Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to MAX CUT and other problems

Uri Zwick *

Abstract

We present a tool, *outward rotations*, for enhancing the performance of several semidefinite programming based approximation algorithms. Using outward rotations, we obtain an approximation algorithm for MAX CUT that, in many interesting cases, performs better than the algorithm of Goemans and Williamson. We also obtain an improved approximation algorithm for MAX NAE-{3}-SAT. Finally, we provide some evidence that outward rotations can also be used to obtain improved approximation algorithms for MAX NAE-SAT and MAX SAT.

1 Introduction

MAX CUT is perhaps the simplest and most natural APX-complete constraint satisfaction problem (see, e.g., [AL97]). There are various simple ways of obtaining a performance guarantee of 1/2 for the problem. One of them, for example, is just choosing a random cut. No performance guarantee better than 1/2 was known for the problem until Goemans and Williamson [GW95], in a major breakthrough, used semidefinite programming to obtain an approximation algorithm for MAX CUT with a performance guarantee of $\alpha \simeq 0.87856$. Goemans and Williamson show that their algorithm behaves even better on graphs that have very large cuts, or more precisely, on graphs that have cuts that contain more than 85% of the edges of the graph, or more than 85% of the total weight of the edges, if the edges are weighted.

Most graphs, however, do not have cuts that account for more than 85% of their edges. For such graphs, the algorithm of Goemans and Williamson is only guaranteed to produce a cut whose size is $\alpha \simeq 0.87856$ times the size of the the maximum cut in the graph. We present here are variant of the algorithm of Goemans and Williamson with a provably better performance guarantee on graphs that do not have huge cuts. For example, given a graph whose maximum cut contains only 75% of the edges of the graph, the new algorithm will produce a cut containing at least 66% of the edges of the graph, a performance ratio of more than 0.88. Given a graph whose maximum cut contains only 60% of the edges of the graph, the new algorithm will produce a cut containing at least 54.7% of the edges of the graph, a performance ratio of more than 0.91. The performance guarantee of the new algorithm is strictly greater than $\alpha \simeq 0.87856$, unless the maximum cut of the input graph contains about 84.458% of the edges of the graph.

Perhaps more interesting than the specific results mention above is the method used to obtain them. The new MAX CUT algorithm is obtained by employing a new tool, *outward rotations*, during the rounding phase of the algorithm. This tool was previously used by Nesterov [Nes98] but was applied to different problems. We further demonstrate the usefulness of this tool by presenting several other applications. Using outward rotations we obtain a 0.90871-approximation algorithm for MAX NAE-{3}-SAT (Maximum Not-All-Equal {3}-SAT), substantially improving on the previously known result. Finally, we provide some numerical evidence that suggests that outward rotations could also be used to improve the performance guarantees of existing algorithms for MAX NAE-SAT and MAX SAT, though proving it rigorously would require overcoming huge technical difficulties.

2 Outward rotations

Let $v_1, v_2, \ldots, v_n$ be unit vectors obtained as an optimal solution to a semidefinite programming relaxation of a combinatorial optimization problem. In many cases we would like to 'round' the vectors into 0-1 values

*Department of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. E-mail: zwick@math.tau.ac.il.

$x_1, \ldots, x_n$. One common way of doing it, first used by Goemans and Williamson [GW95] in their approximation algorithm for MAX CUT, is to choose a random vector r and assign x_i the value 1 if $r \cdot v_i \geq 0$, and the value 0 otherwise. This is equivalent to choosing a random hyperplane and assigning vectors that fall on its positive side the value 1, and those that fall on its negative side the value 0.

In some cases, the solution of the semidefinite relaxation also includes a vector v_0 that plays a special role. This is the case, for example, in the semidefinite relaxations of MAX 2-SAT and MAX DICUT [GW95],[FG95], and MAX 3-SAT, MAX 3-CSP and MAX 4-SAT [KZ97],[Zwi98a],[HZ99]. In this case, it is possible to rotate the vectors $v_1, \ldots, v_n$ toward or away from v_0 before rounding them using a random hyperplane. This was first used by Feige and Goemans [FG95] and later in [Zwi98a],[Zwi98b] and [HZ99].

In some cases, it is advantageous to combine the hyperplane rounding with independent random choice. For such examples, see [Zwi98a]. The naive way of doing this is to produce two 0-1 assignments to the variables $x_1, \ldots, x_n$, one by rounding the vectors $v_1, \ldots, v_n$ using a random hyperplane, and the other by simply choosing a random assignment, and then return the better assignment. A variant of this approach is to return the first assignment with probability p, and the second assignment with probability $1-p$, for some $0 \leq p \leq 1$. (It is better, of course, to take the better assignment, but the variant is usually easier to analyze and in many cases gives the same performance guarantee.)

In this paper, we describe a new way of combining hyperplane rounding with independent random choice. This way makes a better use of the information contained in the solution of the semidefinite program.

Let $u_1, \ldots, u_n$ be an orthonormal system of vectors. This means that $u_1, \ldots, u_n$ are unit vectors and that u_i and u_j are orthogonal, i.e., $u_i \cdot u_j = 0$, for every $i \neq j$. It is easy to see that rounding $u_1, \ldots, u_n$ using a random hyperplane generates a completely random assignment.

This suggests the following way of combining random hyperplane rounding with random choice. Let $v_1, \ldots, v_n$ be the solution of the program. We may assume, without loss of generality, that $v_1, \ldots, v_n$ lie in $\mathbb{R}^n$. It is easy, however, to embed $v_1, \ldots, v_n$ in $\mathbb{R}^{2n}$. This is easily done by adding n zero coordinates to each one of these vectors. Let $u_1, \ldots, u_n$ be an orthonormal system of unit vectors that is also orthogonal to the vectors $v_1, \ldots, v_n$. In other words, $u_i \cdot u_j = 0$ for every $i \neq j$ and $v_i \cdot u_j = 0$ for every $1 \leq i, j \leq n$. An easy way of choosing $u_1, \ldots, u_n$ is to let u_i be a vector whose $(n+i)$-th coordinate is 1 and all other coordinates are 0. Let $0 \leq \gamma \leq \pi/2$ be an angle. We can rotate each vector v_i by an angle of γ toward u_i in the plane spanned by v_i and u_i. (There are actually two ways of doing this, both are equivalent, however.) Let $v'_1, \ldots, v'_n$ be the vectors obtained in this way. We can now round $v'_1, \ldots, v'_n$ using a random hyperplane. By letting γ vary from 0 to $\pi/2$, we get various combinations of random hyperplane rounding and independent random choice.

What is the effect of these rotations on the angles between the vectors? Let $\theta_{ij} = \arccos(v_i \cdot v_j)$ be the angle between v_i and v_j. Let $\theta'_{ij} = \arccos(v'_i \cdot v'_j)$ be the angle between v'_i and v'_j. As $v'_i = (\cos\gamma)v_i + (\sin\gamma)u_i$, for $1 \leq i \leq n$, we immediately get that $v'_i \cdot v'_j = (\cos^2\gamma)(v_i \cdot v_j)$, for every $i \neq j$, and therefore

$$\cos\theta'_{ij} = \cos^2\gamma \cdot \cos\theta_{ij} \quad , \quad \text{for } i \neq j \; .$$

Assuming that $0 < \gamma < \pi/2$, so that $0 < \cos^2\gamma < 1$, we get that if $\theta_{ij} > \pi/2$, then $\pi/2 < \theta'_{ij} < \theta_{ij}$, and if $\theta_{ij} < \pi/2$, then $\theta_{ij} < \theta'_{ij} < \pi/2$.

There is an alternative way of viewing these 'outward' rotations. (We call them 'outward' rotations as the vectors $v_1, \ldots, v_n$ are rotated out of the original $\mathbb{R}^n$ space in which they were defined.) Let $Y = (y_{ij})$ and $Y' = (y'_{ij})$ be the matrices of inner products of $v_1, \ldots, v_n$ and $v'_1, \ldots, v'_n$, i.e., $y_{ij} = v_i \cdot v_j$ and $y'_{ij} = v'_i \cdot v'_j$. Clearly Y and Y' are positive semidefinite. Note that

$$Y' = (\cos^2\gamma)Y + (\sin^2\gamma)I \; .$$

In other words, outward rotations simply multiply each non-diagonal element in the matrix Y by a constant $0 \leq c \leq 1$, where $c = \cos^2\gamma$. As Y' is a convex combination of the two positive semidefinite matrices Y and I, and as the set of positive semidefinite matrices is a convex set, we get an alternative way of seeing that the matrix Y' is positive semidefinite.

We have seen above that outward rotations decrease angles that are larger than $\pi/2$ and increase angles that are below $\pi/2$. Outward rotations are only expected to help, therefore, for problems in which the worst performance guarantees are attained at configurations in which some of the angles are below $\pi/2$. We give several such examples in the following sections.

3 MAX CUT

Let $G = (V, E)$ be a weighted undirected graph. We may assume, without loss of generality, that $V = \{1, 2, \ldots, n\}$. We also assume that the weights of the edges are non-negative. A *cut* in G is a partition of the vertices of G into two sets S and $V - S$. The weight of the cut is $\sum_{e:|e \cap S|=1} w(e)$, where $w(e)$ is the weight of the edge e. MAX CUT is the problem of finding a cut of maximal weight. MAX CUT is APX-complete so there is no polynomial time approximation scheme for it, unless P=NP. Goemans and Williamson

$W = \sum_{i<j} w_{ij}$	–	The total weight of the edges of the graph.
$Z = \sum_{i<j} w_{ij} \frac{1-v_i \cdot v_j}{2}$	–	The value of an optimal solution of the semidefinite relaxation.
M	–	The maximal weight of a cut in the graph.
$E[C] = \sum_{i<j} w_{ij} \frac{\theta_{ij}}{\pi}$	–	The expected weight of a cut obtained by rounding an optimal solution of the semidefinite program using a random hyperplane.
$A = Z/W$	–	The ratio between the solution of the relaxation and the total weight of the edges in the graph.
$B = M/W$	–	The ratio between the maximal weight of a cut in the graph and the total weight of the edges in the graph.

Figure 1: Useful notation for the analysis of the Goemans and Williamson MAX CUT algorithm.

[GW95] consider the following semidefinite relaxation of the MAX CUT problem:

$$\begin{array}{ll} \text{Maximize} & \displaystyle\sum_{i<j} w_{ij} \frac{1 - v_i \cdot v_j}{2} \\ \text{subject to :} & v_i \cdot v_i = 1 \ , \ \text{for } 1 \le i \le n \ , \end{array}$$

where $w_{ij} \ge 0$ is the weight of the edge (i,j). They show that if an optimal solution of this semidefinite program is rounded using a random hyperplane, then the expected weight of the cut obtained is at least $\alpha \simeq 0.87856$ of the weight of the optimal cut. More precisely, $\alpha = \min_{0 \le \theta \le \pi} \frac{2}{\pi} \frac{\theta}{1-\cos\theta}$. It is not difficult to check the this minimum is attained at $\theta_0 \simeq 2.331122$, the non-zero root of the equation $\cos\theta + \theta \sin\theta = 1$.

The result of Goemans and Williamson is in fact stronger. To state is precisely, it is convenient to use the notation given in Figure 1.

Clearly $E[C] \le M \le Z \le W$. As $M \ge W/2$, we get that $1/2 \le B \le A \le 1$. The main result of Goemans and Williamson [GW95], stated above, is

$$E[C] \ \ge \ \alpha Z \ \ge \ \alpha M \ .$$

Let $h(t) = \arccos(1-2t)/\pi$ and let $t_0 = (1-\cos\theta_0)/2 \simeq 0.84458$ be the value of t attaining the minimum of $h(t)/t$ in the interval $(0,1]$. (Note that $h(t_0)/t_0 = \alpha \simeq 0.87856$.) Let

$$\alpha(A) = \begin{cases} h(A)/A & \text{if } A \ge t_0 \ , \\ h(t_0)/t_0 & \text{if } A \le t_0 \ . \end{cases}$$

Goemans and Williamson [GW95] show, in fact, that

$$E[C] \ \ge \ \alpha(A) \cdot Z \ \ge \ \alpha(A) \cdot M \ \ge \ \alpha(B) \cdot M \ .$$

As $\alpha(A) \ge \alpha \simeq 0.87856$, this result is always at least as strong as the previous bound on $E[C]$. For $A > t_0 \simeq 0.84458$, this result is strictly stronger. The function $\alpha_{GW}(A) = \alpha(A)$ is represented by the solid line in Figure 2.

Note that as A tends to 1, $\alpha(A)$ also tends to 1. This means that as the relative size of the maximum cut in the graph gets larger and larger it is easier to approximate it. At the limit, when $A = B = 1$, the graph is bipartite and a cut that cuts all the edges can be easily found in linear time.

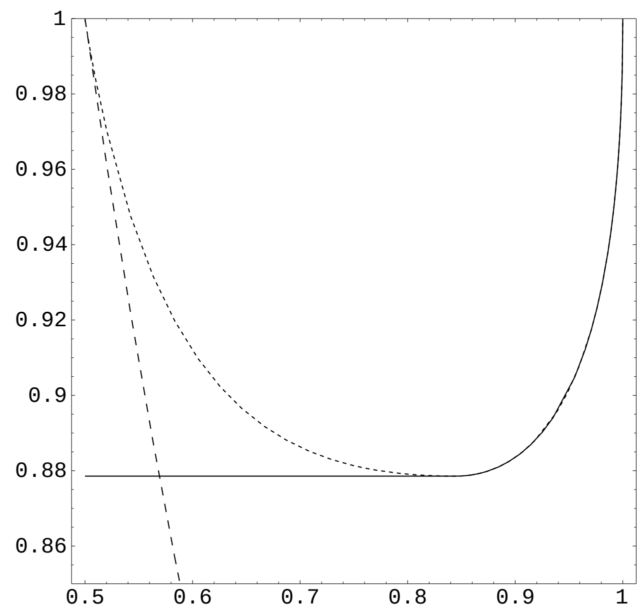

Figure 2: The functions $\alpha_{\mathrm{GW}}(A), \alpha_{\mathrm{RAND}}(A)$ and $\alpha_{\mathrm{ROT}}(A)$ giving the performance ratios of the GW algorithm (solid line), the random algorithm (dashed line) and the new algorithm that uses outward rotations (dotted line) as a function of A.

Alon and Sudakov [AS98], extending a result of Karloff [Kar96], show that the simple analysis of the algorithm of Goemans and Williamson is tight for every $A \ge t_0 \simeq 0.84458$. More precisely, for any *rational* $A \ge t_0 \simeq 0.84458$, there are infinitely many graphs for which $E[C] = \alpha(A)Z = \alpha(B)M$. Alon, Sudakov and Zwick [ASZ98] extended this result even further and showed that the analysis of of the algorithm of Goemans and Williamson is tight for every $1/2 \le A \le 1$.

The performance ratio of the GW algorithm stays unchanged for $1/2 \leq A \leq t_0 \simeq 0.84458$. When A is close enough to $1/2$, the weight of a *random* cut is expected to be larger than the weight of the cut produced by the GW algorithm. As the expected weight of a random cut is $W/2$, the performance ratio achieved by choosing a random cut is $\alpha_{\text{RAND}}(A) = 1/(2A)$. The function $\alpha_{\text{RAND}}(A)$ is represented by the dashed line in Figure 2. When $1/2 \leq A \leq 1/(2\alpha) \simeq 0.569113$, the algorithm that produces a random cut is better than the GW algorithm.

Using outward rotations, we present here a third algorithm whose performance ratio $\alpha_{\text{ROT}}(A)$ is given by the dotted line in Figure 2. As can be seen from the figure, the performance ratio of the new algorithm is better than both the GW and the random algorithm for every $1/2 < A < t_0 \simeq 0.84458$.

When $A \geq t_0$, the worst case of the algorithm of Goemans and Williamson is achieved when for every edge $(i,j) \in E$ of the graph we have $\theta_{ij} = \arccos(1-2A) \geq \theta_0 \simeq 2.331122$. As all the angles here are larger than $\pi/2$, outward rotations cannot help in this case. However, when $A < t_0$, the worst case of the algorithm of Goemans and Williamson is achieved when for every edge $(i,j) \in E$ of the graph, either $\theta_{ij} = \theta_0$ or $\theta_{ij} = 0$. Now some of the angles are smaller than $\pi/2$ and outward rotations may potentially help. We show in the sequel that they do.

The new approximation algorithm for MAX CUT is given in Figure 3. The algorithm begins, as the algorithm of Goemans and Williamson by solving the semidefinite programming relaxation of the MAX CUT instance. [1] Given the solution $v_1, \ldots, v_n$, the algorithm computes $A = (\sum_{i,j} w_{ij} \frac{1-v_i \cdot v_j}{2})/(\sum_{ij} w_{ij})$, the ratio between the value of the optimal solution of the relaxation and the total weight of the edges of the graph. Recall that we always have $1/2 \leq A \leq 1$. Given A, the algorithm computes an angle $\gamma = \gamma(A)$ satisfying $0 \leq \gamma \leq \frac{\pi}{2}$. The algorithm then outward rotates the vectors $v_1, \ldots, v_n$ by an angle of γ and rounds the resulting vectors using a random hyperplane. If $A \geq t_0 \simeq 0.84458$ then $\gamma(A) = 0$. In this case, no outward rotations are performed and the algorithms does not differ from the algorithm of Goemans and Williamson. However, if $A < t_0 \simeq 0.84458$, then $\gamma(A) > 0$ and the algorithm rotates the vectors before rounding them using a random hyperplane. The angle of rotation $\gamma = \gamma(A)$ is chosen to maximize the worst-case performance ratio of the algorithm. The definition of $\gamma(A)$, for $A < t_0 \simeq 0.84458$, relies on the following lemma:

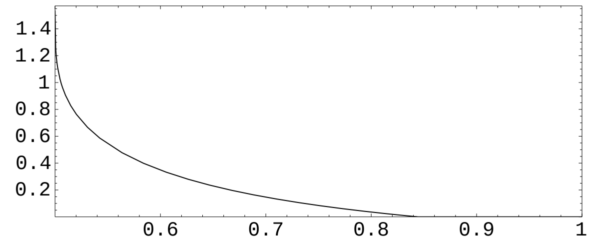

Figure 4: The function $\gamma(A)$ that gives the angle by which the vectors $v_1, \ldots, v_n$ are rotated by the new approximation algorithm for MAX CUT. Note that $\gamma(A) = 0$ for $A \geq t_0 \simeq 0.84458$ and that $\gamma(A)$ tends to $\pi/2 \simeq 1.5708$ as A tends to 0.

Lemma 3.1 *Let $1/2 \leq A \leq t_0$, where $t_0 \simeq 0.84458$ is the unique $0 < t < 1$ satisfying the equation $\frac{\arccos(t)}{t} = \frac{2}{\sqrt{1-(1-2t)^2}}$. Then, the two equations*

$$\frac{\arccos(c(1-2t)) - \arccos(c)}{t} = \frac{2c}{\sqrt{1-c^2(1-2t)^2}}$$

$$\frac{1-\frac{t}{A}}{\sqrt{1-c^2}} = \frac{1-2t}{\sqrt{1-c^2(1-2t)^2}}$$

have a unique solution $c = c(A)$ and $t = t(A)$ such that $0 \leq c \leq 1$ and $\frac{3}{4} \leq t \leq t_0$. Furthermore, $A \leq t(A)$.

The above lemma implicitly defines the function $c(A)$ and $t(A)$. The function $\gamma(A)$ is defined as $\gamma(A) = \arccos(\sqrt{c(A)})$, for $1/2 \leq A \leq t_0$. For $A \geq t_0$, we let $\gamma(A) = 0$. There seems to be no closed form description of the function $\gamma(A)$. It is easy, however, to compute $\gamma(A)$ to an arbitrary precision by numerically solving the equations that define the functions $c(A)$ and $t(A)$. A plot of the function $\gamma(A)$ is given in Figure 4. As expected, $\gamma(A)$ tends to $\pi/2$ as A tends to $1/2$. Given A, the algorithm given in Figure 3 computes $\gamma(A)$, or more precisely, an approximation of $\gamma(A)$. This can be done by either solving the equations defining $\gamma(A)$ numerically. A more efficient method is to approximate $\gamma(A)$ using a precomputed interpolation table. The size of the interpolation table depends on the required accuracy.

For every $0 \leq c \leq 1$, let

$$h_c(t) = \frac{1}{\pi} \arccos(c(1-2t)) \; .$$

We are finally able to define the function which gives the performance guarantee of the new algorithm:

$$\alpha_{\text{ROT}}(A) = \begin{cases} \frac{h(A)}{A} & \text{if } A \geq t_0, \\ (\frac{1}{A} - \frac{1}{t(A)}) h_{c(A)}(0) + \frac{h_{c(A)}(t(A))}{t(A)} & \text{if } 1/2 \leq A \leq t_0. \end{cases}$$

[1] Actually, the algorithm can only find, in polynomial time, an *approximate* solution of the semidefinite relaxation. The approximate solution can be taken to be arbitrary close to the optimal solution which is sufficient for our purposes. See [GW95] for a discussion of this point.

Algorithm **APPROX-MAX-CUT**

1. Solve the semidefinite programming relaxation of the MAX CUT instance and obtain the vectors $v_1, \ldots, v_n$ and $A = (\sum_{i,j} w_{ij} \frac{1-v_i \cdot v_j}{2}) / (\sum_{ij} w_{ij})$.
2. If $A \geq t_0 \simeq 0.84458$ then let $\gamma \leftarrow 0$.
3. If $A < t_0 \simeq 0.84458$ then solve the following equations for c and t

$$\frac{\arccos(c(1-2t)) - \arccos(c)}{t} = \frac{2c}{\sqrt{1-c^2(1-2t)^2}} \quad , \quad \frac{1-\frac{t}{A}}{\sqrt{1-c^2}} = \frac{1-2t}{\sqrt{1-c^2(1-2t)^2}}$$

   and let $\gamma \leftarrow \arccos(\sqrt{c})$. (Note that $\gamma = \gamma(A)$ depends only on A. It is therefore possible to approximate $\gamma = \gamma(A)$ using a precomputed interpolation table.)
4. Let $v'_1, \ldots, v'_n$ be vectors obtained by outward rotating the vectors $v_1, \ldots, v_n$ by an angle of γ. In other words, let $v'_1, \ldots, v'_n$ be unit vectors such that $v'_i \cdot v'_j = (\cos^2 \gamma)(v_i \cdot v_j)$, for every $1 \leq i < j \leq n$.
5. Round the vectors $v'_1, \ldots, v'_n$ using a random hyperplane.

Figure 3: The new approximation algorithm for MAX CUT.

A plot of $\alpha_{\text{ROT}}(A)$ is given by the dotted line in Figure 2. The relation between $\alpha_{\text{ROT}}(A)$ and $\alpha_{\text{GW}}(A)$ and $\alpha_{\text{RAND}}(A)$, which can be seen from Figure 2, is explicitly stated in the following lemma.

Lemma 3.2
(i) For $t_0 \leq A \leq 1$ we have $\alpha_{\text{ROT}}(A) = \alpha_{\text{GW}}(A)$.
(ii) For $1/2 \leq A < t_0$ we have $\alpha_{\text{ROT}}(A) > \alpha_{\text{GW}}(A)$.
(iii) For $1/2 < A \leq t_0$ we have $\alpha_{\text{ROT}}(A) > \alpha_{\text{RAND}}(A)$.

We now claim:

Theorem 3.3 *For a given instance of the MAX CUT problem, let W be the total weight of the edges in the input graph, let Z be the value of the semidefinite relaxation of the instance, let M be the weight of the maximum cut in the graph, and let $A = Z/W$ and $B = M/W$. Let $E[C_{\text{ROT}}]$ be the expected size of the cut returned by the new approximation algorithm. Then,*

$$E[C_{\text{ROT}}] \;\geq\; \alpha_{\text{ROT}}(A) \cdot Z \;\geq\; \alpha_{\text{ROT}}(B) \cdot M \;.$$

Proofs of Lemmas 3.1 and 3.2 and of Theorem 3.3 will appear in the full version of the paper. As presented, the new approximation algorithm is randomized. It can be derandomized, however, as can the algorithm of Goemans and Williamson, using the general technique of Mahajan and Ramesh [MR95].

Algorithm **APPROX-MAX-CUT** chooses γ as a function of A. In practice, it may be beneficial to choose γ as a function of the vectors $v_1, \ldots, v_n$ themselves, or simply try 'all' values of γ, to within some finite resolution, and see which one of them works best. A further extension is to choose a different rotation angle γ_i for each vector v_i.

4 MAX NAE-{3}-SAT

An instance of MAX NAE-{3}-SAT in the variables $x_1, x_2, \ldots, x_n$ is a weighted collection of clauses of the form $\text{NAE}_3(z_1, z_2, z_3)$, where each z_i is a *literal*, i.e., one of the variables $x_1, x_2, \ldots, x_n$ or a negation of one of the variables. The three literals appearing together in a clause must be distinct. A clause $\text{NAE}_3(z_1, z_2, z_3)$ is satisfied by an assignment of 0-1 values to the variables $x_1, x_2, \ldots, x_n$ if at least one of the literals is assigned the value 0 and at least one literal is assigned the value 1. The goal is to find a 0-1 assignment that maximizes the total weight of the satisfied clauses.

MAX NAE-{3}-SAT is also a well known APX-complete problem. It is easily seen to be a generalization of MAX {2}-SAT (each clause $z_1 \vee z_2$ is replaced by a clause $\text{NAE}_3(O, z_1, z_2)$, where O is a variable that appears in all the clauses). The version of the MAX NAE-{3}-SAT problem in which no negations are allowed is equivalent to the problem of 2-coloring the vertices of a 3-uniform hypergraph so as to maximize the number of non-monochromatic edges. This problem is also referred to as MAX 3-SET SPLITTING, see [KLP96],[AE98]. The MAX 3-SET SPLITTING problem, in turn, may be viewed as the MAX CUT problem restricted to graphs whose (weighted) edges can be (fractionally) covered by triangles.

If we adopt the notation $x_{n+i} = \bar{x}_i$, for $1 \le i \le n$, and assume that $w_{ijk} \ge 0$ is the weight of the clause $\mathrm{NAE}_3(x_i, x_j, x_k)$, where $1 \le i,j,k \le 2k$, then the following is easily seen to be a semidefinite programming relaxation of MAX NAE-{3}-SAT:

$$\text{Maximize} \qquad \sum_{i,j,k} w_{ijk} \frac{3 - v_i \cdot v_j - v_i \cdot v_k - v_j \cdot v_k}{4}$$

subject to:

$$v_i \cdot v_i = 1 \;,\; v_i \cdot v_{n+i} = -1 \;,\; \text{for } 1 \le i \le n \;,$$
$$v_i \cdot v_j + v_i \cdot v_k + v_j \cdot v_k \ge -1 \;,\; \text{for } 1 \le i,j,k \le 2n \;.$$

If we solve this SDP relaxation and round the solution using a random hyperplane, we obtain an α-approximation algorithm for MAX NAE-{3}-SAT, where $\alpha \simeq 0.87856$ is the performance ratio of the MAX CUT approximation algorithm of Goemans and Williamson. The ratio α is attained if for every clause $\mathrm{NAE}_3(x_i, x_j, x_k)$ for which $w_{ijk} > 0$ we have $\theta_{ij} = \theta_{ik} = \theta_0 \simeq 2.331122$ and $\theta_{jk} = 0$, possibly after a suitable rearrangement of the indices.

A marginally improved performance guarantee of $\frac{3}{2+\sqrt{2}} \simeq 0.87867$ for MAX NAE-{3}-SAT was obtained by the author in [Zwi98a]. It was also shown there that a performance ratio of $\frac{3}{2\pi}\arccos(-\frac{1}{3}) \simeq 0.91226$ can be obtained for *satisfiable* instances of MAX NAE-{3}-SAT. The improved ratio of $\frac{3}{2+\sqrt{2}}$ was obtained using a simple combination of random hyperplane rounding and random choice. Here we obtain a more substantial improvement using outward rotations.

Theorem 4.1 *Let* $\theta \simeq 2.408874$, $\gamma \simeq 0.145455$ *and* $\beta \simeq 0.908718$ *be solutions of the equations*

$$\cos^4\gamma \cdot (\cos^2\theta + \tfrac{4}{\beta^2\pi^2}) = 1 \;,$$
$$\frac{2\arccos(\cos^2\gamma \cdot \cos\theta) + \arccos(\cos^2\gamma)}{2\pi} = \beta \cdot \frac{1-\cos\theta}{2} \;,$$
$$\frac{3}{2\pi}\arccos(-\tfrac{1}{3}\cos^2\gamma) = \beta \;.$$

The approximation algorithm that solves the semidefinite programming relaxation of a MAX NAE-{3}-SAT *instance and obtains the vectors* $v_1, \ldots, v_n$, *outward rotates the vectors* $v_1, \ldots, v_n$ *by an angle of* γ *and then rounds them using a random hyperplane, is a* β*-approximation algorithm for* MAX NAE-{3}-SAT.

The worst case ratio of $\beta \simeq 0.90871$ is achieved if the solution $v_1, \ldots, v_n$ of the SDP relaxation is such that for every clause $\mathrm{NAE}_3(x_i, x_j, x_k)$ for which $w_{ijk} > 0$ we have $\{\theta_{ij}, \theta_{ik}, \theta_{jk}\} = \{\theta, \theta, 0\}$, where $\theta \simeq 2.408874$, or $(\theta_{ij}, \theta_{ik}, \theta_{jk}) = (\arccos(-\frac{1}{3}), \arccos(-\frac{1}{3}), \arccos(-\frac{1}{3}))$. (Here $\theta_{ij} = \arccos(v_i \cdot v_j)$ is the angle between the vectors v_i and v_j.) A proof of Theorem 4.1 will appear in the full version of the paper.

The algorithm described in Theorem 4.1 is randomized. It can again be derandomized using the technique of Mahajan and Ramesh [MR95]. It is also possible to describe a variant of the algorithm, similar to the approximation algorithm for MAX CUT given in the previous section, in which the angle by which the vectors are rotated depends on $A = (\sum_{i,j,k} \frac{3 - v_i \cdot v_j - v_i \cdot v_k - v_j \cdot v_k}{4}) / (\sum_{i,j,k} w_{ijk})$. This algorithm will have a performance guarantee of $\beta \simeq 0.90871$ on all instances, and an even larger performance guarantee on instances in which most of the clauses can be simultaneously satisfied. For instances of MAX NAE-{3}-SAT in which all clauses can be satisfied, the algorithm would have a performance guarantee of $\beta' = \frac{3}{2\pi}\arccos(-\frac{1}{3}) \simeq 0.91226$. We omit the details of this algorithm from this extended abstract.

5 MAX NAE-SAT and MAX SAT

5.1 Background

An instance of MAX NAE-SAT in the variables $x_1, \ldots, x_n$ is a weighted collection of clauses of the form $\mathrm{NAE}(z_1, \ldots, z_k)$, where the z_i's are literals and $k \ge 2$. The clauses may be arbitrarily large. A clause $\mathrm{NAE}(z_1, \ldots, z_k)$ is satisfied by an assignment of 0-1 values to the literals $z_1, \ldots, z_k$ if at least one of these literals gets the value 0, and at least one of these literals gets the value 1. MAX NAE-SAT is a generalization of the MAX SET SPLITTING problem (see again [KLP96],[AE98]). We let MAX NAE-$\{k\}$-SAT be the restriction of MAX NAE-SAT to instances in which all clauses are of size k. We let MAX NAE-k-SAT be the restriction of MAX NAE-SAT to instances in which all clauses are of size *at most* k.

An instance of MAX SAT in the variables $x_1, \ldots, x_n$ is a weighted collection of clauses of the form $z_1 \vee z_2 \vee \ldots \vee z_k$, where the z_i's are literals and $k \ge 1$. The clauses may again be arbitrarily large. We let MAX $\{k\}$-SAT be the restriction of MAX SAT to instances in which all clauses are of size k. We let MAX k-SAT be the restriction of MAX SAT to instances in which all clauses are of size *at most* k.

A lot of work has been done on MAX SAT. MAX NAE-SAT is much less studied. In addition to being interesting in its own right, MAX NAE-SAT is also interesting as it is a generalization of both MAX CUT and MAX SAT. MAX CUT is simply the restriction of MAX NAE-2-SAT to instances without negations. Each MAX SAT instance can be converted into an equivalent MAX NAE-SAT instance by replacing each clause $z_1 \vee z_2 \vee \ldots \vee z_k$ by the clause $\mathrm{NAE}(O, z_1, \ldots, z_k)$, where O is a variable that appears in all the NAE clauses. A solution to the resulting MAX NAE-SAT instance is easily converted into a solution, with the same value, of the original MAX SAT instance in the following way: If the variable O is assigned the value 0, then take the values assigned to the variables $x_1, \ldots, x_n$ unchanged. If the variable O is assigned the value 1, then complement the values assigned to the variables $x_1, \ldots, x_n$. We thus get:

Lemma 5.1 *If there is a β-approximation algorithm for MAX NAE-SAT, then there is also a β-approximation algorithm for MAX SAT. If there is a β_k-approximation algorithm for MAX NAE-k-SAT, then there is also a β_k-approximation algorithm for MAX $(k-1)$-SAT.*

A summary of the existing approximation algorithms for MAX SAT and MAX NAE-SAT is given in Figure 5. The $\frac{7}{8}$-approximation algorithm for MAX 3-SAT [2] and the conjectured $\frac{7}{8}$-approximation algorithm for satisfiable instances of MAX 4-SAT are optimal, as follows from the following seminal result of Håstad:

Theorem 5.2 (Håstad [Hås97]) *For every $k \geq 3$ and every $\epsilon > 0$, if there is a $(1-2^{-k}+\epsilon)$-approximation algorithm for satisfiable instances of MAX $\{k\}$-SAT then P=NP.*

As MAX SAT and MAX NAE-SAT are both generalizations of MAX $\{3\}$-SAT, we get that for every $\epsilon > 0$, there is no $(\frac{7}{8}+\epsilon)$-approximation algorithms for these problems, unless P=NP.

5.2 New results

A way of obtaining *canonical* SDP relaxations for any constraint satisfaction problem is described in [KZ97]. Using this method we can obtain an SDP relaxation for MAX NAE-k-SAT, for every fixed k. This relaxation contains a unit vector v_i for each one of the literals $x_1, \ldots, x_{2n}$, where we use the convention that $x_{n+i} = \bar{x}_i$, and a scalar $z(C)$ for each clause $C = \mathrm{NAE}(z_1, \ldots, z_k)$. The objective is to maximize the sum $\sum w(C)z(C)$, where $w(C) \geq 0$ is the weight assigned to the clause C. For each clause C, the relaxation contains many inequalities involving the scalar $z(C)$ and inner products of the unit vectors that correspond to the literals that appear in C. For example, for the clause $\mathrm{NAE}(x_1, \ldots, x_k)$, the relaxation will contain the constraint $z \leq 1$, as well as the constraint

$$z \leq \frac{k - \sum_{i=1}^{k} v_{\pi(i)} \cdot v_{\pi(i+1)}}{4} ,$$

for every permutation π on the indices $1, 2, \ldots, k$ (in the summation above, $\pi(k+1) = \pi(1)$). The relaxation also contains other constraints. The reader can easily verify that the specified constraints are satisfied when the vectors v_i are all either $(-1, 0, \ldots, 0)$ or $(1, 0, \ldots, 0)$, and z is 1 if a clause is satisfied and 0 if it is not. Note that the number of constraints that we have per clause grows exponentially with k. As we want the total number of constraints to be polynomial in n, the number of variables that appear in the instance, we write these constraints only for clauses of length $k = O(\log n / \log\log n)$. Larger clauses are simply ignored. The rounding procedure that we will use ensures that these large clauses will be satisfied, in any case, with probabilities very close to 1.

Let $relax(v_1, \ldots, v_k)$ be the largest value of z that satisfies all the constraints associated with the clause $\mathrm{NAE}(x_1, \ldots, x_k)$. We let $prob_\gamma(v_1, \ldots, v_k)$ be the probability that the clause $\mathrm{NAE}(x_1, \ldots, x_k)$ is satisfied by the assignment obtained by outward rotating the vectors $v_1, \ldots, v_k$ by an angle of γ and then rounding them using a random hyperplane. Let

$$\beta_k(\gamma) = \min_{v_1, \ldots, v_k} \frac{prob_\gamma(v_1, \ldots, v_k)}{relax(v_1, \ldots, v_k)} ,$$

where the minimum is taken over all unit vectors $v_1, \ldots, v_k$ for which $relax(v_1, \ldots, v_k) > 0$. It is easy to see that $\beta_k(\gamma)$ is a lower bound on the performance ratio that we get for clauses of size k by using outward rotations by an angle of γ. Evaluating the probabilities $prob_\gamma(v_1, \ldots, v_k)$, even when $k = 4$ and $\gamma = 0$, is a difficult task (see [KZ97]). There seems to be no closed form formula for these probabilities. To compute $prob_\gamma(v_1, \ldots, v_k)$, we have to (numerically) evaluate a $\lfloor \frac{k-2}{2} \rfloor$-dimensional integral. Extensive numerical experiments led us to the following conjecture:

Conjecture 5.3 *For every $k \geq 4$ and $0 \leq \gamma \leq \frac{\pi}{2}$, the value of the minimum $\beta_k(\gamma)$ is attained when for every $1 \leq i < j \leq k$ we have $v_i \cdot v_j = 1 - \frac{4}{k}$.*

The general expression for $prob_\gamma(v_1, \ldots, v_k)$ is greatly simplified when all the inner products $v_i \cdot v_j$, for $i \neq j$, are equal. Let $p_k(c)$, where $-1 \leq c \leq 1$, be $prob_0(v_1, \ldots, v_k)$ when $v_i \cdot v_j = c$, for every $1 \leq i < j \leq k$. We can show:

Lemma 5.4 *For every $k \geq 2$ and $-1 \leq c < 1$ we have*

$$p_k(c) = 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} N\left(-r\sqrt{\frac{c}{1-c}}\right)^k \mathbf{e}^{-r^2/2}\, dr ,$$

where $N(x) = \frac{1}{\sqrt{2\pi}} \int_x^{+\infty} \mathbf{e}^{-r^2/2}\, dr$ is the probability distribution of a standard normal random variable.

As a corollary to Conjecture 5.3 we get that $\beta_k(\gamma) = p_k((\cos^2\gamma)(1 - \frac{4}{k}))$, for every $k \geq 4$ and $0 \leq \gamma \leq \frac{\pi}{2}$.

Theorem 5.5 (i) *If Conjecture 5.3 is valid then the algorithm that solves the canonical SDP relaxation of a given MAX NAE-SAT instance, outward rotates*

[2] This result appears in [KZ97] as a conjecture. It is now a theorem. The algorithm given in [KZ97] is in fact a $\frac{7}{8}$-approximation algorithm for MAX NAE-4-SAT.

General Instances

problem	ratio	reference
MAX 2-SAT	0.931	[FG95]
MAX 3-SAT	$\frac{7}{8}$	[KZ97]
MAX 4-SAT	$\boxed{0.872}$	[HZ99]
MAX SAT	$\frac{3}{4}$	[Yan94]
MAX SAT	0.758	[GW95]
MAX SAT	0.770	[Asa97]
MAX SAT	$\boxed{0.797}$	$\star$

General Instances

problem	ratio	reference
MAX NAE-2-SAT	0.878	[GW95]
MAX NAE-{3}-SAT	0.908	$\star$
MAX NAE-4-SAT	$\frac{7}{8}$	[KZ97]
MAX NAE-SAT	0.724	[AE98]
MAX NAE-SAT	$\boxed{0.797}$	$\star$

Satisfiable Instances

problem	ratio	reference
MAX 4-SAT	$\boxed{\frac{7}{8}}$	[HZ99]
MAX SAT	0.800	[Tre97]
MAX SAT	$\boxed{0.863}$	$\star$

Satisfiable Instances

problem	ratio	reference
MAX NAE-{3}-SAT	0.912	[Zwi98a]
MAX NAE-SAT	$\boxed{0.863}$	$\star$

Figure 5: Approximation algorithms for MAX SAT and MAX NAE-SAT. Results appearing inside a box are conjectures based on numerical evidence.

the obtained vectors by an angle of $\gamma = 0.4555$ *and then rounds them using a random hyperplane is a 0.7977-approximation algorithm for MAX NAE-SAT.*

(*ii*) *If Conjecture 5.3 is valid then the algorithm that solves the canonical SDP relaxation of a given MAX NAE-SAT instance, outward rotates the vectors by an angle of* $\gamma = 0.5720$ *and then rounds them using a random hyperplane is a 0.8636-approximation algorithm for* satisfiable *instances of MAX NAE-SAT.*

A proof of Lemma 5.4 and Theorem 5.5 will appear in the full version of the paper. It is interesting to note that if Conjecture 5.3 is valid, then the algorithm described in part (i) of Theorem 5.5 achieves its worst case ratio on instances in which all clauses are of sizes 2 and 36(!). Additional results for the problems MAX NAE-k-SAT, for $5 \le k \le 36$ will again appear in the full version of the paper.

The conjectured 0.7977-approximation algorithm for MAX NAE-SAT would substantially improve the 0.724-approximation algorithm given by Andersson and Engebretsen [AE98]. The conjectured 0.8636-approximation algorithm for satisfiable instances of MAX NAE-SAT would substantially improve the 0.8-approximation algorithm given by Trevisan [Tre97] for satisfiable instance of MAX SAT problem. An 0.8636-approximation algorithm would be almost optimal as no performance guarantee better than $\frac{7}{8}$ can be obtained for the problem, unless P=NP.

6 Concluding remarks

We have described a simple tool, *outward rotations*, that can be used to enhance the performance of several semidefinite programming based approximation algorithms. We exhibited the usefulness of this tool by using it to obtain improved approximation algorithms for MAX CUT, MAX NAE-{3}-SAT and probably also for MAX NAE-SAT and MAX SAT.

The improved approximation algorithm for MAX CUT performs, in many cases, better than the approximation algorithm of Goemans and Williamson [GW95]. Its overall performance guarantee, however, is the same as that of the GW algorithm. Obtaining a better performance guarantee for MAX CUT remains a challenging open problem.

We have also examined the usefulness of outward rotations in the design of improved approximation algorithms for MAX NAE-SAT. We obtained a 0.90871-approximation algorithm for MAX NAE-{3}-SAT. Analyzing the behavior of semidefinite programming based approximation algorithms for MAX NAE-SAT, or even for MAX NAE-k-SAT for $k \ge 5$, seems very difficult at present. Based on extensive numerical experiments we have formulated a conjecture that, if true, implies that outward rotations can be used to obtain a 0.8636-approximation algorithm for satisfiable instances of MAX NAE-SAT and hence also for satisfiable instances of MAX SAT. We chose to focus here on MAX NAE-SAT as outward rotations can be applied to this more general problem while previous tools, such as the rotations introduced by Feige and Goemans [FG95], cannot.

We conjecture that there is a $\frac{7}{8}$-approximation algorithm for MAX SAT. Outward rotations on their own will not suffice to prove this conjecture. An attempt to move toward a proof of this conjecture is made in [HZ99] were other techniques, that cannot be applied to MAX NAE-SAT, are used to obtain an almost optimal algorithm for MAX 4-SAT. Another intriguing open problem is whether there is a $\frac{7}{8}$-approximation algorithm for MAX NAE-SAT.

References

[AE98] G. Andersson and L. Engebretsen. Better approximation algorithms for SET SPLITTING and NOT-ALL-EQUAL SAT. *Information Processing Letters*, 65:305–311, 1998.

[AL97] S. Arora and C. Lund. Hardness of approximation. In D. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, pages 399–446. PWS Publishing Company, 1997.

[AS98] N. Alon and B. Sudakov. Bipartite subgraphs and the smallest eigenvalue. Manuscript, 1998.

[Asa97] T. Asano. Approximation algorithms for MAX SAT: Yannakakis vs. Goemans-Williamson. In *Proceedings of the 3nd Israel Symposium on Theory and Computing Systems, Ramat Gan, Israel*, pages 24–37, 1997.

[ASZ98] N. Alon, B. Sudakov, and U. Zwick. Work in progress, 1998.

[FG95] 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, Tel Aviv, Israel*, pages 182–189, 1995.

[GW95] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[Hås97] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing, El Paso, Texas*, pages 1–10, 1997. Full version available as E-CCC Report number TR97-037.

[HZ99] E. Halperin and U. Zwick. Approximation algorithms for MAX 4-SAT and rounding procedures for semidefinite programs. In *Proceedings of the 7th Conference on Integer Programming and Combinatorial Optimization, Graz, Austria*, 1999. To appear.

[Kar96] H. Karloff. How good is the Goemans-Williamson MAX CUT algorithm? In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing, Philadelphia, Pennsylvania*, pages 427–434, 1996.

[KLP96] V. Kann, J. Lagergren, and A. Panconesi. Approximability of maximum splitting of k-sets and some other APX-complete problems. *Information Processing Letters*, 58:105–110, 1996.

[KZ97] H. Karloff and U. Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proceedings of the 38rd Annual IEEE Symposium on Foundations of Computer Science, Miami Beach, Florida*, pages 406–415, 1997.

[MR95] S. Mahajan and H. Ramesh. Derandomizing semidefinite programming based approximation algorithms. In *Proceedings of the 36rd Annual IEEE Symposium on Foundations of Computer Science, Milwaukee, Wisconsin*, pages 162–169, 1995.

[Nes98] Y. E. Nesterov. Semidefinite relaxation and nonconvex quadratic optimization. *Optimization Methods and Software*, 9:141–160, 1998.

[Tre97] L. Trevisan. Approximating satisfiable satisfiability problems. In *Proceedings of the 5th European Symposium on Algorithms, Graz, Austria*, 1997. 472–485.

[Yan94] M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17:475–502, 1994.

[Zwi98a] U. Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, California*, pages 201–210, 1998.

[Zwi98b] U. Zwick. Finding almost-satisfying assignments. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing, Dallas, Texas*, pages 551–560, 1998.