Rounding Techniques for Semidefinite Relaxations

6-856 Randomized algorithms

Jerome Le Ny

May 11, 2005

Abstract

This report reviews some approximation algorithms for combinatorial optimization problems, based on a semidefinite relaxation followed by randomized rounding.

Contents

Introduction

In a now celebrated paper[GW96], Goemans and Williamson introduced the use of semidefinite programming for the design of approximation algorithms for NP-hard problems. Their idea inspired a great deal of research activity, since it had the potential to generalize the standard methods based on linear programming to a broader class of problems as well as to improve known approximation ratios dramatically. As with linear programming relaxations, one needs to have a way to round the solution obtained from the semidefinite program. Goemans and Williamson showed that this could be done by separating the solution vectors using random hyperplanes. Their algorithm was subsequently derandomized in [MR99]. As we will see, the hyperplane separation is quite powerful, and has been the basis for a number of improvements and generalizations. Part 2 will review some of this work, in increasing order of generality. In part 3, I discuss a recent paper of Alon and Naor [AN04]. They present three algorithms: two of them are randomized and do not really depart from the random hyperplane rounding, but their analysis is more powerful and show that we can handle more general problems. Quite remarkably, they establish the connection between the probabilistic point of view (choosing a random hyperplane and taking

expectation), and techniques useful in functional analysis (averaging with a Gaussian measure). The important link between semidefinite relaxations of integer quadratic programs and functional analysis lies in Grothendieck's inequality.

1 Preliminaries

In this part, I review some notions and notation that will be useful in the following. This report is concerned with NP-hard maximization problems, for which we are interested in polynomial time approximation algorithms that for every instance produce solutions whose value (or expected value if the algorithm is randomized) is guaranteed to be within a ratio of at least α ($0 \leq \alpha \leq 1$) from the optimal solution. In fact, the problems that we will study are even $MAX - SNP$ hard: this implies, by the results in [PY91, ALM+92], that there exists some $\rho < 1$ such that the existence of a ρ-approximation, polynomial time algorithm for any of these problems would imply $P = NP$.

A common method for obtaining an approximation algorithm for a combinatorial optimization problem is based on linear programming. Goemans and Williamson [GW96] successfully extended this approach to semidefinite programming. The approach can be divided into the following steps:

- Formulate the problem as an integer quadratic program.
- Relax the problem to a semidefinite program.
- Solve the relaxation in polynomial time, obtaining a vector solution $v_1, \ldots, v_n$.
- Round the vector solution to an integer solution $x_1, \ldots, x_n$ (somehow).

In this report, we will investigate *randomized* rounding techniques that have been developped following the original method of Goemans and Williamson, which used a *random hyperplane* rounding. First however, we say a few words about semidefinite programs.

Semidefinite programming is a special case of convex programming where the feasibility region is an affine subspace of the cone of positive semidefinite matrices. Semidefinite programs are much more general that linear programs, but they are not much harder to solve. Most interior-point methods for linear programming have been generalized to semidefinite programs. As in linear programming, these methods have polynomial worst-case complexity, and perform well in practice. Semidefinite programming has been the object of much attention, partly because of applications in combinatorial optimization and in control theory [GLS88, Ali95, VB96].

We will not discuss semidefinite programming itself in this report. Suffices to say that all relaxation formulations presented can be solved via semidefinite programming, in the sense that an algorithm returns a solution which is within $\epsilon > 0$ of the optimum, in time polynomial in n and $\log(1/\epsilon)$. This epsilon should be included in the approximation ratio (i.e. strictly speaking, we obtain an $\alpha - \epsilon$ approximation), but since it can be arbitrarily small we will most often omit it in the discussion. For an overview of the semidefinite programs arising in combinatorial optimization and the duality theory of SDP, one can refer to [VB96]. It is interesting to note the close connections between relaxations used in combinatorial optimization (and other current "hot" topics in optimization) and relaxations of hard control problems, which date back at least to 1944 in the Soviet litterature.

We will write $Y \succcurlyeq 0$ to specify the constraint that the matrix Y must be a semidefinite matrix. Recall that the Gram matrix $[y_{ij}]$ of a set of vectors $v_1, \ldots, v_n$ in $\mathbb{R}^n$ is the matrix whose elements are given by $y_{ij} = v_i \cdot v_j$, where "." denotes the inner product. A Gram matrix is positive semidefinite. Conversely, given a semidefinite matrix Y, we can perform a Cholesky decomposition to write $Y = B^T B$, which gives a representation of Y as a Gram matrix, for which the vectors are the columns of B. Strictly speaking, we can only perform numerically an incomplete Cholesky factorization since we can encounter irrational numbers, but again we can approximate the result up to an arbitrary close number.

Here are some additional notations. If not stated otherwise, $\|.\|$ will denote the euclidian norm. When considering a graph $G = (V, E)$ (undirected), n is number of vertices and m is the number of edges. We will note for the standard normal distribution

$$\phi(r) = \frac{1}{\sqrt{2\pi}} e^{-r^2/2} \quad \text{and } \Phi(x) = \int_x^\infty \phi(r) dr$$

We will also use several times facts about random vectors chosen from the n-dimensional standard normal distribution. Let r be such a vector. Then the projections of r onto two lines l_1 and l_2 are independant and normally distributed iff l_1 and l_2 are orthogonal. Alternatively, we can say that under any rotation of the coordinate axis, the projections of r along these axes are independant standard normal random variables. The intersection of r with the unit sphere defines a point which has uniform distribution on the unit sphere.

2 Rounding via Random Hyperplane Partitions and Improvements

2.1 Random Hyperplane Partition [GW96]

In their seminal paper, Goemans and Williamson present randomized approximation algorithms based on a semidefinite relaxation for MAX CUT, MAX 2SAT, MAX DICUT and MAX SAT, when the weights are non-negative. The algorithms randomly round the solution of the semidefinite program using a hyperplane separation technique, which has proved to be an important tool, for either direct application or extension.

The method can be illustrated on the MAX CUT problem. Given an undirected graph $G = (V, E)$, with vertices $V = 1, \ldots, n$ and weights $w_{ij} = w_{ji} \geq 0$ on the edges $(i, j) \in E$, the maximum cut problem is that of finding the set of vertives S that maximizes the weight of the edges in the cut $(S, \overline{S})$; that is, the weight of the edges with one endpoint in S and the other in $\overline{S}$. We set $w_{ij} = 0$ if $(i, j) \notin E$ and denote the weight of a cut $(S, \overline{S})$ by $w(S, \overline{S}) = \sum_{i \in S, j \notin S} w_{ij}$. Note that assigning each vertex to S with probability $1/2$ produces a cut with expected value $1/2$ of the total weight (by linearity of expectation), and therefore provides a $1/2$-approximation algorithm. We are interested in improving on this result.

The weight of the maximum cut is given by the following integer quadratic program:

$$\begin{aligned} Z^*_{MC} = \ & \text{Max} \frac{1}{2} \sum_{i<j} w_{ij}(1 - y_i y_j) \\ & \text{subject to: } y_i \in \{-1, 1\} \quad \forall i \in V. \end{aligned} \tag{1}$$

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

The semidefinite relaxation of 1 is obtained by allowing y_i to be a multi-dimensional vector v_i of *unit Euclidian norm*. Since the linear space spanned the these vectors has dimension at most n, we can assume that these vectors belong to $\mathbb{R}^n$. If we replace $(1 - y_i y_j)$ by $(1 - v_i \cdot v_j)$, we recover the objective function when the vectors lie in a 1-dimensional space. We can then formulate the relaxation in terms of the Gram matrix $Y = [y_{ij}]$ of the vectors v_i, as follows:

$$\begin{aligned} Z^*_{R,MC} = \ & \text{Max} \frac{1}{2} \sum_{i<j} w_{ij}(1 - y_{ij}) \\ & \text{subject to: } y_{ii} = 1 \quad \forall i \in V \\ & Y \succcurlyeq 0. \end{aligned} \tag{2}$$

Once we have solved this relaxation, the vectors v_i are obtained via Cholesky factorization. Then an integral solution is found using the following randomized rounding:

- Choose a vector r uniformly distributed on the unit sphere S_n. This can be done by drawing n values $r_1, r_2, \ldots, r_n$ independantly from the standard normal distribution (and normalizing the resulting vector r, but this step can be skipped).
- Set $S = \{i | v_i \cdot r \geq 0\}$.

In words, we choose a random hyperplane through the origin, with normal r, and separate the vertices depending on the "side" of the hyperplane to which their vector belongs. If W is the cut produced this way, and $E[W]$ its expectation, the authors showed:

$$E[W] \geq \alpha_{gw} \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i \cdot v_j) = \alpha_{gw} Z^*_{R,MC} \geq \alpha_{gw} Z^*_{MC},$$

where $\alpha_{gw} = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1 - cos\theta} > .878$. Hence the algorithm has performance guarantee of $\alpha_{gw} - \epsilon$ for any desired ϵ for the MAX CUT problem and this proves that the upper bound obtained from the semidefinite relaxation is at most 14% suboptimal.

To analyze the algorithm, we first use the linearity of expectation to write

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

where $sign(x) = 1$ if $x \geq 0$ and -1 otherwise. The analysis is performed on a *term by term* basis, and the result follows essentially from the following lemma:

Lemma 1. *Given a vector drawn uniformly from the unit sphere S_n and two vectors v_i and v_j, we have*

$$P[sgn(v_i \cdot r) \neq sgn(v_j \cdot r)] = \frac{1}{\pi} \arccos(v_i \cdot v_j). \tag{3}$$

Proof. The lemma states that the probability the random hyperplane separates the two vectors is proportional to the angle between the two vectors $\theta = arccos(v_i \cdot v_j)$. We have $P[sgn(v_i \cdot r) \neq sgn(v_j \cdot r)] = 2P[v_i \cdot r \geq 0, v_j \cdot r < 0]$ by symmetry and the set $\{r : v_i \cdot r \geq 0, v_j \cdot r < 0\}$ has measure $\frac{\theta}{2\pi}$ times the measure of the full sphere. □

Since the weights w_{ij} are non-negative, and we can show $\frac{1}{\pi} \arccos(y) \geq \alpha_{gw} \frac{1}{2}(1-y)$ with α_{gw} defined above, we obtain $E[W] \geq \alpha_{gw} \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i \cdot v_j)$ and the approximation result.

The authors also give a better bound when the maximum cut is large, and we will revisit this point when discussing outward rotations and RPR^2. They briefly discuss the case where edges are negative, and we will explore this issue further in part 3.

2.2 Vector Projection [KMS98]

The results of Goemans and Williamson have immediately inspired other researchers for the design of new approximation algorithms. Karger, Motwani and Sudan used semidefinite programming for the problem of coloring k-colorable graphs. To round the solution, they used first random hyperplane partitions and then proved that for this problem, it is possible to optimize the method to obtain a better approximation ratio: this leads to the "vector projection" method, as a direct generalization of random hyperplace partition (this method is also called simply random projection in RPR^2, see below). Their algorithm colors a k-colorable graph on n vertices with $\min\{O(\Delta^{1-2/k} \log^{1/2} \Delta \log n), O(n^{1-3/(k+1)} \log^{1/2} n)\}$ colors, where Δ is the maximum degree of any vertex.

A legal coloring of a graph $G = (V, E)$ with k colors is a partition of its vertices into k independant sets: we can view this as an assignment of colors to the vertices such that no two adjacent vertices receive the same color. The minimum number of colors needed for such a coloring is called the chromatic number of G. As for the MAX CUT problem, the relaxation is based on assigning (n-dimensional) vectors to the vertices, instead of assigning colors directly. Again the vectors assigned to adjacent vertices should be different, in the sense that they form a large angle; then it is easy to separate them using a technique similar to the random hyperplane partition.

More precisely, given a real number $k \geq 1$, a vector k-coloring of G is an assignment of unit vectors $v_i \in \mathbb{R}^n$ to each vertex $i \in V$ such that for any two adjacent vertices i and j the inner product of their vectors satisfies the inequality

$$v_i \cdot v_j \leq -\frac{1}{k-1}$$

Here is how to relate a vector k-coloring to the solution of the original problem. First, we can show that for all positive integers k and n such that $k \leq n+1$, there exist k unit vectors in $\mathbb{R}^n$ such that the inner product of any distinct pair is $-1/(k-1)$. The proof of this fact is outside the scope of this report and can be found in the paper. This implies that if G is k-colorable then G has a vector k-coloring, since we can bijectively map the k colors to such k vectors. In particular, finding the minimum k such that G has a vector k-coloring provides a lower bound on the chromatic number of G. The harder part is to show that this lower bound is not too far from our objective, and for this a semidefinite formulation is introduced.

Let Y be the Gram matrix of the vectors $v_i, i \in V$. As for MAX CUT, this positive semidefinite matrix has ones on its diagonal. We solve the semidefinite program

$$\begin{aligned} Z^*_{R,C} = \ & \text{Min } t \\ & \text{subject to: } y_{ij} \leq t \quad \forall (i,j) \in E \\ & \qquad\qquad y_{ii} = 1 \quad \forall i \in V \\ & \qquad\qquad Y \succcurlyeq 0. \end{aligned} \tag{4}$$

and recover the vectors $\{v_i\}_{i=1}^n$ via (incomplete) Cholesky factorization. If the graph has a vector k-coloring, we obtain a vector $(k+\epsilon)$-coloring from this procedure, in time polynomial in k, n and $\log(1/\epsilon)$. As usual, we omit the ϵ in the discussion.

Let us focus first on the case $k = 3$ for clarity. We will show that using the random hyperplane technique we can transform a vector 3-coloring into a $O(\Delta^{\log_3 2})$-semicoloring as described below. A k-semicoloring of G is an assignment of k colors to at least half its vertices such that no two adjacent

vertices are assigned the same color. This gives an algorithm for semicoloring, which in turn leads to a coloring algorithm that uses up at most a logarithmic factor more colors than the semicoloring algorithm. This can be shown by recursively semicoloring the subset of vertices with no assignment yet, using a new set of colors at each step. In our case, this gives a $O(\Delta^{\log_3 2} \log n)$-coloring of G.

So we want to separate the vertices by choosing hyperplanes randomly. In our vector 3-coloring two vectors associated with an adjacent pair of vertices i and j have a dot product of at most $-1/2$, so the angle between the two vectors is at least $2\pi/3$. By picking a random hyperplane, lemma 1 tells us that we have a probability at least $2/3$ of separating adjacent vertices, i.e. of cutting a given edge. Picking h hyperplanes independantly at random, the probability that an edge is not cut by one of these hyperplanes if $(1/3)^h$ and so the expected fraction of the edges no cut is also $(1/3)^h$. Note also that the h hyperplanes can partition $\mathbb{R}^n$ in at most 2^h distinct regions. We give the same color to all vertices whose vectors lie in the same region. In particular, two adjacent vertices will receive the same color if the edge joining them is not cut. With these elements, we have

Theorem 2. *If a graph has a vector 3-coloring, then it has an $O(\Delta^{\log_3 2})$-semicoloring that can be constructed from the vector 3-coloring in polynomial time with high probability.*

Proof. Take $h = 2 + \lceil \log_3 \Delta \rceil$. Then $(1/3)^h \leq 1/(9\Delta)$ and $2^h = O(\Delta^{\log_3 2})$ (this is the number of color used). The expected number of edges not cut is at most $m/(9\Delta) \leq n/18 < n/8$ since m is at most $n\Delta/2$. So by Markov's inequality, the probability that $n/4$ edges are not cut is at most $1/2$. Deleting one endpoint of each uncut edge, we obtain a set of $3n/4$ vertices colored such that no two adjacent vertices have the same color (their edge is cut), i.e. a semicoloring. Repeating the process t times, we obtain a $O(\Delta^{\log_3 2})$-semicoloring with probability at least $1 - 1/2^t$. □

The result of theorem 2 can be strengthened via a refinement of the hyperplane partition. The new algorithm uses the vector k-coloring to extract from G a large independent set of vertices, namely of size $\Omega(n/(\Delta^{1-2/k}\sqrt{\ln \Delta}))$. We assign one color to this set and recurse on the rest of the vertices, so we use $O(\Delta^{1-2/k}\sqrt{\ln \Delta})$ colors to color half of the vertices. To find this independant set, we actually find an induced subgraph with a large number n' of vertices and a number $m' \ll n'$ of edges. Given such a graph, we can delete one endpoint of each edge to leave an independent set of $n' - m'$ vertices, i.e. of size roughly n'.

We use a randomized procedure to select an induced subgraph with n' vertices and m' edges such that $E[n' - m'] = \Omega(n/(\Delta^{1-2/k}\sqrt{\ln \Delta}))$. Repeating the procedure several times, we can get a subgraph such that $n' - m' = \Omega(n/(\Delta^{1-2/k}\sqrt{\ln \Delta}))$ with high probability.

The method to select the induced subgraph is a generalization of the random hyperplane technique. We choose a random n-dimensional vector r according to the same spherically symmetric n-dimensional normal distribution, and select the vertices i for which $v_i \cdot r \geq c$, for a threshold c to be optimized. That is, we look at the projection of the vectors $\{v_i\}_{i=1}^n$ on r, and as before, we hope that two adjacent vertices will be separated because their vectors are far apart. Vector projection rounding is thus very similar to the random hyperplane partition, and indeed if we fix $c = 0$ we recover the hyperplane partition method. Optimizing over c however allows us to improve on the previous result and to get our sparse induced subgraph. We have:

Lemma 3. *Let $a = \sqrt{\frac{2(k-1)}{k-2}}$. Then for any c,*

$$E[n' - m'] \geq n\left(\Phi(c) - \frac{\Delta\Phi(ac)}{2}\right).$$

Proof. We have $E[n'] = nP(v_i \cdot r \geq c) = n\Phi(c)$. Also looking at the probability that two adjacent vertices are selected, we have

$$P(v_1 \cdot r \geq c,\ v_2 \cdot r \geq c) \leq P((v_1 + v_2) \cdot r \geq 2c) = \Phi\left(\frac{2c}{\|v_1 + v_2\|}\right)$$

Now

$$\begin{aligned}\|v_1 + v_2\| &= \sqrt{v_1^2 + v_2^2 + 2v_1 \cdot v_2} \\ &\leq \sqrt{2 - 2/(k-1)} = 2/a\end{aligned}$$

It follows that $E[m'] \leq m\Phi(ac) \leq (n\Delta/2)\Phi(ac)$, and combining the two expressions we obtain the result. □

With some calculus we can show that setting $c = \sqrt{2\frac{k-2}{k}\ln\Delta}$ gives $n\Phi(c)/2$ as a lower bound of the right hand side of the expression in lemma 3 (if Δ is large enough; but otherwise a greedy coloring algorithm works). Some more calculus show that $n\Phi(c)/2 = \Omega(n/(\Delta^{1-2/k}\sqrt{\ln\Delta}))$ as needed. Therefore with the recursion argument given at the beginning of this discussion, we have a stronger version of theorem 2.

Theorem 4. *For every integer function $k = k(n)$, a vector k-colorable graph with maximum degree Δ can be semi-colored with at most $O(\Delta^{1-2/k}\sqrt{\ln\Delta})$ colors in probabilistic polynomial time.*

As with the simple random hyperplane algorithm, an algorithm for approximate coloring follows immediately from this result. The algorithm presented in the paper combines the previous arguments with a technique due to Wigderson [Wig83] to obtain the approximation ratio announced at the beginning of this paragraph.

It is natural to ask then what characteristic of the problem makes the choice of c different from 0 better. With respect to MAX CUT, here we have more information on the angle formed by the vectors obtained from the semidefinite relaxation. Changing the threshold c is a way to use this information. As we will see in the following, this idea of extracting more information from the angle between the vectors is recurrent in the successive improvements of hyperplane rounding.

2.3 Outward Rotations [Zwi99]

As it was briefly mentioned in paragraph 2.1, Goemans and Willimamson proved that their algorithm behaves even better on graphs with large maximum cuts, more precisely on graphs with cuts that contain 85% of the total weight of the graph. If not, the algorithm is only guaranteed to produce a cut whose size is α_{gw} times the size of the maximum cut in the graph. Zwick [Zwi99] generalized their algorithm, with a tool used previously by Nesterov [Nes98] called *outward rotations*. The method tries to improve on the original algorithm by increasing the small angles.

Let us first review the more general version of the analysis of Goemans and Williamson for MAX CUT. Let $W_{tot} = \sum_{i<j} w_{ij}$ be the total weight of the edges of the graph, and define $A = Z^*_{R,MC}/W_{tot}$ and $B = Z^*_{MC}/W_{tot}$ (recall that A and B are at least $1/2$). Clearly $E[W] \leq Z^*_{MC} \leq Z^*_{R,MC} \leq W_{tot}$. The result stated in paragraph 2.1 was $E[W] \geq \alpha_{gw} Z^*_{R,MC} \geq \alpha_{gw} Z^*_{MC}$.

Let $h(t) = \arccos(1-2t)/\pi$ and let $t_0 = \operatorname{argmin}_{t\in(0,1]} h(t)/t \approx 0.84459$. We have in fact $h(t_0)/t_0 = \alpha_{gw}$. Let

$$\alpha(A) = \begin{cases} h(A)/A & \text{if } A \geq t_0. \\ h(t_0)/t_0 = \alpha_{gw} & \text{if } A \leq t_0. \end{cases}$$

Theorem 5. *The Goemans-Williamson algorithm produces a cut with expected weight*

$$E[W] \geq \alpha(A) Z^*_{R,MC} \geq \alpha(A) Z^*_{MC} \geq \alpha(B) Z^*_{MC}.$$

Proof. The only non trivial inequality is the first one, and only the case $A \geq t_0$ (large maximum cut) has yet to be proved. Letting $\lambda_e = w_{ij}/W_{tot}$ and $x_e = \frac{1-v_i\cdot v_j}{2}$ for $e = i, j$, we can write $A = \sum_e \lambda_e x_e$. From lemma 1, the expected weight of the cut produced by the GW algorithm is:

$$E[W] = W_{tot} \sum_e \lambda_e \frac{\arccos(1-2x_e)}{\pi} = W_{tot} \sum_e \lambda_e h(x_e)$$

Define

$$\tilde{h}(t) = \begin{cases} \alpha_{gw} t, & \forall t \in [0, t_0]. \\ h(t), & \forall t \in [t_0, 1]. \end{cases}$$

We can readily show that $\tilde{h}$ is convex and that $\tilde{h}(t) \leq h(t)\ \forall t \in [0,1]$. Since $\sum_e \lambda_e = 1$ we have by convexity of $\tilde{h}$ and for $A \geq t_0$

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

This shows

$$E[W] \geq W_{tot} h(A) = h(A) Z^*_{R,MC}/A.$$

□

As A varies from t_0 to 1, $h(A)/A$ varies from α_{gw} to 1 so the performance guarantee of the GW algorithm improves. The result is strictly stronger than the previous bound for $A > t_0$. This means that as the relative size of the maximum cut in the graph gets larger, it is easier to approximate it. At the limit when $A = B = 1$, the graph is bipartite and a maximum cut that cuts all the edges can be found in linear time. Alon, Sudakov and Zwick [ASZ01] showed that this analysis of the algorithm is tight, i.e. they constructed graphs for which the algorithm returns cuts of size $E[W] = \alpha(A)Z^*_{R,MC} = \alpha(B)Z^*_{MC}$, for $1/2 \leq A \leq 1$.

Note that for $1/2 \leq A \leq t_0$, the performance of the GW algorithm stays unchanged. On the other hand, the expected weight of a random cut is $W_{tot}/2 = Z^*_{R,MC}/(2A)$, and so achieves a performance ratio of $1/(2A)$, which is close to 1 when A is close to $1/2$ and therefore better than the GW algorithm. Zwick's algorithm gives an approximation ratio which is always at least as good as the better ratio of these two algorithms. As mentioned before, the algorithm performs outward rotation of the vectors obtained from the semidefinite relaxation before rounding them, as described in the following.

We assumed that the solution $\{v_i\}_{i=1}^n$ of the program lies in $\mathbb{R}^n$. We can embed these vectors in $\mathbb{R}^{2n}$ by adding n 0 coordinates to each of them. Then we choose n vectors $u_1, \ldots, u_n$ orthonormal in $\mathbb{R}^{2n}$ such that these vectors are also orthogonal to the vectors $v_1, \ldots, v_n$. For instance, take u_i with 1 for coordinate $(n+i)$ and 0 for the other coordinates. Let $0 \leq \gamma \leq (\pi/2)$ be an angle. We can rotate each vector v_i by an angle γ toward u_i in the plane spanned by u_i and v_i, to obtain the vectors $v'_1, \ldots, v'_n$. These vectors are then rounded using a random hyperplane.

The idea is as follows. Using a random hyperplane to round the vectors $\{u_i\}_{i=1}^n$ produces a uniform random assignment, because $u_i \cdot u_j = 0$ in lemma 1. So by letting γ vary from 0 to $\pi/2$, we get various *combinations* of the GW random hyperplane rounding (obtained exactly for $\gamma = 0$) and the uniform random assignment (obtained exactly for $\gamma = \pi/2$). Hence the performance of the algorithm is expected to be a combination of the performance of the two algorithm, and at least as good as any of the two if we search for the best γ. We have $v'_i = (\cos\gamma)v_i + (\sin\gamma)u_i$ and therefore $v'_i \cdot v'_j = (cos^2\gamma)(v_i \cdot v_j)$ for $i \neq j$. So by letting $\theta_{ij} = \arccos(v_i \cdot v_j)$ and $\theta'_{ij} = \arccos(v'_i \cdot v'_j)$ we immediately get

$$\cos\theta'_{ij} = \cos^2\gamma \ \cos\theta_{ij}, \quad \text{for } i \neq j. \tag{5}$$

In terms of the Y matrix, if we also define $Y' = [y'_{ij}] = [v'_i \cdot v'_j]$, we have

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

From (5) and if $0 < \cos^2\gamma < 1$, we get that if $\theta_{ij} > \pi/2$, then $\theta_{ij} > \theta'_{ij} > \pi/2$, and if $\theta_{ij} < \pi/2$, then $\theta_{ij} < \theta'_{ij} < \pi/2$. Therefore outward rotations decrease angles larger than $\pi/2$ and increase angles smaller than $\pi/2$. Since the performance of the GW algorithm depends on how large the angles are, we expect an improvement on this algorithm only in the case where worst performance guarantees are for configurations in which some of the angles are below $\pi/2$. Zwick show that this happens for $A \leq t_0$. More precisely, for any value of A, his algorithm computes an angle $\gamma(A)$, perform outward rotations of all the vectors $\{v_i\}_{i=1}^n$ by this angle and then rounds the solution using a random hyperplane. If $A < t_0$ then $\gamma(A) > 0$ and we can improve strictly on the GW algorithm. If $A \geq t_0$ then $\gamma(A) = 0$, i.e. we recover the GW algorithm. Zwick provides a formula for determining $\gamma(A)$, however the formula depends on the size of the maximum cut. So in practice we would have to search for a good value of γ numerically. Note that a more general scheme would be to rotate different vectors by different angles.

2.4 Random Projection + Randomized Rounding (RPR^2) [FL01]

Outward rotation is an example of a possible generalization of the hyperplane rounding technique. Further generalization was introduced by Feige and Langberg [FL01] with the RPR^2 (Random Projection followed by Randomized Rounding). This technique is again applied to the light MAX CUT problem. However, in problems where the semidefinite program contains more information, such as in MAX DICUT or MAX 2-SAT, generalizing the rounding procedure also leads to a significant improvement of the approximation ratio of the original GW algorithm in the general case, and not only for special cases such as when the maximum cut is relatively small. We will explore this issue further in paragraph 2.5.

RPR^2 contains the random hyperplane rounding technique and its variant involving outward rotation as special cases. The goal, which started to appear already with outward rotations, is to perform a systematic search for the best approximation, possibly within a restricted family of rounding procedures. The random hyperplane rounding is the basic idea. Vector projection was a "baby version" of searching for an improvement over random hyperplane. With techniques such that RPR^2 or as already in [FG95], the search for improvement relies heavily on numerical computations. Note also that these improvements in

general are concerned with problems close to MAX CUT, for which the semidefinite program is relatively simple and the analysis is not too complicated. Extending these ideas to a problem like approximate graph coloring would probably imply overcoming lots of technical difficulties.

In the spirit of the vector projection formulation, we can describe the more general randomized hyperplane rounding as follows:

- Project the vector solution on a random line through the origin, obtaining a fractional solution $x_1, \ldots, x_n$.
- Round the fractional solution $x_1, \ldots, x_n$ to a 0/1 solution using threshold rounding (i.e. x_i is rounded to 1 if x_i is greater or equal to the specified threshold, to 0 otherwise). The threshold chosen by Goemans and Williamson is 0. The threshold chosen by Karger et al. is the parameter c.

The RPR^2 technique is simply based on the remark that the threshold rounding in the second step can be replaced by a randomized rounding, as in the usual approach for linear programming relaxation. More precisely, the RPR^2 family is a family of rounding procedures parametrized by a function $f : \mathbb{R} \to [0, 1]$. An RPR^2 procedure using f has two steps defined as:

Step 1 (Projection): Project the vectors $v_1, \ldots, v_n$ onto a random line, as in the vector projection method. That is, pick a random vector r drawn from the n-dimensional standard normal distribution and let $x_i = v_i \cdot r$ be the directed distance of the projected vector from the origin.

Step 2 (Randomized rounding): Define the 0/1 solution $a_1, \ldots, a_n$: for each i, set a_i to be 1 independently with probability $f(x_i) = f(v_i \cdot r)$.

Clearly, the standard random hyperplane rounding is a member of the RPR^2 family, with f being the step function which is 1 on $\mathbb{R}_+$ and 0 otherwise. Maybe more surprisingly, the version including outward rotation is also a member of the family RPR^2.

Theorem 6. *For any $\gamma \in [0, \pi/2]$, there is a function f_γ such that using RPR^2 with f_γ is equivalent to γ-outward rotation followed by random hyperplane rouding.*

Proof. We use the notation introduced in paragraph 2.3, and define the coordinate system so that the vector u_i has a 1 for coordinate $(n+i)$ and 0 for the other coordinates. Let $r = (r_1, \ldots, r_n, \ldots, r_{2n})$ be the random vector chosen after the outward rotation, from the 2n-dimensional standard normal distribution. Since the components of r are drawn independantly from the 1-dimensional standard normal distribution, we can write

$$v_i' \cdot r = (\cos\gamma)\ v_i \cdot (r_1, \ldots, r_n) + (\sin\gamma)\ r_{n+i} = (\cos\gamma)\ x_i + (\sin\gamma)\ r_{n+i}$$

(where we consider v_i in the original n-dimensional space). Then in the following random hyperplane rounding step, we set y_i to 1 iff $(v_i' \cdot r) \geq 0$, i.e. iff $r_{n+i} \geq -x_i \cot\gamma$ (for $\gamma \neq 0$; recall that the case $\gamma = 0$ corresponds just to the GW algorithm).

Since r_{n+i} is a one-dimensional standard normal random variable, and r_{n+i} is independent of r_{n+j} for $i \neq j$, this gives an equivalent way to describe the algorithm: first determine x_i via projection of v_i on a random line in $\mathbb{R}^n$, and then set each a_i independantly to 1 with probability $f_\gamma(x_i) = \Phi(-x_i \cot\gamma)$. Therefore this is a special case of RPR^2. □

So RPR^2 offers a generalization of outward rotation. By considering a larger set of functions, we hope to be able to improve on the results obtained via the previous method. On the other hand, the interesting characteristic of the functions corresponding to outward rotations is that the analysis is quite easy to perform, and by considering more general functions this might not always be the case. Let us describe the general analysis of RPR^2 for MAX CUT. As usual, it reduces to computing the probability that an edge is cut, i.e. $P(a_i \neq a_j)$, and use the linearity of expectation to obtain the expected size of the cut. Here is a trick to simplify the calculation, which was actually already used in the proof of lemma 1:

Remark 7. *Let $\theta_{ij} = \arccos(v_i \cdot v_j)$ be the angle between v_i and v_j (recall that these are unit vectors). Since the distribution of r is spherically symmetric, we can assume $v_i = (1, 0, \ldots, 0)$ and $v_j = (\cos\theta, \sin\theta, 0, \ldots, 0)$ in the computation of $P(a_i \neq a_j)$, in particular $v_i \cdot r = r_1$ and $v_j \cdot r = r_1 \cos\theta + r_2 \cos\theta = z(\theta, r_1, r_2)$.*

From this, the following lemma follows directly for MAX CUT, and just says that $(y_i \neq y_j)$ happens exactly when $a_i(1 - a_j) = 1$ or $a_j(1 - a_i) = 1$.

Lemma 8. *Let $\theta \in [0, \pi]$. Given two vectors v_i and v_j that form an angle θ, the probability that the corresponding values a_i and a_j differ after using RPR^2 with a function f is:*

$$P_f(\theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} [f(r_1)(1 - f(z(\theta, r_1, r_2))) + f(z(\theta, r_1, r_2))(1 - f(r_1))]\phi(r_1)\phi(r_2)dr_1dr_2 \qquad (6)$$

Now note that the expected weight of the cut produced by RPR^2 is $E[W] = \sum_{\{i,j\}\in E} P_f(\theta_{ij})$. Using this general analysis scheme, Feige and Langberg showed that outward rotation is not optimal in the family RPR^2 for the light MAX CUT case, that is, they could improve the performance curve of outward rotation when $A < t_0$. Their analysis involves the numerical computation of (6) as well as deriving some necessary conditions that can ensure that f attains a certain level of optimality. They used piecewise linear functions, for which they claim no optimality result but state that numerical results give performance close to optimal.

2.5 Semidefinite Relaxations with More Information [FG95, LLZ02]

In some cases, such as for MAX 2-SAT and MAX DICUT, the semidefinite relaxation includes a vector v_0 that plays a special role. Then it is possible to rotate the vectors $v_1, \ldots, v_n$ toward or away from v_0 before rounding them using a random hyperplane. Feige and Goemans [FG95] showed that this idea could improve the approximation ratio compared to the standard hyperplane rounding. In this article, they also suggested several ideas for modifying the random hyperplane rounding, which led to the developments presented in the two previous paragraph. Feige and Goemans suggested the following ideas (among several others) for problems like MAX 2SAT:

- first, obtain a stronger semidefinite relaxation, adding valid inequalities to the formulation of [GW96].
- in the rounding step, the structure of the problems suggest to first *rotate* the vectors obtained from the semidefinite relaxation.
- instead of using a uniform distribution for r, use the structure of the problem to chose r according to a *skewed* distribution.

Following these suggestions, [LLZ02] obtained a 0.94 approximation ratio for MAX 2SAT and 0.874 for MAX DICUT, compared to 0.878 and 0.796 respectively in the original work of Goemans and Williamson. Note that MAX CUT can be seen as a subproblem of MAX DICUT, and the ratio for MAX DICUT is quite close to the best ratio of 0.878 obtained for MAX CUT, which is believed to be tight. Note also that choosing r from a non-uniform distribution is related to the idea of the vector projection method: indeed, the distribution chosen is skewed in a certain direction, which looks similar to rounding the vectors depending on the size of their projection on r. However, as we have already noted, by using a more general formulation, a systematic search for the best solution becomes possible. This is essentially what the paper of Lewin, Livnat and Zwick does, in the spirit of the RPR^2 approach.

In the following I illustrate the ideas mentioned above for the MAX 2SAT problem, following [LLZ02]. Recall that an instance of MAX 2SAT in the Boolean variables $x_1, \ldots, x_n$ is composed of a collection of clauses $C_1, \ldots, C_m$ with non-negative weights $w_1, \ldots, w_m$ assigned to them. Each clause C_i is of the form z_i or $z_i \vee z_j$, where each z_j is either a variable x_j or its negation $\overline{x_j}$ (z_j is a literal). The goal is to assign the variables $x_1, \ldots, x_n$ Boolean values 0 or 1 so that the total weight of the satisfied clauses is maximized. We let $x_{n+i} = \overline{x_i}$, for $1 \le i \le n$, and also $x_0 = 0$. Each clause is then of the form $x_i \vee x_j$ where $0 \le i, j \le 2n$, and we associate the weight w_{ij} to the clause $x_i \vee x_j$. As usual, we associate a vector v_i to the variable x_i, and the scalar product becomes a dot product in the relaxation. Since $x_0 = 0$, the vector v_0 corresponding to the constant 0 (`false`). The semidefinite relaxation used by Feige and Goemans was:

$$\begin{aligned} Z^*_{R,M2SAT} = \ & \text{Max} \frac{1}{4} \sum_{i,j} w_{ij}(3 - v_0 \cdot v_i - v_0 \cdot v_j - v_i \cdot v_j) \\ & \text{subject to: } v_0 \cdot v_i + v_0 \cdot v_j + v_i \cdot v_j \ge -1, \quad 1 \le i, j \le 2n \\ & \qquad v_i \cdot v_{n+i} = -1, \quad 1 \le i, j \le n \\ & \qquad v_i \in \mathbb{R}^{n+1}, \ \ v_i \cdot v_i = 1, \quad 0 \le i \le 2n \end{aligned} \qquad (7)$$

In the algorithm of Goemans and Williamson, x_i is set to `false` if v_i and v_0 lie on the same side of the random hyperplane. The constraint $v_i \cdot v_{n+i} = -1$ is to ensure consistency. Also, since the formulation depends only on the inner product between the vectors, we can assum $v_0 = (1, 0, \ldots, 0)$. Note that it

is sufficient to consider a vector space of dimension $(n+1)$ instead of $(2n+1)$ to be able to formulate (7) as a semidefinite program since v_i and v_{n+i} are linearly dependant (we can even use a vector space of dimension $\sqrt{2n+1}$ by using a result of Barvinok [Bar95] and Pataki [Pat94]). In the MAX DICUT problem, a similar formulation is possible, with the vector v_0 representing the S side of the cut, i.e. the set corresponding to the tails of the arcs.

[LLZ02] give a formal generalization of the ideas suggested in [FG95] and search (numerically) for the best choice within a family of rounding procedures. They define several families of procedures. The most general is essentially RPR^2, that they call $\mathcal{GEN}^+$. They used combinations of the restricted families $\mathcal{ROT}$ (used by Feige and Goemans) and $\mathcal{SKEW}$, as described below:

$\mathcal{GEN}^+$: let $F : \mathbb{R}^2 \times [-1,1] \to [0,1]$ be an arbitrary function. Let $r = (r_0, \ldots, r_n)$ be a $(n+1)$-dimensional standard normal random variable. For $1 \leq i \leq n$, set x_i to 1 independently with probability $F(v_0 \cdot r, v_i \cdot r, v_0 \cdot v_i)$.

$\mathcal{GEN}$: let $F : \mathbb{R}^2 \times [-1,1] \to \{0,1\}$ be an arbitrary function. Let $r = (r_0, \ldots, r_n)$ be a $(n+1)$-dimensional standard normal random variable. For $1 \leq i \leq n$, let $x_i = F(v_0 \cdot r, v_i \cdot r, v_0 \cdot v_i)$.

$\mathcal{ROT}$: let $f : [0,\pi] \to [0,\pi]$ be an arbitrary rotation function. Let $\theta_i = \arccos(v_0 \cdot v_i)$. Rotate v_i into a vector v_i' that forms an angle $\theta_i' = f(\theta_i)$ with v_0. Round the vectors $v_0, v_1', \ldots, v_n'$ using the random hyperplane technique, i.e. let $x_i = 1$ iff $(v_0 \cdot r)(v_i' \cdot r) \leq 0$.

$\mathcal{SKEW}$: Let $r = (r_0, \ldots, r_n)$ be a skewed normal vector, i.e. $r_1, \ldots, r_n$ are still independent standard normal random variables, but r_0 is chosen according to a different distribution. Again, let $x_i = 1$ iff $(v_0 \cdot r)(v_i' \cdot r) \leq 0$.

It can be shown that $\mathcal{ROT}$ and $\mathcal{SKEW}$ are indeed subfamilies of $\mathcal{GEN}$. I will not present the details of the paper here, since the challenge is in being able to analyze these general families of rounding procedures, but no really new idea is introduced. In particular, as usual the analysis is local, i.e. for MAX 2SAT we only have to compute the probability that a certain clause is satisfied. Let $prob(v_i, v_j)$ be the probability that x_i and x_j both receive the value 1 when the vectors v_i and v_j are rounded using the chosen rounding procedure. Then the probability that the 2-SAT clause $x_i \vee x_j$ is satisfied is $P(v_i, v_j) = 1 - prob(-v_i, -v_j)$. The performance ratio of the MAX 2-SAT algorithm is then at least (this was shown already for the GW algorithm)

$$\alpha = \min_{(v_i,v_j)\in\Omega} \frac{P(v_i, v_j)}{\frac{1}{4}(3 - v_0 \cdot v_i - v_0 \cdot v_j - v_i \cdot v_j)} \ ,$$

where Ω is the set of pairs of vectors satisfying the constraints of the semidefinite program. The evaluation of $P(v_i, v_j)$ boils down to the computation of an integral similar to (6), and the computation of the minimums for various families of rounding procedures is performed numerically.

3 Algorithms Based on Results from Functional Analysis [AN04]

In the previous part, we have explained how the random hyperplane rounding technique formed the basis for a number of more general rounding procedures for semidefinite relaxations. A common feature of the numerous variants of the MAX CUT algorithm is that they look for a lower bound on the probability that each individual edge is cut and obtain a global lower bound via linearity of expectation. Clearly this is possible only when the edge weights are non-negative. In a recent paper, Alon and Naor [AN04] introduced new techniques and analysis methods, based on proofs of Grothendieck's inequality, that can handle arbitrary edge weights for graph problems.

3.1 Approximating the Cut-Norm of a Matrix

Alon and Naor considered the problem of approximating the cut-norm $\|A\|_C$ of a real matrix $A = [a_{ij}]_{i\in R, j\in S}$, i.e. the maximum, over all $I \subset R, J \subset S$ of the quantity $|\sum_{i\in I, j\in J} a_{ij}|$. Let us call CUT NORM the problem of computing the cut-norm of a given real matrix. Alon and Naor show that this problem is MAX SNP hard, hence look for an efficient approximation algorithm. They describe three different algorithms, which output two subsets $I \subset R, J \subset S$ such that $|\sum_{i\in I, j\in J} a_{ij}| \geq \rho\|A\|_C$. The best ratio ρ obtained is 0.56 with a randomized algorithm. In this part we discuss their semidefinite relaxation of the problem.

Assume that A is an n by m matrix. Define another norm that we will relate to the cut-norm, whose value is given by the quadratic program

$$\begin{aligned} \|A\|_{\infty\to 1} = &\text{Max} \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}x_iy_j \\ &\text{subject to } x_i, y_j \in \{-1, 1\} \end{aligned} \tag{8}$$

The obvious semidefinite relaxation is

$$\begin{aligned} Z^*_{R,CN} = &\text{Max} \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}u_i \cdot v_j \\ &\text{subject to } \|u_i\| = \|v_j\| = 1, \quad 1 \leq i \leq n,\ 1 \leq j \leq m \end{aligned} \tag{9}$$

In (9), we allow the vectors to lie in an arbitrary Hilbert space (i.e. we do not restrict the dimension), as this will be convenient in the following. But clearly, to solve the semidefinite program we can assume without loss of generality that they lie in $\mathbb{R}^{n+m}$. The fact that some entries a_{ij} may be positive and some entries may be negative makes the techniques encountered so far fail: even if we can approximate well enough each term $a_{ij}u_i \cdot v_j$ by its integral rounding $a_{ij}x_iy_j$, the approximation of the sum might be bad due to cancellations. Before we look at this problem further, let us first see why this quadratic program is interesting at all for our problem of estimating the cut norm.

Lemma 9. *For any real matrix $A = [a_{ij}]$,*

$$\|A\|_C \leq \|A\|_{\infty\to 1} \leq 4\|A\|_C \tag{10}$$

Moreover, if the sum of each row and the sum of each column of A is zero, then $\|A\|_{\infty\to 1} = 4\|A\|_C$.

Proof. Consider a particular assignment of values $x_i, y_j \in \{-1, 1\}$,

$$\sum_{ij} a_{ij}x_iy_j = \sum_{i:x_i=1, j:x_j=1} a_{ij} - \sum_{i:x_i=1, j:x_j=-11} a_{ij} - \sum_{i:x_i=-1, j:x_j=1} a_{ij} + \sum_{i:x_i=-1, j:x_j=-1} a_{ij}$$

Since each term is bounded in absolute value by $\|A\|_C$, the triangle inequality implies

$$\|A\|_{\infty\to 1} \leq 4\|A\|_C. \tag{11}$$

Now suppose that $\|A\|_C = \sum_{i\in I, j\in J} a_{ij}$ (the case when it is $-\sum_{i\in I, j\in J}$ is handled similarly). Define $x_i = 1$ for $i \in I$ and $x_i = -1$ otherwise, $y_j = 1$ if $j \in J$ and $y_j = -1$ otherwise. Then

$$\|A\|_C = \sum_{i,j} a_{ij}\frac{1+x_i}{2}\frac{1+y_j}{2} = \frac{1}{4}\left(\sum_{i,j} a_{ij} + \sum_{i,j} a_{ij}x_i + \sum_{i,j} a_{ij}y_j + \sum_{i,j} a_{ij}x_iy_j\right)$$

Each sum on the right hand side is bounded by $\|A\|_{\infty\to 1}$ so $\|A\|_C \leq \|A\|_{\infty\to 1}$ follows. If the sum of each row and the sum of each column is zero, then the right hand side is exactly $\frac{1}{4}\sum_{i,j} a_{ij}x_i y_j$, which implies $4\|A\|_C \leq \|A\|_{\infty\to 1}$ and therefore we have equality from (11). □

So we have shown that the cut norm is related to the quadratic program (8). Moreover, using (10), it is possible to show that a ρ-approximation algorithm for approximating $\|A\|_{\infty\to 1}$ provides an approximation algorithm for approximating the cut-norm as well, we the same approximation ratio guaranteed. In the following, we present two randomized algorithms for approximating $\|A\|_{\infty\to 1}$.

3.2 Averaging with a Gaussian Measure

Although we have seen that a term by term analysis for the rounding of the solution of (9) would not work, there is an inequality by Grothendieck [Gro53] which, recast in the language of convex optimization, asserts that the semidefinite program (9) and that of the integer program (8) can differ only by a constant factor. This constant is the Grothendieck's constant K_G, which is unknown but for which we know

$$\pi/2 \leq K_G \leq \frac{\pi}{2\ln(1+\sqrt{2})}$$

So the integrality gap is at most K_G (i.e. $Z^*_{R,CN} \leq K_G\|A\|_{\infty\to 1}$) and therefore it is worth trying to find an integral solution whose value is comparable to the optimal value of the semidefinite relaxation. The first algorithm we describe is based on a proof of Grothendieck's inequality by Rietz [Rie74]. Quite remarkably, the final algorithm boils down to using the technique that we would have naturally used after studying that of Goemans and Williamson, even if their analysis does not carry through. Namely, we will choose a $(n+m)$-dimensional Gaussian vector r, i.e. with standard, independent, normal random variables $r_1, \ldots, r_{n+m}$ as components, and take

$$x_i = \text{sign}(u_i \cdot r), y_j = \text{sign}(v_j \cdot r). \tag{12}$$

Also, in its proof Rietz improves on that method with ideas that resemble some of the ideas seen in part 2 ! We will need the following lemma, whose proof requires calculations that we have already seen. To simplify the notation let $p = n + m$.

Lemma 10. *Let $r = (r_1, \ldots, r_p)$ be a p-dimensional standard normal random vector, and let b, c be unit vectors $\in \mathbb{R}^p$. We have the following identity:*

$$\frac{\pi}{2}E[sign(b\cdot r)sign(c\cdot r)] = b\cdot c + E\left\{\left[b\cdot r - \sqrt{\frac{\pi}{2}}sign(b\cdot r)\right]\left[c\cdot r - \sqrt{\frac{\pi}{2}}sign(c\cdot r)\right]\right\} \tag{13}$$

Proof. The proof is based on expanding the right hand side and verifying that the result holds. We will need

$$E[(b\cdot r)(c\cdot r)] = b\cdot c \tag{14}$$

This follows from $E[r_i r_j] = \delta_{ij}$. Indeed if $b = (b_1, \ldots, b_n)$ and $c = (c_1, \ldots, c_n)$, then

$$E[(b\cdot r)(c\cdot r)] = E\left[\sum_{i=1}^p b_i r_i \sum_{i=1}^p c_i r_i\right] = \sum_{i,j} b_i c_j E[r_i r_j] = \sum_{i=1}^p b_i c_i = b\cdot c$$

To compute $E[(b\cdot r)\text{sign}(c\cdot r)]$ we use the rotation invariance idea stated in remark 7 to write $c = (1, 0\ldots, 0)$ and $b = (b_1, b_2, 0, \ldots, 0)$. Then

$$\begin{aligned}
E[(b\cdot r)\text{sign}(c\cdot r)] &= E[(b_1 r_1 + b_2 r_2)sign(r_1)] \\
&= E[b_1 r_1 \text{sign}(r_1)] + E[b_2 r_2]E[\text{sign}(r_1)] \quad \text{(by independence)} \\
&= E[b_1 r_1 \text{sign}(r_1)] = 2b_1 \int_0^\infty \frac{1}{\sqrt{2\pi}} x e^{-x^2/2} dx = \sqrt{\frac{\pi}{2}} b_1 \\
&= \sqrt{\frac{\pi}{2}}(b\cdot c)
\end{aligned} \tag{15}$$

With (14) and (15), the result follows readily. □

Lemma 10 provides a nice way to analyse globally the quality of the rounding (12).

Theorem 11. *Solving the semidefinite program (9) followed by the rounding scheme (12) provides an approximation ratio of* $(\frac{4}{\pi} - 1)$ *for* $\|A\|_{\infty \to 1}$.

Proof. Let $u_i, v_j \in \mathbb{R}^p$ be the vectors solution of (9). Summing (13) over these vectors

$$\begin{aligned} &\frac{\pi}{2} E\left[\sum_{i,j} \text{sign}(b \cdot r)\text{sign}(c \cdot r)\right] \\ &= Z^*_{R,CN} + \sum_{i,j} a_{ij} E\left\{\left[u_i \cdot r - \sqrt{\frac{\pi}{2}}\text{sign}(u_i \cdot r)\right]\left[v_j \cdot r - \sqrt{\frac{\pi}{2}}\text{sign}(v_j \cdot r)\right]\right\} \end{aligned} \tag{16}$$

Now for any unit vector b, let $z(b,r) = b \cdot r - \sqrt{\frac{\pi}{2}}\text{sign}(b \cdot r)$: these quantities are random variables. For any two (real valued) random variables X and Y, let us write $< X, Y >= E[XY]$: this defines an inner product on the space of random variables. From (13) again, by taking $b = c = u_i$, we obtain $< z(u_i, r), z(u_i, r) >=< z(v_j, r), z(v_j, r) >= \frac{\pi}{2} - 1$. Now since we have defined an inner product in a Hilbert space (the space of random variables, which is infinite dimensional) and the relaxation (9) was defined in this full generality, we have for any two random variables X, Y with norm $\sqrt{\pi/2 - 1}$:

$$\sum_{i,j} a_{ij} < X, Y) > \ \leq \ \left(\frac{\pi}{2} - 1\right) Z^*_{R,CN}$$

In particular for $-z(u_i, r), z(v_j, r)$

$$\sum_{i,j} a_{ij} < -z(u_i, r), z(v_j, r) > \ \leq \ \left(\frac{\pi}{2} - 1\right) Z^*_{R,CN}$$

which provides the following bound

$$\sum_{i,j} a_{ij} E[z(u_i, r) z(v_j, r)] \geq \left(1 - \frac{\pi}{2}\right) Z^*_{R,CN}$$

From this inequality and (16), we obtain the bound

$$\frac{\pi}{2} E\left[\sum_{i,j} \text{sign}(b \cdot r)\text{sign}(c \cdot r)\right] \geq \left(2 - \frac{\pi}{2}\right) Z^*_{R,CN}$$

which gives the result stated in the theorem. □

In this proof, it is interesting to note that the definition of the relaxation on a general Hilbert space looks quite powerful and provides us with an elegant argument. At the same time, it does not restrict our ability to solve the program in practice since the solution will only involve p vectors, which therefore lie in a space of dimension at most p.

3.3 Constructing New Vectors

In this section, we show how to improve the approximation ratio from $(\frac{4}{\pi} - 1) \approx 0.27$ to $\frac{2\ln(1+\sqrt{2})}{\pi} \approx 0.56$. The algorithm is based on another proof of Grothendieck's inequality by Krivine [Kri77]. We first need a technical lemma in the spirit of lemma 1.

Lemma 12. *For every two unit vectors* u, v *in a Hilbert space, if* r *is chosen randomly and uniformly on the unit sphere of the space, then*

$$E[\textit{sign}(u \cdot r)\ \textit{sign}(v \cdot r)] = \frac{2}{\pi} \arcsin(u \cdot v). \tag{17}$$

Proof. As usual by rotation invariance (see remark 7) we can look at the situation in the plane spanned by u and v. We refer to the notation on Fig. 1. We have

$$\begin{aligned} E[\text{sign}(u \cdot r)\ \text{sign}(v \cdot r)] &= P(\text{sign}(u \cdot r) = \text{sign}(v \cdot r)) - P(\text{sign}(u \cdot r) \neq \text{sign}(v \cdot r)) \\ &= 1 - 2P(\text{sign}(u \cdot r) \neq \text{sign}(v \cdot r)) \end{aligned}$$

The event $\{\text{sign}(u \cdot r) \neq \text{sign}(v \cdot r)\}$, as we have already seen in lemma 1, happens exactly when the projection of r falls between the dashed line and the vertical axis, which has a probability $\theta/\pi = 1/2 - \psi/\pi$. But $\psi = \arcsin(u \cdot v)$, so we have (17). □

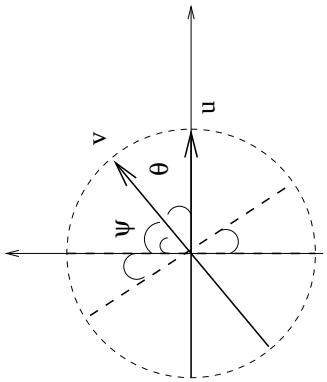

Figure 1: Figure for the proof of identity (17)

Using this identity, the main part of the proof is contained in the following lemma (I skipped some details, which can be found in [Jam87]):

Lemma 13. *For any sets of unit vectors* $\{u_i\}_{i=1}^n$ *and* $\{v_j\}_{j=1}^m$ *in a Hilbert space* H*, and for* $c = \sinh^{-1}(1) = \ln(1+\sqrt{2})$*, there exist sets of unit vectors* $\{u_i'\}_{i=1}^n$ *and* $\{v_j'\}_{j=1}^m$ *in a Hilbert space* H'*, such that if* r *is chosen randomly and uniformly in the unit sphere of* H' *then*

$$E[sign(u_i' \cdot r)\ sign(v_j' \cdot r)] = \frac{2}{\pi} c\,(u_i \cdot v_j) \quad \forall\, 1 \le i \le n,\ 1 \le j \le m.$$

Proof. From identity (17), all that we have to prove is that we can find vectors u_i', v_j' in a Hilbert space such that

$$u_i' \cdot v_j' = \sin(c\, u_i \cdot v_j) \tag{18}$$

Note that we use the same notation for inner products in different Hilbert spaces, as well as for the norm of a vector which will be written $\|.\|$ without specifying the underlying vector space. For k integer, let l_2^k denote $\mathbb{R}^k$ with the euclidian norm, and l_2 denote the space of all infinite square summable sequences. We will construct the vectors $u_i', v_j' \in l_2$ and describe the mappings $T : l_2^n \to l_2$ and $S : l_2^m \to l_2$ such that $T(u_i) = u_i'$ and $S(v_j) = v_j'$.

Writing the Taylor expansion of sin:

$$\sin(c\, u \cdot v) = \sum_{k=0}^{\infty} (-1)^{k+1} c_k\ (u \cdot v)^{2k+1}$$

where $c_k = c^{2k+1}/(2k+1)!$. Assume $n = m$ (we can always consider the vectors to be in the highest dimensional space). We can show that for every positive integer k, there is a mapping $w_k : l_2^n \to l_2^N$ (where $N = 2^{nk}$) such that for all u, v

$$(u \cdot v)^k = w_k(u) \cdot w_k(v)$$

Therefore we can write

$$\sin(c\, u \cdot v) = \sum_{k=1}^{\infty} (-1)^{k-1} c_k\ (w_{2k+1}(u) \cdot w_{2k+1}(v))$$

Define the mappings T, S by

$$T(u) = \sum_k (-1)^k \sqrt{c_k} [w_{2k+1}(u)] I_k \quad \text{and} \quad S(u) = \sum_k \sqrt{c_k}, [w_{2k+1}(u)] I_k \tag{19}$$

where I_k denotes the sequence in l_2 with a 1 as its k^{th} coordinate and 0 otherwise. For these operators we have

$$T(u) \cdot S(v) = \sin(c\, u \cdot v)$$

Also, (19) implies $\|T(u)\|^2 = \sinh(c\|u\|^2)$ and $\|S(v)\|^2 = \sinh(c\|v\|^2)$ (by reversing the argument above and recognizing the Taylor's expansion of sinh). So with our choice of $c = \sinh^{-1}(1)$, if u and v are unit vectors, so are $u' = T(u)$ and $v' = T(v)$. □

We can now provide the description of the algorithm. First, we solve the semidefinite program (9), which returns unit vectors $\{u_i\}_{i=1}^n$ and $\{v_j\}_{j=1}^m$. Now we need to find the vectors u_i' and v_j'. But lemma 13 asserts that the following semidefinite program has a feasible solution (the decision parameters are the vectors u_i' and v_j'):

$$\begin{aligned} &\text{maximize } 0 \\ &\text{subject to } u_i' \cdot v_j' = \sin(c\, u_i \cdot v_j), \quad 1 \le i \le n,\ 1 \le j \le m \\ &\qquad\qquad \|u_i'\| = \|v_j'\| = 1, \quad 1 \le i \le n,\ 1 \le j \le m \end{aligned} \tag{20}$$

This program involves $p = n + m$ vectors so can be formulated and solved as a semidefinite program in $\mathbb{R}^p$ (even if the lemma gave a description in an infinite dimensional space). Then pick r at random from the p-dimensional standard normal distribution, and let

$$x_i = \text{sign}(u_i' \cdot r) \quad \text{and} \quad y_j = \text{sign}(v_j' \cdot r)$$

From the lemma and by linearity of expectation we have

$$E[\sum_{i,j} a_{ij} x_i y_j] = \frac{2\ln(1+\sqrt{2})}{\pi} \sum_{i,j} a_{ij} u_i \cdot v_j \ge \frac{2\ln(1+\sqrt{2})}{\pi} \|A\|_{\infty\to 1}.$$

We restate this as a theorem

Theorem 14. *There is a randomized ρ-approximation algorithm for computing $\|A\|_{\infty\to 1}$, where $\rho = \frac{2\ln(1+\sqrt{2})}{\pi} > 0.56$.*

Conclusion

Obviously this report does not pretend to offer a complete view of the use of semidefinite programming relaxations for combinatorial optimization problems. One important missing element is to discuss in the various cases encountered how far we are from the best solution achievable. For example, Håstad [Hås97] has shown that it is NP-hard to approximate MAX CUT within $16/17+\epsilon \approx 0.94$ for any $\epsilon > 0$. The GW rounding technique is tight however, i.e. there are graphs where their algorithm performs exactly with the approximation ratio α_{gw}. So one could think that there is still room for improvement using a new technique. However, Khot et al. [KKMO04] give an analysis which suggests that the ratio achieved by the Goemans-Williamson algorithm could be in fact optimal, assuming the "Unique Games conjecture" (it assumes another conjecture which has been proven in [MOO05]). This is important to understand the generality of the semidefinite programming formulation.

As we have seen, the random hyperplane rounding technique was the source of a number of generalizations, which seem to have now exhausted the power of the basic method for simple combinatorial problems. Translating these methods and analysis techniques to more complex problems appears to be a challenge. We have also shown that the ideas introduced by Alon and Naor lead to a more powerful analysis of the rounding technique, and at the same time draws interesting connections with important topics in functional analysis. This convergence looks promising, as it has already inspired a number of papers solving quite general integer quadratic programs (see [CW04] and the references therein).

References

[Ali95] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5:13–51, 1995.

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 14–23, 1992.

[AN04] N. Alon and A. Naor. Approximating the cut-norm via Grothendieck's inequality. In *Proceedings of the 36th ACM STOC*, pages 72–80, 2004.

[ASZ01] N. Alon, B. Sudakov, and U. Zwick. Constructing worst case instances for semidefinite programming based approximation algorithms. In *Symposium on Discrete Algorithms*, pages 92–100, 2001.

[Bar95] A. Barvinok. Problems of distance geometry and convex properties of quadratic maps. *Discrete and Computational Geometry*, 13:189–202, 1995.

[CW04] M. Charikar and A. Wirth. Maximizing quadratic programs: extending Grothendieck's inequality. In *Proceedings of the 45th annual IEEE symposium on Foundations of Computer Science*, 2004.

[FG95] U. Feige and M. Goemans. Approximating the value of two prover proof systems, with applications to max 2sat and max dicut. In *Proceedings of the Third Israel Symposium on Theory of Computing and Systems*, pages 182–189, 1995.

[FL01] U. Feige and M. Langberg. The RPR2 rounding technique for semidefinite programs. In *ICALP '01: Proceedings of the 28th International Colloquium on Automata, Languages and Programming*, pages 213–224, 2001.

[GLS88] M. Grotschel, L. Lovasz, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, 1988.

[Gro53] A. Grothendieck. Résumé de la théorie métrique des produits tensoriels topologiques. *Boletim da Sociedade de matematica de Sao Paulo*, 8:1–79, 1953.

[GW96] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming programs. *Journal of the ACM*, 42:1115–1145, 1996.

[Hås97] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th ACM Symposium on the Theory of Computing*, 1997.

[Jam87] G. Jameson. *Summing and nuclear norms in Banach space theory*, volume 8. London Mathematical Society Student Texts, 1987.

[KKMO04] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for max-cut and other 2-variable CSPs? In *Proceedings of the 45th annual IEEE symposium on Foundations of Computer Science*, 2004.

[KMS98] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM*, 45:246–265, 1998.

[Kri77] J. Krivine. Sur la constante de Grothendieck. *Compte Rendu de l'Academie des Sciences de Paris*, Serie A-B 284:445–445, 1977.

[LLZ02] M. Lewin, D. Livnat, and U. Zwick. Improved rounding techniques for the max 2-sat and max di-cut problems. In *Integer Programming and Combinatorial Optimization (IPCO)*, pages 67–82, 2002.

[MOO05] E. Mossel, R. O'Donnell, and K. Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. Manuscript, 2005.

[MR99] S. Mahajan and H. Ramesh. Derandomizing approximation algorithms based on semidefinite programming. *SIAM Journal on Computing*, 28(5):1641–1663, 1999.

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

[Pat94] G. Pataki. On the multiplicity of optimal eigenvalues. Technical report, Carnegie-Mellon University, 1994.

[PY91] C.H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and Systems Sciences*, 43:425–440, 1991.

[Rie74] R. Rietz. A proof of the Grothendieck inequality. *Isreal Journal of Mathematics*, 19:271–276, 1974.

[VB96] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[Wig83] A. Wigderson. Improving the performance guarantee for approximate graph coloring. *Journal of the ACM*, 30:729–735, 1983.

[Zwi99] U. Zwick. Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to MAX CUT and other problems. In *Proceedings of the 31th ACM Symposium on Theory of Computing*, pages 679–687, 1999.