Improved approximation of Max-Cut on graphs of bounded degree

Uriel Feige* Marek Karpinski† Michael Langberg‡

April 16, 2000

Abstract

We analyze the addition of a simple local improvement step to various known randomized approximation algorithms. Let $\alpha \simeq 0.87856$ denote the best approximation ratio currently known for the Max Cut problem on general graphs [GW95]. We consider a semidefinite relaxation of the Max Cut problem, round it using the random hyperplane rounding technique of ([GW95]), and then add a local improvement step. We show that for graphs of degree at most Δ, our algorithm achieves an approximation ratio of at least $\alpha + \epsilon$, where $\epsilon > 0$ is a constant that depends only on Δ. In particular, using computer assisted analysis, we show that for graphs of maximal degree 3, our algorithm obtains an approximation ratio of at least 0.921, and for 3-regular graphs, the approximation ratio is at least 0.924. We note that for the semidefinite relaxation of Max Cut used in [GW95], the integrality gap is at least 1/0.884, even for 2-regular graphs.

1 Introduction

Given a graph $G = (V, E)$, the Max-Cut problem on G is the problem of finding a partition (X, Y) of the vertex set V which maximizes the amount of edges with one endpoint in X and another in Y. Let A be an approximation algorithm for the Max-Cut problem with an approximation ratio of r. That is, running A on a given graph G we obtain a partition (X, Y) of value at least r times the value of the optimal partition.

Usually when considering **random** approximation algorithms A as above, one cannot be sure that the partition (X, Y), obtained by running A on G, is maximal with respect to *local* improvements. For instance, it might be the case that a vertex i of G and more than half its neighbors end up on the same side of the partition (X, Y) above. We denote such vertices as *misplaced* vertices. Clearly such a partition can be improved by moving the misplaced vertex i from one side of the partition to the other, thus yielding a new partition of higher weight.

*Department of Computer Science and Applied Mathematics, the Weizmann Institute.
†Department of Computer Science, University of Bonn.
‡Department of Computer Science and Applied Mathematics, the Weizmann Institute.

In the following work, we analyze the expected approximation ratio of a number of randomized approximation algorithms A which are enhanced by an additional local step that moves misplaced vertices from one side of the partition to the other, until none such vertices are left.

This approach generalizes to other problems such as the Max-Sat and Max-CSP problems, where we are given a Boolean formula φ and we are to find a truth assignment to the variables of φ which maximizes the number of clauses satisfied. In this case, misplaced variables can be defined analogously to misplaced vertices in the Max-Cut problem. Given a truth assignment $\mathbf{s} = \{s_1 \dots s_n\}$ to the variables of φ, we say that a variable x_i is misplaced in $\mathbf{s}$ if the number of clauses satisfied by $\mathbf{s}$ is strictly less than the number of clauses satisfied by the modified assignment achieved by flipping the value of s_i.

We address the restriction of the Max-Cut and Max-CSP problems to instances in which each variable (vertex) shares a bounded number of constraints with other variables (vertices). In the case of Max-Cut, we define the bounded Max-Cut problem as the restriction of Max-Cut to graphs of bounded maximal degree, and in the case of Max-CSP we define the bounded Max-CSP problem as the restriction of Max-CSP to instances in which each clause is of bounded length and each variable appears in a bounded number of clauses.

Our main results are achieved on the bounded Max-Cut problem when we consider the algorithm based on semidefinite programming presented in [GW95] as a base for the enhancement described above. In this algorithm a semidefinite relaxation of the Max-Cut problem on a given graph G is solved, resulting in a set of n unit vectors in R^n corresponding to the n vertices of G. These vectors are then rounded into a partition of G by choosing a random hyperplane passing through the origin, and then setting all vertices corresponding to vectors that lie 'above' the hyperplane to be on one side of the partition, and all remaining vertices to be on the other side. It is shown by [GW95] that for each edge e in G the ratio between the expected contribution of the edge to the final cut and the contribution of the edge to the objective function of the semidefinite program is at least 0.87856, thus implying (using linearity of expectation) that the expected approximation ratio of this algorithm is at least 0.87856.

The Max-Cut problem on bounded degree graphs is mentioned in [BK98] where it is shown that it is *NP-hard* to approximate the Max-Cut problem on regular graphs of degree 3 beyond the ratio of 0.997. To the best of our knowledge, the best known approximation ratio for the Max-Cut problem on bounded degree graphs is that of the general problem which is 0.87856. We improve this approximation ratio to 0.921 on graphs of maximal degree three, to 0.924 on regular graphs of degree three, and to $0.87856 + \varepsilon_\Delta$ on graphs of maximal degree Δ, where ε_Δ is a positive constant depending on Δ alone.

These results are achieved using an algorithm that differs from the one presented in [GW95] not only in the additional improvement step, but also in the fact that an *improved* semidefinite relaxation of the Max-Cut problem is used as a base of our algorithm. Specifically, we add *triangle* constraints (mentioned in [FG95]) to our relaxation. These additional constraints have been studied in the past in the context of several problems including the Max-Cut problem. The results of [BM86] imply that the value of the Max-Cut semidefinite relaxation with triangle constraints is **equal** to the value of the optimal cut on planar graphs. Without these constraints it is shown in [GW95] that the original semidefinite relaxation has an integrality gap of 1/0.884 even on two regular graphs (the 5 cycle). In [Zwi99] these constraints are used in order to achieve an improved approximation ratio on the Max-Cut problem on graphs which have a fractional triangle covering. The addition of such triangle constraints to the semidefinite relaxation of the Max-Cut problem on general graphs is not known to yield an improved approximation algorithm nor is it known to improve the *worst case* integrality gap of such a relaxation. In our work, the addition of such triangle constraints to the semidefinite relaxation of the Max-Cut problem on graphs of maximal degree three yields an improved approximation algorithm with a ratio of 0.921. As the original semidefinite relaxation of [GW95] has an integrality gap of 1/0.884 on a two regular graph, we conclude that the addition of such triangle constraints to our semidefinite relaxation is crucial to the success of our algorithm.

As in the algorithm of [GW95], many other algorithms based on semidefinite programming use the *random hyperplane* rounding technique, and are analyzed in a *local* manner (for instance [FG95, KZ97, Zwi99]). *I.e.* for each edge a local expected approximation ratio is computed, this ratio then holds as an expected approximation ratio for the algorithm as a whole due to linearity of expectation. In our algorithm for the Max-Cut problem on graphs with maximal degree three, we present an analysis in which we compute a local approximation ratio not on single edges but on clusters of edges. In particular we concentrate on pairs of edges which share a common vertex. Such an analysis allows us to evaluate the contribution of our additional improvement step as a function of the vector configuration obtained by the semidefinite relaxation.

The paper is structured as follows. We address three different randomized approximation algorithms as a base for the above enhancement. That is, we analyze the expected approximation ratio of these algorithms when a new step is added which flips the value of misplaced variables, until none such are left. In Section 2 we consider the trivial random algorithm for the bounded Max-Cut problem which given a graph $G = (V, E)$ constructs a partition (X, Y) by independently choosing each vertex to be in X with probability 1/2, and achieve an approximation ratio strictly greater than 1/2. In Section 3 we consider the algorithm presented in [GW94] for the Max-Sat problem in which we fix an assignment to the variables of a given instance φ by independently setting each variable x_i in φ to be *true* with some probability p_i derived as a solution of a linear program. By enhancing this algorithm with a local improvement step, we slightly improve the original approximation ratio of 3/4 stated in [GW94] when we restrict ourselves to instances of bounded Max-Sat. Finally in our main Section 4, we return to the bounded Max-Cut problem and consider the [GW95] algorithm discussed above.

2 The trivial algorithm for Max-Cut

Let $G = (V, E)$ be a graph of maximal degree Δ where $V = \{1 \ldots n\}$. For a partition (X, Y) of V let $w(X) = |E \wedge (X \times Y)|$ be the number of edges cut by the partition, and $|E|$ be the total number of edges in E. In the following we denote $w(X)$ and $|E|$ as the *weight* of the partition (X, Y) and the weight of the total edge set respectively. Finally, denote the number of edges cut by the optimal partition as $Opt(G)$. Note that in this paper we consider unweighted graphs only. Our methods can be extended to the case of weighted graphs of bounded degree, though the analysis becomes more complicated, and is omitted from the current version of the paper.

Consider the following trivial random algorithm for the Max-Cut problem on G which constructs a partition (X, Y) by independently choosing each vertex to be in X with probability $1/2$. It is clear that the expected number of edges cut by the resulting partition is exactly half the total number of edges in G, thus this trivial algorithm has an expected approximation ratio of $1/2$. We now improve this algorithm by adding a simple local improvement step which moves misplaced vertices from one side of the partition to another in aim to increase the number of edges cut by the partition.

Consider a random partition (X, Y) of the vertices of G obtained by the above trivial algorithm. We say that a vertex v is *misplaced* in (X, Y) if it has more neighbors on its side of the partition than on the opposite side. Moving such a vertex from one side of the partition to the other will increase the value of the cut. Roughly speaking we will prove that in a random partition of the vertex set V the expected fraction of misplaced vertices is constant. Combining this with the fact that moving each misplaced vertex may affect at most Δ other vertices, we conclude that the expected contribution of the additional step to the trivial algorithm is a constant fraction of the total number of edges in E (assuming Δ is constant). This yields an approximation algorithm with an expected approximation ratio strictly greater than $1/2$ by some constant. Details follow.

We start by computing the probability that a vertex v will be misplaced in a random partition. Consider all neighbors of v. With probability at least $1/2$, the random partition partitions them into two sets of unequal size. Thereafter, with probability at least $1/2$, vertex v is placed on the same side as the majority of its neighbors. Hence the probability that v is misplaced is at least $1/4$, and the expected number of misplaced vertices is at least $n/4$. Moving a misplaced vertex v from one side of the partition to the other contributes at least one edge to the partition. Such a movement can reduce the number of misplaced vertices by at most $\Delta+1$, because only v and its neighbors are affected. We conclude that enhancing the trivial random algorithm described above by adding a step which moves misplaced vertices from one side of the partition to the other until none such vertices are left will achieve a cut of expected weight at least

$$\frac{|E|}{2} + \frac{n}{4(\Delta+1)} \geq |E| \left(\frac{1}{2} + \frac{1}{2\Delta(\Delta+1)}\right) \geq Opt(G) \left(\frac{1}{2} + \frac{1}{2\Delta(\Delta+1)}\right),$$

thus yielding an improved expected approximation ratio of $\frac{1}{2} + \frac{1}{2\Delta(\Delta+1)}$. We remark that the term $\frac{1}{2\Delta(\Delta+1)}$ can be significantly improved by tighter analysis, but we wish to keep the analysis simple so as to clearly illustrate the method.

A similar argument can be used in order to improve the approximation ratio of the trivial random algorithm on the maximization of any bounded constraint satisfaction problem (Max-CSP) in which each variable appears in at most Δ clauses, and each clause is of length at most k. In such cases, instead of moving a vertex from one side of the partition to the other we *flip* the value of a variable in aim to increase the total amount of clauses satisfied. If the trivial algorithm yields an expected approximation ratio of r, our enhanced algorithm will yield an expected approximation ratio of $r + \frac{1}{\Delta^2 2^k \Delta}$. Independently, Håstad ([Hås99]) presents an algorithm for a related restriction of Max-CSP and achieves a better approximation ratio of $r + \frac{1-r}{2^{3k/2} 2k\Delta}$ using algebraic techniques.

3 The LP algorithm for Max-Sat

Let φ be an instance of the bounded Max-Sat problem in which each clause of φ is of length at most k and each variable in φ appears in at most Δ clauses. We denote this restricted problem as the Max-k-Sat(Δ) problem. Let $\{x_1 \dots x_n\}$ be the variables of φ and $\{C_1 \dots C_m\}$ be the set of clauses in φ.

Consider the following relaxation of the Max-k-Sat(Δ) problem on φ :

(LP)	Maximize	$\sum_{j=0}^{m} z_j$	
	subject to:		
	(1)	$z_j \leq \sum_{x_i \in C_j} x_i + \sum_{\bar{x}_i \in C_j} 1 - x_i$	for $j \in [m]$
	(2)	$x_i, z_j \in [0, 1]$	for $i \in [n],\ j \in [m]$

Where $[n] = \{1 \dots n\}$.

Let $\{x_1^* \dots x_n^*\}$ be the optimal fractional solution obtained by solving *(LP)*. In [GW94] an algorithm which obtains an assignment $\mathbf{s} = \{s_1 \dots s_n\}$ to the variables of φ by independently choosing each s_i to be *true* with probability x_i^* is analyzed. In particular, it is shown in [GW94] that combining such an algorithm with a trivial random algorithm, an approximation ratio of 3/4 on the Max-Sat problem is achieved.

Applying the same principles described in Section 2 to the above algorithm based on linear programming we obtain the following proposition.

Proposition 3.1 *Adding a local improvement step to the algorithm based on relaxation (LP) yields an expected approximation ratio of* $3/4+\varepsilon$ *on the Max*-k-*Sat(Δ) problem for some* $\varepsilon > 0$ *dependent on* Δ *and* k *alone.*

The proof is omitted from the current version of the paper.

4 Semidefinite programming

Consider the well known Max-Cut algorithm based on semidefinite programming presented in [GW95]. In this algorithm, given a graph G, a semidefinite relaxation of the Max-Cut problem on G is solved yielding an embedding of G on the n dimensional sphere. This embedding is then rounded using the random hyperplane rounding technique, into a partition of G. It is shown in [GW95] that the expected weight of this partition is at least 0.87856 the weight of the optimal cut in G.

We enhance the above Max-Cut algorithm by adding a local improvement step, analogous to the one presented in the previous sections, which improves the expected approximation ratio of 0.87856 on graphs of bounded degree Δ. In contrast to the previous sections, the random hyperplane rounding technique does not round vertices independently. Hence we need more sophisticated methods in order to argue that there will be misplaced vertices after applying such a rounding technique. These more sophisticated methods use the fact that the graph is of bounded degree, a fact that was not used for this purpose in Section 2. In the following section, we present a detailed analysis in the case of graphs with maximal degree three, and sketch the analysis for graphs of higher maximal degree.

4.1 Max-Cut on graphs of maximal degree 3

Using the notation of Section 2, let $G = (V, E)$ be a graph of maximal degree 3 and let $w(X) = |E \wedge (X \times Y)|$ denote the number of edges cut by the partition (X, Y). Denote the total number of edges by $|E|$, and the number of edges cut by the optimal partition of G as $Opt(G)$. We present an algorithm which given $G = (V, E)$ as above will output a partition (X, Y) of V such that $w(X) \geq rOpt(G)$ for $r \geq 0.921$. Our algorithm is an extension of the well known Max-Cut algorithm presented in [GW95].

In the first step of our algorithm a semidefinite relaxation of the Max-Cut problem on G is solved to achieve a set of n unit vectors $\{v_1 \dots v_n\}$.

(SDP-Cut)	Maximize	$\sum_{e_{ij} \in E} \frac{1-v_iv_j}{2}$	
	subject to:		
	(1)	$v_i \in S_n$	for $1 \leq i \leq n$
	(2)	$v_iv_j + v_iv_k + v_jv_k \geq -1$	
		$v_iv_j - v_iv_k - v_jv_k \geq -1$	for all $i, j, k \in [n]$

By restricting the vectors $\{v_1 \dots v_n\}$ to be one dimensional, and setting the vertex i to be in X if and only if the vector $v_i = 1$ it can be seen that *(SDP-Cut)* corresponds to the Max-Cut problem on G. Note that we have added additional triangle constraints (mentioned in [FG95]) that do not appear in the original relaxation presented in [GW95]. Without these constraints it is shown in [GW95] that the above semidefinite relaxation has an integrality gap of $1/0.884$ even on 2 regular graphs (the 5 cycle). Hence these additional triangle constraints are crucial to our analysis which results with an approximation ratio of 0.921 on such graphs. The value of the integrality gap for this semidefinite relaxation is open. See [FG95] for a discussion of this issue.

It is shown in [GW95], that obtaining a cut (X, Y) in G by rounding the vector configuration obtained by *(SDP-Cut)* using the random hyperplane rounding technique, the expected contribution of each edge e_{ij} to the cut is $\frac{\theta_{ij}}{\pi}$, where θ_{ij} is the angle between the vectors v_i and v_j. As the contribution of each edge to the objective function of *(SDP-Cut)* is $\frac{1-\cos\theta_{ij}}{2}$ we conclude that the expected approximation ratio achieved on each edge is $\frac{2\theta_{ij}}{\pi(1-\cos\theta_{ij})}$. This expected ratio is minimal only when $\theta_{ij} = \theta_0 = 2.3311$, and in this case obtains the value $\alpha = 0.87856$. We conclude that the expected weight $w(X)$ of the partition (X, Y) obtained by the random hyperplane rounding technique, is at least $\alpha Opt(G)$, and will be exactly $\alpha Opt(G)$ when for all edges e_{ij} we have that the angle θ_{ij} is θ_0 or zero.

Our main observation is the fact that in both *worst* cases (where for all edges in G the angle between the corresponding vectors is zero or θ_0) there is some constant probability that a vertex and two of its neighbors lie on the same side of the partition obtained by the random hyperplane rounding technique. In such cases moving the vertex from one side of the partition to the other will increase the weight of the partition. As earlier, we denote such vertices as misplaced ones. Using this observation we add a second step to our algorithm in which misplaced vertices are moved from one side of the partition to the other, until none such vertices are left. This step is done in a *greedy* manner, thus we denote this second step as the greedy phase of our algorithm.

Geometrically speaking, one can view our observation in the following way. Let v be some vertex in V, and y_1, y_2 be two of its neighbors. Denote the corresponding vectors in the optimal vector configuration as v, y_1, y_2 respectively. If the angles between the vectors v and y_1, v and y_2 are exactly θ_0, (*i.e.* the expected approximation ratio achieved on the edges (v, y_1) and (v,y_2) is exactly α) then by constraint *(2)* of *(SDP-Cut)* above, it **cannot** be the case that all three vectors lie on the same plane. Furthermore, this constraint implies that the angle between the vector v and the plane containing the vectors y_1 and y_2 is at least 0.57. Implying a probability of at least 0.07 that after the randomized rounding the vertex v will be misplaced.

We now add an additional constraint to the relaxation *(SDP-Cut)* which will simplify the analysis yet to come, and fill in the details regarding the greedy phase of our algorithm.

Semidefinite relaxation :
Consider an **optimal** partition (X, Y) of a given graph $G = (V, E)$ of maximal degree three. For every vertex v it cannot be the case that v and two of its neighbors lie on the same side of the partition (*i.e.* v is misplaced). In such a case moving v to the other side of the partition would increase the weight of the partition, which is a contradiction to its optimality. Hence we may add a corresponding constraint to *(SDP-Cut)* which rules out the possibility of misplaced vertices, when the corresponding vectors of *(SDP-Cut)* are restricted to be one dimensional.

Let T be the set of all triplets (i, j, k) such that $i, j, k \in [n]$, $j < k$ and $e_{ij}, e_{ik} \in E$, we enhanced the previous semidefinite relaxation by adding the new constraint *(3)* below. The resulting relaxation will be used in our algorithm :

(SDP-Cut)	Maximize	$\sum_{e_{ij}\in E} \frac{1-v_iv_j}{2}$	
	subject to:		
	(1)	$v_i \in S_n$	for $1 \le i \le n$
	(2)	$v_iv_j + v_iv_k + v_jv_k \ge -1$	
		$v_iv_j - v_iv_k - v_jv_k \ge -1$	for all $i, j, k \in [n]$
	(3)	$v_iv_j + v_iv_k + v_jv_k = -1$	for all $(i, j, k) \in T$

Greedy phase :
As mentioned earlier after the first step of our algorithm it might be the case that for some triplet $(i, j, k) \in T$, defined above, the vertices i, j and k lie on the same side of the partition. In such a case, moving the misplaced vertex i to the other side of the partition will increase its weight. We denote such a triplet $(i, j, k) \in T$ in which i is misplaced as a *good* triplet.

Given a partition (X, Y) we are interested in moving all misplaced vertices until none are left. In general at each step of this greedy process we could decide to move the one misplaced vertex that increases the weight of the partition by most. But as moving one misplaced vertex affects other vertices, we are also interested that the vertex moved does not *destroy* many good triplets (a good triplet is destroyed if it is no longer good). In order to combine these two interests, at each stage of our greedy process we move the vertex for which the ratio between the weight added to the partition by moving it from one side of the partition to another and the number of good triplets destroyed by this act, is maximal. We are now ready to define our algorithm on a given graph G of maximal degree 3.

Algorithm A_{cut} **:**

1. Solve *(SDP-Cut)* to obtain an optimal vector configuration $\{v_1 \dots v_n\}$ of value Z. Round the vector configuration using the random hyperplane rounding technique from [GW95]. Denote the partition obtained by (X, Y).

2. Greedily move misplaced vertices from one side of the partition to the other according to the procedure above.

Theorem 4.1 *A_{cut} has an expected approximation ratio of $\alpha \geq 0.921$.*

Proof : First we note that w.l.o.g. we may assume that G does not have any vertices of degree 1. Otherwise we may run algorithm A_{cut} on the graph $\hat{G}$ obtained by iteratively removing all vertices of degree one in G. It can be seen that an approximation algorithm with ratio r on $\hat{G}$ yields an approximation algorithm with ratio at least r on G.

Assume G is as above, and let (X, Y) be the partition after step *(1)* of algorithm A_{cut}. Let $W = w(X)$ be the weight of this partition. The upcoming lemma analyzes the contribution of the second greedy step of algorithm A_{cut}, and uses the following refinement of the set T.

For each edge e_{ij} let d_{ij} be the number of triplets in T in which e_{ij} appears. It can be seen that in a graph of minimum degree 2 and maximum degree 3, $d_{ij} = 2$ if the degree of v_i and v_j are 2, $d_{ij} = 3$ if the degree of v_i differs from the degree of v_j, and $d_{ij} = 4$ otherwise. Let T be the set of triplets in G, for $l = 4 \dots 8$ denote the set of triplets $(i, j, k) \in T$ in which $d_{ij} + d_{ik} = l$ by T_l. Given a partition (X, Y) we denote by S_l the number of good triplets in T_l.

Lemma 4.2 *Let (X, Y) be some partition in G of weight W. Let T, T_l, and S_l for $l = 4 \dots 8$ be defined as above. Executing step (2) of algorithm A_{cut} on the partition (X, Y) will yield a new partition of weight at least*

$$W + \frac{2}{3}S_4 + \frac{1}{2}S_5 + \frac{2}{5}S_6 + \frac{3}{8}S_7 + \frac{1}{3}S_8$$

Proof : Roughly speaking, we define the contribution of each triplet (i, j, k) that is good in (X, Y) as the number of edges it adds to the partition the moment it is destroyed by the greedy phase of our algorithm. Let τ be one of the vertices i, j, or k. If (i, j, k) is destroyed by the movement of the vertex τ, and by this act p edges are added to the partition and q triplets (including (i, j, k)) are destroyed, we fix the contribution of the triplet (i, j, k) to be p/q. Hence, due to the nature of our greedy phase, we may bound the contribution of the good triplet (i, j, k) from below by computing the ratio between the number of edges added to the partition and the number of triplets destroyed by the act of moving any one of the vertices i, j, or k from one side of the partition to another. By finding such a lower bound for each good triplet in T_l $(l = 4 \dots 8)$ we conclude our assertion.

Note that for any good triplet (i, j, k) and any vertex τ as above, the contribution of moving τ from one side of the partition to another is at least one edge, while this act will destroy at most 9 triplets. It follows that we may trivially claim that executing step *(2)* of algorithm A_{cut} on the given partition (X, Y) will yield a new partition of weight at least

$$W + \frac{1}{9}(S_4 + S_5 + S_6 + S_7 + S_8).$$

The following case analysis refines this trivial analysis and provides full proof of the asserted lemma.

Case 1 : Let (i, j, k) be a good triplet in T_4, *i.e.* the vertices i, j and k are of degree 2, and all lie on the same side of the partition (X, Y). As the greedy step *(2)* of A_{cut} continues as long as there is some misplaced vertex, the triplet (i, j, k) will be destroyed sometime during our greedy procedure. Denote the vertex moved when (i, j, k) is destroyed by τ (τ is either i, j or k). Let p be the number of edges added to the partition when (i, j, k) is destroyed, and q be the number of good triplets destroyed along with (i, j, k).

As mentioned above the degree of i, j and k are 2. Hence, if (i, j, k) is a good triplet, then by moving the vertex i we destroy at most 3 triplets and increase the weight of the partition by exactly 2 edges. Due to the nature of our greedy phase, we conclude that by moving τ from one side of the partition to the other, the ratio between the number of edges added to the partition and the number of triplets destroyed is at least $2/3$. We conclude that when (i, j, k) is destroyed it contributes at least $2/3$ to the partition.

Case 2 : Let (i, j, k) be a good triplet in T_5, using the same line of analysis displayed in the previous case, it is enough to analyze the movement of the vertex i from one side of the partition to the other. Assume that the vertex j is of degree 3 and vertices i, k are of degree 2 (it cannot be the case that i is of degree 3). In Figure 1 a schematic view of our case is presented. Solid lines represent edges that **preserve** sides of the partition (for instance the vertices i, j and k lie on the same side of the partition), dotted lines represent edges that are **not known to preserve** sides of the partition, and bold lines represent edges that are known **not to preserve** sides of the partition. It can be seen that moving i from one side of the partition to another we gain exactly 2 edges to the partition, but destroy at most 4 triplets. We conclude that (i, j, k) contributes at least $1/2$ edges to the partition.

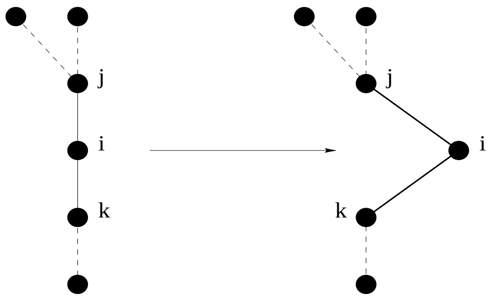

Figure 1: Case 2.

Case 3 : Let (i, j, k) be a good triplet in T_6. There are two possibilities, either the vertex

i is of degree 2 and the vertices j, k are of degree 3, or the vertex i is of degree 3 and j, k are of degree 2. Both cases are presented in Figure 2. In the first case *(a)*, moving i to the other

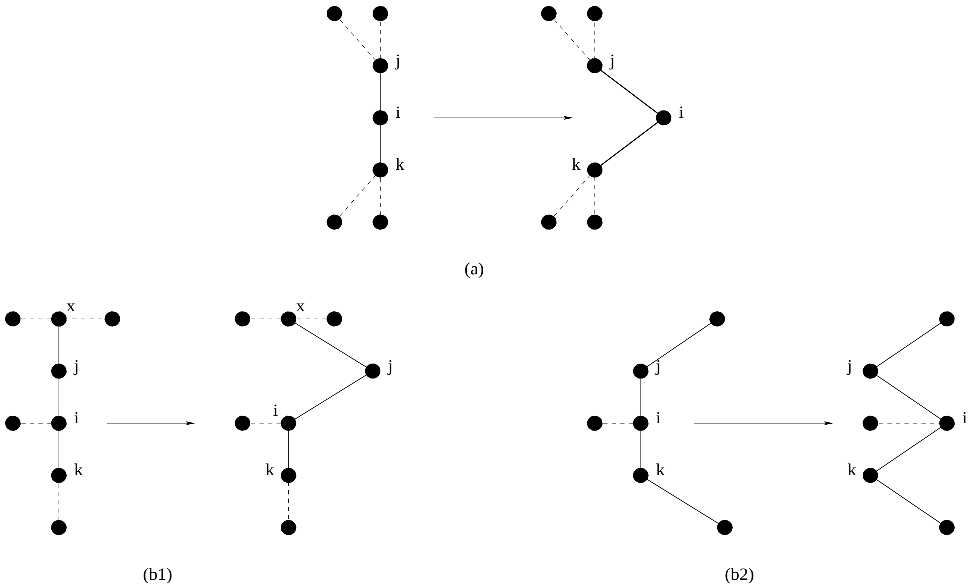

Figure 2: Case 3.

side of the partition increases the weight of the partition by 2 edges and destroys at most 5 triplets. Hence, if we are in this case then (i, j, k) contributes $2/5$ when it is destroyed.

In the second case we consider two sub-cases : *(b1)* the case in which one of the vertices j or k are on the same side as their additional neighbor x, *(b2)* the case in which the additional neighbors to j and k are both on the opposite side of the partition. In the first case we have that moving j from one side of the partition to the other will contribute 2 edges to the cut and destroy at most 5 triplets, yielding a ratio of $2/5$, and in the second we obtain a ratio of $3/5$ by again moving j (note that in the case *(b2)* one must fix the side of the third neighbor x of i and only then analyze), thus we conclude our assertion.

Case 4 : Let (i, j, k) be a good triplet in T_7. It must be the case that the vertex i is of degree 3 and the vertices j, k are of different degree (we assume that j is of degree 2). We consider three cases that are presented in Figure 3. In the first case *(a)* all three neighbors of i are on the same side as i. In such a case moving i one can see that a ratio of at least $3/8$ is achieved. In the second and third cases we assume that the additional neighbor of i is on the opposite side of the partition.

In the second case *(b)* we assume that the additional neighbor x of j is on the same side as j. In this case moving j we achieve a ratio of at least $2/4$.

At last in the third case *(c1)* and *(c2)* we assume that the additional neighbor of j is also on the opposite side of the partition, and we consider two situations. The first when all additional neighbors of k are on the same side of the partition as k, and the second when there is some neighbor x of k on a different side. It can be seen that moving k in the first case results with a ratio of at least $3/8$, and moving i in the second results with a ratio of at least $1/2$.

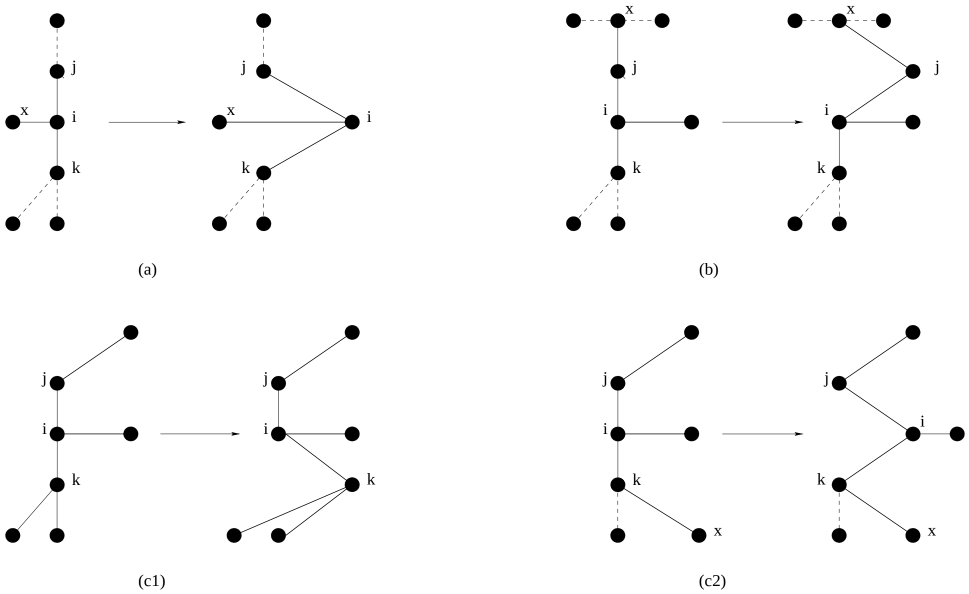

Figure 3: Case 4.

Case 5 : Let (i,j,k) be a good triplet in T_8. It must be the case that all three vertices are of degree 3. In Figure 4 three sub-cases are considered. In case *(a)* all three neighbors of i are on the same side of the partition as i, in this case moving i we obtain a ratio of 3/9. In the remaining cases *(b), (c)* we assume that the additional neighbor of i is on the opposite side of the partition.

In case *(b)* we assume that for one of the vertices j or k, all three of its neighbors are on the same side of the partition as it is (in Figure 4 we assume the above for j). In such a case moving j results with a ratio $\geq 3/8$.

In the last case *(c)* both i, j, and k have at least one neighbor on an opposite side. In this case moving i we have a ratio $\geq 1/3$. $\square$

Corollary 4.3 *Let W be the weight of the partition (X,Y) obtained by step (2) of algorithm A_{Cut}, and let S_l $(l = 4\ldots 8)$ be the number of good triplets in (X,Y) according to the above definition. The expected weight of the partition received by algorithm A_{Cut} is :*

$$\begin{aligned}
E[w(A_{Cut})] &= E\left[W + \frac{2}{3}S_4 + \frac{1}{2}S_5 + \frac{2}{5}S_6 + \frac{3}{8}S_7 + \frac{1}{3}S_8\right] = E[W] + \sum_{l=4}^{8}\alpha_l E[S_l] \\
&= \sum_{e_{ij}} \Pr(e_{ij} \text{ is cut}) + \sum_{l=4}^{8}\sum_{(i,j,k)\in T_l} \alpha_l \Pr(v_i, v_j, v_k \text{ are not separated}) \\
&= \sum_{l=4}^{8}\sum_{(i,j,k)\in T_l} \frac{\Pr(e_{ij} \text{ is cut})}{d_{ij}} + \frac{\Pr(e_{ik} \text{ is cut})}{d_{ik}} + \alpha_l \Pr(v_i, v_j, v_k \text{ are not separated}).
\end{aligned}$$

For $\alpha_{4\ldots 8} = (2/3, 1/2, 2/5, 3/8, 1/3)$.

Proof : All probabilities above are taken over random hyperplanes cutting the unit sphere of R^n. The second and third equalities are due to linearity of expectation, and the fourth is due to the fact that each edge e_{ij} is in exactly d_{ij} triplets in T. $\square$

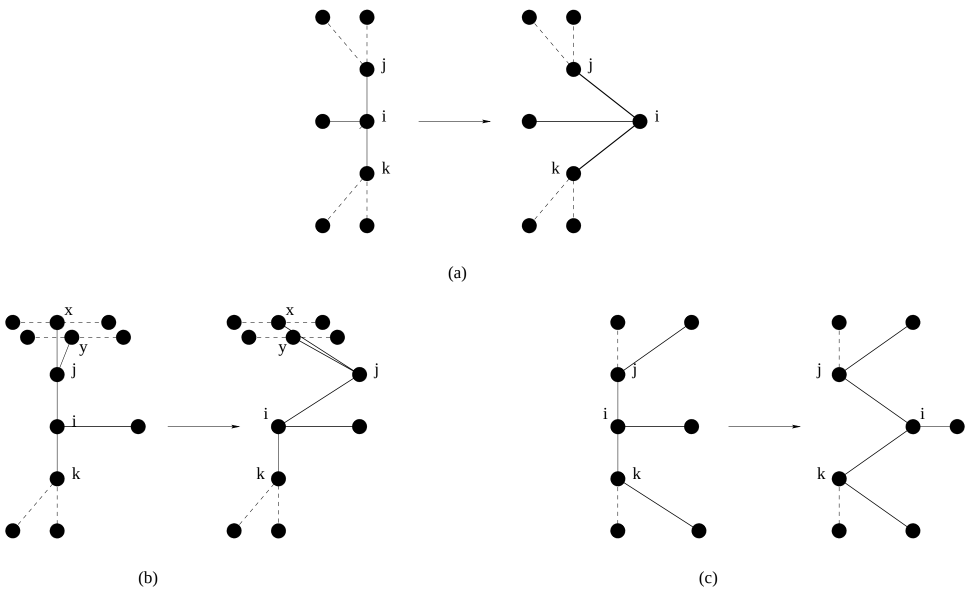

Figure 4: Case 5.

Lemma 4.4 *For every triplet $(i,j,k) \in T_l$, using the fact that $d_{ij}, d_{ik} \in \{2,3,4\}$, $l = d_{ij} + d_{ik} \in \{4 \dots 8\}$ we have that :*

$$\frac{\Pr(e_{ij} \text{ is cut})}{d_{ij}} + \frac{\Pr(e_{ik} \text{ is cut})}{d_{ik}} + \alpha_l \Pr(v_i, v_j, v_k \text{ are not separated}) \geq r\left(\frac{1 - v_i v_j}{2d_{ij}} + \frac{1 - v_i v_k}{2d_{ik}}\right)$$

for $r = 0.921$.

Proof : Let $l = d_{ij} + d_{ik}$, define the following functions :

$$\begin{aligned}
f(\theta_{ij}, \theta_{ik}) &= \frac{\theta_{ij}}{d_{ij}\pi} + \frac{\theta_{ik}}{d_{ik}\pi} + \alpha_l\left(1 - \frac{1}{2\pi}(\theta_{ij} + \theta_{ik} + \arccos(-1 - \cos(\theta_{ij}) - \cos(\theta_{ik})))\right) \\
g(\theta_{ij}, \theta_{ik}) &= \frac{1 - \cos(\theta_{ij})}{2d_{ij}} + \frac{1 - \cos(\theta_{ik})}{2d_{ik}} \\
h(\theta_{ij}, \theta_{ik}) &= \frac{f(\theta_{ij}, \theta_{ik})}{g(\theta_{ij}, \theta_{ik})}
\end{aligned}$$

Where θ_{ij} (θ_{ik}) is the angle between v_i and v_j (v_i and v_k).

The function $f(\theta_{ij}, \theta_{ik})$ represents the expected contribution of the triplet (i,j,k) to the cut. As stated earlier summing over $f(\theta_{ij}, \theta_{ik})$ for all triplets (i,j,k) in T yields the expected size of the cut obtained by A_{Cut}. Similarly, the function $g(\theta_{ij}, \theta_{ik})$ represents the contribution of each triplet (i,j,k) to the objective function of *(SDP-Cut)*.

In order to prove the above claim it is suffice to show that $h(\theta_{ij}, \theta_{ik}) > r$ in the range $\theta_{ij} \in [0, \pi]$, $\theta_{ik} \in [\pi - \theta_{ij}, \pi]$ (the triangle constraints added to *(SDP-Cut)* imply that $\cos(\theta_{ij}) + \cos(\theta_{ik}) \leq 0$). By evaluating the value of h over a number of points in the above range and bounding the derivative of h over the whole range our lemma is proven. $\square$

By Corollary 4.3 and Lemma 4.4 we have that the expected weight of the partition obtained by algorithm A_{Cut} is

$$E[w(A_{Cut})] \geq r \sum_{l=4}^{8} \sum_{(i,j,k) \in T_l} \left(\frac{1 - v_i v_j}{2d_{ij}} + \frac{1 - v_i v_k}{2d_{ik}} \right) = rZ \geq rOpt(G),$$

for $r = 0.921$.

This completes the proof of Theorem 4.1. The same line of analysis yields an 0.924 approximation ratio on 3 regular graphs. □

4.2 Max-Cut on graphs of maximal degree Δ

In the following section we deal with the Max-Cut problem on graphs of maximal degree Δ. We present a slightly different algorithm and analysis than the ones presented in the case where $\Delta = 3$, and achieve an improved approximation ratio of $\alpha + \varepsilon_\Delta$ where $\alpha = 0.87856$ as before and $\varepsilon_\Delta > 0$ decreases as Δ increases.

As in the previous section, the starting point of our enhanced algorithm is the following semidefinite relaxation of the Max-Cut problem on a given graph $G = (V, E)$.

(SDP-Cut)	Maximize	$\sum_{e_{ij} \in E} \frac{1 - v_i v_j}{2}$	
	subject to:		
	(1)	$v_i \in S_n$	for $1 \leq i \leq n$
	(2)	$v_i v_j + v_i v_k + v_j v_k \geq -1$	
		$v_i v_j - v_i v_k - v_j v_k \geq -1$	for all $i, j, k \in [n]$

Note that constraint *(3)* used in the previous section is no longer valid.

As earlier, solving *(SDP-Cut)* on a given graph G, we obtain a set of n vectors corresponding to the vertices of G. We are interested in the case in which for all edges e_{ij} in G the angle between the corresponding vectors v_i and v_j is $\theta_0 = 2.3311$, for in this case the random hyperplane technique will yield a partition (X, Y) of expected weight exactly α times the weight of the optimal partition in G. [1]

In this *worst* case, we will show that there is some probability (depending on Δ alone) that a vertex and more than half its neighbors lie on the same side of the partition obtained by the random hyperplane rounding technique. As earlier, we denote such vertices as misplaced, and by adding an additional step which moves misplaced vertices from one side of the partition to the other, we are able to achieve an improved approximation ratio.

Consider the following algorithm for the Max-Cut problem on graphs with bounded degree Δ.

Algorithm A_{Cut} **:**

[1] As noted earlier, there are other worst cases for the [GW95] algorithm, namely when for some edges e_{ij} the angles between the corresponding vectors is zero. In [Zwi99] it is shown how to deal with these cases by the use of an improved rounding technique which uses *outward rotations*. We deal with such cases in a different manner which will be explained in the proof of Theorem 4.7.

1. Solve the semidefinite relaxation *(SDP-Cut)* and round the resulting optimal vector configuration using the random hyperplane rounding technique. Denote the partition obtained by (X, Y).

2. Move misplaced vertices in (X, Y) from one side of the partition to another in aim to increase the value of the partition, until none such vertices are left.

As mentioned above, in order to prove that A_{Cut} has an improved approximation ratio, one must prove that given an optimal vector configuration $\{v_1 \dots v_n\}$ in which for each edge e_{ij} the angle between the vectors v_i and v_j is θ_0, there is some non-negligible probability that any vertex v_i will be misplaced after the rounding mentioned above. We are able to prove such a claim using the triangle constraints appearing in relaxation *(SDP-Cut)*.

Consider a vertex v_i and all its neighbors $\{y_1 \dots y_\Delta\}$. If the angle between v_i and y_j is θ_0 for all $j \in [\Delta]$, then constraint *(2)* of *(SDP-Cut)* above implies that the angle between y_j and $y_{j'}$ for any couple $j, j' \in [\Delta]$ is at most 1.184 (significantly smaller than $\pi/2$). We are interested in the probability that v_i along with more than half its neighbors end up on the same side of a random hyperplane. Let $\{\hat{y}_1 \dots \hat{y}_\Delta\}$ be the projection of the vectors $\{y_1 \dots y_\Delta\}$ on the subspace orthogonal to v_i. Roughly speaking, the above probability can be bounded by the probability that a random hyperplane containing v_i partitions the vectors $\{\hat{y}_1 \dots \hat{y}_\Delta\}$ in an unbalanced manner (*i.e.* more than half of the vectors $\hat{y}_j$ lie on the same side of the random hyperplane). As the angle between y_j and $y_{j'}$ is bounded by 1.184, it can be seen that the angle between $\hat{y}_j$ and $\hat{y}_{j'}$ is bounded by 1.763 (a bit above $\pi/2$). Hence we are interested in the probability that a set of Δ vectors $\{\hat{y}_1 \dots \hat{y}_\Delta\}$ which are relatively *close* to each other (*i.e.* the angle between each pair $\hat{y}_j$ and $\hat{y}_{j'}$ is bounded) are separated in an unbalanced manner by a random hyperplane. Intuitively, this probability seems non-negligible (for instance consider the related case in which the angle between each pair $\hat{y}_j$ and $\hat{y}_{j'}$ is $\pi/2$ or less, in such a configuration it can be seen that with probability at least $2^{-\Delta}$ **all** vectors $\hat{y}_j$ will lie on the **same** side of a random hyperplane).

In the following we present a few technical lemmas which confirm this intuition. Let $\{v_1 \dots v_n\}$ be the vector configuration obtained after solving the semidefinite relaxation *(SDP-Cut)* on G. Let (X, Y) be the partition received after rounding this vector configuration using the random hyperplane rounding technique ([GW95]). Define an edge e_{ij} to be *bad* if the inner product (v_i, v_j) is equal to -0.688 (*i.e.* $\theta_{ij} = \theta_0$). Define a vertex i to be *all bad* if for all neighbors j of i we have that the edge e_{ij} is bad. Denote by $N(i)$ the set of vertices adjacent to i.

Lemma 4.5 *Let v be some vector in R^n of norm $\|v\|$, and let $r = (r_1, \dots, r_n) \in R^n$ be the normal of a random hyperplane in R^n (i.e. each r_i has normal distribution $N(0,1)$). The size of the projection of v on r can be bounded as follows :*

$$\Pr[|(v, r)| \geq \delta\|v\|] = \Pr[|r_1| \geq \delta] \geq 1 - \delta.$$

$$\Pr[(v, r) \in [0, \delta]\|v\|] = \Pr[r_1 \in [0, \delta]] \in [\delta/8, \delta/2].$$

Where $\delta \in [0, 1]$.

Proof : As the distribution of r is spherically symmetric ([Ren70] Theorem **IV**.16.3), we may assume that v is the vector $(\|v\|, 0, \dots, 0)$, thus (v, r) is exactly $r_1\|v\|$. $\square$

Lemma 4.6 *Let (X, Y) be the partition obtained after the random hyperplane rounding technique. Let i be an* **all bad** *vertex of degree d. With probability at least δ it is the case that* **more** *that half of the vertices in $N(i)$ lie on the same side of the partition (X, Y) as the vertex i, where $\delta = \delta_d \geq \frac{1}{2^{19}d^2}$.*

Proof : Let $N(i) = \{u_1 \dots u_d\}$ and denote the vector v_i corresponding to the vertex i as v. As the vectors $v, \{u_1 \dots u_d\}$ lie in a common $d+1$ dimensional space, the following representation of $v, \{u_1 \dots u_d\}$ may be assumed without loss of generality. Let $v = (1, 0, \dots, 0) \in R^{d+1}$ and $u_j = (\alpha_j, \beta_j, \gamma_j)$ for $j \in [d]$. Where $\alpha_j \in R$, $\beta_j \in R$, $\gamma_j \in R^{d-1}$, and α_j, β_j, γ_j are vectors in mutually orthogonal subspaces. The vertex i is **all bad**, thus for all $j \in [d]$ we have that the inner product $(v, u_j) = \alpha_j$ equals -0.688, and that the vector $(\beta_j, \gamma_j) \in R^d$ is of norm 0.725. Furthermore, we may assume for the vector u_1 that $\beta_1 = 0.725$ and $\gamma_1 = 0$.

For $i, j \in [d]$ the following constraint appears in our semidefinite program :

$$(v, u_i) + (v, u_j) + (u_i, u_j) \geq -1.$$

We conclude that $(u_i, u_j) = \alpha_i\alpha_j + \beta_i\beta_j + (\gamma_i, \gamma_j) \geq 0.376$. Hence $\beta_i\beta_j + (\gamma_i, \gamma_j) \geq -0.097$ meaning that the angle between the vectors $(\beta_i, \gamma_i), (\beta_j, \gamma_j) \in R^d$ is at most 1.763. Note that the vectors (β_j, γ_j) are the projection of u_i to the subspace orthogonal to v and are norm 0.725.

Let $r = (r_1 \dots r_{d+1})$ be a $d+1$ dimensional random variable representing the normal to a random hyperplane, *i.e* every r_i is an independent standard normal random variable. As above, denote r by $(\alpha_r, \beta_r, \gamma_r)$ where $\alpha_r = r_1$, $\beta_r = r_2$, and $\gamma_r = (r_3, \dots, r_{d+1})$.

In general we intend to prove that with some probability δ_d over r, the vector v and more than half its neighbors lie on the same side of the hyperplane corresponding to r. Let δ_1 be the probability that the vector v lies very *close* but *above* the random hyperplane corresponding to r, *i.e.* the inner product of (v, r) is small but positive. Note that this probability is dependent on r_1 alone, and corresponds to a positive r_1 of low magnitude. Conditioning on such r (*i.e* r_1), we are interested in the probability that more than half of the vectors in $N(i)$ also lie above the hyperplane corresponding to r. As r_1 is small, this is roughly the probability that more than half of the projected vectors $(\beta_j, \gamma_j) \in R^d$ for $j \in [d]$ will be above the random hyperplane corresponding to the random vector (r_2, r_3).

If the number of vectors in $N(i)$ is odd (*i.e.* d is odd), then the above will happen with probability 1/2, and we may conclude that δ_d is roughly $\delta_1/2$.

Otherwise we condition on an additional projection. Let δ_2 be the probability that the vector u_1 is also very close but above a random hyperplane corresponding to a random vector r. As before this probability is dependent on r_1 and r_2 alone (recall that $u_1 = (\alpha_1, \beta_1, 0)$). Now conditioning the random vector r on r_1 and r_2 (in order to assure that both v and u_1 lie close to and above the random hyperplane corresponding to r), we are interested in the probability that more than half of the remaining vectors in $N(i)$ also lie above the hyperplane corresponding to r. As above, due to the fact that r_1 and r_2 are of small magnitude this probability is roughly the probability that more than half of the projected vectors $\gamma_j \in R^{d-1}$ for $j \in \{2 \dots d\}$ will be above the random hyperplane corresponding to the random vector r_3. We are now in a case similar to the case in which d is odd, and conclude that δ_d is roughly $\delta_1\delta_2/2$. Detailed proof follows.

By Lemma 4.5 we have that with probability at least $\delta/16$ (over $\alpha_r \in R$) it is the case that $(v,r) = \alpha_r \in [\delta/8, \delta]$, and with probability at least $\delta/2$ (over $\beta_r \in R$) we have that $\beta_r \in [\delta, 8\delta]$ (we assume that $\delta < 1/8$). We conclude that with probability at least $\frac{\delta^2}{2^5}$ over the random vector r we have that both v and u_1 lie above the hyperplane corresponding to r.

Furthermore, considering a vector γ_j, we conclude using Lemma 4.5 that with probability at least $1 - \frac{1}{4d}$ (over γ_r) the value of $|(\gamma_j, \gamma_r)|$ is at least $\frac{1}{4d}\|\gamma_j\|$. Hence with probability at least $3/4$ (over γ_j) we have for all vectors γ_j that $|(\gamma_j, \gamma_r)| \geq \frac{1}{4d}\|\gamma_j\|$.

Finally, consider the probability over γ_r that **more** than half the inner products (γ_j, γ_r) are non-negative ($j \in [d]$). As described above, this probability is at least half, due to the fact that for any γ_r we have $(\gamma_1, \gamma_r) = 0$. We conclude, that with probability $1/4$ over γ_r, we are in the case in which $(\gamma_r, \gamma_j) \geq \frac{1}{4d}\|\gamma_j\|$ for **more** than half of the vectors γ_j. Denote the set of these vectors as $\hat{N}(i)$. Note that $u_1 \in \hat{N}(i)$.

Let r be a random vector $r = (\alpha_r, \beta_r, \gamma_r)$ with $\alpha_r,\ \beta_r,\ \gamma_r$ as above (this happens with probability at least $\frac{\delta^2}{2^7}$). We now claim that choosing an appropriate δ, we obtain for all vectors u_j in $\hat{N}(i)$ that the inner product $(u_j, r) > 0$, as $|\hat{N}(i)| > d/2$ this completes the proof of our lemma.

Let $u_j = (\alpha_j, \beta_j, \gamma_j)$ be some vector in $\hat{N}(i)$. If $\beta_j \geq 0.7$ then we have that

$$(u_j, r) = \alpha_j\alpha_r + \beta_j\beta_r + (\gamma_j, \gamma_r) \geq (\beta_j - 0.688)\delta + \frac{1}{4d}\|\gamma_j\| > 0.$$

Otherwise assume that for $u_j \in \hat{N}(i)$ we have $\beta_j < 0.7$. As we have shown earlier, the norm of (β_j, γ_j) is 0.725 for all vectors in $N(i)$, thus we conclude that in this case $\|\gamma_j\|$ must be at least $1/8$. Furthermore, for any vector $u_j \in N(i)$ the inner product $(u_1, u_j) = \alpha_1\alpha_j + \beta_1\beta_j$ is at least 0.376, thus implying that β_j must be at least -0.14 (recall that $u_1 = (\alpha_1, \beta_1, 0)$). We conclude that

$$(u_j, r) = \alpha_j\alpha_r + \beta_j\beta_r + (\gamma_j, \gamma_r) \geq (-1.12 - 0.688)\delta + \frac{1}{4d}\|\gamma_j\| > \frac{1}{2^5 d} - 2\delta.$$

Setting δ to be less than $\frac{1}{2^6 d}$ the above inner product is positive.

All in all, the proof of our lemma is complete with $\delta_d \geq \frac{1}{2^{19}d^2}$. □

Theorem 4.7 (Sketch) *There exists a semidefinite based algorithm that for every $\Delta > 0$ approximates the Max-Cut problem on graphs with bounded degree Δ within an expected ratio of $\alpha + \varepsilon_\Delta$ where $\varepsilon_\Delta = \frac{1}{2^{33}\Delta^4}$.*

Proof : Consider the optimal vector configuration $\{v_1 \dots v_n\}$, and the partition (X, Y) achieved after step *(1)* of algorithm A_{Cut}. Let Z be the value of the above optimal vector configuration, and $w(X)$ the weight of the partition (X, Y). Using the analysis of [GW95] it can be seen that the expected value of $w(X)$ is at least αZ where $\alpha = 0.87856$. Define an edge e_{ij} to be *bad* if the inner product (v_i, v_j) is *close* -0.688, *i.e.* $(v_i, v_j) \in [-0.688 - 0.01, -0.688 + 0.01]$.

We start by assuming that a $(1 - \frac{1}{2\Delta})$ fraction of edges in E are bad. As each edge that is *good* (*i.e.* not bad) may affect at most two vertices, we conclude that at most $|E|/\Delta$ vertices are **not** all bad. Hence at least $n - |E|/\Delta \geq n/2$ vertices are all bad. Moving an all bad vertex from one side of the partition to another has an expected contribution of δ edges to the cut. Where by Lemma 4.6 we have that $\delta \geq \delta_\Delta = \frac{1}{2^{19}\Delta^2}$. Note that in Lemma 4.6 we dealt with the case in which e_{ij} was assumed to be bad if (v_i, v_j) was exactly -0.688. As our analysis in Lemma 4.6 was slack, it can be seen that we obtain the same results for the above definition of a bad edge as well. As each vertex moved may affect at most Δ other vertices, we conclude that the additional step of our algorithm has an expected contribution of $\frac{n}{2^{20}\Delta^3}$ edges, which is at least $\frac{Opt(G)}{2^{19}\Delta^4}$. In such a case we achieve an approximation ratio of at least $\alpha + \frac{1}{2^{19}\Delta^4}$.

Otherwise there are at least $\frac{|E|}{2\Delta}$ edges e_{ij} such that $v_i v_j \notin [-0.688 - \varepsilon_1, -0.688 + \varepsilon_1]$. In general, using the analysis of [GW95] we conclude that the expected approximation ratio on each of these edges is strictly greater than α by at least $\frac{1}{2^{12}}$.

Let Z_{good} be the contribution of these edges to the value Z defined above, and ε be some small positive constant. If Z_{good} is greater than εZ we have that

$$E[w(X)] \geq \alpha(Z - Z_{good}) + \left(\alpha + \frac{1}{2^{12}}\right) Z_{good} \geq \left(\alpha + \frac{\varepsilon}{2^{12}}\right) Z.$$

We are left with the case in which the contribution of the good edges, Z_{good}, is less that εZ. In such a case neglecting these edges and running step *(2)* of A_{Cut} on the original graph G without the set of good edges, we are able to achieve a cut of value

$$\alpha\left(1 + \frac{1}{2^{19}\Delta^4}\right)(1 - \varepsilon)Z$$

Setting ε to be $\frac{1}{2^{21}\Delta^4}$ our proof is complete. □

Acknowledgements

The first author is the Incumbent of the Joseph and Celia Reskin Career Development Chair. This research was supported in part by a Minerva grant, and by DFG grant 673/4-1, Esprit BR grants 7079, 21726, and EC-US 030, and by the Max-Planck Research Prize.

References

[BK98] P. Berman and M. Karpinski. On some tighter inapproximability results, further improvements. *ECCC*, TR98-065, 1998. Extended abstract appears in *ICALP* 1999, pages 200-209.

[BM86] F. Barahona and A.R. Mahjoub. On the cut polytope. *Mathematical Programming*, 36:157–173, 1986.

[FG95] U. Feige and M.X. Goemans. Approximating the value of two prover proof systems with applications to Max-2-Sat and Max-Dicut. *Proceedings of th 3rd IEEE Israel Symposium on Theory of Computing and Systems*, pages 182–189, 1995.

[GW94] M.X. Goemans and D.P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7(4):656–666, 1994.

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

[Hås99] J. Håstad. On approximating CSP-B. *ECCC*, TR99-039, 1999.

[KZ97] B. Karloff and U. Zwick. A 7/8-approximation algorithm for Max-3-Sat? *In Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 406–415, 1997.

[Ren70] A. Renyi. Probability theory. *Elsevier, New York*, 1970.

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