.

Let T be all the possible triplets (i, j, k) of vertices in V such that $(i, j), (i, k) \in E$.

The FKL-algorithm is modified in two aspects from the algorithm in [4]. First the authors of [2] use an improved relaxation $(\tilde{P})$:

$$\begin{aligned} (\tilde{P}) \quad & \max \quad \frac{1}{2} \sum_{(i,j) \in E} w_{ij} (1 - v_i^T v_j) \\ & s.t. \quad v_i \in S_n \quad \forall i \in V \\ & \quad v_i^T v_j + v_i^T v_k + v_j^T v_k \geq -1, \\ & \quad v_i^T v_j - v_i^T v_k - v_j^T v_k \geq -1 \quad \forall i, j, k \in V \quad (9) \\ & \quad v_i^T v_j + v_i^T v_k + v_j^T v_k = -1 \quad \forall (i, j, k) \in T. \quad (10) \end{aligned}$$

The new constraints (9) are called *triangle constraints* and are valid when the v_i 's are restricted to be one-dimensional. The new constraints (10) are called *tight triangle constraints* and this has to be fulfilled for maximum cut, if it is not then the vertex i is misplaced.

The program $(\tilde{P})$ is equivalent to a semidefinite program $(\tilde{SD})$ using the reasoning in Section 2.3.

The second modification is a *greedy phase* used after the hyperplane rounding phase in [4]. The greedy phase involves recognizing if there is a misplaced vertex and moving it to the other set until no more misplaced vertices can be found.

It is shown in [2] that the final partition $(S, \bar{S})$ given by the modified randomized algorithm satisfies

$$E[S, \bar{S}] \geq 0.919 \cdot Z_P^*.$$

In [2] it is also shown that for regular graphs³ of degree 3 we have

$$E[S, \bar{S}] \geq 0.924 \cdot Z_P^*.$$

For a graph with a bounded maximum degree Δ , Feige, Karpinski and Langberg prove a performance guarantee of

$$\alpha + \frac{1}{2^{33} \cdot \Delta^4}.$$

It was shown in [9] that it is impossible to improve the performance ratio by only adding valid linear constraints. That is the reason for the use of the greedy phase in [2].

In 2004, Halperin, Livnat and Zwick [6] proved a 0.9326 for graphs of maximal degree 3 by adding a preprocessing phase to the FKL-algorithm. In this preprocessing phase the graph is converted into a cubic graph.

4.2 A Deterministic Result

Williamson and Goemans' algorithm is, as mentioned before, a randomized approximation algorithm, i.e. it returns solutions whose expected value is at least α times the optimal. However, it was shown by Mahajan and Ramesh in [12] that this algorithm can be derandomized to give a deterministic α -approximation algorithm.

5 Summary

This report is a close review of Williamson and Goemans' randomized approximation algorithm for MAX CUT presented in 1995 [4]. This was the first time semidefinite programming was used in approximation algorithm design. Through their analysis they found a 0.87856-approximation algorithm for MAX CUT. The best previously known result was a $\frac{1}{2}$ -approximation algorithm.

The randomized algorithm is based on the following steps:

- Solve the semidefinite relaxation (SD) of the MAX CUT problem (Q).
- Use incomplete Cholesky decomposition to obtain the vectors $v_1, \dots, v_n \in S_m$, for some $m \leq n$. These are the vectors that solve (P).
- Generate a random vector r , uniformly distributed over S_n .
- Let $S = \{i | v_i^T r \geq 0\}$.
- Then $E[w(S, \bar{S})] \geq \alpha \cdot Z_P^* \geq \alpha \cdot Z_{mc}^*$, where $\alpha > 0.87856$.

But since we can not solve the semidefinite program (SD) exactly we in fact get a $(\alpha - \epsilon)$ -approximation algorithm for some predetermined precision ϵ .

It can be shown that (P) is equivalent to an eigenvalue minimization problem.

³A regular graph is a graph where each vertex has the same number of neighbors, i.e. every vertex has the same degree.

Further analysis of Williamson and Goemans' randomized approximation algorithm shows that the performance guarantee is larger than α when the maximum cut is large. The analysis can also be extended to the case when the edge weights can be negative.

For special cases of MAX CUT improvements made to the algorithm in [4] have rendered better performance guarantees. But in the general case no one have yet been able to prove a better performance guarantee than α for the general MAX CUT problem with nonnegative edge weights.

References

- [1] C. Delorme and S. Poljak. Laplacian eigenvalues and the maximum cut problem. *Math. Programming*, 62(3, Ser. A):557–574, 1993.
- [2] U. Feige, M. Karpinski, and M. Langberg. Improved approximation of Max-Cut on graphs of bounded degree. *J. Algorithms*, 43(2):201–219, 2002.
- [3] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoret. Comput. Sci.*, 1(3):237–267, 1976.
- [4] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.
- [5] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3 of *Johns Hopkins Series in the Mathematical Sciences*. Johns Hopkins University Press, Baltimore, MD, 1983.
- [6] E. Halperin, D. Livnat, and U. Zwick. MAX CUT in cubic graphs. *J. Algorithms*, 53(2):169–185, 2004.
- [7] J. Håstad. Some optimal inapproximability results. In *STOC '97 (El Paso, TX)*, pages 1–10 (electronic). ACM, New York, 1999.
- [8] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz. An interior-point method for semidefinite programming. *SIAM J. Optim.*, 6(2):342–361, 1996.
- [9] H. Karloff. How good is the Goemans-Williamson MAX CUT algorithm? *SIAM J. Comput.*, 29(1):336–350 (electronic), 1999.
- [10] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103. Plenum, New York, 1972.
- [11] M. Laurent and F. Rendl. Semidefinite programming and integer programming. In K. Aardal, G. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, chapter 8, pages 393–514. Elsevier B.V., 2005.
- [12] S. Mahajan and H. Ramesh. Derandomizing semidefinite programming based approximation algorithms. In *36th Annual Symposium on Foundations of Computer Science (Milwaukee, WI, 1995)*, pages 162–169. IEEE Comput. Soc. Press, Los Alamitos, CA, 1995.
- [13] G. I. Orlova and Y. G. Dorfman. Finding the maximum cut in a graph. *Izv. Akad. Nauk SSSR Tehn. Kibernet.*, (3):155–159, 1972.
- [14] S. Poljak and F. Rendl. Nonpolyhedral relaxations of graph-bisection problems. *SIAM J. Optim.*, 5(3):467–487, 1995.
- [15] S. Sahni and T. Gonzalez. P -complete approximation problems. *J. Assoc. Comput. Mach.*, 23(3):555–565, 1976.