Check for updates

.878-Approximation Algorithms for MAX CUT and MAX 2SAT

*Michel X. Goemans**
M.I.T.

David P. Williamson†
Cornell University

Abstract

We present randomized approximation algorithms for the MAX CUT and 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. We then show how to derandomize the algorithm to obtain approximation algorithms with the same performance guarantee of .87856. The previous best-known approximation algorithms for these problems had performance guarantees of $\frac{1}{2}$ for MAX CUT and $\frac{3}{4}$ for MAX 2SAT. A slight extension of our analysis leads to a .79607-approximation algorithm for the maximum directed cut problem, where a $\frac{1}{4}$-approximation algorithm was the previous best-known algorithm. Our algorithm gives the first substantial progress in approximating MAX CUT in nearly twenty years, and, to the best of our knowledge, represents the first use of semidefinite programming in the design of approximation algorithms.

*Address: Dept. of Mathematics, Room 2-372, M.I.T., Cambridge, MA 02139. Email: goemans@math.mit.edu. Research supported in part by NSF contract 9302476-CCR, Air Force contract F49620-92-J-0125 and DARPA contract N00014-92-J-1799.

†Address: School of Operations Research and Industrial Engineering, 237 ETC Building, Cornell University, Ithaca, NY 14853. Email: dpw@cs.cornell.edu. Research supported by an NSF Postdoctoral Fellowship. This research was conducted while the author was visiting MIT.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association of Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

STOC 94- 5/94 Montreal, Quebec, Canada
© 1994 ACM 0-89791-663-8/94/0005..$3.50

Introduction

Given an undirected graph $G = (V, E)$ and nonnegative weights $w_{ij} = w_{ji}$ on the edges $(i, j) \in E$, the maximum cut problem (MAX CUT) is that of finding the set of vertices S that maximizes the weight of the edges in the *cut* $(S, \bar{S})$; 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 [19], and has long been known to be NP-complete even if the problem is unweighted; that is, if $w_{ij} = 1$ for all $(i, j) \in E$ [9]. The MAX CUT problem is solvable in polynomial time for some special classes of graphs (e.g. if the graph is planar [29, 14]). Besides its theoretical importance, the MAX CUT problem has applications in circuit layout design and statistical physics (Barahona et al. [3]). For a comprehensive survey of the MAX CUT problem, the reader is referred to Poljak and Tuza [38].

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. The constant ρ is sometimes called the *performance guarantee* of the algorithm. In 1976, Sahni and Gonzales [40] 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 $\frac{1}{2} + \frac{1}{2m}$ [42], $\frac{1}{2} + \frac{n-1}{4m}$ [37], and $\frac{1}{2} + \frac{1}{2n}$ [17] (where $n = |V|$ and $m = |E|$), 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. We also show how to derandomize the algorithm to obtain a deterministic $(\alpha - \epsilon)$-approximation algorithm, for any $\epsilon > 0$. The algorithm for MAX CUT also leads directly to randomized and deterministic $(\alpha - \epsilon)$-approximation algorithms for the maximum 2-satisfiability problem (MAX 2SAT). The previously best known algorithm for this problem has a performance guarantee of $\frac{3}{4}$ and is due to Yannakakis [44] (see also Goemans and Williamson [10]). Coppersmith and Shmoys have independently observed that the improved 2SAT algorithm leads to a slightly better than $\frac{3}{4}$-approximation algorithm for the overall MAX SAT problem. Finally, a slight extension of our analysis yields a .79607-approximation algorithm for the maximum directed cut problem (MAX DICUT). The previous best known algorithm for MAX DICUT has a performance guarantee of $\frac{1}{4}$ [32].

Our algorithm depends on a means of randomly rounding a solution to a nonlinear relaxation. 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 $\sum_{i<j} w_{ij}$.

A semidefinite program is the optimization problem of 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 [43] and the exposition by Alizadeh [1]). The simplex method and primal-dual algorithms can be generalized to semidefinite programs (Pataki [33]). 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. [13]) and other polynomial-time algorithms for convex programming (Vaidya [41]) as well as interior-point methods (Nesterov and Nemirovskii [27, 28] 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. [13] and Section 3.3 of Alizadeh [1].

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 [22] 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 (or the largest clique) in a perfect graph (Grötschel et al. [12]). More recently, there has been increased interest in semidefinite programming [24, 25, 1, 33, 35, 8, 23]. This started with the work of Lovász and Schrijver [24, 25], who developed a machinery to define tighter and tighter relaxations of any integer program based on quadratic and semidefinite programming. This demonstrated the wide applicability and the power of semidefinite programming for combinatorial optimization problems. A consequence of our algorithm is a proof that a semidefinite program is a much tighter relaxation in the worst case than any known linear programming relaxation for the MAX CUT problem.

An eigenvalue minimization problem consists of minimizing a linear combination of the k largest eigenvalues of a matrix subject to equality constraints on the matrix. These problems can be solved in polynomial time by the ellipsoid algorithm [13], since the objective function can be seen to be convex. There is an abundant literature on spectral bounds for combinatorial optimization problems (see the survey paper by Mohar and Poljak [26]). Building on work by Overton and Womersley [31, 30], Alizadeh [1] has shown that eigenvalue minimization problems can be formulated as semidefinite programs.

For MAX CUT, the nonlinear relaxation we consider is equivalent to a spectral bound proposed by Delorme and Poljak [5, 4]. This equivalence to the semidefinite program we consider was established by Poljak and Rendl [35]. As shown by Poljak and Rendl [34, 36] and Delorme and Poljak [6], the spectral bound provides a very good bound on the maximum cut in practice. Delorme and Poljak [5, 4] study the worst-case ratio between the maximum cut and their spectral 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 $\alpha = 0.87856\ldots$, 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 [32], 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]. Nevertheless, since the appearance of an abstract of this paper, Feige and Goemans [7] have extended our technique to yield a .931-approximation algorithm for MAX 2SAT and a .859-approximation algorithm for MAX DICUT. In addition, by using semidefinite programming and similar rounding ideas, Karger, Motwani, and Sudan [18] 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. Thus it seems likely that the techniques in this paper will continue to prove useful in designing approximation algorithms.

The abstract is structured as follows. In Section 1, we present the randomized algorithm for MAX CUT and its analysis, as well as connections of the semidefinite programming bound we use to other work on the MAX CUT problem. In Section 2, we show how to extend the algorithm to an algorithm for MAX 2SAT, MAX DICUT, and other problems. Section 3 presents the derandomization of the algorithm, and we conclude with a few remarks in Section 4.

1 The Randomized Approximation Algorithm for MAX CUT

1.1 The Algorithm

Given a graph with vertex set $V = \{1, \ldots, n\}$ and nonnegative 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:

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

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. 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 norm. Since the linear space spanned by the vectors v_i has dimension at most n, we can assume that these vectors belong to R^n (or R^m for some $m \leq n$), or more precisely to the n-dimensional unit sphere S_n (or S_m for $m \leq n$). 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 two very natural ways of guaranteeing this property. One can either 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, or by $\frac{2\arccos(v_i \cdot v_j)}{\pi}$ which corresponds to the angle between v_i and v_j scaled to range between 0 and 2. The resulting relaxations are denoted by (P) and (R):

$$\begin{array}{lll} & \text{Maximize} & \frac{1}{2}\sum_{i<j} w_{ij}(1 - v_i \cdot v_j) \\ (P) & \text{subject to:} & v_i \in S_n \qquad \forall i \in V \end{array}$$

and

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

Relaxation (P) plays a crucial role in the design of our approximation algorithm, while the analysis of the algorithm will show that (R) is in fact equivalent to the MAX CUT problem. We will show in Section 1.3 we can solve the relaxation (P) 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 nonnegative 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) (and (R) too) 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 Section 1.2 that, given any set of vectors $v_i \in S_n$, the expected weight of the cut defined by a random hyperplane is

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

Therefore there must exist a cut of value at least $E[W]$, implying (somewhat surprisingly) that the relaxation (R) is in fact a reformulation of the MAX CUT problem. Moreover, we will also show 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 defer the discussion on solving the relaxation (P) to Section 1.3. We note here, however, that strictly speaking, we cannot solve (P) to optimality in polynomial time; both the vectors and the optimal value Z^*_P might in fact be irrational. In Section 1.3, we describe two equivalent formulations to (P). One is based on semidefinite programming, while the other is defined in terms of eigenvalues. Using an algorithm for semidefinite programming (see introduction), one 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 $\alpha(Z^*_P - \epsilon) \geq (\alpha - \epsilon)Z^*_{MC}$, assuming the weights are integral. The resulting algorithm is thus a randomized $(\alpha - \epsilon)$-approximation algorithm for MAX CUT, whose running time dependence on ϵ is polynomial in $\log \frac{1}{\epsilon}$.

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 [20, 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 [20, p. 117]). Hence the algorithm can be made to run in polynomial time.

1.2 The Analysis

Let $\{v_1, \ldots, v_n\}$ be 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 subsection. In this subsection we will show the following theorem.

Theorem 1.1

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

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

As we have argued above, this theorem implies that the algorithm has a performance guarantee of $\alpha - \epsilon$. The proof of this theorem is amazingly elementary.

Given a vector r drawn uniformly from the unit sphere S_n, we know by the linearity of expectations 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 1.1 is then implied by the following two lemmas.

Lemma 1.2

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

Lemma 1.3 For $-1 \leq y \leq 1$, $\frac{1}{\pi} \arccos(y) \geq \alpha \cdot \frac{1}{2}(1 - y)$ where $\alpha = \min_{0 < \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1-\cos\theta} > 0.87856$.

We will now prove each lemma in turn.

Proof of Lemma 1.2: 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 has thus 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. ■

Proof of Lemma 1.3: The lemma follows straightforwardly by using the change of variables $\cos\theta = y$. One can use simple calculus to verify that $\frac{2}{\pi}\theta > .87856(1 - \cos\theta)$. Also, using simple calculus, one can see that α achieves its value for θ the non-zero root of $\cos\theta + \theta \sin\theta = 1$. ■

1.3 Relaxations

The relaxation (P) can be reformulated in two (seemingly very different) ways.

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 (psd) if for every vector $x \in \mathrm{R}^n$, $x^T A x \geq 0$. The following statements are equivalent for a symmetric matrix A (see e.g. [21]): (i) A is positive semidefinite, (ii) all eigenvalues of A are nonnegative, 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 $m \leq n$. Given a symmetric positive semidefinite matrix A, an $m \times n$ matrix B of full row-rank satisfying (iii) can be obtained in polynomial time using an incomplete Cholesky decomposition [11, 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 exactly 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 $\{v_1, \ldots, v_n\}$ [21, p. 110].Using this equivalence, we can reformulate (P) as a semidefinite program:

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

where $Y = (y_{ij})$. As mentioned in the introduction, semidefinite programs can be solved to within an additive error of ϵ in time polynomial in the input size and $\log \frac{1}{\epsilon}$. 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 $\frac{1}{2}\sum_{i<j} w_{ij}(1 - v_i \cdot v_j) \geq Z_P^* - \epsilon$.

The relaxation (P) is also equivalent to an eigenvalue upper bound on the value of the maximum cut Z_{MC}^* introduced by Delorme and Poljak [5, 4]. 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 $l_{ii} = \sum_{k=1}^n w_{ik}$. Given a vector u, $diag(u)$ denotes the diagonal matrix with entries u_i for $i = 1, \ldots, n$ on the main diagonal. The maximum eigenvalue of a matrix A is denoted by $\lambda_{max}(A)$.

Lemma 1.4 [[5]] Let $u \in \mathrm{R}^n$ satisfy $u_1 + \ldots + u_n = 0$. Then

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

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*. The bound proposed by Delorme and Poljak [5] is to optimize $g(u)$ over all correcting vectors:

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

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 [35]. Although the equality between Z_P^* and Z_{EIG}^* is not crucial for the result of this paper, we nevertheless elaborate on this topic in the following paragraphs.

The equivalence between (SD) and (EIG) follows in fact from their correspondence to a dual pair of semidefinite programs. As usual, weak duality is fairly easy to show. To see this, consider a correcting vector u and a matrix Y feasible for (SD). Let $\lambda = \lambda_{max}(L + diag(u))$. By definition, $M = \lambda I - L - diag(u)$ is positive semidefinite, as is Y. Thus $Tr(MY) \geq 0$ where Tr denotes the trace (see [21, p. 218, ex. 14]). But, $Tr(MY) = n\lambda - Tr(LY)$, where we have used the facts that $y_{ii} = 1$ for all i and that u is a correcting vector. Noticing that $Tr(LY)$ is precisely four times the objective function of (SD), one derives weak duality, showing that $Z_P^* \leq Z_{EIG}^*$.

For the optimum correcting vector and the optimum matrix Y, strong duality for semidefinite programming guarantees that the duality gap is equal to 0 and, thus, $Tr(MY) = 0$, where M is defined as above. Since both M and Y are positive semidefinite, we derive that $MY = 0$ (see [21, p. 218, ex. 14]). From the definition of M and the fact that $MY = 0$, we derive that $\lambda Y - LY = diag(u)Y$, implying that $u_i = \lambda - \sum_j l_{ij} y_{ij}$ for all i. Since $\lambda = \frac{4}{n} Z_{EIG}^* = \frac{4}{n} Z_P^*$, we observe that we can easily deduce the optimum correcting vector from the optimum matrix Y for (SD).

1.4 Quality of the Relaxation

Theorem 1.1 implies the following corollary:

Corollary 1.5 For any instance of MAX CUT,

$$\frac{Z_{MC}^*}{Z_P^*} \geq \alpha > 0.87856.$$

For the 5-cycle, Delorme and Poljak [5] 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, we have constructed instances for

which $Z_P^*/E[W] < 0.8786$, showing that the analysis of our algorithm is practically tight.

Poljak and Rendl [34, 36] (see also Delorme and Poljak [6]) report computational results showing that the bound Z_{EIG}^* is typically less than 2-5% and never worse than 8% away from Z_{MC}^*.

We have implemented the randomized algorithm using a code supplied by R. Vanderbei [39] for a special class of semidefinite programs. Very preliminary experiments have shown that the algorithm typically generates cuts within 4-5% of the semidefinite bound; see the full paper for details.

2 Generalizations

We can use the same technique as above to approximate several other problems.

MAX RES CUT For the 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, we have an $(\alpha - \epsilon)$-approximation algorithm.

MAX 2SAT For the maximum 2-satisfiability problem, we also have an $(\alpha - \epsilon)$-approximation algorithm.

MAX DICUT For the maximum directed cut problem, we have a $(\beta - \epsilon)$-approximation algorithm where $\beta > 0.79607$.

The extension 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 algorithm above 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.

In the next section, we show that the algorithm for MAX CUT can also be used to approximate a more general integer quadratic program, and that MAX 2SAT can be modelled as such. In Section 2.2, we show how to approximate another general integer quadratic program, which can be used to model the MAX DICUT problem. Before describing these extensions, we should point out that nothing prevents the combination of these three generalizations.

2.1 MAX 2SAT

We consider the integer quadratic program

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

where a_{ij} and b_{ij} are non-negative. The objective function of (Q') is thus a nonnegative linear form in $1 \pm y_i y_j$. We relax (Q') to:

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

We approximate (Q') by using exactly the same algorithm as before (except that (P) is replaced by (P')). Practically the same analysis shows that the algorithm is a randomized $(\alpha - \epsilon)$-approximation algorithm for (Q'); we only need the following additional lemma.

Lemma 2.1 For $-1 \leq y \leq 1, 1 - \frac{1}{\pi}\arccos(y) \geq \alpha \cdot \frac{1}{2}(1 + y)$ where $\alpha = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos\theta} > 0.87856$.

Proof: The lemma follows from Lemma 1.3 by using a change of variables $z = -y$ and noting that $\pi - \arccos(x) = \arccos(-x)$. ■

We now show that the maximum 2-satisfiability problem can be modelled by (Q'). 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 $\bar{x}$. 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 contains no more than two literals. MAX 2SAT is NP-complete [9]; the best approximation algorithm known previously has a performance guarantee of $\frac{3}{4}$ and is due to Yannakakis [44] (see also Goemans and Williamson [10]). As with the MAX CUT problem, MAX 2SAT is known to be MAX SNP-hard [32]; thus there exists some constant $c < 1$ such that the existence of a c-approximation algorithm implies that $P = NP$ [2]. Haglin [15, 16] has shown that any α-approximation algorithm for MAX RES CUT can be translated into an α-approximation algorithm for MAX 2SAT, but we will show a direct algorithm here.

We will model MAX 2SAT using the integer quadratic program (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, $v(x_i) = \frac{1+y_0y_i}{2}$ and $v(\bar{x}_i) = 1 - v(x_i) = \frac{1-y_0y_i}{2}$. Observe that

$$\begin{aligned} v(x_i \vee x_j) &= 1 - v(\bar{x}_i \wedge \bar{x}_j) \\ &= 1 - v(\bar{x}_i)v(\bar{x}_j) \\ &= 1 - \frac{1-y_0y_i}{2}\frac{1-y_0y_j}{2} \\ &= \frac{1}{4}(3 + y_0y_i + y_0y_j - y_0^2y_iy_j) \\ &= \frac{1+y_0y_i}{4} + \frac{1+y_0y_j}{4} + \frac{1-y_iy_j}{4}. \end{aligned}$$

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

$$\text{Maximize} \quad \sum_{C_j \in \mathcal{C}} w_j v(C_j)$$

$$(SAT) \quad \text{subject to:} \quad y_i \in \{-1, 1\} \qquad \forall i \in \{0, 1, \ldots, n\},$$

where the $v(C_j)$ are non-negative linear combinations of $1 + y_iy_j$ and $1 - y_iy_j$. The (SAT) program is in the same form as (Q'), so by the theorem above, our algorithm is an $(\alpha - \epsilon)$-approximation algorithm for the MAX 2SAT problem.

Don Coppersmith and David Shmoys have observed independently that the improved MAX 2SAT algorithm leads to slightly improved MAX SAT algorithm. Given a MAX SAT instance, formulate the program (SAT) for clauses of length 1 and 2 as above. Then take the better of the two solutions given by our algorithm on (SAT) and the Goemans/Williamson $\frac{3}{4}$-approximation algorithm [10] for MAX SAT. This can be shown to yield a .755-approximation algorithm for MAX SAT. Other slight improvements can be made; see the full paper for details.

2.2 MAX DICUT

We consider the integer quadratic program

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

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 nonnegative 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 relax (Q'') to:

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

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

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

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

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

$$+ d_{ijk} \cdot \Pr[\text{sgn}(v_i \cdot r) = \text{sgn}(v_j \cdot r) = \text{sgn}(v_k \cdot r)]].$$

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

Lemma 2.2

$$\Pr[\text{sgn}(v_i \cdot r) = \text{sgn}(v_j \cdot r) = \text{sgn}(v_k \cdot r)] = 1 - \frac{1}{2\pi}(a + b + c).$$

Lemma 2.3 For any $v_i, v_j, v_k \in S_n$,

$$1 - \frac{1}{2\pi}(a + b + c) \ge \frac{\beta}{4}[1 + \cos(a) + \cos(b) + \cos(c)],$$

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

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 previous best known approximation algorithm for MAX DICUT has a performance guarantee of $\frac{1}{4}$ [32].

We can model the MAX DICUT problem using the program (Q''). We introduce a variable y_i for each $i \in V$, and, as with the MAX 2SAT program, we introduce 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 $\frac{1}{4}w_{ij}(1+y_iy_0)(1-y_jy_0)$ to the cut. Summing over all arcs $(i,j) \in A$ gives a program of the same form as (Q''). Hence we obtain a .79607-approximation algorithm for MAX DICUT. Finally, if the directed graph has weighted indegree of every node equal to weighted outdegree, the program (Q'') reduces to one of the form (Q'), and therefore our approximation algorithm has a performance guarantee of $(\alpha - \epsilon)$.

3 Derandomization

The algorithm can be derandomized, resulting in a deterministic approximation algorithm with the same performance guarantee. The method we use is the classical method of conditional expectations. We illustrate the derandomization on MAX CUT, and present a deterministic $(\alpha - \epsilon)$-approximation algorithm for any $\epsilon > 0$.

Assume we are given a set of unit vectors $v_1, \ldots, v_n$ spanning a linear space of dimension m. We can thus assume that the vectors v_i belong to S_m. Define $f(v_1, \ldots, v_n) = \frac{1}{\pi}\sum_{i<j} w_{ij} \arccos(v_i \cdot v_j) = E[W]$. In the randomized algorithm, we directly obtain from the v_i's a set of y_i's belonging to S_1 (i.e. scalars taking values in $\{-1,+1\}$), proving the existence of $y_1, \ldots, y_n$ such that $f(y_1, \ldots, y_n) \geq f(v_1, \ldots, v_n)$. In the deterministic algorithm, we shall decrease the dimension m one unit at a time.

Consider a vector r drawn uniformly from S_m. For simplicity of exposition, we shall assume that r is obtained by generating independent standard normal variables $x_1, \ldots, x_m$. We express the last two coordinates of x in polar coordinates: $(x_{m-1}, x_m) = (s\cos\theta, s\sin\theta)$. We restrict θ to lie in $[-\pi/2, \pi/2)$, implying that s is allowed to be negative. Observe that θ is undefined on a set of measure 0 (when $s = 0$). Although we don't really need this fact, it is easy to see that θ is uniformly distributed in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$. (Indeed, (x_{m-1}, x_m) is a bivariate normal variable and its density depends only on $s^2 = x_{m-1}^2 + x_m^2$.) We shall condition on the value of θ. For a given value of θ, let $f_\theta(v_1, \ldots, v_n) = E[W|\theta]$. We know that

$$\begin{aligned} f(v_1, \ldots, v_n) &= E[W] = E_\theta[E[W|\theta]] \\ &= \int_{-\pi/2}^{\pi/2} f_\theta(v_1, \ldots, v_n) g(\theta) d\theta \\ &\leq \max_{-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}} f_\theta(v_1, v_2, \ldots, v_n), \end{aligned}$$

where $g(\theta)$ is the probability density function. As argued above, $g(\theta) = \frac{1}{\pi}$, but the fact that θ is uniformly distributed is not needed to derive the inequality.

We first describe a set of vectors $u_1(\theta), \ldots, u_n(\theta) \in S_{m-1}$ such that $f_\theta(v_1, \ldots, v_n) = f(u_1(\theta), \ldots, u_n(\theta))$ and, thus, for a fixed value of θ, f_θ can easily be evaluated. We will then describe a discrete set Θ of size polynomial in n and $1/\delta$ such that

$$\begin{aligned} &\max_{-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}} f_\theta(v_1, v_2, \ldots, v_n) - \delta W_{tot} \\ &\quad \leq \max_{\theta \in \Theta} f_\theta(v_1, v_2, \ldots, v_n), \end{aligned}$$

where $W_{tot} = \sum_{i<j} w_{ij}$. Selecting for θ the value maximizing f_θ over Θ, we obtain vectors $u_1, \ldots, u_n \in S_{m-1}$ such that $f(u_1, \ldots, u_n) \geq f(v_1, \ldots, v_n) - \delta W_{tot}$. We can then continue the process until the dimension is equal to 1. At that point, we have scalars $y_i \in \{-1, +1\}$ such that

$$f(y_1, \ldots, y_n) \geq f(v_1, \ldots, v_n) - n\delta W_{tot}.$$

We can assume without loss of generality that $f(v_1, \ldots, v_n) \geq W_{tot}/2$ since this is true for an orthonormal basis of n vectors. Thus

$$f(y_1, \ldots, y_n) \geq (1 - 2n\delta) f(v_1, \ldots, v_n).$$

Recalling that $f(v_1, \ldots, v_n) \geq \alpha(Z^*_{MC} - \epsilon) \geq \alpha(1 - \epsilon)Z^*_{MC}$, we see that the derandomized algorithm achieves a performance guarantee of $(1 - 2n\delta)\alpha(1-\epsilon)$, which can be made to be $(\alpha - \epsilon')$ for any $\epsilon' > 0$, by setting $\epsilon = \frac{\epsilon'}{2}$ and $\delta = \frac{\epsilon'}{4n}$. The running time of the resulting deterministic $(\alpha - \epsilon')$-approximation algorithm is thus polynomial in $\frac{1}{\epsilon'}$. Our derandomized algorithm heavily involves square roots and trigonometric functions. As usual, we implicitly assume that all these operations are computed with precision polynomial in the input size and $\log \frac{1}{\epsilon'}$, in order to guarantee polynomiality and to avoid losing in the performance guarantee.

We first show how to define $u_i(\theta)$ from v_i and θ. Given a vector $z \in \mathrm{R}^m$, we define, for any scalar a, $z|a$ to be the vector in R^{m-1} having coordinates $z_1, z_2, \ldots, z_{m-2}$ and a. Recall that $(x_{m-1}, x_m) = (s\cos\theta, s\sin\theta)$. We also express the last two coordinates of v_i in polar form: $(v_{i,m-1}, v_{im}) = (t_i\cos\gamma_i, t_i\sin\gamma_i)$, where $t_i = \sqrt{v_{i,m-1}^2 + v_{im}^2} \geq 0$ and $\gamma_i \in [0, 2\pi)$. We observe that

$$\begin{aligned} x_{m-1}v_{i,m-1} + x_m v_{im} &= st_i(\cos\gamma_i\cos\theta + \sin\gamma_i\sin\theta) \\ &= st_i\cos(\gamma_i - \theta). \end{aligned}$$

Setting $x' = x|s$ and

$$a_i(\theta) = v_i|[t_i\cos(\gamma_i - \theta)],$$

we derive that $x \cdot v_i = x' \cdot a_i(\theta)$. Let $u_i(\theta) = a_i(\theta)/\|a_i(\theta)\|$. The above discussion shows that $\mathrm{sgn}(x \cdot$

$v_i) = \mathrm{sgn}(x' \cdot u_i(\theta))$. Moreover, once we condition on θ, s is normally distributed (and independent of $x_1, \ldots, x_{m-2}$), implying that $x'/\|x'\|$ is uniformly distributed on S_{m-1}. Therefore, $f_\theta(v_1, \ldots, v_n)$ can be evaluated as $f(u_1(\theta), \ldots, u_n(\theta))$.

In order to compute approximately the maximum of $f(u_1(\theta), \ldots, u_n(\theta))$ over $\theta \in [-\pi/2, \pi/2]$, we discretize the interval in the following way. For every i, let $\alpha_i(\theta)$ denote the angle between the vectors $v_i|0$ and $a_i(\theta)$. Since $v_i|0$ and $a_i(\theta) - (v_i|0)$ are orthogonal, we derive that

$$\tan \alpha_i(\theta) = \frac{\|a_i(\theta) - (v_i|0)\|}{\|v_i|0\|} = \frac{t_i |\cos(\gamma_i - \theta)|}{\sqrt{\sum_{j=1}^{m-2} v_{ij}^2}},$$

and from this that $\alpha_i(\theta)$ ranges between 0 and

$$m_i = \arctan \frac{t_i}{\sqrt{\sum_{j=1}^{m-2} v_{ij}^2}} \leq \frac{\pi}{2}.$$

As θ varies, $\alpha_i(\theta)$ covers the interval $[0, m_i]$ twice, because the dependence of $\alpha_i(\theta)$ on θ can be stated in terms of $\cos(\gamma_i - \theta)$. Therefore, we can choose $2m_i/\nu \leq \pi/\nu$ values of θ, say $\theta_j^{(i)}$ for $j \leq 2m_i/\nu$, such that for all $\theta_j^{(i)} \leq \theta \leq \theta_{j+1}^{(i)}$ we have $|\alpha_i(\theta) - \alpha_i(\theta_j^{(i)})| \leq \nu$. Let Θ denote the union of these angles over $i = 1, \ldots, n$. Notice that $|\Theta| \leq \pi n/\nu$.

Lemma 3.1 There exists Θ with $|\Theta| \leq \frac{2n}{\delta}$ such that

$$\max_{-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}} f_\theta(v_1, v_2, \ldots, v_n) - \delta W_{tot} \leq \max_{\theta \in \Theta} f_\theta(v_1, v_2, \ldots, v_n),$$

where $W_{tot} = \sum_{i<j} w_{ij}$.

Proof of Lemma 3.1: Let $\Theta = \{\theta_1 < \ldots < \theta_p\}$ be the set corresponding to $\nu = \delta\pi/2$. We claim that for any $i < j$ and any $\theta_k \leq \theta \leq \theta_{k+1}$, we have

$$|\arccos(u_i(\theta) \cdot u_j(\theta)) - \arccos(u_i(\theta_k) \cdot u_j(\theta_k))| \leq 2\nu = \delta\pi.$$

Summing the above inequality over all $i < j$ and using the definition of f, one derives the lemma.

To prove the claim, notice that the angle between $u_i(\theta)$ and $u_j(\theta)$ differs from the angle between $u_i(\theta_k)$ and $u_j(\theta_k)$ by at most $|\alpha_i(\theta) - \alpha_i(\theta_k)| + |\alpha_j(\theta) - \alpha_j(\theta_k)|$, which by construction of Θ is at most 2ν. ■

4 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 [10]). Given the work of Lovász and Schrijver [24, 25], 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.

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.

References

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. To appear in *SIAM Journal on Optimization*. A preliminary version appeared in the *Proc. 2nd IPCO*, 1992.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc. 33rd FOCS*, pages 14–23, 1992.

[3] 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.

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

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

[6] 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.

[7] U. Feige and M. X. Goemans. Personal communication, 1994.

[8] U. Feige and L. Lovász. Two-prover one-round proof systems: Their power and their problems. In *Proc. 24th STOC*, pages 733–744, 1992.

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

[10] M. X. Goemans and D. P. Williamson. New 3/4-approximation algorithms for MAX SAT. To appear in *SIAM J. Disc. Math.* Earlier version appeared in the *Proc. 3rd IPCO*, pages 313–321, 1993.

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

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

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

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

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

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

[17] 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.

[18] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. In preparation, 1994.

[19] 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.

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

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

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

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

[24] 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.

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

[26] B. Mohar and S. Poljak. Eigenvalues in combinatorial optimization. Technical report, University of Ljubljana, 1992.

[27] 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.

[28] Y. Nesterov and A. Nemirovskii. *Interior Point Polynomial Methods in Convex Programming.* Society for Industrial and Applied Mathematics, 1993.

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

[30] 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.

[31] 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.

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

[33] G. Pataki. Algorithms for cone-optimization problems and semi-definite programming. Manuscript, 1993.

[34] S. Poljak and F. Rendl. Solving the max-cut problem using eigenvalues. Report 91735-OR, Institute für Diskrete Mathematik, Universität Bonn, 1991.

[35] S. Poljak and F. Rendl. Nonpolyhedral relaxations of graph-bisection problems. DIMACS Technical Report 92-55, 1992.

[36] S. Poljak and F. Rendl. Computational experiments with node and edge relaxations of the max-cut problem. Report 266, Technische Universität Graz, 1993.

[37] 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.

[38] S. Poljak and Z. Tuza. The max-cut problem — a survey. Manuscript, 1993.

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

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

[41] P. M. Vaidya. A new algorithm for minimizing convex functions over convex sets. In *Proc. 30th FOCS*, pages 338–343, 1989.

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

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

[44] M. Yannakakis. On the approximation of maximum satisfiability. In *Proc. 3rd SODA*, pages 1–9, 1992.