Journal of Computational Mathematics, Vol.21, No.3, 2003, 357–366.

APPROXIMATION ALGORITHM FOR MAX-BISECTION PROBLEM WITH THE POSITIVE SEMIDEFINITE RELAXATION[*1)]

Da-chuan Xu   Ji-ye Han
(*Institute of Applied Mathematics, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing 100080, China*)

Abstract

Using outward rotations, we obtain an approximation algorithm for Max-Bisection problem, i.e., partitioning the vertices of an undirected graph into two blocks of equal cardinality so as to maximize the weights of crossing edges. In many interesting cases, the algorithm performs better than the algorithms of Ye and of Halperin and Zwick. The main tool used to obtain this result is semidefinite programming.

Key words: Approximation algorithm, Max-Bisection problem, Semidefinite programming, Approximation ratio.

1. Introduction

Given an undirected graph $G = (V, E)$ and nonnegative weights $w_{ij} = w_{ji}$ on the edges $(i, j) \in E$, the maximum cut problem (Max-Cut) is that of finding the set of vertices S that maximizes the weight of the edges in the *cut* $(S, V \setminus S)$; that is, the weight of the edges with one endpoint in S and the other in $V \setminus S$. For simplicity, we usually set $w_{ij} = 0$ for $(i, j) \notin E$ and denote the weight of a cut $(S, \bar{S})$ by $w(S) = \sum_{i \in S, j \in V \setminus S} w_{ij}$. The Maximum Bisection of $G = (V, E)$ is a partition of the vertex set V into two equally sized sets S and $V \setminus S$ so that $w(S)$ is maximized. In Max-Bisection (MB) problem, $n = |V|$ is assumed to be even.

Max-Cut is well-known to be NP-hard and so is Max-Bisection. This means that one should not expect to find a polynomial time algorithm for solving it exactly. Therefore many experts are interested in developing polynomial time approximation algorithms for Max-Cut and Max-Bisection. A (randomized) algorithm for the maximization problem is called (randomized) r-approximation algorithm, where $0 < r \le 1$, if it outputs a feasible solution with its (expected) value at least r times the optimum value for all instances of the problem.

The best known approximation ratio for Max-Cut is 0.87856 ([7]). Using outward rotations, Zwick [14] obtain an approximation algorithm for Max-Cut that, in many interesting cases, the algorithm performs better than the algorithm of Goemans and Williamson [7]. Feige, Karpinski and Langberg [5] show that for graphs of degree at most Δ, their algorithm achieces an approximation ratio of at least $0.87856 + \varepsilon$, where $\varepsilon > 0$ is a constant that depends only on Δ. In particular, using computer assisted analysis, they show that for graphs of maximal degree 3, their algorithm obtains an approximation ratio of at least 0.921, and for 3-regular graphs, the approximation ratio is at least 0.924. We refer to [2] for the survey of Max-Cut.

[*] Received January 6, 2001; final revised October 11, 2001.

[1)] Research partly supported by Chinese NSF grant 19731001 and National 973 Information Technology and High-Performance Software Program of China with grant No. G1998030401. The first author gratefully acknowledges the support of K. C. Wong Education Foundation, Hong Kong.

To our knowledge, the best known approximation ratio for Max-Bisection is 0.7016 ([8]). This work is an extension of the works of [1] and [12]. Feige Karpinski and Langberg [4] design a 0.795 approximation algorithm for Max-Bisection restricted to regular graphs. In the case of three regular graphs their results imply an approximation ratio of 0.834.

All the above works adopt the SDP relaxation for Max-Cut or Max-Bisection. SDP relaxations have been successfully applied to various graph optimization problems ([1]-[10], [12]-[14]). Many experts put forward different techiques to get a good approximation solution from the SDP relaxation (for example, see [7], [8], [12], and [14]). Using these techiques carefully, we obtain an approximation algorithm for Max-Bisection that, in many interesting cases, the algorithm performs better than the algorithm of Halperin and Zwick [8].

2. SDP Relaxation of MB

The MB problem can be formulated as the following binary integer program

$$\begin{aligned} w^* := \quad \text{Max} \quad & \frac{1}{2}\sum_{i<j} w_{ij}(1 - x_i x_j) \\ \text{s.t.} \quad & \sum_{j=1}^{n} x_j = 0 \\ & x_j^2 = 1,\ j = 1, \cdots, n. \end{aligned} \tag{2.1}$$

Using the "triangle inequalities", the SDP relaxation of MB becomes

$$\begin{aligned} w^{\text{SDP}} := \quad \text{Max} \quad & \frac{1}{2}\sum_{i<j} w_{ij}(1 - v_i \cdot v_j) \\ \text{s.t.} \quad & \sum_{1\le i,j\le n} v_i \cdot v_j = 0, \\ & v_i \cdot v_j + v_i \cdot v_k + v_j \cdot v_k \ge -1, \quad 1 \le i,j,k \le n \\ & -v_i \cdot v_j - v_i \cdot v_k + v_j \cdot v_k \ge -1, \quad 1 \le i,j,k \le n \\ & -v_i \cdot v_j + v_i \cdot v_k - v_j \cdot v_k \ge -1, \quad 1 \le i,j,k \le n \\ & v_i \cdot v_j - v_i \cdot v_k - v_j \cdot v_k \ge -1, \quad 1 \le i,j,k \le n \\ & ||v_j|| = 1, \quad v_j \in R^{n+1}, \quad j = 1, \cdots, n. \end{aligned} \tag{2.2}$$

To state it precisely, it is convenient to use the following notation.

$W = \sum_{i<j} w_{ij}$ — The total weight of the edges of the graph.

$A = w^{\text{SDP}}/W$ — The ratio between the solution of the relaxation (2.2) and the total weight of the edges in the graph.

$B = w^*/W$ — The ratio between the solution of the relaxation (2.1) and the total weight of the edges in the graph.

Clearly $w^* \le w^{\text{SDP}} \le W$. By inductive method, one can prove that $w^* \ge W/2$. So we get that $1/2 \le B \le A \le 1$.

We present an SDP-based approximation algorithm for MB as follows.

SDP-Algorithm

Step 1. **SDP Solving:** Solve (2.2) and obtain the vectors $v_1, \cdots, v_n$.

Step 2. **Rotating:** Rotate the vectors $v_1, \cdots, v_n$ into new vectors $v_1', \cdots, v_n'$.

Step 3. **Randomized Rounding:** Choose a random hyperplane (by choosing randomly its normal r which is a vector uniformly distributed on the unit sphere), and set $S = \{i | v_i' \cdot r \ge 0\}$.

Step 4. **Vertex Swapping:** Let $\tilde{S} = S$. Do one of the following two cases:

If $|\tilde{S}| \geq \frac{n}{2}$, for each $i \in \tilde{S}$, let $\zeta(i) = \sum_{j \notin \tilde{S}} w_{ij}$ and $\tilde{S} := \{i_1, i_2, \cdots, i_{|\tilde{S}|}\}$, where $\zeta(i_1) \geq \zeta(i_2) \geq \cdots \geq \zeta(i_{|\tilde{S}|})$. Then, remove vertex $i_{|\tilde{S}|}$ from $\tilde{S}$ and reassign $\tilde{S} := \{i_1, i_2, \cdots, i_{|\tilde{S}|-1}\}$. Repeat this swapping procedure till $|\tilde{S}| = n/2$.

If $|\tilde{S}| < \frac{n}{2}$, then for each $i \in V \setminus \tilde{S}$, let $\zeta(i) = \sum_{j \in \tilde{S}} w_{ji}$ and $V \setminus \tilde{S} = \{i_1, i_2, \cdots, i_{|V\setminus\tilde{S}|}\}$, where $\zeta(i_1) \geq \zeta(i_2) \geq \cdots \geq \zeta(i_{|V\setminus\tilde{S}|})$. Then, add node $i_{|V\setminus\tilde{S}|}$ to $\tilde{S}$ and reassign $V \setminus \tilde{S} := \{i_1, i_2, \cdots, i_{|V\setminus\tilde{S}|-1}\}$. Repeat this swapping process till $|\tilde{S}| = \frac{n}{2}$.

The rotations of the vectors in Step 2 will be outward rotations (see [11], [12] and [14]). Let $\rho \in [0,1]$ be a constant. Let $u_1, \cdots, u_n$ be n unit vectors such that for each $i \neq j$, u_i and u_j are orthogonal, and u_i is orthogonal to the space spanned by $v_1, \cdots, v_n$. We then set $v_i' = \sqrt{\rho} v_i + \sqrt{1-\rho} u_i$, for $1 \leq i \leq n$. Thus, $||v_i'|| = 1$, and for $i \neq j$, $v_i' \cdot v_j' = \rho(v_i \cdot v_j)$.

For any $U \subset V$, denote the total weights within the cut $(U, V \setminus U)$ as $w(U, V \setminus U)$ or $w(U)$, that is:

$$w(U) := \sum_{(i,j) \in E, i \in U, j \in V \setminus U} w_{ij}.$$

Clearly, the construction of bisection $\tilde{S}$ guarantees that

Lemma 2.1.

$$w(\tilde{S}) \geq \begin{cases} \frac{n/2}{|S|} \cdot w(S) & \text{if} \quad |S| \geq \frac{n}{2}, \\ \frac{n/2}{n - |S|} w(S) & \text{if} \quad |S| < \frac{n}{2}. \end{cases}$$

3. Analysis of the SDP-Algorithm

Goemans and Williamson [7] define $h(t) = \arccos(1-2t)/\pi$. Let t_1 be the value of t attaining the minimum of $h(t)/t$ in the interval $(0,1]$. The value of t_1 is approximately 0.84458. For every $0 \leq \rho \leq 1$, let

$$h_\rho(t) = \frac{1}{\pi} \arccos(\rho(1-2t)). \tag{3.1}$$

Let t_ρ be the value of t attaining the minimum of $\frac{h_\rho(t) - h_\rho(0)}{t}$ in the interval $(0,1]$.

We are finally able to define the function which gives the performance guarantee of the SDP-Algorithm:

$$\alpha(A, \rho) = \begin{cases} \frac{h_\rho(A)}{A} & \text{if } A \geq t_\rho, \\ (\frac{1}{A} - \frac{1}{t_\rho}) h_\rho(0) + \frac{h_\rho(t_\rho)}{t_\rho} & \text{if } \frac{1}{2} \leq A \leq t_\rho. \end{cases} \tag{3.2}$$

Let $E[w]$ be the expected value of the cut $w(S, V \setminus S)$ produced by the randomized algorithms given in the previous section. Given a vector r drawn uniformly from the unit sphere, we know by the linearity of expectation that

$$\begin{aligned} E[w(S)] &= \sum_{i<j} w_{ij} \cdot \Pr[\operatorname{sgn}(v_i' \cdot r) \neq \operatorname{sgn}(v_j' \cdot r)] \\ &= \sum_{i<j} w_{ij} \cdot \frac{\arccos(\rho(v_i \cdot v_j))}{\pi}, \end{aligned} \tag{3.3}$$

where $\operatorname{sgn}(x) = 1$ if $x \geq 0$, and -1 otherwise.

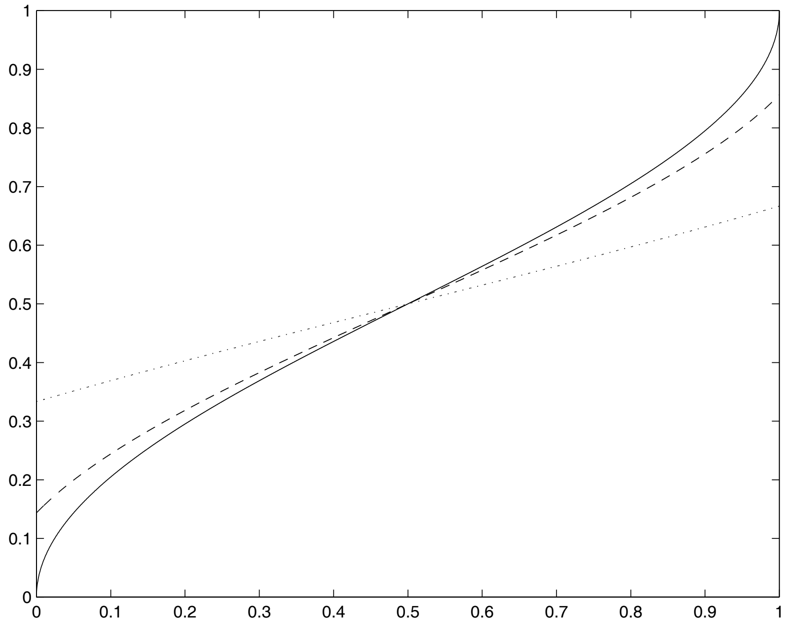

Figure 1

In the above figure, the solid line corresponds to $h_1(t)$, the dashed line corresponds to $h_{0.9}(t)$, and the dotted line corresponds to $h_{0.5}(t)$.

Lemma 3.2. *If $A \geq t_\rho$, then*

$$E[w(S)] \geq \frac{h_\rho(A)}{A} w^{\text{SDP}}.$$

Proof. Letting $\lambda_e = w_{ij}/W$ and $x_e = (1 - v_i \cdot v_j)/2$ for $e = (i,j)$, we can rewrite A as $A = \sum_e \lambda_e x_e$. The expected weight of the cut $w(S)$ produced by the algorithm is equal to

$$\begin{aligned} E[w(S)] &= \sum_{i<j} w_{ij} \cdot \frac{\arccos(\rho(v_i \cdot v_j))}{\pi} \\ &= W \sum_e \lambda_e \frac{\arccos(\rho(1 - 2x_e))}{\pi} \\ &= W \sum_e \lambda_e h_\rho(x_e). \end{aligned} \tag{3.4}$$

To bound $E[w(S)]$, we evaluate

$$\begin{aligned} \min \quad & \sum_e \lambda_e h_\rho(x_e) \\ \text{s.t.} \quad & \sum_e \lambda_e x_e = A \\ & 0 \leq x_e \leq 1. \end{aligned} \tag{3.5}$$

Consider the relaxation obtained by replacing $h_\rho(t)$ by the largest (pointwise) convex function $\tilde{h}_\rho(t)$ smaller or equal to $h_\rho(t)$ (see Figure 2). It is easy to see that $\tilde{h}_\rho(t)$ is linear with slope $\dfrac{h_\rho(t_\rho)-h_\rho(0)}{t_\rho}$ between 0 and t_ρ, and then equal to $h_\rho(t)$ for any t greater than t_ρ. But for $A \geq t_\rho$,

$$\sum_e \lambda_e h_\rho(x_e) \geq \sum_e \lambda_e \tilde{h}_\rho(x_e) \geq \tilde{h}_\rho(\sum_e \lambda_e x_e) = \tilde{h}_\rho(A) = h_\rho(A), \tag{3.6}$$

where we have used the fact that $\sum_e \lambda_e = 1$, $\lambda_e \geq 0$ and that $\tilde{h}$ is a convex function. This shows that

$$E[w(S)] \geq W h_\rho(A) = \frac{h_\rho(A)}{A} w^{\mathrm{SDP}}, \tag{3.7}$$

proving the result.

Lemma 3.3. *If* $\frac{1}{2} \leq A \leq t_\rho$, *then*

$$E[w(S)] \geq [(\frac{1}{A} - \frac{1}{t_\rho}) h_\rho(0) + \frac{h_\rho(t_\rho)}{t_\rho}] w^{\mathrm{SDP}}.$$

Proof. When $\frac{1}{2} \leq A \leq t_\rho$, we have that (see Figure 2)

$$\begin{aligned} \tilde{h}_\rho(A) &= \tilde{h}_\rho[(1 - \frac{A}{t_\rho}) \cdot 0 + \frac{A}{t_\rho} \cdot t_\rho] \\ &= (1 - \frac{A}{t_\rho}) \tilde{h}_\rho(0) + \frac{A}{t_\rho} \tilde{h}_\rho(t_\rho) \\ &= (1 - \frac{A}{t_\rho}) h_\rho(0) + \frac{A}{t_\rho} h_\rho(t_\rho). \end{aligned}$$

Similar to (3.4)-(3.7), we can prove the lemma.

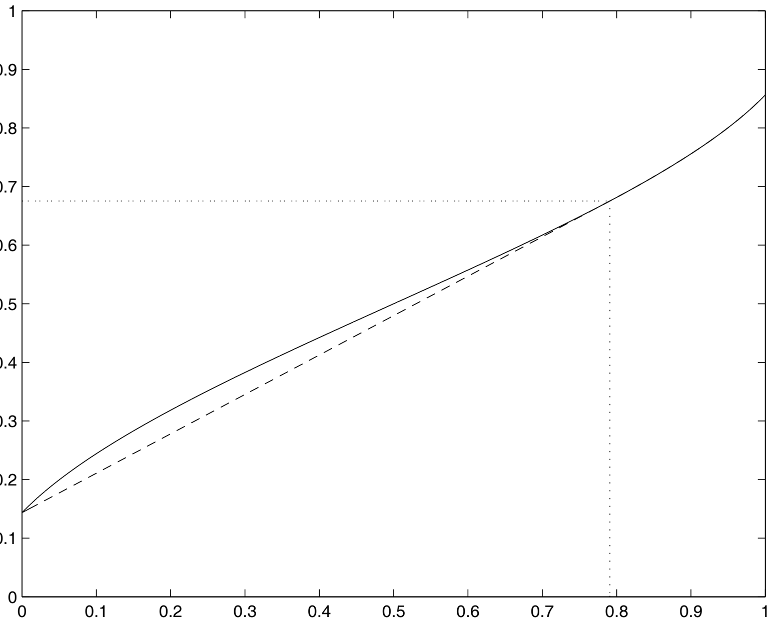

Figure 2

In the above figure, the solid line corresponds to h_ρ, the dashed line corresponds to $\tilde{h}_\rho$, the horizontal dotted line corresponds to $t = t_\rho$, and the vertical dotted line corresponds to $h_\rho(t) \equiv h_\rho(t_\rho)$.

Lemma 3.4. *The expectation of $w(S)$ of the SDP-Algorithm satisfies the following inequality*

$$E[w(S)] \geq \alpha(A,\rho) w^{\mathrm{SDP}} \geq \alpha(A,\rho) w^*.$$

Proof. It follows from Lemma 3.3 and 3.4.

In order to analyze the algorithm, we will need estimation on the expectation of the random variable $C = |S|(n-|S|)$. This random variable is a function of the configuration of the vectors and the specific outward rotation used. We need to find a constant $\gamma(\rho)$, which satisfy that for every configuration of the vectors that satisfy the constraints of the semidefinite program the following holds:

$$E[C] \geq \gamma(\rho)\frac{n^2}{4}.$$

Note that

$$\begin{aligned} E[|S|(n-|S|)] &= \sum_{i<j} \Pr[\operatorname{sgn}(v_i' \cdot r) \neq \operatorname{sgn}(v_j' \cdot r)] \\ &= \sum_{i<j} \frac{\arccos(\rho(v_i \cdot v_j))}{\pi}. \end{aligned}$$

Consider the following two numbers $\gamma_1(\rho)$, $\gamma_2(\rho)$ defined by:

$$\begin{aligned} \gamma_1(\rho) = &\min_{-\frac{1}{3}\leq y \leq 0} \frac{2}{\pi(1-y)}(\arccos(\rho y) - (y + \frac{1-y}{n}) \cdot \arccos(\rho)), \\ \gamma_2(\rho) = &\min_{-1\leq x \leq -\frac{1}{3}} \frac{1}{\pi}(\frac{(1-3x)\arccos(\rho)}{4} - \frac{2\arccos(\rho)}{n} \\ &+ \frac{3(x+1)\arccos(-\frac{\rho}{3})}{4} + \arccos(\rho x)). \end{aligned}$$

By the triangle inequality, Halperin and Zwick [8] obtain the followings lemma:

Lemma 3.5. *The SDP-Algorithm yields S satisfying the following inequalities:*

$$E[|S|(n-|S|)] \geq \frac{n^2}{4}\gamma(\rho),$$

where $\gamma(\rho) = \min\{\gamma_1(\rho), \gamma_2(\rho)\}$.

Define a new random variable as

$$z(c) := \frac{w(S)}{w^*} + c\frac{4|S|(n-|S|)}{n^2}, \quad c > 0.$$

Then we have

$$E[z(c)] \geq \alpha + c\gamma.$$

On the other hand,

$$z \leq 2 + c.$$

Similar to [12], we have the following lemma:

Lemma 3.6. *If random variable $z(c)$ fulfills its expectation, i.e., $z(c) \geq \alpha + c\gamma$, then*

$$w(\tilde{S}) \geq R(A, \rho) \cdot w^*,$$

where

$$R(A, \rho) = \frac{\alpha}{1 + \sqrt{1-\gamma}}.$$

We solve the maximization problem

$$R(A) = \max_{\rho \in [0,1]} R(A, \rho).$$

Note that α and γ are also functions of ρ.

Then together with Lemma 3.6 and Frieze and Jerrum's analysis [1] yield the final result:

Theorem 3.7. *The worst-case performance ratio of the SDP-Algorithm for the MB problem is at least $R(A)$ for sufficiently large n.*

Performance ratio of the SDP-Algorithm

We have calculated the following table, which shows that $R(A) > 0.7016$ in many cases. Also see Figure 3.

In the above figure, the solid line corresponds to our $R(A)$, the dashed line corresponds to Halperin and Zwick's ratio.

Figure 3 indicates that the hard case of the SDP-Algorithm is $A \in [0.90, 0.92]$. When A reduces from 0.90 to 0.50, the performance ratio increases from 0.7019 to 1.0. Considering the case $A = 0.5$, we deduce that $w^{\mathrm{SDP}} = w^*$. So it is natural that the performance ratio equals 1.0. We guess that A is close to B and w^{SDP} is close to w^* when A reduces from 0.90 to 0.50. When A increases from 0.92 to 1.0, the performance ratio increases from 0.7018 to 0.7456. We guess that A is close to B when A increases from 0.92 to 1.0.

Acknowledgments. The authors would like to thank Prof. Yinyu Ye for his useful comments, Dr. Jiawei Zhang for his helpful dicussions, and Prof. Uriel Feige and Prof. Uri Zwick for kindly giving us their nice papers. Thanks are also due to the two anonymous referees for their suggestions to improve the presentation of our paper.

Table 1: Worst-case performance ratios of the SDP-Algorithm for $A \in [0.50, 1.00]$.

A	ρ	t_ρ	α	γ	$R(A)$
0.50	0.00	0.9999	1	1.0000	1.0000
0.51	0.00	0.9999	0.9804	1.0000	0.9804
0.52	0.01	0.7500	0.9618	1.0000	0.9616
0.53	0.02	0.7500	0.9441	1.0000	0.9437
0.54	0.03	0.7500	0.9273	1.0000	0.9266
0.55	0.06	0.7501	0.9126	1.0000	0.9104
0.56	0.09	0.7502	0.8990	1.0000	0.8951
0.57	0.13	0.7505	0.8873	0.9999	0.8807
0.58	0.15	0.7506	0.8752	0.9999	0.8671
0.59	0.19	0.7510	0.8658	0.9998	0.8544

A	ρ	t_ρ	α	γ	$R(A)$
0.60	0.23	0.7515	0.8576	0.9997	0.8425
0.61	0.27	0.7521	0.8505	0.9995	0.8314
0.62	0.31	0.7528	0.8445	0.9992	0.8211
0.63	0.36	0.7539	0.8407	0.9987	0.8115
0.64	0.38	0.7544	0.8339	0.9985	0.8025
0.65	0.43	0.7557	0.8322	0.9977	0.7942
0.66	0.45	0.7563	0.8269	0.9973	0.7864
0.67	0.49	0.7576	0.8254	0.9965	0.7792
0.68	0.52	0.7587	0.8230	0.9957	0.7725
0.69	0.55	0.7599	0.8214	0.9948	0.7662
0.70	0.58	0.7612	0.8204	0.9938	0.7604
0.71	0.60	0.7622	0.8181	0.9930	0.7549
0.72	0.63	0.7638	0.8184	0.9916	0.7498
0.73	0.65	0.7649	0.8172	0.9906	0.7450
0.74	0.67	0.7661	0.8164	0.9895	0.7405
0.75	0.69	0.7674	0.8161	0.9883	0.7363
0.76	0.71	0.7688	0.8163	0.9869	0.7324
0.77	0.73	0.7703	0.8169	0.9853	0.7287
0.78	0.74	0.7711	0.8154	0.9845	0.7252
0.79	0.76	0.7728	0.8168	0.9827	0.7220
0.80	0.77	0.7737	0.8161	0.9817	0.7190
0.81	0.78	0.7747	0.8156	0.9807	0.7162
0.82	0.80	0.7767	0.8184	0.9785	0.7137
0.83	0.80	0.7767	0.8157	0.9785	0.7113
0.84	0.82	0.7789	0.8194	0.9759	0.7093
0.85	0.82	0.7789	0.8172	0.9759	0.7074
0.86	0.84	0.7814	0.8218	0.9730	0.7058
0.87	0.84	0.7814	0.8201	0.9730	0.7044
0.88	0.86	0.7841	0.8258	0.9696	0.7033
0.89	0.87	0.7856	0.8286	0.9678	0.7025
0.90	0.88	0.7872	0.8318	0.9658	0.7019
0.91	0.89	0.7890	0.8356	0.9636	0.7017
0.92	0.90	0.7908	0.8401	0.9612	0.7018
0.93	0.91	0.7929	0.8453	0.9585	0.7023
0.94	0.92	0.7951	0.8514	0.9556	0.7033
0.95	0.93	0.7975	0.8586	0.9524	0.7048
0.96	0.94	0.8002	0.8672	0.9487	0.7071
0.97	0.95	0.8032	0.8777	0.9445	0.7104
0.98	0.96	0.8068	0.8909	0.9395	0.7151
0.99	0.98	0.8163	0.9193	0.9259	0.7226
1.00	1.00	0.8446	1	0.8836	0.7456

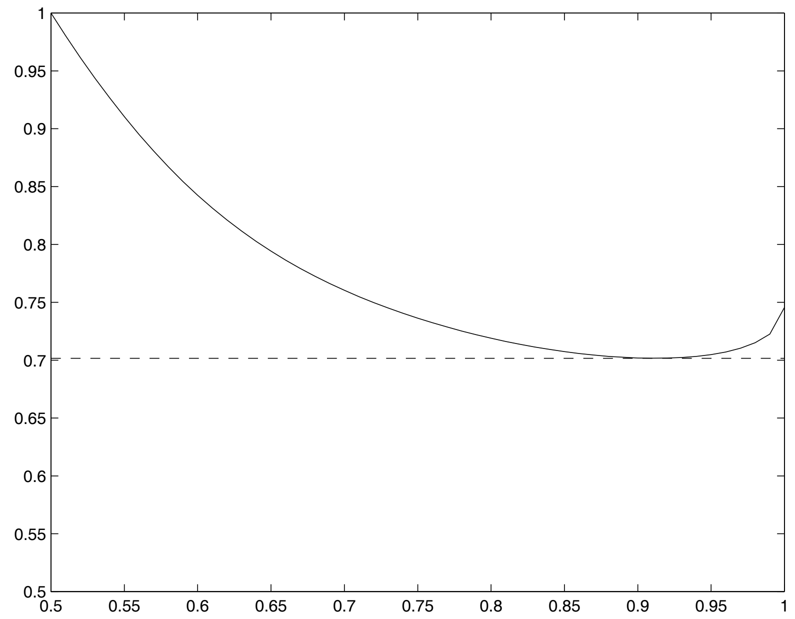

Figure 3

References

[1] A. Frieze and M. Jerrum, Improved approximation algorithms for max k-cut and max bisection, in Proc. 4th IPCO Conference, pages 1-13, 1995.

[2] U. Feige, Randomized rounding of semidefinite programs – variations on the MAX CUT example, Randomization, Approximation, and Combinatorial Optimization, Proceedings of Random-Approx'99, Lecture Notes in Computer Science 1671, Springer 1999, pp. 189–196.

[3] U. Feige and M. X. Goemans, Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT, In Proceedings of the 3nd Israel Symposium on Theory and Computing Systems, pages 182-189, Tel Aviv, Israel, 1995.

[4] U. Feige, M. Karpinski, and M. Langberg, A note on approximating MAX-BISECTION on regular graphs, Electronic Colloquium on Computational Complexity, Report No. 43, 2000.

[5] U. Feige, M. Karpinski, and M. Langberg, Improved approximation of Max-Cut on graphs of bounded degree, available at *URL http://www.wisdom.weizmann.ac.il/home/feige/public_html/index.html*, 2000.

[6] U. Feige and M. Langberg, Approximation algorithms for maximization problems arising in graph partitioning, *Journal of Algorithms,* **41** (2001), 174-211.

[7] 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.

[8] E. Halperin and U. Zwick, Improved approximation algorithms for maximum graph bisection problems, *Proc. of 8th IPCO*, pages 210-225, 2001, To appear in *Random Structures and Algorithms.*

[9] Q. Han, Y. Ye, H. Zhang, and J. Zhang, On approximation of Max-Vertex-Cover, *European Journal of Operational Research*, **143**:2 (2002), 207-220.

[10] Q. Han, Y. Ye, and J. Zhang, An improved rounding method and semidefinite programming relaxation for graph partition, *Mathematical Programming*, **92**:3 (2002), 509-535.

[11] Y. Nesterov, Semidefinite relaxation and nonconvex quadratic optimization, *Optimization Methods and Software,* **9** (1998), 141-160.

[12] Y. Ye, A .699-approximation algorithm for Max-Bisection, *Mathematical Programming*, **90** (2001), 101-111.

[13] Y. Ye and J. Zhang, Approximation of Dense-n/2-Subgraph and the Complement of Min-Bisection, *Journal of Global Optimization*, **25** (2003), 55-73.

[14] U. Zwick, Outward rotations: a new tool for rounding solutions of semidefinite programming relaxations, with applications to Max-Cut and other problems, in Proceedings of the 31th Symposium on Theory of Computing, pages 679-687, 1999.