In J. ACM., 42, 1115-1145, 1995.

Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming*

Michel X. Goemans[†] *David P. Williamson*[‡]
M.I.T. IBM Watson

Abstract

We present randomized approximation algorithms for the maximum cut (MAX CUT) and maximum 2-satisfiability (MAX 2SAT) problems that always deliver solutions of expected value at least .87856 times the optimal value. These algorithms use a simple and elegant technique that randomly rounds the solution to a nonlinear programming relaxation. This relaxation can be interpreted both as a semidefinite program and as an eigenvalue minimization problem. The best previously known approximation algorithms for these problems had performance guarantees of $\frac{1}{2}$ for MAX CUT and $\frac{3}{4}$ for MAX 2SAT. Slight extensions of our analysis lead to a .79607-approximation algorithm for the maximum directed cut problem (MAX DICUT) and a .758-approximation algorithm for MAX SAT, where the best previously known approximation algorithms had performance guarantees of $\frac{1}{4}$ and $\frac{3}{4}$ respectively. Our algorithm gives the first substantial progress in approximating MAX CUT in nearly twenty years, and represents the first use of semidefinite programming in the design of approximation algorithms.

Introduction

Given an undirected graph $G = (V, E)$ and non-negative 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* $(S, \bar{S})$; that is, the weight of the edges with one endpoint in S and the other in $\bar{S}$. For simplicity, we usually set $w_{ij} = 0$ for $(i, j) \notin E$ and denote the weight of a cut $(S, \bar{S})$ by $w(S, \bar{S}) = \sum_{i\in S, j\notin S} w_{ij}$. The MAX CUT problem is one of the Karp's original NP-complete problems [37], and has long been known to be NP-complete even if the problem is unweighted; that is, if $w_{ij} = 1$ for all $(i, j) \in E$ [19]. The MAX CUT problem is solvable in polynomial time for some special classes of graphs (e.g. if the graph is planar [52, 27]). Besides its theoretical importance, the MAX CUT problem has applications in circuit layout design and statistical physics (Barahona et al. [4]). For a comprehensive survey of the MAX CUT problem, the reader is referred to Poljak and Tuza [62].

Because it is unlikely that there exist efficient algorithms for NP-hard maximization problems, a typical approach to solving such a problem is to find a *ρ-approximation algorithm*; that is, a polynomial-time algorithm that delivers a solution of value at least ρ times the optimal value.

---

*A preliminary version has appeared in the Proceedings of the 26th Annual ACM Symposium on Theory of Computing, Montréal, pages 422–431, 1994.

†Address: Dept. of Mathematics, Room 2-382, M.I.T., Cambridge, MA 02139. Email: `goemans@math.mit.edu`. Research supported in part by NSF contract 9302476-CCR and DARPA contract N00014-92-J-1799.

‡Address: IBM T.J. Watson Research Center, Room 33-219, P.O. Box 218, Yorktown Heights, NY, 10598. Email: `dpw@watson.ibm.com`. Research supported by an NSF Postdoctoral Fellowship. This research was conducted while the author was visiting MIT.

The constant ρ is sometimes called the *performance guarantee* of the algorithm. We will also use the term "ρ-approximation algorithm" for randomized polynomial-time algorithms that deliver solutions whose expected value is at least ρ times the optimal. In 1976, Sahni and Gonzales [66] presented a $\frac{1}{2}$-approximation algorithm for the MAX CUT problem. Their algorithm iterates through the vertices and decides whether or not to assign vertex i to S based on which placement maximizes the weight of the cut of vertices 1 to i. This algorithm is essentially equivalent to the randomized algorithm that flips an unbiased coin for each vertex to decide which vertices are assigned to the set S. Since 1976, a number of researchers have presented approximation algorithms for the unweighted MAX CUT problem with performance guarantees of $\frac{1}{2}+\frac{1}{2m}$ [69], $\frac{1}{2}+\frac{n-1}{4m}$ [61], $\frac{1}{2}+\frac{1}{2n}$ [30], and $\frac{1}{2}+\frac{1}{2\Delta}$ [33] (where $n=|V|$, $m=|E|$ and Δ denotes the maximum degree), but no progress was made in improving the constant in the performance guarantee beyond that of Sahni and Gonzales's straightforward algorithm.

We present a simple, randomized $(\alpha-\epsilon)$-approximation algorithm for the maximum cut problem where

$$\alpha = \min_{0\leq\theta\leq\pi} \frac{2}{\pi}\frac{\theta}{1-\cos\theta} > 0.87856,$$

and ϵ is any positive scalar. The algorithm represents the first substantial progress in approximating the MAX CUT problem in nearly twenty years. The algorithm for MAX CUT also leads directly to a randomized $(\alpha-\epsilon)$-approximation algorithm for the maximum 2-satisfiability problem (MAX 2SAT). The best previously known algorithm for this problem has a performance guarantee of $\frac{3}{4}$ and is due to Yannakakis [71]. A somewhat simpler $\frac{3}{4}$-approximation algorithm was given in Goemans and Williamson [22]. The improved 2SAT algorithm leads to .7584-approximation algorithm for the overall MAX SAT problem, fractionally better than Yannakakis' $\frac{3}{4}$-approximation algorithm for MAX SAT. Finally, a slight extension of our analysis yields a $(\beta-\epsilon)$-approximation algorithm for the maximum directed cut problem (MAX DICUT), where

$$\beta = \min_{0\leq\theta<\arccos(-1/3)} \frac{2}{\pi}\frac{2\pi-3\theta}{1+3\cos\theta} > 0.79607.$$

The best previously known algorithm for MAX DICUT has a performance guarantee of $\frac{1}{4}$ [55].

Our algorithm depends on a means of randomly 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*. To our knowledge, this is the first time that semidefinite programs have been used in the design and analysis of approximation algorithms. The relaxation plays a crucial role in allowing us to obtain a better performance guarantee: previous approximation algorithms compared the value of the solution obtained to the total sum of the weights $\sum_{i<j} w_{ij}$. This sum can be arbitrarily close to twice the value of the maximum cut.

A semidefinite program is the problem of optimizing a linear function of a symmetric matrix subject to linear equality constraints and the constraint that the matrix be positive semidefinite. Semidefinite programming is a special case of convex programming and also of the so-called *linear programming over cones* or *cone-LP* since the set of positive semidefinite matrices constitutes a convex cone. To some extent, semidefinite programming is very similar to linear programming; see Alizadeh [1] for a comparison. It inherits the very elegant duality theory of cone-LP (see Wolkowicz [70] and the exposition by Alizadeh [1]). The simplex method can be generalized to semidefinite programs (Pataki [57]). Given any $\epsilon>0$, semidefinite programs can be solved within an additive error of ϵ in polynomial time (ϵ is part of the input, so the running time dependence on ϵ is polynomial in $\log\frac{1}{\epsilon}$). This can be done through the ellipsoid algorithm (Grötschel et al. [26]) and other polynomial-time algorithms for convex programming (Vaidya [67]) as well as interior-point methods (Nesterov and Nemirovskii [50, 51] and Alizadeh [1]). To terminate in polynomial time, these algorithms implicitly assume some requirement on the feasible space or on the size of the optimum solution; for details see Grötschel et al. [26] and Section 3.3 of Alizadeh [1]. Since the work of Nesterov and Nemirovskii, and Alizadeh, there has been much development in the design and analysis of interior-point methods for semidefinite programming; for several references available at the time of writing of this paper, see the survey paper by Vandenberghe and Boyd [68].

Semidefinite programming has many interesting applications in a variety of areas including control theory, nonlinear programming, geometry and combinatorial optimization; see [51, 9, 68, 1], the references therein and the references below. In combinatorial optimization, the importance of semidefinite programming is that it leads to tighter relaxations than the classical linear programming relaxations for many graph and combinatorial problems. A beautiful application of semidefinite programming is the work of Lovász [43] on the Shannon capacity of a graph. In conjunction with the polynomial-time solvability of semidefinite programs, this leads to the only known polynomial-time algorithm for finding the largest stable set in a perfect graph (Grötschel et al. [25]). More recently, there has been increased interest in semidefinite programming from a combinatorial point-of-view [46, 47, 1, 58, 17, 45]. This started with the work of Lovász and Schrijver [46, 47], who developed a machinery to define tighter and tighter relaxations of any integer program based on quadratic and semidefinite programming. Their papers demonstrated the wide applicability and the power of semidefinite programming for combinatorial optimization problems. Our use of semidefinite programming relaxations was inspired by these papers, and by the paper of Alizadeh [1].

For MAX CUT, the semidefinite programming relaxation we consider is equivalent to an eigenvalue minimization problem proposed by Delorme and Poljak [13, 12]. An eigenvalue minimization problem consists of minimizing a decreasing sum of the eigenvalues λ_i of a matrix subject to equality constraints on the matrix; that is, minimizing $\sum_i m_i \lambda_i$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ and $m_1 \geq m_2 \geq \cdots \geq m_n \geq 0$. The equivalence of the semidefinite program we consider and the eigenvalue bound of Delorme and Poljak was established by Poljak and Rendl [58]. Building on work by Overton and Womersley [54, 53], Alizadeh [1] has shown that eigenvalue minimization problems can in general be formulated as semidefinite programs. This is potentially quite useful, since there is an abundant literature on eigenvalue bounds for combinatorial optimization problems; see the survey paper by Mohar and Poljak [49].

As shown by Poljak and Rendl [60, 59] and Delorme and Poljak [14], the eigenvalue bound provides a very good bound on the maximum cut in practice. Delorme and Poljak [13, 12] study the worst-case ratio between the maximum cut and their eigenvalue bound. The worst instance they are aware of is the 5-cycle for which the ratio is $\frac{32}{25+5\sqrt{5}} = 0.88445\ldots$, but they were unable to prove a bound better than 0.5 in the worst-case. Our result implies a worst-case bound of α, very close to the bound for the 5-cycle.

The above discussion on the worst-case behavior indicates that straightforward modifications of our technique will not lead to significant improvements in the MAX CUT result. Furthermore, MAX CUT, MAX 2SAT, and MAX DICUT are MAX SNP-hard [55], and so it is known that there exists a constant $c < 1$ such that a c-approximation algorithm for any of these problems would imply that $P = NP$ [2]. Bellare, Goldreich, and Sudan [6] have shown that c is as small as 83/84 for MAX CUT and 95/96 for MAX 2SAT. Since bidirected instances of MAX DICUT are equivalent to instances of MAX CUT, the bound for MAX CUT also applies to MAX DICUT.

Since the appearance of an abstract of this paper, Feige and Goemans [16] have extended our technique to yield a .931-approximation algorithm for MAX 2SAT and a .859-approximation algorithm for MAX DICUT. By using semidefinite programming and similar rounding ideas, Karger, Motwani, and Sudan [36] have been able to show how to color a k-colorable graph with $\tilde{O}(n^{1-\frac{3}{k+1}})$

colors in polynomial time. Frieze and Jerrum [18] have used the technique to devise approximation algorithms for the maximum k-way cut problem that improve on the previously best known $1-1/k$ performance guarantee. Chor and Sudan [10] apply ideas from this paper to the "betweeness" problem. Thus it seems likely that the techniques in this paper will continue to prove useful in designing approximation algorithms.

We expect that in practice the performance of our algorithms will be much better than the worst-case bounds. We have performed some preliminary computational experiments with the MAX CUT algorithm which show that on a number of different types of random graphs the algorithm is usually within .96 of the optimal solution.

A preliminary version of this paper [21] presented a method to obtain deterministic versions of our approximation algorithm with the same performance guarantees. However, the method given had a subtle error, as was pointed out to us by several researchers. Mahajan and Ramesh [48] document the error and propose their own derandomization scheme for our algorithms.

The paper is structured as follows. We present the randomized algorithm for MAX CUT in Section 1, and its analysis in Section 2. We elaborate on our semidefinite programming bound and its relationship with other work on the MAX CUT problem in Section 3. The quality of the relaxation is investigated in Section 4, and computational results are presented in Section 5. In Section 6, we show how to extend the algorithm to an algorithm for MAX 2SAT, MAX SAT, MAX DICUT, and other problems. We conclude with a few remarks and open problems in Section 7.

1 The Randomized Approximation Algorithm for MAX CUT

Given a graph with vertex set $V=\{1,\ldots,n\}$ and non-negative weights $w_{ij}=w_{ji}$ for each pair of vertices i and j, the weight of the maximum cut $w(S,\bar{S})$ is given by the following integer quadratic program:

$$
\begin{array}{llll}
 & \text{Maximize} & \displaystyle\frac{1}{2}\sum_{i<j} w_{ij}(1-y_i y_j) & \\
(Q) & \text{subject to:} & y_i \in \{-1,1\} & \forall i \in V.
\end{array}
$$

To see this, note that the set $S=\{i|y_i=1\}$ corresponds to a cut of weight $w(S,\bar{S})=\frac{1}{2}\sum_{i<j} w_{ij}(1-y_i y_j)$.

Since solving this integer quadratic program is NP-complete, we consider relaxations of (Q). Relaxations of (Q) are obtained by relaxing some of the constraints of (Q), and extending the objective function to the larger space; thus all possible solutions of (Q) are feasible for the relaxation, and the optimal value of the relaxation is an upper bound on the optimal value of (Q). We can interpret (Q) as restricting y_i to be a 1-dimensional vector of unit norm. Some very interesting relaxations can be defined by allowing y_i to be a multi-dimensional vector v_i of unit Euclidean norm. Since the linear space spanned by the vectors v_i has dimension at most n, we can assume that these vectors belong to $\mathbb{R}^n$ (or $\mathbb{R}^m$ for some $m \le n$), or more precisely to the n-dimensional unit sphere S_n (or S_m for $m \le n$). To ensure that the resulting optimization problem is indeed a relaxation, we need to define the objective function in such a way that it reduces to $\frac{1}{2}\sum_{i<j} w_{ij}(1-y_i y_j)$ in the case of vectors lying in a 1-dimensional space. There are several natural ways of guaranteeing this property. For example, one can replace $(1-y_i y_j)$ by $(1-v_i \cdot v_j)$ where $v_i \cdot v_j$ represents the inner product (or dot product) of v_i and v_j. The resulting relaxation is denoted by (P):

$$
\text{Maximize} \quad \frac{1}{2}\sum_{i<j} w_{ij}(1-v_i \cdot v_j)
$$

$$(P) \qquad \text{subject to:} \quad v_i \in S_n \qquad\qquad \forall i \in V.$$

We will show in Section 3 we can solve this relaxation using semidefinite programming. We can now present our simple randomized algorithm for the MAX CUT problem.

1. Solve (P), obtaining an optimal set of vectors v_i.
2. Let r be a vector uniformly distributed on the unit sphere S_n.
3. Set $S = \{i | v_i \cdot r \geq 0\}$.

In other words, we choose a random hyperplane through the origin (with r as its normal) and partition the vertices into those vectors that lie "above" the plane (i.e. have a non-negative inner product with r) and those that lie "below" it (i.e. have a negative inner product with r). The motivation for this randomized step comes from the fact that (P) is independent of the coordinate system: applying any orthonormal transformation to a set of vectors results in a solution with the same objective value.

Let W denote the value of the cut produced in this way, and $E[W]$ its expectation. We will show in Theorem 2.1 in Section 2 that, given any set of vectors $v_i \in S_n$, the expected weight of the cut defined by a random hyperplane is

$$E[W] = \sum_{i<j} w_{ij} \frac{\arccos(v_i \cdot v_j)}{\pi}.$$

We will also show in Theorem 2.3 that

$$E[W] \geq \alpha \cdot \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i \cdot v_j),$$

where $\alpha = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos\theta} > .878$. If Z^*_{MC} is the optimal value of the maximum cut and Z^*_P is the optimal value of the relaxation (P), then since the expected weight of the cut generated by the algorithm is equal to $E[W] \geq \alpha Z^*_P \geq \alpha Z^*_{MC}$, the algorithm has a performance guarantee of α for the MAX CUT problem.

We must argue that the algorithm can be implemented in polynomial time. We assume that the weights are integral. In Section 3 we show that the program (P) is equivalent to a semidefinite program. Then we will show that by using an algorithm for semidefinite programming we can obtain, for any $\epsilon > 0$, a set of vectors v_i's of value greater than $Z^*_P - \epsilon$ in time polynomial in the input size and $\log \frac{1}{\epsilon}$. On these approximately optimal vectors, the randomized algorithm will produce a cut of expected value greater than or equal to $\alpha(Z^*_P - \epsilon) \geq (\alpha - \epsilon) Z^*_{MC}$. The point on the unit sphere S_n can be generated by drawing n values $x_1, x_2, \ldots, x_n$ independently from the standard normal distribution, and normalizing the vector obtained (see Knuth [38, p. 130]); for our purposes, there is no need to normalize the resulting vector x. The standard normal distribution can be simulated using the uniform distribution between 0 and 1 (see Knuth [38, p. 117]). The algorithm is thus a randomized $(\alpha - \epsilon)$-approximation algorithm for MAX CUT.

2 Analysis of the Algorithm

In this section, we analyze the performance of the algorithm. We first analyze the general case and then consider the case in which the maximum cut is large and the generalization to negative edge weights. We conclude this section with a new formulation for the MAX CUT problem.

Let $\{v_1, \ldots, v_n\}$ be any vectors belonging to S_n, and let $E[W]$ be the expected value of the cut $w(S, \bar{S})$ produced by the randomized algorithm given in the previous section. We start by characterizing the expected value of the cut.

Theorem 2.1

$$E[W] = \frac{1}{\pi} \sum_{i<j} w_{ij} \arccos(v_i \cdot v_j).$$

Given a vector r drawn uniformly from the unit sphere S_n, we know by the linearity of expectation that

$$E[W] = \sum_{i<j} w_{ij} \cdot \Pr[\mathrm{sgn}(v_i \cdot r) \neq \mathrm{sgn}(v_j \cdot r)],$$

where $\mathrm{sgn}(x) = 1$ if $x \geq 0$, and -1 otherwise. Theorem 2.1 is thus implied by the following lemma.

Lemma 2.2

$$\Pr[\mathrm{sgn}(v_i \cdot r) \neq \mathrm{sgn}(v_j \cdot r)] = \frac{1}{\pi} \arccos(v_i \cdot v_j).$$

Pf: A restatement of the lemma is that the probability the random hyperplane separates the two vectors is directly proportional to the angle between the two vectors; that is, it is proportional to the angle $\theta = \arccos(v_i \cdot v_j)$. By symmetry, $\Pr[\mathrm{sgn}(v_i \cdot r) \neq \mathrm{sgn}(v_j \cdot r)] = 2 \Pr[v_i \cdot r \geq 0, v_j \cdot r < 0]$. The set $\{r : v_i \cdot r \geq 0, v_j \cdot r < 0\}$ corresponds to the intersection of two half-spaces whose dihedral angle is precisely θ; its intersection with the sphere is a spherical digon of angle θ and, by symmetry of the sphere, thus has measure equal to $\frac{\theta}{2\pi}$ times the measure of the full sphere. In other words, $\Pr[v_i \cdot r \geq 0, v_j \cdot r < 0] = \frac{\theta}{2\pi}$, and the lemma follows. ■

Our main theorem is the following. As we have argued above, this theorem applied to vectors v_i's of value greater than $Z_P^* - \epsilon$ implies that the algorithm has a performance guarantee of $\alpha - \epsilon$. Recall that we defined

$$\alpha = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos\theta}.$$

Theorem 2.3

$$E[W] \geq \alpha \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i \cdot v_j).$$

By using Theorem 2.1 and the nonnegativity of w_{ij}, the result follows from the following lemma applied to $y = v_i \cdot v_j$.

Lemma 2.4 For $-1 \leq y \leq 1$, $\frac{1}{\pi} \arccos(y) \geq \alpha \cdot \frac{1}{2}(1 - y)$.

Pf: The expression for α follows straightforwardly by using the change of variables $\cos\theta = y$. See Figure 1, part (i). ■

The quality of the performance guarantee is established by the following lemma.

Lemma 2.5 $\alpha > .87856$.

Pf: Using simple calculus, one can see that α achieves its value for $\theta = 2.331122\ldots$, the non-zero root of $\cos\theta + \theta \sin\theta = 1$. To formally prove the bound of 0.87856, we first observe that $\frac{2}{\pi} \frac{\theta}{1-\cos\theta} \geq 1$ for $0 < \theta \leq \pi/2$. The concavity of $f(\theta) = 1 - \cos\theta$ in the range $\pi/2 \leq \theta \leq \pi$ implies that, for any θ_0, we have $f(\theta) \leq f(\theta_0) + (\theta - \theta_0) f'(\theta)$, or $1 - \cos\theta \leq 1 - \cos\theta_0 + (\theta - \theta_0) \sin\theta_0 =$

$\theta \sin\theta_0 + (1 - \cos\theta_0 - \theta_0 \sin\theta_0)$. Choosing $\theta_0 = 2.331122$ for which $1 - \cos\theta_0 - \theta_0 \sin\theta_0 < 0$, we derive that $1 - \cos\theta < \theta \sin\theta_0$, implying that $\alpha > \frac{2}{\pi \sin\theta_0} > 0.87856$. ■

We have analyzed the expected value of the cut returned by the algorithm. For randomized algorithms, one can often also give high probability results. In this case, however, even computing the variance of the cut value analytically is difficult. Nevertheless, one could use the fact that W is bounded (by $\sum_{i<j} w_{ij}$) to give lower bounds on the probability that W is less than $(1-\epsilon)E[W]$.

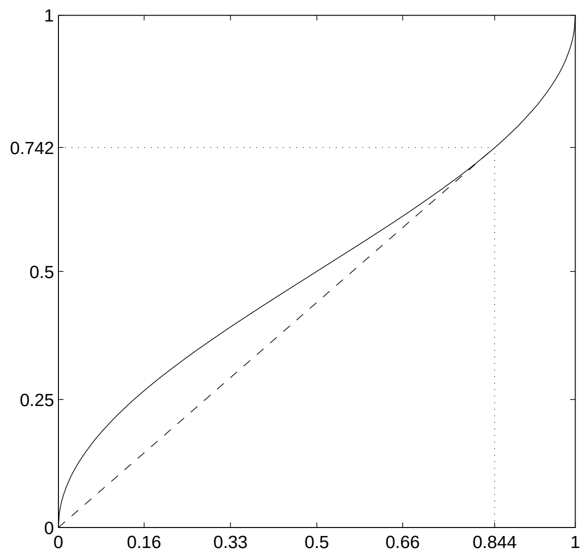

Figure 1: (i) Plot of $z = \arccos(y)/\pi$ as a function of $t = \frac{1}{2}(1-y)$. The ratio z/t is thus the slope of the line between $(0,0)$ and (z,t). The minimum slope is precisely $\alpha = 0.742/0.844$ and corresponds to the dashed line. (ii) The plot also represents $h(t) = \arccos(1-2t)/\pi$ as a function of t. As t approaches 1, $h(t)/t$ also approaches 1. (iii) The dashed line corresponds to $\tilde{h}$ between 0 and $\gamma \approx .844$.

2.1 Analysis when the maximum cut is large

We can refine the analysis and prove a better performance guarantee whenever the value of the relaxation (P) constitutes a large fraction of the total weight. Define $W_{tot} = \sum_{i<j} w_{ij}$ and $h(t) = \arccos(1-2t)/\pi$. Let γ be the value of t attaining the minimum of $h(t)/t$ in the interval (0,1]. The value of γ is approximately .84458.

Theorem 2.6 Let $A = (\sum_{i<j} w_{ij} \frac{1 - v_i \cdot v_j}{2})/W_{tot}$. If $A \geq \gamma$, then

$$E[W] \geq \frac{h(A)}{A} \sum_{i<j} w_{ij} \frac{1 - v_i \cdot v_j}{2}.$$

The theorem implies a performance guarantee of $h(A)/A - \epsilon$ when $A \geq \gamma$. As A varies between γ and 1, one easily verifies that $h(A)/A$ varies between α and 1 (see Figure 1, part (ii)).

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

$$E[W] = \sum_{i<j} w_{ij} \frac{\arccos(v_i \cdot v_j)}{\pi} = W_{tot} \sum_e \lambda_e \frac{\arccos(1-2x_e)}{\pi} = W_{tot} \sum_e \lambda_e h(x_e).$$

To bound $E[W]$, we evaluate

$$\begin{aligned} \text{Min} \quad & \sum_e \lambda_e h(x_e) \\ \text{subject to:} \quad & \sum_e \lambda_e x_e = A \\ & 0 \leq x_e \leq 1. \end{aligned}$$

Consider the relaxation obtained by replacing $h(t)$ by the largest (pointwise) convex function $\tilde{h}(t)$ smaller or equal to $h(t)$. It is easy to see that $\tilde{h}(t)$ is linear with slope α between 0 and γ, and then equal to $h(t)$ for any t greater than γ. See Figure 1, part (iii). But for $A \geq \gamma$,

$$\sum_e \lambda_e h(x_e) \geq \sum_e \lambda_e \tilde{h}(x_e) \geq \tilde{h}(\sum_e \lambda_e x_e) = \tilde{h}(A) = h(A),$$

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

$$E[W] \geq W_{tot} h(A) = \frac{h(A)}{A} \sum_{i<j} w_{ij} \frac{1 - v_i \cdot v_j}{2},$$

proving the result. ∎

2.2 Analysis for negative edge weights

So far, we have assumed that the weights are non-negative. In several practical problems, some edge weights are negative [4]. In this case the definition of the performance guarantee has to be modified since the optimum value could be positive or negative. We now give a corresponding generalization of Theorem 2.3 to arbitrary weights.

Theorem 2.7 Let $W_- = \sum_{i<j} w_{ij}^-$, where $x^- = \min(0,x)$. Then

$$\{E[W] - W_-\} \geq \alpha \left\{ \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i \cdot v_j) - W_- \right\}.$$

Pf: The quantity $E[W] - W_-$ can be written as

$$\sum_{i<j: w_{ij}>0} w_{ij} \frac{\arccos(v_i \cdot v_j)}{\pi} + \sum_{i<j: w_{ij}<0} |w_{ij}| \left(1 - \frac{\arccos(v_i \cdot v_j)}{\pi} \right).$$

Similarly, $\left\{ \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i \cdot v_j) - W_- \right\}$ is equal to

$$\sum_{i<j: w_{ij}>0} w_{ij} \frac{1 - v_i \cdot v_j}{2} + \sum_{i<j: w_{ij}<0} |w_{ij}| \frac{1 + v_i \cdot v_j}{2}.$$

The result therefore follows from Lemma 2.4 and the following variation of it. ∎

Lemma 2.8 For $-1 \le z \le 1, 1 - \frac{1}{\pi}\arccos(z) \ge \alpha \cdot \frac{1}{2}(1+z)$.

Pf: The lemma follows from Lemma 2.4 by using a change of variables $z = -y$ and noting that $\pi - \arccos(z) = \arccos(-z)$. ■

2.3 A new formulation of MAX CUT

An interesting consequence of our analysis is a new nonlinear formulation of the maximum cut problem. Consider the following nonlinear program:

$$\begin{array}{llll} & \text{Maximize} & \sum_{i<j} w_{ij} \frac{\arccos(v_i \cdot v_j)}{\pi} & \\ (R) & \text{subject to:} & v_i \in S_n & \forall i \in V. \end{array}$$

Let Z_R^* denote the optimal value of this program.

Theorem 2.9 $Z_R^* = Z_{MC}^*$.

Pf: We first show that $Z_R^* \ge Z_{MC}^*$. This follows since (R) is a relaxation of (Q): the objective function of (R) reduces to $\frac{1}{2}\sum_{i<j} w_{ij}(1 - v_i v_j)$ in the case of vectors v_i lying in a 1-dimensional space.

To see that $Z_R^* \le Z_{MC}^*$, let the vectors v_i denote the optimal solution to (R). From Theorem 2.1, we see that the randomized algorithm gives a cut whose expected value is exactly Z_R^*, implying that there must exist a cut of value at least Z_R^*. ■

3 Relaxations and Duality

In this section, we address the question of solving the relaxation (P). We do so by showing that (P) is equivalent to a semidefinite program. We then explore the dual of this semidefinite program and relate it to the eigenvalue minimization bound of Delorme and Poljak [13, 12].

3.1 Solving the Relaxation

We begin by defining some terms and notation. All matrices under consideration are defined over the reals. An $n \times n$ matrix A is said to be positive semidefinite if for every vector $x \in \mathbb{R}^n$, $x^T A x \ge 0$. The following statements are equivalent for a symmetric matrix A (see e.g. [39]): (i) A is positive semidefinite, (ii) all eigenvalues of A are non-negative, and (iii) there exists a matrix B such that $A = B^T B$. In (iii), B can either be a (possibly singular) $n \times n$ matrix, or an $m \times n$ matrix for some $m \le n$. Given a symmetric positive semidefinite matrix A, an $m \times n$ matrix B of full row-rank satisfying (iii) can be obtained in $O(n^3)$ time using an incomplete Cholesky decomposition [23, p. 90, P5.2-3].

Using the decomposition $Y = B^T B$, one can see that a positive semidefinite Y with $y_{ii} = 1$ corresponds precisely to a set of unit vectors $v_1, \ldots, v_n \in S_m$: simply correspond the vector v_i to the ith column of B. Then $y_{ij} = v_i \cdot v_j$. The matrix Y is known as the Gram matrix of $\{v_1, \ldots, v_n\}$ [39, p. 110]. Using this equivalence, we can reformulate (P) as a semidefinite program:

$$\begin{array}{llll} & Z_P^* = \text{Max} & \frac{1}{2}\sum_{i<j} w_{ij}(1 - y_{ij}) & \\ (SD) & \text{subject to:} & y_{ii} = 1 & \forall i \in V \\ & & Y \text{ symmetric positive semidefinite} & \end{array}$$

where $Y = (y_{ij})$. The feasible solutions to (SD) are often referred to as *correlation matrices* [24]. Strictly speaking, we cannot solve (SD) to optimality in polynomial time; the optimal value Z_P^* might in fact be irrational. However, using an algorithm for semidefinite programming, one can obtain, for any $\epsilon > 0$, a solution of value greater than $Z_P^* - \epsilon$ in time polynomial in the input size and $\log \frac{1}{\epsilon}$. For example, Alizadeh's adaptation of Ye's interior-point algorithm to semidefinite programming [1] performs $O(\sqrt{n}(\log W_{tot} + \log \frac{1}{\epsilon}))$ iterations. By exploiting the simple structure of the problem (SD) as is indicated in [64] (see also [68, Section 7.4]), each iteration can be implemented in $O(n^3)$ time. Once an almost optimal solution to (SD) is found, one can use an incomplete Cholesky decomposition to obtain vectors $v_1, \ldots, v_n \in S_m$ for some $m \leq n$ such that $\frac{1}{2}\sum_{i<j} w_{ij}(1 - v_i \cdot v_j) \geq Z_P^* - \epsilon$.

Among all optimum solutions to (SD), one can show the existence of a solution of low rank. Grone, Pierce and Watkins [24] show that any extreme solution of (SD) (i.e. which cannot be expressed as the strict convex combination of other feasible solutions) has rank at most l where $\frac{l(l+1)}{2} \leq n$, i.e. $l \leq \frac{\sqrt{8n+1}-1}{2} < \sqrt{2n}$. For related results, see [41, 11, 42, 40]. This means that there exists a primal optimum solution Y^* to (SD) of rank less than $\sqrt{2n}$, and that the optimum vectors v_i of (P) can be embedded in $\mathbb{R}^m$ with $m < \sqrt{2n}$. This result also follows from a more general statement about semidefinite programs due to Barvinok [5] and implicit in Pataki [56]: any extreme solution of a semidefinite program with k linear equalities has rank at most l where $\frac{l(l+1)}{2} \leq k$.

3.2 The Semidefinite Dual

As mentioned in the introduction, there is an elegant duality theory for semidefinite programming. We now turn to discussing the dual of the program (SD). It is typical to assume that the objective function of a semidefinite program is symmetric. For this purpose, we can rewrite the objective function of (SD) as $\frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(1 - y_{ij})$, or even as $\frac{1}{2}W_{tot} - \frac{1}{4}\sum_i\sum_j w_{ij}y_{ij}$. In matrix form, the objective function can be conveniently written as $\frac{1}{2}W_{tot} - \frac{1}{4}Tr(WY)$, where $W = (w_{ij})$ and Tr denotes the trace.

The dual of (SD) has a very simple description:

$$\begin{array}{llll} & Z_D^* & = \frac{1}{2}W_{tot} + \frac{1}{4}\text{Min} & \sum_i \gamma_i \\ (D) & & \text{subject to:} & W + diag(\gamma) \text{ positive semidefinite,} \end{array}$$

where $diag(\gamma)$ denotes the diagonal matrix whose ith diagonal entry is γ_i. The dual has a simple interpretation. Since $W + diag(\gamma)$ is positive semidefinite, it can be expressed as C^TC; in other words, the weight w_{ij} can be viewed as $c_i \cdot c_j$ for some vectors c_i's and $\gamma_i = c_i \cdot c_i = \|c_i\|^2$. The weight of any cut is thus $w(S, \bar{S}) = \left(\sum_{i\in S} c_i\right) \cdot \left(\sum_{j\notin S} c_j\right)$, which is never greater than

$$\left\|\frac{\sum_{i\in V} c_i}{2}\right\|^2 = \frac{1}{2}W_{tot} + \frac{1}{4}\sum_{i\in V}\gamma_i.$$

Showing weak duality between (P) and (D), namely that $Z_P^* \leq Z_D^*$, is easy. Consider any primal feasible solution Y and any dual vector γ. Since both Y and $W + diag(\gamma)$ are positive semidefinite, we derive that $Tr((diag(\gamma) + W)Y) \geq 0$ (see [39, p. 218, ex. 14]). But $Tr((diag(\gamma) + W)Y) = Tr(diag(\gamma)Y) + Tr(WY) = \sum_i \gamma_i + Tr(WY)$, implying that the difference of the dual objective function value and the primal objective function value is non-negative.

For semidefinite programs in their full generality, there is no guarantee that the primal optimum value is equal to the dual optimum value. Also, the maximum (resp. minimum) is in fact a supremum (resp. infimum) and there is no guarantee that the supremum (resp. infimum) is attained. These, however, are pathological cases. Our programs (SD) and (D) behave nicely; both programs attain their optimum values, and these values are equal (i.e. $Z_P^* = Z_D^*$). This can be shown in a variety of ways (see [58, 1, 5]).

Given that strong duality holds in our case, the argument showing weak duality implies that, for the optimum primal solution Y^* and the optimum dual solution γ^*, we have $Tr((diag(\gamma^*) + W)Y^*) = 0$. Since both $diag(\gamma^*) + W$ and Y^* are positive semidefinite, we derive that $(diag(\gamma^*) + W)Y^* = 0$ (see [39, p. 218, ex. 14]). This is the strong form of complementary slackness for semidefinite programs (see Alizadeh [1]); the component-wise product expressed by the trace is replaced by matrix multiplication. This implies, for example, that Y^* and $diag(\gamma^*) + W$ share a system of eigenvectors and that $rank(Y^*) + rank(diag(\gamma^*) + W) \leq n$.

3.3 The Eigenvalue Bound of Delorme and Poljak

The relaxation (D) (and thus (P)) is also equivalent to an eigenvalue upper bound on the value of the maximum cut Z_{MC}^* introduced by Delorme and Poljak [13, 12]. To describe the bound, we first introduce some notation. The Laplacian matrix $L = (l_{ij})$ is defined by $l_{ij} = -w_{ij}$ for $i \neq j$ and $l_{ii} = \sum_{k=1}^n w_{ik}$. The maximum eigenvalue of a matrix A is denoted by $\lambda_{max}(A)$.

Lemma 3.1 [Delorme and Poljak [13]] Let $u \in \mathbb{R}^n$ satisfy $u_1 + \ldots + u_n = 0$. Then

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

is an upper bound on Z_{MC}^*.

The proof is simple. Let y be an optimal solution to the integer quadratic program (Q). Notice that $y^T L y = 4Z_{MC}^*$. By using the Rayleigh principle ($\lambda_{max}(M) = \max_{||x||=1} x^T M x$), we obtain

$$\begin{aligned} \lambda_{max}(L + diag(u)) &\geq \frac{y^T(L + diag(u))y}{y^T y} \\ &= \frac{1}{n}\left(y^T L y + \sum_{i=1}^n y_i^2 u_i\right) \\ &= \frac{4Z_{MC}^*}{n}, \end{aligned}$$

proving the lemma. A vector u satisfying $\sum_{i=1}^n u_i = 0$ is called a *correcting vector*. Let $g(u) = \frac{n}{4}\lambda_{max}(L + diag(u))$. The bound proposed by Delorme and Poljak [13] is to optimize $g(u)$ over all correcting vectors:

$$(EIG) \qquad \begin{aligned} Z_{EIG}^* &= \text{Inf} \quad g(u) \\ &\quad \text{subject to: } \sum_{i=1}^n u_i = 0. \end{aligned}$$

As mentioned in the introduction, eigenvalue minimization problems can be formulated as semidefinite programs. For MAX CUT, the equivalence between (SD) and (EIG) was established by Poljak and Rendl [58].

For completeness, we derive the equivalence between (EIG) and the dual (D). For the optimum dual vector γ^*, the smallest eigenvalue of $diag(\gamma^*)+W$ must be 0, since otherwise we could decrease all γ_i^* by some $\epsilon > 0$ and thus reduce the dual objective function. This can be rewritten as $\lambda_{max}(-W - diag(\gamma^*)) = 0$. Define λ as $(\sum_i \gamma_i^* + 2W_{tot})/n$. By definition, $Z_D^* = n\lambda/4$. Moreover, defining $u_i = \lambda - \gamma_i^* - \sum_j w_{ij}$, one easily verifies that $\sum_i u_i = 0$ and that $-W - diag(\gamma^*) = L + diag(u) - \lambda I$, implying that $\lambda_{max}(L + diag(u)) = \lambda$. This shows that $Z_{EIG}^* \leq Z_D^*$. The converse inequality follows by reversing the argument.

4 Quality of the Relaxation

In this section we consider the tightness of our analysis and the quality of the semidefinite bound Z_P^*. Observe that Theorem 2.3 implies the following corollary:

Corollary 4.1 For any instance of MAX CUT,

$$\frac{Z_{MC}^*}{Z_P^*} \geq \alpha.$$

For the 5-cycle, Delorme and Poljak [13] have shown that $Z_{MC}^*/Z_{EIG}^* = \frac{32}{25+5\sqrt{5}} = 0.88445\ldots$, implying that our worst-case analysis is almost tight. One can obtain this bound from the relaxation (P) by observing that for the 5-cycle $1-2-3-4-5-1$, the optimal vectors lie in a 2-dimensional subspace and can be expressed as $v_i = (\cos(\frac{4i\pi}{5}), \sin(\frac{4i\pi}{5}))$ for $i = 1, \ldots, 5$ corresponding to $Z_P^* = \frac{5}{2}(1 + \cos\frac{\pi}{5}) = \frac{25+5\sqrt{5}}{8}$. Since $Z_{MC}^* = 4$ for the 5-cycle, this yields the bound of Delorme and Poljak. Delorme and Poljak have shown that $Z_{MC}^*/Z_{EIG}^* \geq \frac{32}{25+5\sqrt{5}}$ holds for special subclasses of graphs, such as planar graphs or line graphs. However, they were unable to prove a bound better than 0.5 in the absolute worst-case.

Although the worst-case value of Z_{MC}^*/Z_P^* is not completely settled, there exist instances for which $E[W]/Z_P^*$ is very close to α, showing that the analysis of our algorithm is practically tight. Leslie Hall [31] has observed that $E[W]/Z_P^* \approx .8787$ for the Petersen graph [8, p. 55]. In Figure 2, we give an unweighted instance for which the ratio is less than .8796 in which the vectors have a nice three-dimensional representation. We have also constructed a weighted instance on 103 vertices for which the ratio is less than .8786. These two instances are based on strongly self-dual polytopes due to Lovász [44]. A polytope P in $\mathbb{R}^n$ is said to be strongly self-dual [44] if (i) P is inscribed in the unit sphere, (ii) P is circumscribed around the sphere with origin as center and with radius r for some $0 < r < 1$, and (iii) there is a bijection σ between vertices and facets of P such that, for every vertex v of P, the facet $\sigma(v)$ is orthogonal to the vector v. For example, in $\mathbb{R}^2$, the strongly self-dual polytopes are precisely the regular odd polygons. One can associate a graph $G = (V, E)$ to a self-dual polytope P: the vertex set V corresponds to the vertices of P and there is an edge (v, w) if w belongs to the facet $\sigma(v)$ (or, equivalently, v belongs to $\sigma(w)$). For the regular odd polygons, these graphs are simply the odd cycles. Because of conditions (ii) and (iii), the inner product $v \cdot w$ for any pair of adjacent vertices is equal to $-r$. As a result, a strongly self-dual polytope leads to a feasible solution of (P) of value $\frac{1+r}{2}W_{tot}$. Lovász [44] gives a recursive construction of a class of strongly self-dual polytopes. One can show that, by choosing the dimension n large enough, his construction leads to strongly self-dual polytopes for which r is arbitrarily close to the critical value giving a bound of α. However, it is unclear whether, in general, for such polytopes, non-negative weights can be selected such that the vectors given by the polytope constitute an *optimum* solution to (P). Nevertheless, we conjecture that such instances lead to a proof that $E[W]/Z_P^*$ can be arbitrarily close to α. Even if this could be shown, this would not imply anything for the ratio Z_{MC}^*/Z_P^*.

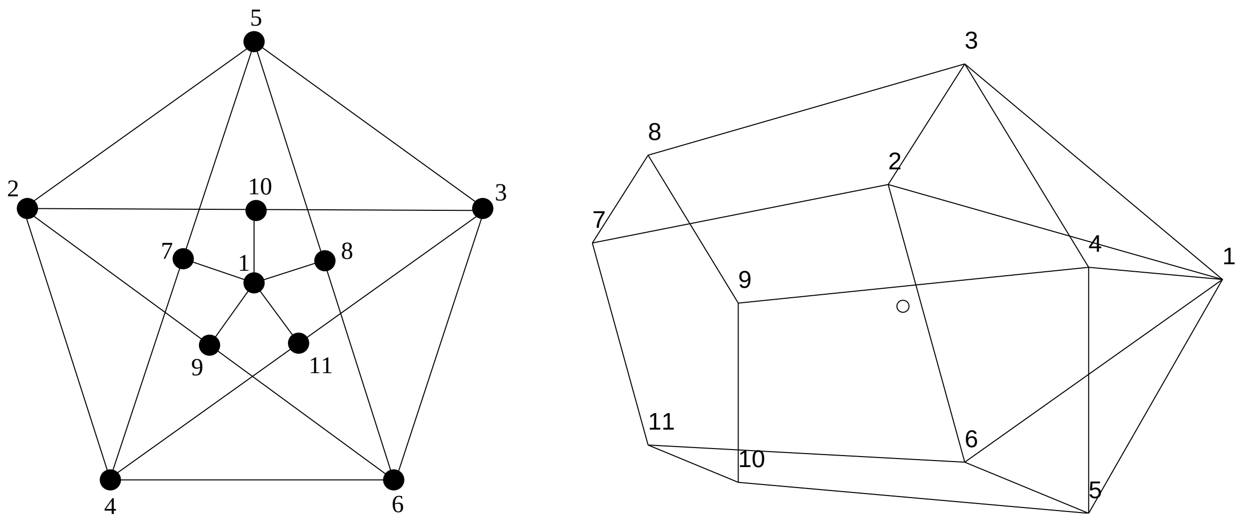

Figure 2: Graph on 11 vertices for which the ratio $E[W]/Z_P^*$ is less than .8796 in the unweighted case. The convex hull of the optimum vectors is depicted on the right; the circle represents the center of the sphere.

5 Computational Results

In practice, we expect that the algorithm will perform much better than the worst-case bound of α. Poljak and Rendl [60, 59] (see also Delorme and Poljak [14]) report computational results showing that the bound Z_{EIG}^* is typically less than 2-5% and, in the instances they tried, never worse than 8% away from Z_{MC}^*. We also performed our own computational experiments, in which the cuts computed by the algorithm were usually within 4% of the semidefinite bound Z_P^*, and never less than 9% from the bound. To implement the algorithm, we used code supplied by R. Vanderbei [64] for a special class of semidefinite programs. We use Vanderbei's code to solve the semidefinite program, then we generate 50 random vectors r. We output the best of the 50 cuts induced. We applied our code to a small subset of the instances considered by Poljak and Rendl [60]. In particular, we considered several different types of random graphs, as well as complete geometric graphs defined by Traveling Salesman Problem (TSP) instances from the TSPLIB (see Reinelt [63]).

For four different types of random graphs, we ran 50 instances on graphs of 50 vertices, 20 on graphs of size 100, and 5 on graphs of size 200. In the Type A random graph, each edge (i,j) is included with probability 1/2. In the Type B random graph, each edge is given a weight drawn uniformly from the interval [-50,50]; the ratio of Theorem 2.7 is used in reporting nearness to the semidefinite bound. In the Type C random graph of size $n \geq 10$, an edge (i,j) is included with probability $10/n$, leading to constant expected degree. Finally, in the Type D random graphs, an edge (i,j) is included with probability .1 if $i <= n/2$ and $j > n/2$ and probability .05 otherwise, leading to a large cut between the vertices in $[1,\ldots,n/2]$ and those in $[n/2+1,\ldots,n]$. We summarize the results of our experiments in Table 1 below. CPU Times are given in CPU seconds on a Sun SPARCstation 1.

In the case of the TSP instances, we used Euclidean instances from the TSPLIB, and set the edge weight w_{ij} to the Euclidean distance between the points i and j. We summarize our results in

Type of Graph	Size	Num Trials	Ave Int Gap	Ave CPU Time
Type A	50	50	.96988	36.28
	100	20	.96783	323.08
	200	5	.97209	4629.62
Type B	50	50	.97202	23.06
	100	20	.97097	217.42
	200	5	.97237	2989.00
Type C	50	50	.95746	23.53
	100	20	.94214	306.84
	200	5	.92362	2546.42
Type D	50	50	.95855	27.35
	100	20	.93984	355.32
	200	5	.93635	10709.42

Table 1: Summary of results of algorithm on random instances. Ave Int Gap is the average ratio of the value of the cut generated to the semidefinite bound, except for Type B graphs, where it is the ratio of the value of the cut generated minus the negative edge weights to the semidefinite bound minus the negative edge weights.

Table 2. In all 10 instances, we compute the optimal solution; for 5 instances, the value of the cut produced is equal to Z_P^*, and for the others, we have been able to exploit additional information from the dual semidefinite program to prove optimality (for the problem dantzig42, gr48 and hk48, Poljak and Rendl [60] also show that our solution is optimal). For all TSPLIB instances, the maximum cut value is within .995 of the semidefinite bound. Given these computational results, it is tempting to speculate that a much stronger bound can be proven for these Euclidean instances. However, the instance defined by a unit length equilateral triangle has a maximum cut value of 2, but $Z_P^* = \frac{9}{4}$, for a ratio of $\frac{8}{9} = 0.8889$.

Homer and Peinado [34] have implemented our algorithm on a CM-5, and have shown that it produces optimal or very nearly optimal solutions to a number of MAX CUT instances derived from via minimization problems. These instances were provided by Michael Jünger [35] and have between 828 and 1366 vertices.

6 Generalizations

We can use the same technique as in Section 1 to approximate several other problems. In the next section we describe a variation of MAX CUT and give an $(\alpha - \epsilon)$-approximation algorithm for it. In Section 6.2 we give an $(\alpha - \epsilon)$-approximation algorithm for the MAX 2SAT problem, and show that it leads to a slightly improved algorithm for MAX SAT. Finally, in Section 6.3, we give a $(\beta - \epsilon)$-approximation algorithm for the maximum directed cut problem (MAX DICUT), where $\beta > .79607$. In all cases, we will show how to approximate more general integer quadratic programs that can be used to model these problems.

6.1 MAX RES CUT

The MAX RES CUT problem is a variation of MAX CUT in which pairs of vertices are forced to be on either the same side of the cut or on different sides of the cut. The extension of the algorithm

Instance	Size	SD Val	Cut Val	Time
dantzig42	42	42638	42638	43.35
gr120	120	2156775	2156667	754.87
gr48	48	321815	320277	26.17
gr96	96	105470	105295	531.50
hk48	48	771712	771712	66.52
kroA100	100	5897392	5897392	420.83
kroB100	100	5763047	5763047	917.47
kroC100	100	5890760	5890760	398.78
kroD100	100	5463946	5463250	469.48
kroE100	100	5986675	5986591	375.68

Table 2: Summary of results of algorithm on TSPLIB instances. SD Val is the value produced by the semidefinite relaxation. Cut Val is the value of the best cut output by the algorithm.

to the MAX RES CUT problem is trivial. We merely need to add the following constraints to (P): $v_i \cdot v_j = 1$ for $(i,j) \in E^+$ and $v_i \cdot v_j = -1$ for $(i,j) \in E^-$, where E^+ (resp. E^-) corresponds to the pair of vertices forced to be on the same side (resp. different sides) of the cut. Using the randomized algorithm of Section 1 and setting $y_i = 1$ if $r \cdot v_i \geq 0$ and $y_i = -1$ otherwise gives a feasible solution to MAX RES CUT, assuming that a feasible solution exists. Indeed, it is easy to see that if $v_i \cdot v_j = 1$, then the algorithm will produce a solution such that $y_i y_j = 1$. If $v_i \cdot v_j = -1$ then the only case in which the algorithm produces a solution such that $y_i y_j \neq -1$ is when $v_i \cdot r = v_j \cdot r = 0$, an event that happens with probability 0. The analysis of the expected value of the cut is unchanged and, therefore, the resulting algorithm is a randomized $(\alpha - \epsilon)$-approximation algorithm.

Another approach to the problem is to use a standard reduction of MAX RES CUT to MAX CUT based on contracting edges and "switching" cuts (see e.g. [60]). This reduction introduces negative edge weights and so we do not discuss it here, although Theorem 2.7 can be used to show that our MAX CUT algorithm applied to a reduced instance has a performance guarantee of $(\alpha - \epsilon)$ for the original MAX RES CUT instance. In fact, a more general statement can be made: any ρ-approximation algorithm (in the sense of Theorem 2.7) for MAX CUT instances possibly having negative edge weights leads to a ρ-approximation algorithm for MAX RES CUT.

6.2 MAX 2SAT and MAX SAT

An instance of the maximum satisfiability problem (MAX SAT) is defined by a collection $\mathcal{C}$ of boolean clauses, where each clause is a disjunction of literals drawn from a set of variables $\{x_1, x_2, \ldots, x_n\}$. A *literal* is either a variable x or its negation $\bar{x}$. The *length* $l(C_j)$ of a clause C_j is the number of distinct literals in the clause. In addition, for each clause $C_j \in \mathcal{C}$ there is an associated non-negative weight w_j. An optimal solution to a MAX SAT instance is an assignment of truth values to the variables $x_1, \ldots, x_n$ that maximizes the sum of the weight of the satisfied clauses. MAX 2SAT consists of MAX SAT instances in which each clause has length at most two. MAX 2SAT is NP-complete [19]; the best approximation algorithm known previously has a performance guarantee of $\frac{3}{4}$ and is due to Yannakakis [71] (see also Goemans and Williamson [22]). As with the MAX CUT problem, MAX 2SAT is known to be MAX SNP-hard [55]; thus there exists some constant $c < 1$ such that the existence of a c-approximation algorithm implies that $P = NP$

[2]. Bellare, Goldreich, and Sudan [6] have shown that a 95/96-approximation algorithm for MAX 2SAT would imply $P = NP$. Haglin [28, 29] has shown that any ρ-approximation algorithm for MAX RES CUT can be translated into a ρ-approximation algorithm for MAX 2SAT, but we will show a direct algorithm here. Haglin's observation together with the reduction from MAX RES CUT to MAX CUT mentioned in the previous section shows that any ρ-approximation for MAX CUT with negative edge weights translates into a ρ-approximation algorithm for MAX 2SAT.

6.2.1 MAX 2SAT

In order to model MAX 2SAT, we consider the integer quadratic program

$$
\begin{array}{lll}
 & \text{Maximize} & \sum_{i<j} [a_{ij}(1 - y_i y_j) + b_{ij}(1 + y_i y_j)] \\
(Q') & \text{subject to:} & y_i \in \{-1, 1\} \qquad \forall i \in V,
\end{array}
$$

where a_{ij} and b_{ij} are non-negative. The objective function of (Q') is thus a non-negative linear form in $1 \pm y_i y_j$. To model MAX 2SAT using (Q'), we introduce a variable y_i in the quadratic program for each boolean variable x_i in the 2SAT instance; we also introduce an additional variable y_0. The value of y_0 will determine whether -1 or 1 will correspond to "true" in the MAX 2SAT instance. More precisely, x_i is true if $y_i = y_0$ and false otherwise. Given a boolean formula C, we define its value $v(C)$ to be 1 if the formula is true and 0 if the formula is false. Thus, $v(x_i) = \frac{1+y_0 y_i}{2}$ and $v(\bar{x}_i) = 1 - v(x_i) = \frac{1-y_0 y_i}{2}$. Observe that

$$
\begin{aligned}
v(x_i \vee x_j) &= 1 - v(\bar{x}_i \wedge \bar{x}_j) = 1 - v(\bar{x}_i)v(\bar{x}_j) = 1 - \frac{1 - y_0 y_i}{2}\frac{1 - y_0 y_j}{2} \\
&= \frac{1}{4}\left(3 + y_0 y_i + y_0 y_j - y_0^2 y_i y_j\right) \\
&= \frac{1 + y_0 y_i}{4} + \frac{1 + y_0 y_j}{4} + \frac{1 - y_i y_j}{4}.
\end{aligned}
$$

The value of other clauses with 2 literals can be similarly expressed; for instance, if x_i is negated one only needs to replace y_i by $-y_i$. Therefore, the value $v(C)$ of any clause with at most two literals per clause can be expressed in the form required in (Q'). As a result, the MAX 2SAT problem can be modelled as

$$
\begin{array}{lll}
 & \text{Maximize} & \sum_{C_j \in \mathcal{C}} w_j v(C_j) \\
(SAT) & \text{subject to:} & y_i \in \{-1, 1\} \qquad \forall i \in \{0, 1, \ldots, n\},
\end{array}
$$

where the $v(C_j)$ are non-negative linear combinations of $1 + y_i y_j$ and $1 - y_i y_j$. The (SAT) program is in the same form as (Q').

We relax (Q') to:

$$
\begin{array}{lll}
 & \text{Maximize} & \sum_{i<j} [a_{ij}(1 - v_i \cdot v_j) + b_{ij}(1 + v_i \cdot v_j)] \\
(P') & \text{subject to:} & v_i \in S_n \qquad \forall i \in V.
\end{array}
$$

Let $E[V]$ be the expected value of the solution produced by the randomized algorithm. By the linearity of expectation,

$$
E[V] = 2\sum_{i<j} a_{ij} \Pr[\operatorname{sgn}(v_i \cdot r) \neq \operatorname{sgn}(v_j \cdot r)] + 2\sum_{i<j} b_{ij} \Pr[\operatorname{sgn}(v_i \cdot r) = \operatorname{sgn}(v_j \cdot r)].
$$

Using the analysis of the max cut algorithm, we note that $\Pr[\mathrm{sgn}(v_i \cdot r) = \mathrm{sgn}(v_j \cdot r)] = 1 - \frac{1}{\pi}\arccos(v_i \cdot v_j)$, and thus the approximation ratio for the more general program follows from Lemmas 2.4 and 2.8.

Hence we can show the following theorem, which implies that the algorithm is an $(\alpha - \epsilon)$-approximation algorithm for (Q') and thus for MAX 2SAT.

Theorem 6.1

$$E[V] \geq \alpha \sum_{i<j} \left[a_{ij}(1 - v_i \cdot v_j) + b_{ij}(1 + v_i \cdot v_j)\right].$$

6.2.2 MAX SAT

The improved MAX 2SAT algorithm leads to a slightly improved approximation algorithm for MAX SAT. In [22], we developed several randomized $\frac{3}{4}$-approximation algorithms for MAX SAT. We considered the following linear programming relaxation of the MAX SAT problem, where I_j^+ denotes the set of non-negated variables in clause C_j and I_j^- is the set of negated variables in C_j:

$$\begin{array}{lll}
\text{Max} & \displaystyle\sum_{C_j \in \mathcal{C}} w_j z_j & \\
\text{subject to:} & & \\
& \displaystyle\sum_{i \in I_j^+} y_i + \sum_{i \in I_j^-} (1 - y_i) \geq z_j & \forall C_j \in \mathcal{C} \\
& 0 \leq y_i \leq 1 & 1 \leq i \leq n \\
& 0 \leq z_j \leq 1 & \forall C_j \in \mathcal{C}.
\end{array}$$

By associating $y_i = 1$ with x_i set true, $y_i = 0$ with x_i set false, $z_j = 1$ with clause C_j satisfied, and $z_j = 0$ with clause C_j not satisfied, the program exactly corresponds to the MAX SAT problem. We showed that for any feasible solution (y, z), if x_i is set to be true with probability y_i, then the probability that clause j will be satisfied is at least $(1 - (1 - \frac{1}{k})^k) z_j$, for $k = l(C_j)$. We then considered choosing randomly between the following two algorithms: (1) set x_i true independently with probability y_i; (2) set x_i true independently with probability $\frac{1}{2}$. Given this combined algorithm, the probability that a length k clause is satisfied is at least $\frac{1}{2}(1 - 2^{-k}) + \frac{1}{2}(1 - (1 - \frac{1}{k})^k) z_j$. This expression can be shown to be at least $\frac{3}{4} z_j$ for all lengths k. Thus if an optimal solution to the linear program is used, the algorithm results in a $\frac{3}{4}$-approximation algorithm for MAX SAT, since the expected value of the algorithm is at least $\frac{3}{4}\sum_j w_j z_j$.

We formulate a slightly different relaxation of the MAX SAT problem. Let $u(C)$ denote a relaxed version of the expression v used in the previous section in which the products $y_i y_j$ are replaced by inner products of vectors $v_i \cdot v_j$. Thus $u(x_i) = \frac{1}{2}(1 + v_i \cdot v_0)$, $u(\bar{x}_i) = \frac{1}{2}(1 - v_i \cdot v_0)$, and $u(x_i \vee x_j) = \frac{1}{4}(1 + v_i \cdot v_0) + \frac{1}{4}(1 + v_j \cdot v_0) + \frac{1}{4}(1 - v_i \cdot v_j)$. We then consider the following relaxation,

$$\begin{array}{lll}
\text{Max} & \displaystyle\sum_{C_j \in \mathcal{C}} w_j z_j & \\
\text{subject to:} & & \\
& \displaystyle\sum_{i \in I_j^+} u(x_i) + \sum_{i \in I_j^-} u(\bar{x}_i) \geq z_j & \forall C_j \in \mathcal{C} \\
& u(C_j) \geq z_j & \forall C_j \in \mathcal{C}, l(C_j) = 2 \\
& v_i \cdot v_i = 1 & 0 \leq i \leq n \\
& 0 \leq z_j \leq 1 & \forall C_j \in \mathcal{C}.
\end{array}$$

Thus if we set x_i to be true with probability $u(x_i)$ for the optimal solution to the corresponding semidefinite program, then by the arguments of [22], we satisfy a clause C_j of length k with probability at least $(1-(1-\frac{1}{k})^k)z_j$.

To obtain the improved bound, we consider three algorithms: (1) set x_i true independently with probability $\frac{1}{2}$; (2) set x_i true independently with probability $u(x_i)$ (given the optimal solution to the program); (3) pick a random unit vector r and set x_i true iff $\mathrm{sgn}(v_i \cdot r) = \mathrm{sgn}(v_0 \cdot r)$. Suppose we use algorithm i with probability p_i, where $p_1 + p_2 + p_3 = 1$. From the previous section, for algorithm (3) the probability that a clause C_j of length 1 or 2 is satisfied is at least $\alpha u(C_j) \geq \alpha z_j$. Thus the expected value of the solution is at least

$$\sum_{j:l(C_j)=1} w_j(.5p_1 + (p_2 + \alpha p_3)z_j) + \sum_{j:l(C_j)=2} w_j(.75p_1 + (.75p_2 + \alpha p_3)z_j)$$
$$+ \sum_{j:l(C_j)\geq 3} w_j((1-2^{-l(C_j)})p_1 + (1-(1-\frac{1}{l(C_j)})^{l(C_j)})p_2 z_j).$$

If we set $p_1 = p_2 = .4785$ and $p_3 = .0430$, then the expected value is at least $.7554\sum_j w_j z_j$, yielding a .7554-approximation algorithm. To see this, we check the value of the expression for lengths 1 through 4, and notice that $\min_k(1-(1-\frac{1}{k})^k) \geq 1-\frac{1}{e}$ and $.4785((1-2^{-5})+1-\frac{1}{e}) \geq .7554$.

We can obtain even a slightly better approximation algorithm for the MAX SAT problem. The bottleneck in the analysis above is that algorithm (3) contributes no expected weight for clauses of length 3 or greater. For a given clause C_j of length 3 or more, let P_j be a set of length 2 clauses formed by taking the literals of C_j two at a time; thus P_j will contain $\binom{l(C_j)}{2}$ clauses. If at least one of the literals in C_j is set true, then at least $l(C_j)-1$ of the clauses in P_j will be satisfied. Thus the following program is a relaxation of the MAX SAT problem:

$$\text{Max} \quad \sum_{C_j\in\mathcal{C}} w_j z_j$$

subject to:

$$\begin{array}{ll}
\displaystyle\sum_{i\in I_j^+} u(x_i) + \sum_{i\in I_j^-} u(\bar{x}_i) \geq z_j & \forall C_j \in \mathcal{C} \\
u(C_j) \geq z_j & \forall C_j \in \mathcal{C}, l(C_j) = 2 \\
\displaystyle\frac{1}{l(C_j)-1}\sum_{C\in P_j} u(C) \geq z_j & \forall C_j \in \mathcal{C}, l(C_j) \geq 3 \\
v_i \cdot v_i = 1 & 0 \leq i \leq n \\
0 \leq z_j \leq 1 & \forall C_j \in \mathcal{C}.
\end{array}$$

Algorithm (3) has expected value of $\alpha u(C)$ for each $C \in P_j$ for any j, so that its expected value for any clause of length 3 or more becomes at least

$$\begin{aligned}
\alpha \cdot \frac{1}{\binom{l(C_j)}{2}} \sum_{C\in P_j} u(C) &= \alpha \cdot \frac{2}{l(C_j)} \frac{1}{l(C_j)-1} \sum_{C\in P_j} u(C) \\
&\geq \alpha \cdot \frac{2}{l(C_j)} z_j,
\end{aligned}$$

so that the overall expectation of the algorithm will be at least

$$\sum_{j:l(C_j)=1} w_j(.5p_1 + (p_2 + \alpha p_3)z_j) + \sum_{j:l(C_j)=2} w_j(.75p_1 + (.75p_2 + \alpha p_3)z_j)$$

$$+\sum_{j:l(C_j)\geq 3} w_j((1-2^{-l(C_j)})p_1 + ((1-(1-\frac{1}{l(C_j)})^{l(C_j)})p_2 + \alpha \cdot \frac{2}{l(C_j)}p_3)z_j).$$

By setting $p_1 = p_2 = .467$ and $p_3 = .066$, we obtain a .7584-approximation algorithm, which can be verified by checking the expression for lengths 1 through 6, and noticing that $.467((1-\frac{1}{e}) + (1-2^{-7})) \geq .7584$. Other small improvements are possible by tightening the analysis.

6.3 MAX DICUT

Suppose we are given a directed graph $G = (V, A)$ and weights w_{ij} on each directed arc $(i, j) \in A$, where i is the *tail* of the arc and j is the *head*. The maximum directed cut problem is that of finding the set of vertices S that maximizes the weight of the edges with their tails in S and their heads in $\bar{S}$. The problem is NP-hard via a straightforward reduction from MAX CUT. The best previously known approximation algorithm for MAX DICUT has a performance guarantee of $\frac{1}{4}$ [55].

To model MAX DICUT, we consider the integer quadratic program

$$\begin{array}{llll} & \text{Max} & \sum_{i,j,k}[c_{ijk}(1-y_iy_j-y_iy_k+y_jy_k)+d_{ijk}(1+y_iy_j+y_iy_k+y_jy_k)] & \\ (Q'') & \text{subject to:} & y_i \in \{-1,1\} & \forall i \in V, \end{array}$$

where c_{ijk} and d_{ijk} are non-negative. Observe that $1-y_iy_j-y_iy_k+y_jy_k$ can also be written as $(1-y_iy_j)(1-y_iy_k)$ (or as $(1-y_iy_j)(1+y_jy_k)$), and, thus, the objective function of (Q'') can be interpreted as a non-negative restricted quadratic form in $1 \pm y_iy_j$. Moreover, $1-y_iy_j-y_iy_k+y_jy_k$ is equal to 4 if $y_i = -y_j = -y_k$ and 0 otherwise, while $1+y_iy_j+y_iy_k+y_jy_k$ is 4 if $y_i = y_j = y_k$ and is 0 otherwise.

We can model the MAX DICUT problem using the program (Q'') by introducing a variable y_i for each $i \in V$, and, as with the MAX 2SAT program, and introducing a variable y_0 that will denote the S side of the cut. Thus $i \in S$ iff $y_i = y_0$. Then arc (i, j) contributes weight $\frac{1}{4}w_{ij}(1+y_iy_0)(1-y_jy_0) = \frac{1}{4}w_{ij}(1+y_iy_0-y_jy_0-y_iy_j)$ to the cut. Summing over all arcs $(i, j) \in A$ gives a program of the same form as (Q''). We observe that if the directed graph has weighted indegree of every vertex equal to weighted outdegree, the program (Q'') reduces to one of the form (Q'), and therefore our approximation algorithm has a performance guarantee of $(\alpha - \epsilon)$.

We relax (Q'') to:

$$\begin{array}{llll} & \text{Max} & \sum_{i,j,k}[c_{ijk}(1-v_i\cdot v_j-v_i\cdot v_k+v_j\cdot v_k)+d_{ijk}(1+v_i\cdot v_j+v_i\cdot v_k+v_j\cdot v_k)] & \\ (P'') & \text{subject to:} & v_i \in S_n & \forall i \in V. \end{array}$$

We approximate (Q'') by using exactly the same algorithm as before. The analysis is somewhat more complicated. As we will show, the performance guarantee β is slightly weaker, namely

$$\beta = \min_{0\leq\theta<\arccos(-1/3)} \frac{2}{\pi}\frac{2\pi-3\theta}{1+3\cos\theta} > 0.79607.$$

Given a vector r drawn uniformly from the unit sphere S_n, we know by the linearity of expectation that the expected value $E[U]$ of the solution output is

$$4\sum_{i,j,k}[c_{ijk}\cdot \Pr[\operatorname{sgn}(v_i\cdot r)\neq \operatorname{sgn}(v_j\cdot r)=\operatorname{sgn}(v_k\cdot r)] + d_{ijk}\cdot \Pr[\operatorname{sgn}(v_i\cdot r)=\operatorname{sgn}(v_j\cdot r)=\operatorname{sgn}(v_k\cdot r)]] .$$

Consider any term in the sum, say $d_{ijk} \cdot \Pr[\mathrm{sgn}(v_i \cdot r) = \mathrm{sgn}(v_j \cdot r) = \mathrm{sgn}(v_k \cdot r)]$. The c_{ijk} terms can be dealt with similarly by simply replacing v_i by $-v_i$. The performance guarantee follows from the proof of the following two lemmas.

Lemma 6.2

$$\Pr[\mathrm{sgn}(v_i \cdot r) = \mathrm{sgn}(v_j \cdot r) = \mathrm{sgn}(v_k \cdot r)] = 1 - \frac{1}{2\pi}(\arccos(v_i \cdot v_j) + \arccos(v_i \cdot v_k) + \arccos(v_j \cdot v_k)).$$

Lemma 6.3 For any $v_i, v_j, v_k \in S_n$,

$$1 - \frac{1}{2\pi}(\arccos(v_i \cdot v_j) + \arccos(v_i \cdot v_k) + \arccos(v_j \cdot v_k)) \geq \frac{\beta}{4}\left[1 + v_i \cdot v_j + v_i \cdot v_k + v_j \cdot v_k\right],$$

where $\beta = \min_{0 \leq \theta < \arccos(-1/3)} \frac{2}{\pi} \frac{2\pi - 3\theta}{1 + 3\cos\theta} > 0.79607$.

Pf f Lea : A very short proof can be given relying on spherical geometry. The desired probability can be seen to be equal to twice the area of the spherical triangle polar to the spherical triangle defined by v_i, v_j and v_k. Stated this way, the result is a corollary to Girard's formula (1629 [20], see [65]) expressing the area of a spherical triangle with angles θ_1, θ_2 and θ_3 as its excess $\theta_1 + \theta_2 + \theta_3 - \pi$.

We also present a proof of the lemma from first principles. In fact, our proof parallels Euler's proof (1781 [15], see [65]) of Girard's formula. We define the following events:

$$\begin{array}{rl} A: & \mathrm{sgn}(v_i \cdot r) = \mathrm{sgn}(v_j \cdot r) = \mathrm{sgn}(v_k \cdot r) \\ B_i: & \mathrm{sgn}(v_i \cdot r) \neq \mathrm{sgn}(v_j \cdot r) = \mathrm{sgn}(v_k \cdot r) \\ C_i: & \mathrm{sgn}(v_j \cdot r) = \mathrm{sgn}(v_k \cdot r) \\ C_j: & \mathrm{sgn}(v_i \cdot r) = \mathrm{sgn}(v_k \cdot r) \\ C_k: & \mathrm{sgn}(v_i \cdot r) = \mathrm{sgn}(v_j \cdot r). \end{array}$$

Note that $B_i = C_i - A$. We define B_j and B_k similarly, so that $B_j = C_j - A$ and $B_k = C_k - A$. Clearly,

$$\Pr[A] + \Pr[B_i] + \Pr[B_j] + \Pr[B_k] = 1. \tag{1}$$

Also, $\Pr[C_i] = \Pr[A] + \Pr[B_i]$ and similarly for j and k. Adding up these equalities and subtracting (1), we obtain

$$\Pr[C_i] + \Pr[C_j] + \Pr[C_k] = 1 + 2\Pr[A]. \tag{2}$$

By Lemma 2.2, $\Pr[C_i] = 1 - \frac{1}{\pi}\arccos(v_j \cdot v_k)$ and similarly for j and k. Together with (2), we derive

$$\Pr[A] = 1 - \frac{1}{2\pi}(\arccos(v_i \cdot v_j) + \arccos(v_i \cdot v_k) + \arccos(v_j \cdot v_k)),$$

proving the lemma. ■

Pf f Lea : One can easily verify that the defined value of β is greater than 0.79607. Let $a = \arccos(v_i \cdot v_j)$, $b = \arccos(v_i \cdot v_k)$ and $c = \arccos(v_j \cdot v_k)$. From the theory of spherical triangles, it follows that the possible values for (a, b, c) over all possible vectors v_i, v_j and v_k define the set

$$S = \{(a,b,c) : 0 \leq a \leq \pi, 0 \leq b \leq \pi, 0 \leq c \leq \pi, c \leq a+b, b \leq a+c, a \leq b+c, a+b+c \leq 2\pi\}.$$

(see [7, Corollary 18.6.12.3]). The claim can thus be restated as $1 - \frac{1}{2\pi}(a+b+c) \geq \frac{\beta}{4}(1 + \cos(a) + \cos(b) + \cos(c))$ for all $(a,b,c) \in S$.

Let (a,b,c) minimize $h(a,b,c) = 1 - \frac{1}{2\pi}(a+b+c) - \frac{\beta}{4}(1 + \cos(a) + \cos(b) + \cos(c))$ over all $(a,b,c) \in S$. We consider several cases:

1. $a+b+c=2\pi$. We have $1-\frac{1}{2\pi}(a+b+c)=0$. On the other hand,

$$\begin{aligned}
&1+\cos(a)+\cos(b)+\cos(c)\\
&= 1+\cos(a)+\cos(b)+\cos(a+b)\\
&= 1+\cos(a+b)+2\cos(\frac{a+b}{2})\cos(\frac{a-b}{2})\\
&= 2\cos^2(\frac{a+b}{2})+2\cos(\frac{a+b}{2})\cos(\frac{a-b}{2})\\
&= 2\cos(\frac{a+b}{2})\left[\cos(\frac{a+b}{2})+\cos(\frac{a-b}{2})\right]. \qquad (3)
\end{aligned}$$

We now derive that $h(a,b,c)\geq -\frac{\beta}{2}\cos(\frac{a+b}{2})[\cos(\frac{a+b}{2})+\cos(\frac{a-b}{2})]\geq 0$, the last inequality following from the fact that $\frac{\pi}{2}\leq\frac{a+b}{2}\leq\pi-|\frac{a-b}{2}|$ and thus $\cos(\frac{a+b}{2})\leq 0$ and $[\cos(\frac{a+b}{2})+\cos(\frac{a-b}{2})]\geq 0$.

2. $a=b+c$ or $b=a+c$ or $c=a+b$. By symmetry, assume that $c=a+b$. Observe that $1-\frac{1}{2\pi}(a+b+c)=1-\frac{a+b}{\pi}$. On the other hand, by (3) we have that

$$\begin{aligned}
&1+\cos(a)+\cos(b)+\cos(c)\\
&= 2\cos(\frac{a+b}{2})(\cos(\frac{a-b}{2})+\cos(\frac{a+b}{2}))\\
&\leq 2\cos(\frac{a+b}{2})(1+\cos(\frac{a+b}{2})).
\end{aligned}$$

Letting $x=\frac{a+b}{2}$, we observe that the claim is equivalent to $1-\frac{2x}{\pi}\geq\frac{\beta}{2}\cos(x)(1+\cos(x))$ for any $0\leq x\leq\pi/2$. One can in fact verify that $1-\frac{2x}{\pi}\geq\frac{0.81989}{2}\cos(x)(1+\cos(x))$ for any $0\leq x\leq\pi/2$, implying the claim.

3. $a=0$ or $b=0$ or $c=0$. Without loss of generality, let $a=0$. The definition of S implies that $b=c$, and thus $b=a+c$. This case therefore reduces to the previous one.

4. $a=\pi$ or $b=\pi$ or $c=\pi$. Assume $a=\pi$. This implies that $b+c=\pi$ and, thus, $a+b+c=2\pi$. We have thus reduced the problem to case 1.

5. In the last case, (a,b,c) belongs to the interior of S. This implies that the gradient of h must vanish and the hessian of h must be positive semidefinite at (a,b,c). In other words, $\sin a=\sin b=\sin c=\frac{2}{\beta\pi}$, and $\cos a\geq 0,\cos b\geq 0$ and $\cos c\geq 0$. From this, we derive that $a=b=c$. But $h(a,a,a)=1-\frac{3a}{2\pi}-\frac{\beta}{4}(1+3\cos(a))$. The lemma now follows from the fact that $a\leq\frac{2\pi}{3}$, the definition of β and the fact that $1+3\cos a\leq 0$ for $a\geq\arccos\frac{-1}{3}$.

■

Thus we obtain a $(\beta-\epsilon)$-approximation algorithm for (Q'') and for the MAX DICUT problem.

7 Concluding Remarks

Our motivation for studying semidefinite programming relaxations came from a realization that the standard tool of using linear programming relaxations for approximation algorithms had limits which might not be easily surpassed (see the conclusion of Goemans and Williamson [22]). In fact, a classical linear programming relaxation for the maximum cut problem can be shown to be arbitrarily close to twice the value of the maximum cut in the worst case. Given the work of Lovász and Schrijver [46, 47], which showed that tighter and tighter relaxations could be obtained through semidefinite programming, it seemed worthwhile to investigate the power of such relaxations from a worst-case perspective. The results of this paper constitute a first step in this direction. As we mentioned in the introduction, further steps have already been made, with improved results for MAX 2SAT and MAX DICUT by Feige and Goemans, and for coloring by Karger, Motwani, and Sudan. We think that the continued investigation of these methods is promising.

While this paper leaves many open questions, we think there are two especially interesting problems. The first question is whether a .878-approximation algorithm for MAX CUT can be obtained without explicitly solving the semidefinite program. For example, the first 2-approximation algorithms for weighted vertex cover involved solving a linear program [32], but later Bar-Yehuda and Even [3] devised a primal-dual algorithm in which linear programming was used only in the analysis of the algorithm. Perhaps a semidefinite analog is possible for MAX CUT. The second question is whether adding additional constraints to the semidefinite program leads to a better worst-case bound. There is some reason to think this might be true. Linear constraints are known for which the program would find an optimal solution on any planar graph, whereas there is a gap of $\frac{32}{25+5\sqrt{5}}$ for the current semidefinite program for the 5-cycle.

One consequence of this paper is that the situation with several MAX SNP problems is no longer clear-cut. When the best-known approximation results for MAX CUT and MAX SAT had such long-standing and well-defined bounds as $\frac{1}{2}$ and $\frac{3}{4}$, it was tempting to believe that perhaps no further work could be done in approximating these problems, and that it was only a matter of time before matching hardness results would be found. The improved results in this paper should rescue algorithm designers from such fatalism. Although MAX SNP problems cannot be approximated arbitrarily closely, there still is work to do in designing improved approximation algorithms.

Acknowledgements

Since the appearance of an extended abstract of this paper, we have benefited from discussions with a very large number of colleagues. In particular, we would like to thank Don Coppersmith and David Shmoys for pointing out to us that our MAX 2SAT algorithm could lead to an improved MAX SAT algorithm, Jon Kleinberg for bringing reference [44] to our attention, Bob Vanderbei for kindly giving us his semidefinite code, Francisco Barahona and Michael Jünger for providing problem instances, Joel Spencer for motivating Theorem 2.6, Farid Alizadeh, Gabor Pataki and Rob Freund for results on semidefinite programming, and Shang-Hua Teng for bringing reference [38] to our attention. We received other useful comments from Farid Alizadeh, Joseph Cheriyan, Jon Kleinberg, Monique Laurent, Colin McDiarmid, Giovanni Rinaldi, David Shmoys, Éva Tardos, and the two anonymous referees.

References

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5:13–51, 1995.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[3] R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2:198–203, 1981.

[4] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36:493–513, 1988.

[5] A. I. Barvinok. Problems of distance geometry and convex properties of quadratic maps. *Discrete and Computational Geometry*, 13:189–202, 1995.

[6] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCP and non-approximability — Towards tight results. Unpublished manuscript, 1995.

[7] M. Berger. *Geometry II*. Springer-Verlag, Berlin, 1987.

[8] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. American Elsevier Publishing, New York, NY, 1976.

[9] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. SIAM, Philadelphia, PA, 1994.

[10] B. Chor and M. Sudan. A geometric approach to betweenness. In *Proceedings of the European Symposium on Algorithms*, 1995.

[11] J. Christensen and J. Vesterstrøm. A note on extreme positive definite matrices. *Mathematische Annalen*, 244:65–68, 1979.

[12] C. Delorme and S. Poljak. Combinatorial properties and the complexity of a max-cut approximation. *European Journal of Combinatorics*, 14:313–333, 1993.

[13] C. Delorme and S. Poljak. Laplacian eigenvalues and the maximum cut problem. *Mathematical Programming*, 62:557–574, 1993.

[14] C. Delorme and S. Poljak. The performance of an eigenvalue bound on the max-cut problem in some classes of graphs. *Discrete Mathematics*, 111:145–156, 1993.

[15] L. Euler. De mensura angulorum solidorum. Petersburg, 1781.

[16] U. Feige and M. X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proc. of the Third Israel Symposium on Theory of Computing and Systems*, pages 182–189, 1995.

[17] U. Feige and L. Lovász. Two-prover one-round proof systems: Their power and their problems. In *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, pages 733–744, 1992.

[18] A. Frieze and M. Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. In *Proceedings of the Fourth MPS Conference on Integer Programming and Combinatorial Optimization*. Springer-Verlag, 1995. To appear in *Algorithmica*.

[19] M. Garey, D. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[20] A. Girard. De la mesure de la superficie des triangles et polygones sphériques. Appendix to "Invention nouvelle en l'algèbre", Amsterdam, 1629.

[21] M. X. Goemans and D. P. Williamson. .878-approximation algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, pages 422–431, 1994.

[22] M. X. Goemans and D. P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7:656–666, 1994.

[23] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, 1983.

[24] R. Grone, S. Pierce, and W. Watkins. Extremal correlation matrices. *Linear Algebra and its Applications*, 134:63–70, 1990.

[25] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[26] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, Berlin, 1988.

[27] F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4:221–225, 1975.

[28] D. J. Haglin. Approximating maximum 2-CNF satisfiability. *Parallel Processing Letters*, 2:181–187, 1992.

[29] D. J. Haglin. Personal communication, 1994.

[30] D. J. Haglin and S. M. Venkatesan. Approximation and intractability results for the maximum cut problem and its variants. *IEEE Transactions on Computers*, 40:110–113, 1991.

[31] L. Hall. Personal communication, 1994.

[32] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11:555–556, 1982.

[33] T. Hofmeister and H. Lefmann. A combinatorial design approach to MAXCUT. Unpublished manuscript, 1995.

[34] S. Homer and M. Peinado. A highly parallel implementation of the Goemans/Williamson algorithm to approximate MaxCut. Manuscript, 1994.

[35] M. Jünger. Personal communication, 1994.

[36] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 2–13, 1994.

[37] R. M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, NY, 1972.

[38] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, second edition, 1981.

[39] P. Lancaster and M. Tismenetsky. *The Theory of Matrices*. Academic Press, Orlando, FL, 1985.

[40] M. Laurent and S. Poljak. On the facial structure of the set of correlation matrices. Unpublished manuscript, 1995.

[41] C.-K. Li and B.-S. Tam. A note on extreme correlation matrices. *SIAM Journal on Matrix Analysis and Applications*, 15:903–908, 1994.

[42] R. Loewy. Extreme points of a convex subset of the cone of positive definite matrices. *Mathematische Annalen*, 253:227–232, 1980.

[43] L. Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information Theory*, IT-25:1–7, 1979.

[44] L. Lovász. Self-dual polytopes and the chromatic number of distance graphs on the sphere. *Acta Scientiarum Mathematicarum*, 45:317–323, 1983.

[45] L. Lovász. Combinatorial optimization: Some problems and trends. DIMACS Technical Report 92-53, 1992.

[46] L. Lovász and A. Schrijver. Matrix cones, projection representations, and stable set polyhedra. In *Polyhedral Combinatorics*, volume 1 of *DIMACS series in Discrete Mathematics and Theoretical Computer Science*, pages 1–17. American Mathematical Society, 1989.

[47] L. Lovász and A. Schrijver. Cones of matrices and setfunctions, and 0-1 optimization. *SIAM Journal on Optimization*, 1:166–190, 1990.

[48] S. Mahajan and H. Ramesh. Correctly derandomizing Goemans and Williamson's Max CUT algorithm. Unpublished manuscript, 1995.

[49] B. Mohar and S. Poljak. Eigenvalue methods in combinatorial optimization. In R. Brualdi, S. Friedland, and V. Klee, editors, *Combinatorial and Graph-Theoretic Problems in Linear Algebra*, volume 50 of *The IMA Volumes in Mathematics and its Applications*, pages 107–151. Springer-Verlag, 1993.

[50] Y. Nesterov and A. Nemirovskii. *Self-Concordant Functions and Polynomial Time Methods in Convex Programming.* Central Economic and Mathematical Institute, USSR Academy of Science, Moscow, 1989.

[51] Y. Nesterov and A. Nemirovskii. *Interior Point Polynomial Methods in Convex Programming.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1994.

[52] G. I. Orlova and Y. G. Dorfman. Finding the maximal cut in a graph. *Engineering Cybernetics*, pages 502–506, 1972.

[53] M. L. Overton and R. S. Womersley. On the sum of the largest eigenvalues of a symmetric matrix. *SIAM Journal on Matrix Analysis and Applications*, 13:41–45, 1992.

[54] M. L. Overton and R. S. Womersley. Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrices. *Mathematical Programming*, 62:321–357, 1993.

[55] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[56] G. Pataki. On the multiplicity of optimal eigenvalues. Management Science Research Report MSRR-604, GSIA, Carnegie Mellon University, 1994.

[57] G. Pataki. On cone-LP's and semi-definite programs: Facial structure, basic solutions, and the simplex method. GSIA Working Paper WP 1995-03, Carnegie-Mellon University, 1995.

[58] S. Poljak and F. Rendl. Nonpolyhedral relaxations of graph-bisection problems. DIMACS Technical Report 92-55, 1992. To appear in *SIAM Journal on Optimization*.

[59] S. Poljak and F. Rendl. Node and edge relaxations of the max-cut problem. *Computing*, 52:123–137, 1994. Also appears as Report 266, Technische Universität Graz.

[60] S. Poljak and F. Rendl. Solving the max-cut problem using eigenvalues. In M. Lucertini, G. Rinaldi, A. Sassano, and B. Simeone, editors, *Partitioning and Decomposition in Combinatorial Optimization*, Discrete Applied Mathematics. 1995. To appear.

[61] S. Poljak and D. Turzík. A polynomial algorithm for constructing a large bipartite subgraph, with an application to a satisfiability problem. *Canadian Journal of Mathematics*, 34:519–524, 1982.

[62] S. Poljak and Z. Tuza. The max-cut problem — a survey. In W. Cook, L. Lovász, and P. Seymour, editors, *Special Year on Combinatorial Optimization*, DIMACS series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1995. To be published.

[63] G. Reinelt. TSPLIB – A traveling salesman problem library. *ORSA Journal on Computing*, 3:376–384, 1991.

[64] F. Rendl, R. Vanderbei, and H. Wolkowicz. Interior point methods for max-min eigenvalue problems. Report 264, Technische Universität Graz, 1993.

[65] B. Rosenfeld. *A history of non-Euclidean geometry*. Springer-Verlag, Berlin, 1988.

[66] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–565, 1976.

[67] P. M. Vaidya. A new algorithm for minimizing convex functions over convex sets. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 338–343, 1989.

[68] L. Vandenberghe and S. Boyd. Semidefinite programming. Submitted to SIAM Review, 1994.

[69] P. M. Vitányi. How well can a graph be n-colored? *Discrete Mathematics*, 34:69–80, 1981.

[70] H. Wolkowicz. Some applications of optimization in matrix theory. *Linear Algebra and Its Applications*, 40:101–118, 1981.

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