Approximation of Max-Cut on Graphs of Bounded Degree

MIKAEL FLORÉN
Approximation of Max-Cut on Graphs of Bounded Degree

Approximation av Max-Cut i grafer med begränsat gradtal

MIKAEL FLORÉN

Degree Project in Computer Science, Second Cycle - DA225X
Supervisor: Per Austrin
Examiner: Johan Håstad
Email: mfloren@kth.se

April 12, 2016
The Max-Cut problem is a well-known NP-hard problem, for which numerous approximation algorithms have been developed over the years. In this thesis, we examine the special case where the degree of vertices in the graph is bounded. With minor modifications to existing algorithms, we are able to obtain an improved approximation ratio for general bounded-degree graphs. Furthermore we show additional improvements for graphs with at least a constant fraction of odd-degree vertices. We also identify some other possible areas for improvement in the general bounded-degree case.
Referat

## Contents

1 Introduction 1
  1.1 NP-hard problems 1
  1.2 The Max-Cut problem 2
  1.3 Thesis objective 3

2 Background 5
  2.1 Linear Programming 5
  2.2 Semidefinite programming 6
  2.3 Goemans-Williamson algorithm 7
    2.3.1 Algorithm description 7
    2.3.2 Algorithm analysis 9
  2.4 Specialization for graphs of bounded degree 11
    2.4.1 Modified algorithm 11
    2.4.2 Algorithm analysis 12

3 Results 13
  3.1 Summary of results 13
  3.2 Bounded-degree improvement 14
    3.2.1 Notation 14
    3.2.2 Candidate Vertices 14
    3.2.3 Updated Analysis 18
    3.2.4 Approximation Ratio 18
  3.3 Vertices of odd degree 19
    3.3.1 Probability for odd candidate vertex 20
    3.3.2 Ratio of odd-degree vertices 21
    3.3.3 Approximation ratio 22
  3.4 Assuming unbalance around all bad vertices 24
    3.4.1 Utilizing the conjecture 24
    3.4.2 Motivation for the conjecture 26
    3.4.3 An approach for proving the conjecture 28
    3.4.4 Handling balanced pairs 29

4 Discussion 33
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Conclusions</td>
<td>33</td>
</tr>
<tr>
<td>4.2</td>
<td>Future work</td>
<td>33</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Probability of unbalanced neighbors</td>
<td>33</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Utilizing balanced pairs</td>
<td>34</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Using Hierarchies</td>
<td>34</td>
</tr>
</tbody>
</table>

Bibliography 37
Chapter 1

Introduction

1.1 NP-hard problems

Within computer science, there are problems of various difficulties. A common distinction is whether a problem can be solved in polynomial time (often referred to as being efficiently solvable). Efficiently solvable decision problems (i.e. problems with a yes-or-no answer) belong to the class P. Similarly, the class NP contains those decision problems where a solution can be verified efficiently. We know that P ⊆ NP, but it is not known whether the two are equal. In fact, it is widely believed that P ̸= NP, which is used as a working hypothesis.

Based on the above, NP-hard problems are those which are at least as hard as the hardest problems in NP. Formally, we say that a problem X is NP-hard if every problem in NP can be reduced to an instance of X in polynomial time. In other words, a solver for X can be used to solve any problem in NP with similar efficiency as the original solver. Furthermore, NP-complete problems are those that are both in NP and NP-hard, making them the hardest problems in NP. It is worth noting that NP-hard problems are not limited to decision problems, which means the definition extends to other natural problem formulations like optimization and search problems. The fact that there are no known efficient algorithms for NP-hard problems, despite many years of research, provides some anecdotal evidence for the idea that P ̸= NP.

Since NP-hard problems probably cannot be solved efficiently, it is interesting to look at polynomial-time approximation algorithms for such problems. This is motivated by the possibility of finding a “reasonably good” solution in time that is hopefully much smaller than an exact solution would require. For maximization problems, an algorithm that guarantees a result that is at least a factor ρ of the optimal solution is known as a ρ-approximation algorithm. In the case of randomized algorithms, as is the case in this thesis, the same term can be used to refer to algorithms finding
solutions with an expected value that is at least $\rho$ times the optimal value.

\section{1.2 The Max-Cut problem}

The Max-Cut problem is one of Karp’s original NP-complete problems. Given an undirected graph with unweighted edges, the objective is to find a set $S$ of vertices that maximizes the number of edges between $S$ and $\overline{S}$ (i.e. the cut edges). An example problem instance is shown in Figure 1.1.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{max-cut.png}
\caption{Illustration of a Max-Cut problem instance (left) and its solution (right). A filled circle corresponds to a vertex in $S$, while an empty circle corresponds to those in $\overline{S}$. The dashed edges are those that are cut by this assignment.}
\end{figure}

For Max-Cut, the best known polynomial-time approximation algorithm is due to Goemans and Williamson (1995), and achieves an expected approximation ratio of $\alpha_{GW} \approx 0.878$. This ratio is optimal assuming the so called unique games conjecture\footnote{First formulated by Khot (2002). Similarly to how the $P = NP$ conjecture implies the difficulty of obtaining certain exact solutions, this conjecture implies the difficulty of obtaining certain approximation ratios.} but improvements are possible for special instances of the problem. Feige, Karpinski, and Langberg (2002) show that for a graph of maximum degree $\Delta$, the algorithm can be improved to yield a $\alpha_{GW} + \epsilon_{\Delta}$ approximation ratio, where $\epsilon_{\Delta} = \frac{1}{2^{1/2}\Delta}$. Both of these algorithms will be discussed in greater depth in the next section.

It is interesting to look at non-approximability results, to see what results we could hope for at best. The randomized reduction described by Trevisan (2001) can be used to show that it is not possible to efficiently obtain an approximation ratio better than $\alpha_{GW} + O(\frac{1}{\sqrt{\Delta}})$ (assuming the unique games conjecture). In other words, fairly large improvements may be possible.

As this is an NP-hard problem, improvements here have some inherent theoretical relevance. Not only are improvements for the Max-Cut problem itself interesting, it may also be possible to generalize those improvements for usage with other problems. For instance, problems such as Max 3SAT and Max Bisection are in some
1.3. THESIS OBJECTIVE

ways similar to the Max-Cut problem, and may benefit from improvements here. There are also practical applications in circuit layout design and statistical physics (Goemans and Williamson, 1995).

1.3 Thesis objective

The goal of this thesis is to investigate whether further improvements can be made to the approximation ratio for graphs with bounded degree, either by improving the algorithm itself, or through a better analysis.
Chapter 2

Background

2.1 Linear Programming

Linear programming is a technique that can be applied to various optimization problems. First, consider the linear equation systems that can be solved by Gaussian elimination. That is, we are given a matrix $A$ and a vector $b$, and wish to find another vector $x$ such that $Ax = b$. In linear programming, this equation is replaced by the inequality $Ax \geq b$ (where the inequality is applied componentwise). The constraint $x \geq 0$ (which is just a special case of $Ax \geq b$) may be added as well.

These *constraints* define a region of possible solutions. Within this region, we wish to minimize a linear combination of the components of $x$. This linear combination can be written as $c^T x$, referred to as the *objective function*, for some coefficient vector $c$. Both of these concepts have relatively simple geometrical interpretations, as shown in Figure 2.1. Using these definitions, Kleinberg and Tardos (2006) give the following formulation for the standard form of a Linear Programming problem:

Given an $m \times n$ matrix $A$, and vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$, find a vector $x \in \mathbb{R}^n$ to solve the following optimization problem:

$$\min(c^T x \text{ such that } x \geq 0; \ Ax \geq b).$$

Given a problem of this type, there are algorithms that give a solution in polynomial time. This is useful not only because it gives a way to solve certain problems efficiently, but it can also be used as a tool for approximation algorithms.
CHAPTER 2. BACKGROUND

Figure 2.1: Illustration of a 2-dimensional linear programming problem. The grey area is the feasible region of solutions. The objective function is illustrated using the vector $-c$, which points in the direction of the optimal solution.

2.2 Semidefinite programming

The algorithms in the following sections are based on semidefinite programming techniques. A semidefinite program (SDP) is a kind of generalisation of a linear program, which works with matrices rather than vectors.

A core concept is that of a positive semidefinite matrix. A symmetric $n$-by-$n$ matrix $X$ is called positive semi-definite if $v^T X v \geq 0$ for all $v \in \mathbb{R}^n$. This constraint on a semidefinite program essentially replaces the non-negativity constraint of a linear program, and is denoted $X \succeq 0$. There are other ways to state this requirement (see e.g. Lancaster and Tismenetsky 1985):

**Theorem 2.2.1.** Given a symmetric $n$-by-$n$ matrix $X$, the following are equivalent:

1. $X$ is positive semidefinite
2. All eigenvalues of $X$ are non-negative
3. There exists a $m$-by-$n$ (where $m \leq n$) matrix $B$ such that $B^T B = X$

Then, we introduce the following notation for a linear function $C$ of a matrix $X$:
2.3. GOEMANS-WILLIAMSON ALGORITHM

\[ C \cdot X = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij} \]

Using this, we can write a semidefinite program on the following form:

\[
\begin{align*}
\text{Minimize} & \quad C \cdot X \\
\text{subject to:} & \quad A_i \cdot X \geq b_i, \quad i = \{1, \ldots, m\} \\
& \quad X \succeq 0
\end{align*}
\]

Similar to a linear program, we have an objective function that we wish to minimize, defined by the matrix \( C \), as well as a number of linear constraints set by the matrices \( A_i \).

Klerk (2002) describes some complexity results for SDPs. For one, it is not always possible to find an optimal solution to semidefinite programs in polynomial time. However, there are algorithms that can find a solution within \( \epsilon > 0 \) additive error of the optimal solution in time that is polynomial in input size and \( \log(1/\epsilon) \). In the context of efficient approximation algorithms, this is good enough, as we can choose a value of \( \epsilon \) that is small enough for our purposes while still only taking polynomial time to obtain our solution.

2.3 Goemans-Williamson algorithm

The best known approximation algorithm for the general Max-Cut problem is the one presented by Goemans and Williamson (1995). The algorithm utilises semidefinite programming combined with a random hyperplane rounding technique to partition the graph, obtaining an expected approximation ratio of \( \alpha_{GW} \approx 0.878 \).

2.3.1 Algorithm description

To understand this algorithm, it is useful to first look at the Max-Cut problem formulated as an integer quadratic program. Here, we have a graph with the the vertex set \( V = \{1, \ldots, n\} \). Given this, the following program gives the weight of the maximum cut:
Maximize $\frac{1}{2} \sum_{(i,j) \in E} (1 - y_i y_j)$

subject to: $y_i \in \{-1, 1\}$ \quad \forall i \in V

Since this is exactly the Max Cut problem, solving it is still NP-hard, and we need to use some relaxed version of the problem to obtain a polynomial running time. The solution used here is to allow the values $y_i$ to be multidimensional vectors $v_i \in S_n$ (where $S_n$ is the $n$-dimensional unit sphere). In doing this, we also need to modify the objective function in a way that is equivalent to the above formulation for vectors in a single dimension. A natural way to do this is to change the product to a dot product, which gives the following formulation:

Maximize $\frac{1}{2} \sum_{(i,j) \in E} (1 - v_i \cdot v_j)$

subject to: $v_i \in S_n$ \quad \forall i \in V

We then need to show that this formulation is equivalent to a semidefinite program. Using the decomposition $A = B^T B$ mentioned in the previous section, we can simply let each $v_i$ correspond to the $i$th column of $B$. This way, $a_{ij} = v_i \cdot v_j$. Thus, the matrix $A$ being symmetric corresponds to the fact that $v_i \cdot v_j = v_j \cdot v_i$. Additionally, we need the restriction that $a_{ii} = 1$ to ensure that each vector $v_i$ is a unit vector. With this restriction, the following SDP can be formulated:

Maximize $\frac{1}{2} \sum_{(i,j) \in E} (1 - a_{ij})$

subject to: $a_{ii} = 1$ \quad \forall i \in V$

$A \succeq 0$

Solving this problem, we obtain a set of vectors $v_i$ corresponding to the set of vertices in the original graph. Now, we want to use these vectors to determine how to create a partition of the vertices into two sets. The solution is quite simple: we begin by choosing a random vector $r$, uniformly distributed on $S_n$. Finally, we let one side of the partition be $S = \{i | v_i \cdot r \geq 0\}$ (and the other being $\overline{S}$). This is equivalent to choosing a random hyperplane, and partitioning the vectors according to which ones lie “above” and “below” the plane.
2.3. GOEMANS-WILLIAMSON ALGORITHM

2.3.2 Algorithm analysis

We begin by presenting the following lemma regarding the probability of any given edge being cut:

**Lemma 2.3.1.** The probability of an edge \((i, j)\) being cut by the hyperplane rounding is proportional to the angle between the corresponding vectors, and can be explicitly written as:

\[
\Pr[\text{sgn}(v_i \cdot r) \neq \text{sgn}(v_j \cdot r)] = \frac{\arccos(v_i \cdot v_j)}{\pi}
\]

![Figure 2.2: Illustration of the vectors and projected hyperplane.](image)

**Proof.** Picking any edge \((i, j)\), we can consider the plane containing both the vectors \(v_i\) and \(v_j\) (illustrated in Figure 2.2). The edge will be cut precisely when the projection of the hyperplane onto this plane is between the two vectors. Due to the spherical symmetry of the distribution of the vector \(r\), we know that the angle of its projection onto any plane will be uniformly distributed across a circle. Thus, the projection of the hyperplane it defines will also have an angle uniformly distributed over \([0, 2\pi]\). Due to symmetry, it suffices to examine cases where \(\text{sgn}(v_i \cdot r) = 1\). This means that if the angle between the vectors is \(\theta_{ij}\), the probability for the cut is \(\theta_{ij}/\pi\). In terms of \(v_i\) and \(v_j\), we have:

\[
\Pr[\text{sgn}(v_i \cdot r) \neq \text{sgn}(v_j \cdot r)] = \frac{\arccos(v_i \cdot v_j)}{\pi}
\]

**Theorem 2.3.2.** There exists a semidefinite based algorithm that approximates the Max-Cut problem within an expected ratio of at least \(\alpha_{GW} = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos \theta}\).
Proof. The expected weight $\mathbb{E}[W]$ of the cut obtained by the algorithm is simply the sum of the expected contributions of each edge. The contribution of an edge $(i, j)$ is equal to the probability that it is cut, and we can write the sum as follows:

$$
\mathbb{E}[W] = \sum_{i<j} \cdot \Pr[\text{sgn}(v_i \cdot r) \neq \text{sgn}(v_j \cdot r)]
$$

(2.1)

where $\text{sgn}(x) = 1$ if $x \geq 0$ and $-1$ otherwise. Combining this formulation of the expected weight of the cut with Lemma 2.3.1, we obtain

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

To determine the approximation ratio, we need to compare this value to the optimal cut. Since the SDP used above is a relaxation of the original problem, we know that $\text{SDP}(G) \geq \text{OPT}(G)$. In other words, a $\rho$ such that $\mathbb{E}[W] \geq \rho \text{SDP}(G)$ implies that $\mathbb{E}[W] \geq \rho \text{OPT}(G)$. Thus, the value for $\alpha_{GW}$ can be obtained by analyzing the ratio of the contributions of individual edges in $\mathbb{E}[W]$ and $\text{SDP}(G)$ and finding its minimum value:

Figure 2.3: Plot of the functions being compared. The dashed line $\alpha_{GW} \cdot \text{SDP}(G)$ lies tangent to $\mathbb{E}[W]$ at $v_i \cdot v_j = \rho_\ast = -0.688$ (i.e. $\theta = 2.331$)
2.4. SPECIALIZATION FOR GRAPHS OF BOUNDED DEGREE

\[ \alpha_{ij} = \frac{\arccos(v_i \cdot v_j)}{\pi} \cdot \frac{2}{(1 - v_i \cdot v_j)} = \frac{\theta_{ij}}{\pi} \cdot \frac{2}{1 - \cos \theta} \]

\[ \alpha_{GW} = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos \theta} \approx 0.878567 \]

This value is obtained for \( \theta = \theta_0 \approx 2.331 \) (shown in Figure 2.3). We now know that the contribution of each edge is at least a fraction \( \alpha_{GW} \) of the optimal value in expectation. Using linearity of expectation \(^1\) we can conclude that the sum is also at least a fraction \( \alpha_{GW} \) of the optimal value.

2.4 Specialization for graphs of bounded degree

Feige, Karpinski, and Langberg (2002) build upon this method by adding a local improvement step. This addition is shown to improve the approximation ratio for graphs of bounded degree \( \Delta \), obtaining a ratio of at least \( \alpha_{\Delta} = \alpha_{GW} + \frac{1}{2^{3\pi^2} \Delta^4} \).

2.4.1 Modified algorithm

First, additional constraints of the form (2.2) and (2.3) (known as triangle inequalities) are added to the SDP for all \( i, j, k \in [n] \). This is still a relaxation of the original Max-Cut problem, which will be shown in the next subsection.

\[ v_i \cdot v_j + v_i \cdot v_k + v_j \cdot v_k \geq -1 \quad (2.2) \]

\[ v_i \cdot v_j - v_i \cdot v_k - v_j \cdot v_k \geq -1 \quad (2.3) \]

Once this SDP is solved, the hyperplane rounding technique is used as before. Then, a local improvement step is performed, where misplaced vertices are moved to the other side of the partition. The algorithm thus comes down to the following steps:

Algorithm \( A_{Cut} \):

1. Solve the modified SDP, and round the resulting vectors using the random hyperplane technique.

\(^1\)The expected value of the sum is equal to the sum of the individual expected values, regardless of whether they are independent or not (see e.g. Blom et al. (2005)).
2. Each vertex that is on the same side of the partition as more than half its neighbors is moved to the other side of the partition. This is repeated until no such vertices can be found.

When performing the local improvement step, the order in which the vertices are moved may affect the end result. However, to obtain the approximation ratio mentioned earlier in this section, these improvements can be made in an arbitrary order. Essentially, this is because the degree is bounded, meaning each vertex moved may only affect a limited number of other vertices. Thus, given that some number of improvements are initially possible, this means that at least some constant (depending on $\Delta$) fraction of these will yield an improvement regardless of order.

Validity of triangle inequalities

As mentioned by Feige and Goemans (1995) these triangle inequalities can be shown to be valid based on the fact that they hold for boolean variables $\in \{-1, 1\}$. There are some symmetries in these equations that simplify this proof. In both (2.2) and (2.3), any assignment of $v_i$, $v_j$, $v_k$ will give the same result as its inverse (i.e. $v'_i = -v_i$, etc). In (2.2), we also have the case that swapping the values of any two variables will also give the same result. Thus, (2.2) effectively only has two cases: one where all values are equal, and one where they are not.

<table>
<thead>
<tr>
<th>$v_i$</th>
<th>$v_j$</th>
<th>$v_k$</th>
<th>$v_i \cdot v_j + v_i \cdot v_k + v_j \cdot v_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Table 2.1: Possible values for the left-hand side of (2.2).

As for (2.3), the only difference is that $v_k$ is replaced by $-v_k$. As $v_k \in \{-1, 1\}$, this means that the left-hand side of the equation can take on exactly the same values as in (2.2).

Thus, the smallest possible value in the boolean case is $-1$. The upper limit of 3 is already implied by the variables being unit vectors.

2.4.2 Algorithm analysis

As the analysis of this algorithm is very similar to the one used for our own analysis (which can be found in Section 3.2), we will not cover this analysis here. The interested reader can find the original proof in Feige, Karpinski, and Langberg (2002).
Chapter 3

Results

3.1 Summary of results

In this chapter, we will present and prove several results. The first is an improvement of the additive term of the approximation ratio such that the denominator is cubic in $\Delta$:

**Theorem 3.2.5.** There exists a semidefinite based algorithm that for every $\Delta > 0$ approximates the Max-Cut problem on graphs with bounded degree $\Delta$ within an expected ratio of $\alpha_{GW} + \epsilon_\Delta$ where $\epsilon_\Delta = \frac{1}{2\Delta^3}$.

The second theorem deals with graphs that both have a bounded degree and at least some constant fraction of odd-degree vertices. In this case, the additive term is improved further, with a denominator that is quadratic in $\Delta$:

**Theorem 3.3.4.** Assume the graph $G$ has bounded degree $\Delta$, and some fraction $\omega$ of its vertices are of odd degree. Then there is an SDP-based algorithm that approximates the Max-Cut problem on $G$ with an expected ratio of at least $\alpha_{GW} + \epsilon_\Delta$, where $\epsilon_\Delta = \frac{\omega}{2\Delta^2}$.

Furthermore, in Section 3.4 we examine what happens if it is likely that the partitions around an all bad vertex are of unequal size, even if that vertex has an even degree. To do this, we assume the following conjecture:

**Conjecture 3.4.1.** Assume some set of vectors $v_i$, $i \in [d]$, where for each $i, j \in [d]$ we have $\frac{v_i}{|v_i|} \cdot \frac{v_j}{|v_j|} \geq -0.2$, is partitioned using the random hyperplane rounding technique. Then there is at least a probability $c$ that the two sides of the partition are not of equal size, for some positive constant $c$, which does not depend on $d$. 
We present some arguments for why this conjecture is reasonable, but do not prove it. Utilizing this conjecture, we go on to prove the following theorem, which would allow the additive term to be proportional to $\frac{1}{\Delta}$:

**Theorem 3.4.3** Assuming Conjecture 3.4.1, the following holds.

Given a graph $G$ with bounded degree $\Delta$, there is an SDP-based algorithm that approximates the Max-Cut problem on $G$ with an expected ratio of at least $\alpha_{GW} + \epsilon_{\Delta}$, where $\epsilon_{\Delta} = \frac{c_{23}}{\Delta^2}$.

### 3.2 Bounded-degree improvement

In this section, we will show how some slight alterations to the algorithm yield an improved approximation ratio of $\alpha_{GW} + \frac{1}{2^{3/3}}$. 

#### 3.2.1 Notation

First, we explain some of the notation used in the rest of the section.

A **bad edge** $(i, j)$ is one for which the corresponding inner product $v_i \cdot v_j \in [-0.689, -0.687]$. This interval roughly corresponds to the worst-case scenario of the Goemans-Williamson algorithm, in which $\rho_* = -0.688 \ldots$, which occurs when the angle between those vectors is $\theta_0$.

An **all bad** vertex $i$ is one for which every edge $(i, j)$ is bad.

When discussing the neighborhood of an all bad vertex $i$ of degree $d$, we can limit ourselves to a $(d + 1)$-dimensional space in which the corresponding vectors lie. We represent these vectors as $(\alpha, \beta, \gamma) \in \mathbb{R}^{d+1}$, where $\alpha \in \mathbb{R}$, $\beta \in \mathbb{R}$, and $\gamma \in \mathbb{R}^{d-1}$. Specifically, the vertex in question is represented by $v_i = (1, 0, 0)$. One of its neighbors is $u_1 = (\alpha_1, \beta_1, 0)$ (chosen such that $\beta_1 > 0$), and the rest are $u_j = (\alpha_j, \beta_j, \gamma_j)$. Similarly, we have the hyperplane normal $r = (\alpha_r, \beta_r, \gamma_r)$. Notably, for all $j \in [d]$, $\alpha_j \in [-0.689, -0.687]$ (since $i$ is all bad).

#### 3.2.2 Candidate Vertices

The idea of the local improvement step in Feige, Karpinski, and Langberg (2002) is that there are some misplaced vertices, where moving them will increase the value of the cut, as shown in Figure 3.1.
3.2. BOUNDED-DEGREE IMPROVEMENT

In order to estimate how likely it is that a vertex \( i \) is misplaced, they examine the case where the vector \( v_i \) lies above, but very close to, the random hyperplane. Assume this happens with some probability \( \delta_1 \). Since \( \alpha_r \) is small, the position of the neighbors \( u_j \) relative to the hyperplane will largely be decided by the hyperplane corresponding to \((\beta_r, \gamma_r)\). This means that if the degree of \( i \) is odd, it will end up on the same side of the hyperplane as more than half its neighbors with probability about \( 1/2 \). Otherwise, continue with the case that a neighbor \( u_1 \) is also close to and above the random hyperplane, which we assume happens with some probability \( \delta_2 \).

In that case, we also have the probability \( 1/2 \) for more than half of the remaining vertices to end up on the same side as \( i \). In the latter case, we conclude that the probability of a misplaced vertex is at least roughly \( \delta_1 \delta_2 / 2 \).

Based on this idea, we now define a set of candidate vertices. These are essentially the same vertices that were examined by Feige, Karpinski, and Langberg [2002], except some of the constants involved are different. Later in this section we will show that these are indeed misplaced vertices, and make use of their other properties to prove our approximation ratio.

![Figure 3.1](image-url): A vertex \( i \) that is initially misplaced (left) that is then moved to the other side of the partition (right). This increases the cut by 1.

**Definition 3.2.1.** Let \( i \) be an all bad vertex of degree \( d \), with corresponding vector \( v_i \). Then \( i \) is a \( \delta \)-candidate vertex with respect to \( r \) if the following is true:

1. \( v_i \cdot r = \alpha_r \in [\delta/8, \delta] \)
2. \( \beta_r \in [4\delta, 32\delta] \)
3. \( |\gamma_j \cdot r| > \frac{1}{4\delta} \cdot ||\gamma_j|| \) for all \( j \in [d] \).
4. \( \gamma_j \cdot r \geq 0 \) for more than half \( j \in [d] \).

In short, the first requirement says that \( v_i \) most lie close to and above the hyperplane. The second essentially states that one neighbor should also be close to and above the hyperplane. The third and fourth requirements make sure that the final components of more than half the remaining vectors are non-negative and “large enough”, which is important for the proofs later in this section.
Lemma 3.2.1. Given \( \delta \leq \frac{1}{2\Delta} \), a \( \delta \)-candidate vertex with \( d \) neighbors will have at least \( \lfloor d/2 \rfloor + 1 \) neighbors that lie on the same side of the partition and that are not \( \delta \)-candidate vertices themselves.

Proof. First, we consider the vertices \( j \) for which \( \gamma_j \cdot r \geq 0 \). We want to make sure that the inner products \( u_j \cdot r > \delta \), since that means that all of them lie above the hyperplane, and that none of them are candidate vertices (as they do not fulfill the requirement of \( u_j \cdot r \in [\delta/8, \delta] \)). The rest of this proof follows some of the later steps used for Lemma 4.6 in Feige, Karpinski, and Langberg (2002). We first consider the case where \( \beta_j \geq 0 \):

\[
u_j \cdot r = \alpha_j \alpha_r + \beta_j \beta_r + \gamma_j \cdot \gamma_r \geq (4\beta_j - 0.689)\delta + \frac{1}{4\Delta} ||\gamma_j|| > 2\delta
\]

For the case where \( \beta_j < 0 \), we first show a lower limit on \( \beta_j \), using the triangle inequality (2.2) for the vectors \( v_i, u_1, u_j \):

\[
v_i \cdot u_1 + v_i \cdot u_j + u_1 \cdot u_j \geq -1
\]

\[
\implies u_1 \cdot u_j - 2 \cdot 0.689 \geq -1
\]

\[
\implies \alpha_1 \alpha_j + \beta_1 \beta_j + 0\gamma_j \geq 1 + 1.378
\]

\[
\implies \beta_1 \beta_j \geq 0.378 - 0.689^2 \geq -0.097
\]

\[
\implies \beta_j \geq -0.097/(1 - 0.689^2) \geq -0.14
\]

This means that \( \beta_j \in [-0.14, 0.7] \), which we can use to establish a lower limit on \( ||\gamma_j|| \). Using the fact that \( ||u_j|| = 1 \), we get \( ||\gamma_j||^2 = 1 - \alpha_j^2 - \beta_j^2 \geq 1 - 0.689^2 - 0.7^2 \geq 0.038 > \frac{1}{64} \). This means that \( ||\gamma_j|| > 1/8 \), which use in the following way:

\[
u_j \cdot r = \alpha_j \alpha_r + \beta_j \beta_r + \gamma_j \cdot \gamma_r
\]

\[
\geq (32 \cdot \beta_j - 0.689)\delta + \frac{1}{4\Delta} ||\gamma_j||
\]

\[
\geq (-4.48 - 0.689)\delta + \frac{1}{2\Delta}
\]

\[
> -6\delta + 8\delta = 2\delta
\]

where the last step is obtained by using \( \delta \leq \frac{1}{2\Delta} \). Thus, we can conclude that the vertices for which \( \gamma_j \cdot r \geq 0 \) all lie above the hyperplane, and none of them are candidate vertices. By the fourth requirement on candidate vertices, we know that this is true for more than half the neighbor vertices, which completes the proof. \( \Box \)
3.2. BOUNDED-DEGREE IMPROVEMENT

As the above shows that each candidate vertex improves the size of our cut, we now want to examine the probability of an all bad vertex being a candidate vertex. To do this, we first need the following lemma regarding the projection of a vector $v$ on $r$, found in Feige, Karpinski, and Langberg (2002) (which follows from the fact that the components $r_i$ of the vector defining the hyperplane follow a normal distribution $N(0, 1)$):

**Lemma 3.2.2.** Let $v$ be some vector in $\mathbb{R}^n$ of norm $||v||$, and let $r = (r_1, \ldots, r_n)$ be a $n$-dimensional vector where each $r_i$ has normal distribution $N(0, 1)$. The size of the projection of $v$ on $r$ can be bounded as follows:

$$
\Pr[|v \cdot r| \geq \delta ||v||] = \Pr[|r_1| \geq \delta] \geq 1 - \delta
$$

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

We now move on to the actual probability for candidate vertices:

**Lemma 3.2.3.** Given $\delta \leq \frac{1}{2\Delta}$, an all bad vertex is a $d$-candidate vertex with probability at least $\frac{1}{2\Delta^2}$ over the choice of $r$.

**Proof.** We can consider each of the requirements for a candidate vertex separately. This works since the first two requirements are concerned with variables not involved in any others (i.e. $\alpha_r$ and $\beta_r$). The last two requirements both depend on $\gamma_r$, but they are only concerned with its magnitude and sign respectively, and are thus independent.

1. Using Lemma 3.2.2 we know that $\alpha_r \in [\delta/8, \delta]$ occurs with probability $\geq \frac{\delta}{8\Delta}$.

2. Using the same lemma, $\beta_r \in [4\delta, 32\delta]$ occurs with probability $\geq 2\delta$.

3. By Lemma 3.2.2, we have the probability $1 - \frac{1}{\Delta^2}$ for each $j$ that $|\gamma_j \cdot r| > \frac{1}{\Delta} ||\gamma_j||$. By Boole’s inequality\footnote{Also known as the union bound. This states that for any finite set of events, the probability of at least one event happening is at most equal to the sum of the individual probabilities (see e.g. Blom et al. (2005)).}, the probability of at least one inner product being too small is at most $\sum_{j=1}^d \Pr[|\gamma_j \cdot r| < \frac{1}{\Delta} ||\gamma_j||] \leq d \cdot \frac{1}{\Delta^2} \leq 1/4$. In other words, the absolute values of the inner products are large enough with probability at least $3/4$.

4. Since $\gamma_1 = 0$, we know that $\gamma_1 \cdot r \geq 0$. For the other neighbors, we can simply use the fact that $r$ and $-r$ are equally likely. For any $r$ where at least half the neighbors are below the hyperplane, $-r$ would instead place at least half the neighbors above the hyperplane. Taking into account $u_1$, this means that more than half the neighbors are above the hyperplane. Thus, we have a probability that is at least $1/2$ that $\gamma_j \cdot r \geq 0$ for more than half of the neighbors.
With \( \delta = \frac{1}{2\Delta} \), we have a final probability of at least \( \frac{\delta}{2} \cdot 2\delta \cdot \frac{3}{4} \cdot \frac{1}{2} = \frac{3\delta^2}{2\Delta} \geq \frac{1}{21\Delta^2} \). \( \square \)

### 3.2.3 Updated Analysis

**Lemma 3.2.4.** Let \((X, Y)\) be a partition obtained from the hyperplane rounding technique with \( c \) candidate vertices (with respect to the vector \( r \) defining the hyperplane). Moving each of these candidate vertices to the opposite side of the partition will increase the size of the cut by at least \( c \).

**Proof.** By Lemma 3.2.1 more than half the neighbors of a candidate vertex \( i \) will be vertices on the same side as \( i \) that are not themselves candidate vertices. This means that moving any single candidate vertex to the other side of the partition increases the cut by at least 1 (as we are now cutting more than half the edges, while previously we were cutting less than half the neighbors).

Now, we consider what happens when one or more of the neighbors are candidate vertices themselves. In Feige, Karpinski, and Langberg (2002) this is the source of interference between moved nodes; moving one vertex might make it so that it is no longer profitable to move another. With these candidate vertices, that is no longer a risk as the analyzed benefit of moving a candidate vertex only includes non-candidate vertices\(^2\).

Thus, each of the \( c \) candidate vertices will contribute 1 edge to the cut, for a total increase of \( c \) edges. \( \square \)

### 3.2.4 Approximation Ratio

**Theorem 3.2.5.** There exists a semidefinite based algorithm that for every \( \Delta > 0 \) approximates the Max-Cut problem on graphs with bounded degree \( \Delta \) within an expected ratio of \( \alpha_{GW} + \epsilon_\Delta \) where \( \epsilon_\Delta = \frac{1}{21\Delta^2} \).

**Proof.** First, we solve the SDP and choose a random vector \( r \), where each \( r_i \) has normal distribution \( N(0, 1) \), to define our hyperplane for rounding as in the first step of Algorithm 2.4.1. By Goemans and Williamson (1995), if this SDP solution has value \( Z \), we have an expected cut of \( \alpha_{GW}Z \) after hyperplane rounding.

\(^2\)In fact, due to the pessimism in this analysis, we assume that edges to adjacent candidate vertices were cut before the move, and no longer will be afterwards, resulting in a loss of one edge. Since both candidates will be moved, each edge between candidates will remain the same as it was previously, meaning that any adjacent candidate vertices will give a cut 1 higher than this analysis implies.
3.3. VERTICES OF ODD DEGREE

We first assume that at least a fraction \((1 - \frac{1}{2\Delta})\) of the edges in \(E\) are bad. This means that at most \(\frac{|E|}{\Delta}\) vertices are not all bad, and at least \(n - \frac{|E|}{\Delta} \geq \frac{n}{2}\) vertices are all bad. By Lemma 3.2.3, an all bad vertex has probability at least \(\frac{21}{2\Delta^2}\) to be a candidate vertex. Thus, the expected number of candidate vertices is \(\frac{n}{2\Delta^2} \geq \frac{|E|}{2\Delta^2}\). By Lemma 3.2.4, the cut is increased by at least this value in the local improvement step, giving us an approximation ratio that is at least \((\alpha_{GW} + \frac{1}{2\Delta^2})\).

Now, we examine the case where we have a large number of good (i.e. not bad) edges. By the analysis of Goemans and Williamson (1995), each such edge will in expectation give an approximation ratio that is at least \(\alpha_{GW} + \frac{1}{2}\). If the contribution of these edges to the SDP value is at least \(\epsilon Z\), the expected cut is

\[
\mathbb{E}[W] \geq (\alpha_{GW} + \frac{\epsilon}{2\Delta^2})Z
\]

Otherwise, we can ignore the good edges and perform the local improvement on the remaining subgraph. This gives an expected cut of

\[
\mathbb{E}[W] \geq (\alpha_{GW} + \frac{1}{2\Delta^2})(1 - \epsilon)Z \\
\geq (\alpha_{GW} + \frac{1}{2\Delta^2} - \epsilon)Z
\]

Choosing \(\epsilon = \frac{1}{2\Delta^2}\) gives us an expected approximation ratio of at least \(\alpha_{GW} + \frac{1}{\Delta^2}\).

\[\square\]

3.3 Vertices of odd degree

In the original analysis of Feige, Karpinski, and Langberg (2002), the existence of vertices with even degree is somewhat problematic. The whole argument surrounding the vector \(u_1\) is based on this, which directly adds a factor proportional to \(\frac{1}{\Delta}\) to \(\epsilon_\Delta\) (see the proof of Lemma 3.2.3). This means that a graph with only vertices of odd degree will definitely show some improvement in the approximation ratio. Building on the previous improvements, we will show that we can obtain an additive term \(\epsilon_\text{odd} = \Omega(\frac{1}{\Delta^2})\) for such graphs.
3.3.1 Probability for odd candidate vertex

For this section, we will use odd candidate vertices, which are similar to the candidate vertices of the previous section, but with a slightly different definition:

**Definition 3.3.1.** Let $i$ be an all bad vertex of odd degree $d$, with corresponding vector $v_i$. Then $i$ is an odd candidate vertex with respect to $r$ if the following is true:

1. $v_i \cdot r = \alpha_r \in [\delta/8, \delta]$
2. $|\gamma_j \cdot r| > \frac{1}{4\Delta} \cdot ||\gamma_j||$ for all $j \in [d]$.
3. $\gamma_j \cdot r \geq 0$ for more than half $j \in [d]$.

Since we are guaranteed to never have the same number of neighbors on both sides of the hyperplane, the condition on the value of $\beta_r$ has been removed. This also lets us use the notation $u_j = (\alpha_j, \gamma_j)$ for the neighbors of $i$. Other differences are that we require that the vertex has an odd degree, and that the value of $\delta$ has been altered (with the particular value being motivated in the proof of Lemma 3.3.2.

**Lemma 3.3.1.** Given $\delta \leq \frac{1}{2^3 \Delta}$, an odd-degree all bad vertex is an odd $\delta$-candidate vertex with probability at least $\frac{1}{2^{10} \Delta}$ over the choice of $r$.

**Proof.** The analysis uses the same overall ideas as before. We begin by considering an all bad vertex $i$, and in the same way (using Lemma 3.2.2) as before we have the probability $\frac{\delta}{2^5}$ that $v_i \cdot r \in [\delta/8, \delta]$.

As before, we examine the case where $|\gamma_j \cdot r| \geq \frac{1}{4\Delta} ||\gamma_j||$ for all vectors $u_j$. Despite $\gamma_j$ being one dimension larger, the probability bound given before remains unchanged, meaning this happens with probability at least $\frac{3}{4}$. By the same argument as before, we also have that more than half the products $\gamma_j \cdot r$ are positive with probability at least $\frac{1}{2}$. Taken together, we have a probability $\frac{\delta}{2^5} \geq \frac{1}{2^{10} \Delta}$ of this occurring. 

**Lemma 3.3.2.** Given $\delta \leq \frac{1}{2^3 \Delta}$, an odd $\delta$-candidate vertex with $d$ neighbors will have at least $\frac{d+1}{2}$ neighbors on the same side of the partition and that are not odd candidate vertices themselves.

**Proof.** As in the proof of Lemma 3.2.1, we consider the vertices $j$ for which $\gamma_j \cdot r \geq 0$. Again, the goal is to make sure that $u_j \cdot r > \delta$. Since we are no longer concerned with the $\beta_j$ component, the equation is slightly simpler:

$$|u_j \cdot r| \geq -0.689\delta + \frac{1}{4\Delta} \cdot \sqrt{1 - (-0.689)^2} \geq \frac{1}{4\Delta} \cdot 0.724 - 0.689 \cdot \delta$$
3.3. VERTICES OF ODD DEGREE

By using \( \delta \leq \frac{1}{\sqrt{3}} \), this product is greater than \( \delta \). As before, this means that the vertices for which \( \gamma_j \cdot r \geq 0 \) all lie above the hyperplane, and none of them are odd candidate vertices. We also know that these are more than half the neighbor vertices, which completes the proof.

\[ \square \]

3.3.2 Ratio of odd-degree vertices

In order to use a proof similar to that of Theorem 3.2.5 for the approximation ratio in a graph with a given ratio \( \omega \) of odd-degree vertices, we will need some additional results. Specifically, in the case where good edges are ignored, we want to make sure that we can retain a similar ratio (i.e. a ratio \( \Theta(\omega) \)) of odd-degree vertices.

Lemma 3.3.3. Assume \( G \) is a graph with a fraction \( \omega \) vertices that are of odd degree, and some number of “good” edges. Then it is possible to remove “good” edges in a way that retains a fraction of odd-degree vertices that is at least \( \omega \), with an average of less than 2 “good” edges adjacent to each odd-degree vertex.

Proof. Consider the following algorithm:

**Algorithm** \( A_{\text{remove}} \):

1. While there exists a cycle \( C \) of good edges, remove all edges in \( C \).

2. Iteratively remove each good edge that is adjacent to at least one even-degree vertex, until no such edges are left.

We first look at the number of odd-degree vertices after \( A_{\text{remove}} \) is performed. Step 1 of the algorithm doesn’t change the parity of any vertex (since each vertex involved in a cycle is adjacent to two edges of that cycle), so the number of odd-degree vertices is unchanged. In step 2, removing an edge can never decrease the number of odd-degree vertices, since only two vertices are affected and at least one of them is turned from an even degree to an odd degree. Thus, we can conclude that the fraction of odd-degree vertices is still at least \( \omega \).

Since all cycles of good edges are removed, the graph spanned by the remaining good edges is a forest (i.e. some number of disjoint trees). As we also remove all good edges adjacent to even-degree vertices, we know that every vertex in this forest is of odd degree. Since a tree has an average degree of \( <2 \), we can conclude that the average number of good edges adjacent to each odd-degree vertex is less than 2.

\[ \square \]
3.3.3 Approximation ratio

**Theorem 3.3.4.** Assume the graph \( G \) has bounded degree \( \Delta \), and some fraction \( \omega \) of its vertices are of odd degree. Then there is an SDP-based algorithm that approximates the Max-Cut problem on \( G \) with an expected ratio of at least \( \alpha_{GW} + \epsilon \), where \( \epsilon = \frac{\omega}{2^{10} \Delta^2} \).

**Proof.** As in the proof of Theorem 3.2.5, we solve the SDP and choose a random vector \( r \). Given an optimal vector configuration \( \{v_i\} \) with weight \( Z \), we know by Goemans and Williamson (1995) that the expected value of the cut after the hyperplane rounding is at least \( \alpha_{GW} Z \). As before, a bad edge is one where \( v_i \cdot v_j \in [-0.689, -0.687] \).

We first assume that at least a \( (1 - \frac{\omega}{\Delta}) \) fraction of the edges are bad. This means that at most \( \frac{\omega |E|}{\Delta} \leq \frac{\omega n}{2} \) vertices are not all bad. Thus at least \( \omega n - \frac{\omega n}{2} = \frac{\omega n}{2} \) vertices are all bad and of odd degree. Combining Lemmas 3.3.1 and 3.3.2 each of these have an expected contribution of \( \frac{\omega n}{2^{10} \Delta^2} \) edges in the local improvement step. Thus, the approximation ratio in this case is at least

\[
\alpha_{GW} + \frac{n\omega}{2} \cdot \frac{1}{2^{10} \Delta} \cdot \frac{1}{|E|} \geq \alpha_{GW} + \frac{\omega}{2^{10} \Delta^2} \quad (3.1)
\]

where the inequality follows from \( \frac{n}{|E|} \geq \frac{1}{\Delta} \).

Otherwise, we have at least \( \frac{\omega n}{2} |E| \) edges that are not bad. Again, we use the analysis by Feige, Karpinski, and Langberg (2002), stating that these edges each have an approximation ratio that is at least \( \alpha_{GW} + \frac{\epsilon}{212} \). We also assume that the contribution of these edges towards \( Z \) is at least \( \epsilon Z \) for some small, positive \( \epsilon \). Then we have that

\[
\mathbb{E}[W] \geq (\alpha_{GW} + \frac{\epsilon}{212})Z
\]

The final case is when the good edges contribute less than \( \epsilon Z \). In this case, we can ignore some of those edges when analyzing our improvement step. Note that these edges do not have to be ignored when actually running the algorithm, as doing so can only make the cut worse - this is merely a way of analyzing a worst-case scenario. Doing so will allow us to reuse the proof for the case where many edges are bad. However, we also want to make sure that we don’t change too many odd-degree vertices into ones of even degree when doing so.

This is possible by choosing to first ignore those edges \((i, j)\) where the product \( v_i \cdot v_j \) is less than that of a bad edge (i.e. \( v_i \cdot v_j < -0.689 \)). Doing this, we run the risk of decreasing the ratio of odd degree vertices. Here we will show that the impact
3.3. VERTICES OF ODD DEGREE

of this is guaranteed to be relatively small. For each edge ignored, we know that $v_i \cdot v_j < -0.689$. This means that the contribution from each edge towards $Z$ is $Z_{ij} = \frac{1 - v_i \cdot v_j}{2} \geq 0.844$. However, we also know that the good edges (of which the ignored edges are a subset) contribute less than $\epsilon Z$. The number of ignored edges can thus be at most $\frac{\epsilon Z}{0.844}$. Since each edge affects at most 2 vertices, the number of “lost” odd-degree vertices is no more than

$$2 \cdot \frac{\epsilon Z}{0.844} \leq 3\epsilon |E| \leq \frac{3n\Delta\epsilon}{2} \leq 2n\Delta\epsilon$$

which guarantees that we have at least a fraction $(\omega - 2\Delta\epsilon)$ odd-degree vertices remaining.

The edges where the product is greater than −0.687 do not impact the expected contribution negatively, and do not all have to be ignored. This can be seen by simply considering that a larger inner product implies that the vectors are closer together, which in turn means that they are more likely to be on the same side of the random hyperplane. Thus, whenever such an edge is involved, the probability of more than half the neighbors being on the same side increases.

On the other hand, these edges are not guaranteed to not affect other odd candidate vertices (consider e.g. $v_i \cdot v_j = 1$, where $v_i \cdot r = \delta \Leftrightarrow v_j \cdot r = \delta$). Here, we can use Lemma 3.3.3, letting us ignore most of the remaining good edges. Specifically, we will still have at least a fraction $(\omega - 2\Delta\epsilon)$ odd-degree vertices, with an average of less than 2 good edges adjacent to each of them. Since vertices connected by these good edges may affect each other, each of these “nearly all bad” vertices may affect up to 2 other such vertices, giving a factor $\frac{1}{2 + 1}$ in the improvement term. As before, each odd-degree vertex has an expected contribution of $\delta_{odd}^\Delta \geq \frac{1}{2^{11} \Delta}$ edges in the local improvement step, for a total ratio of at least:

$$(1 - \epsilon)\alpha_{GW} + \frac{1}{|E|} \cdot n(\omega - 2\Delta\epsilon) \cdot \frac{1}{2^{11} \Delta^2} \cdot \frac{1}{3}$$

$$\geq (1 - \epsilon)\alpha_{GW} + (\omega - 2\Delta\epsilon) \cdot \frac{1}{2^{11} \Delta^2}$$

$$\geq \alpha_{GW} + \frac{\omega}{2^{11} \Delta^2} - \epsilon(\alpha_{GW} + \frac{1}{2^{11} \Delta})$$

$$\geq \alpha_{GW} + \frac{\omega}{2^{11} \Delta^2} - \epsilon$$

Choosing $\epsilon = \frac{\omega}{2^{11} \Delta^2}$ completes the proof.

$\square$
3.4 Assuming unbalance around all bad vertices

As mentioned in Section 3.3, vertices of even degree are somewhat problematic. Specifically, the problem is that their neighbors may be partitioned in a balanced manner (i.e. where the two partitions are of equal size). In this section, we explore what it would mean if the partitions are unbalanced with a reasonably large probability. Specifically, we assume the following conjecture:

**Conjecture 3.4.1.** Assume some set of vectors \( v_i, i \in [d] \), where for each \( i,j \in [d] \) we have \( \frac{v_i}{|v_i|} \cdot \frac{v_j}{|v_j|} \geq -0.2 \), is partitioned using the random hyperplane rounding technique. Then there is at least a probability \( c \) that the two sides of the partition are not of equal size, for some positive constant \( c \), which does not depend on \( d \).

There are at least some intuitive reasons to believe that this may be the case. In essence, these are based on the fact that any negative correlation must be relatively small, and that in the completely uncorrelated case we know that an unbalanced partition is likely to occur. In Section 3.4.2 we present both some more detailed arguments based on this intuition, as well as some possible approaches for proving the conjecture.

3.4.1 Utilizing the conjecture

Probability for local improvement

**Lemma 3.4.2.** Assuming Conjecture 3.4.1, the following holds.

Let \( a \) be an all bad vertex of degree \( d \). Then there is a probability of at least \( \mathbb{P}[\text{Unbalance}] \geq \frac{c}{2^{10} \Delta^2} \) that more than half of its neighbor vertices lie on the same side of the random hyperplane used for rounding.

**Proof.** Similar to the proof for Lemma 3.2.3, we use Lemma 3.2.2 to determine that there is a \( \frac{\delta}{2\pi} \) probability that \( \gamma_j = (\gamma_{j1}, \gamma_{j2}) \). Again, we denote the neighboring vectors as \( u_j = (\alpha_j, \gamma_j) \).

In order to apply Conjecture 3.4.1 we need to make sure the inner products \( \gamma_i \cdot \gamma_j \) are large enough (doing this for the full products \( u_i \cdot u_j \) is not sufficient, as they are more likely to be on a different side of the hyperplane than \( a \)). Since \( a \) is all bad (again, an edge is bad when \( v_i \cdot v_j \in [-0.689, -0.687] \)), we also have that \( ||\gamma_j|| \geq \sqrt{1 - 0.687^2} \approx 0.724 \). We then combine this with the triangle inequality:
3.4. ASSUMING UNBALANCE AROUND ALL BAD VERTICES

\[ v_a \cdot u_i + v_a \cdot u_j + u_i \cdot u_j \geq -1 \]

\[ \Rightarrow \quad 2(-0.687) + u_i \cdot u_j \geq -1 \]

\[ \Rightarrow \quad u_i \cdot u_j = \alpha_i \alpha_j + \gamma_i \cdot \gamma_j \geq -1 - 2(-0.687) = 0.374 \]

\[ \Rightarrow \quad \gamma_i \cdot \gamma_j \geq 0.374 - (-0.689)^2 \geq 0.374 - 0.475 \]

\[ \Rightarrow \quad \left| \frac{\gamma_i}{|\gamma_i|} \right| \geq \frac{-0.101}{0.724^2} \geq -0.193 \]

Using the union bound, we see that \(|\gamma_j \cdot \gamma_r| \geq \frac{c}{2\Delta} ||\gamma_j|||\) for all vectors \(u_j\) with probability at least \(1 - \frac{c}{2}\) (since the union bound gives the probability \(\sum_{j=1}^d Pr[|\gamma_j \cdot \gamma_r| < \frac{c}{2\Delta} ||\gamma_j|||] \leq d \cdot \frac{c}{2\Delta} \leq c/2\) for the opposite). As Conjecture 3.4.1 gives a \(c\) probability for an unbalanced partition, we have a probability that is at least \(c \cdot (1 - \frac{c}{2}) \geq c \cdot (1 - \frac{1}{2}) = \frac{c}{2}\) for both these events to occur simultaneously. Due to symmetry, we also have \(\frac{c}{2}\) probability that the larger partition is above the hyperplane. Taken together with the condition on \(r_j\) (as in the proof of Lemma 3.2.3, with a probability of \(\frac{c}{2\pi}\)), we have a total probability of at least \(\frac{c}{2\pi}\).

Now, we want to choose a \(\delta\) such that \(|u_j \cdot r| > \delta\). As in Lemma 3.3.2, this means there is no interference between the vertices considered by our analysis. As before, we have

\[ |u_j \cdot r| \geq -0.689\delta + \frac{1}{4\Delta} ||\gamma_j||| \geq \frac{1}{4\Delta} \cdot 0.724 - 0.689 \cdot \delta \]

and choosing \(\delta \leq \frac{1}{2\pi\Delta}\) the product is greater than \(\delta\).

This means that the probability for an all bad vertex to be on the same side of the hyperplane as more than half of its neighbors is \(\mathbb{P}[\text{Unbalance}] \geq \frac{c}{2\pi\Delta}\).

New approximation ratio

**Theorem 3.4.3.** Assuming Conjecture 3.4.1, the following holds.

Given a graph \(G\) with bounded degree \(\Delta\), there is an SDP-based algorithm that approximates the Max-Cut problem on \(G\) with an expected ratio of at least \(\alpha_{GW} + \epsilon_{\Delta}\), where \(\epsilon_{\Delta} = \frac{c}{2\pi\Delta^2}\).

**Proof.** This proof closely follows that of Theorem 3.2.5, with only minor differences. As such, this description is a bit more terse; refer to the former for more details on some of the steps.
First, assume that at most \(\frac{|E|}{2\Delta}\) edges are not bad, implying that at least \(\frac{n}{2}\) vertices are all bad. By Lemma 3.4.2, each such vertex is expected to contribute at least \(\delta \geq \frac{c}{2\Delta^2}\) edges in the local improvement step. Since the vertices covered by the analysis do not affect each other, the total expected contribution is \(\frac{n}{2} \cdot \frac{c}{2\Delta^2} \geq \frac{c}{2\Delta^2} \text{Opt}\), for a total approximation ratio of

\[
\alpha_{GW} + \frac{c}{210 \Delta^2}
\]

If we have more edges that are not bad, we first consider the case where they contribute more than \(\epsilon Z\) to the objective value \(Z\) of the SDP. Each good edge has an expected approximation ratio that is at least \(\alpha_{GW} + \frac{1}{2}\), giving us a total approximation ratio of at least

\[
\alpha_{GW} + \frac{\epsilon}{212}
\]

If the good edges contributed less than \(\epsilon Z\), we can simply ignore them when running the algorithm, for an expected cut of at least

\[
(\alpha_{GW} + \frac{c}{210 \Delta^2})(1 - \epsilon)Z
\]

Choosing \(\epsilon = \frac{c}{211 \Delta^2}\) completes the proof.

\[
\square
\]

### 3.4.2 Motivation for the conjecture

Intuitively, the conjecture seems reasonable. For a pair of vectors, the bound on \(v_i \cdot v_j\) guarantees there is at least a \(1 - \frac{\arccos(-0.2)}{\pi} \geq 0.435\) probability that the two end up on the same side of the plane. Iteratively adding more vector pairs seems like it should only increase the probability of an unbalanced partition; more pairs increase the chance that at least one pair is unbalanced, and if more pairs are unbalanced it seems unlikely that they’d counteract each other with a very high probability (similar to how individual pairs act).

One can also consider the worst-case scenario if the inner products were not limited. In that case, the vectors could be pairwise antipodal (the relationship between different pairs would be irrelevant), meaning there is a 100% probability that the partitions are of equal size. There is no obvious way of translating this scenario to one where the inner products are limited, as the products between different pairs
3.4. ASSUMING UNBALANCE AROUND ALL BAD VERTICES

Figure 3.2: Measured probability of $d$ vectors being distributed evenly across the two partitions by a random hyperplane.

...can now be relevant. One approach that is easy to analyze is where the different pairs are all uncorrelated:

Assume that all but the last pair have already been partitioned by the random hyperplane. As the vectors are uncorrelated to everything outside the pair, the final pair still has the same constant probabilities for each way of partitioning it. Regardless of how the previous pairs have been partitioned, there is at most one way of achieving partitions of equal size (balanced if the previous vectors were balanced, unbalanced in the opposite direction if they were a single pair off of being balanced). Even if the previous vectors were balanced or near-balanced with probability $\frac{1}{2}$, there would still be at least a constant probability for the final partitions to be unbalanced.

Another candidate for a worst-case scenario is where the vectors form a regular simplex. In other words, the inner product between any two vectors is the same: $-\frac{1}{d-1}$. This essentially means the vectors are as far away as possible from as many others as possible. On the other hand, as $d$ increases, this configuration should more and more behave like a completely uncorrelated one, which would imply a probability of $1 - O(1/\sqrt{d})$ (close to 1 for large values of $d$) for an unbalanced configuration. Similar to the previous argument, this would imply at least a constant probability...
for unbalanced partitions.

To support this notion, we performed random hyperplane rounding both on regular
simplexes and the near-antipodal pairs discussed before (with no correlation between
different pairs), the results of which are shown in Figure 3.2.

3.4.3 An approach for proving the conjecture

In this section, we discuss an approach we attempted, but did not manage to come
to any definite conclusion using. The idea is to investigate the variance (and possibly
higher-degree moments) of the sum

\[ X = \sum x_i, \quad x_i \in \{-1, 1\} \]

where \( x_i \) being positive or negative corresponds to which side of the partition \( i \) is
on. Since \( \mathbb{E}[X] = 0 \) (due to the symmetry of the random hyperplane), the variance
of this sum can be written as

\[
\mathbb{E}[X^2] = \sum_i \sum_j \mathbb{E}[x_i x_j] = d + \sum_{i \neq j} \mathbb{E}[x_i x_j] = d + 2 \sum_{i < j} \mathbb{E}[x_i x_j] = d + 2 \left( \frac{d}{2} \right) \text{avg}_{i \neq j}(\mathbb{E}[x_i x_j]) = d + d(d - 1) \text{avg}_{i \neq j}(\mathbb{E}[x_i x_j]) = d \cdot (1 + (d - 1) \cdot \text{avg}_{i \neq j}(\mathbb{E}[f(v_i \cdot v_j)])) \tag{3.2}
\]

where \( d \) is the number of vertices considered and \( \text{avg}_{i \neq j}(E) \) is the average value of
\( E \) for all \( i, j \in [d] \) such that \( i \neq j \). \( f(v_i \cdot v_j) \) can be obtained from Lemma 2.3.1

\[
\mathbb{E}[x_i x_j] = f(v_i \cdot v_j) = -1 \cdot \Pr[\text{Cut}] + 1 \cdot (1 - \Pr[\text{Cut}]) = 1 - 2 \cdot \mathbb{E}[W] = 1 - \frac{2}{\pi} \arccos(v_i \cdot v_j)
\]

Using the first-degree Taylor expansion of \( f(x) \) and the fact that \( \mathbb{E}[v_i \cdot v_j] \geq \frac{1}{d} \)
we can estimate a lower bound for \( \text{avg}_{i \neq j}(\mathbb{E}[f(v_i \cdot v_j)]) \):

\[ \text{This can be shown by using } f(x) = x \text{ in Equation (3.2), combined with } \mathbb{E}[X^2] \geq 0. \]
3.4. ASSUMING UNBALANCE AROUND ALL BAD VERTICES

\[ E[f(v_i \cdot v_j)] \approx E[\frac{2v_i \cdot v_j}{\pi}] = \frac{2}{\pi} \cdot E[v_i \cdot v_j] \geq \frac{2}{\pi} \cdot \frac{-1}{d-1} \]  

(3.3)

Inserting this into (3.2) gives us:

\[ E[X^2] \approx \Theta(d) \]

However, it turns out that this is not necessarily good enough; for instance, we could have \( X \approx \pm d \) with probability about \( \frac{1}{d} \), and roughly \( 1 - \frac{1}{d} \) probability that the partitions are balanced.

The next step is to examine the fourth degree moment, \( E[X^4] \). In the case from the previous paragraph, we would have \( E[X^4] \approx d^3 \). Essentially, it should be enough to show that \( E[X^4] \) is “significantly” smaller than this. What exactly this means depends on how accurate the approximation (3.3) is, but \( E[X^4] \approx d^2 \) is definitely good enough (as that would imply a probability of at most \( \approx \frac{1}{d^2} \) that \( X \approx \pm d \)).

Expanding \( E[X^4] \) and using \( x_i^2 = 1 \), we get:

\[ E[X^4] = \sum_i E[x_i^4] + \sum_{i,j} (E[x_i^2 x_j^2] + E[x_i^4 x_j]) + \sum_{i,j,k} E[x_i x_j x_k x_l] + \sum_{i,j,k,l} E[x_i x_j x_k x_l] \]

\[ \approx d + d^2 + \sum_{i,j} E[x_i x_j] + d \cdot \sum_{i,j} E[x_i x_j] + \sum_{i,j,k,l} E[x_i x_j x_k x_l] \]

For the next step, we would like to show that:

\[ E[x_i x_j] \approx \Theta(\frac{1}{d}) \]

\[ E[x_i x_j x_k x_l] \approx \Theta(\frac{1}{d^2}) \]

However, we were not able to prove that this is the case, leaving this as an open question.

3.4.4 Handling balanced pairs

Another possible approach is to show that even if Conjecture 3.4.1 does not hold, we can obtain an improved approximation ratio. In this section, we will describe an
additional local improvement step. Where the previous improvement step worked on vertices with an uneven distribution of neighbors, this step deals with some of the vertices that are balanced (i.e. there is an equal amount of neighbors on each side of the partition). Do note that the results presented here are not enough to cover every case where the conjecture is false, it may be a useful starting point.

More specifically, the case we are interested in is the one where two adjacent vertices \(i\) and \(j\) are both balanced, and \(i\) and \(j\) are on different sides of the partition. Referring to such \((i, j)\) as a balanced pair, we can perform the following local improvement step:

- For each balanced pair \((i, j)\), move both \(i\) and \(j\) to the opposite sides of the partition

This could be incorporated in the \(A_{\text{Cut}}\) algorithm in a number of different ways, though combining it with the other local improvement step seems good as the two may provide opportunities for each other.

![Figure 3.3: If \(i\) and \(j\) form a balanced pair, \(i\) is initially “positive”, while \(j\) would be “negative”. The sets \(B(i)\) and \(B(j)\) are each balanced, meaning they (individually) have an equal number of vertices on each side of the hyperplane.](image)

**Lemma 3.4.4.** If there are \(\omega\) balanced pairs in a graph with bounded degree \(\Delta\), performing the above local improvement step will increase the approximation ratio by at least \(\frac{\omega}{\Delta^2}\).

**Proof.** First, we look at the improvement to the cut from altering one pair (as illustrated by Figure 3.3). Disregarding the edge between \(i\) and \(j\), both vertices have \(d/2\) (for their individual degrees \(d\) - which are not necessarily equal) neighbors on the same side, and \(d/2 - 1\) on the other side. If we swap the sides of \(i\) and \(j\), these values are reversed. In other words, we gain \(d/2 - (d/2 - 1) = 1\) cut edges on each
3.4. ASSUMING UNBALANCE AROUND ALL BAD VERTICES

Since the edge between $i$ and $j$ will still be cut, we thus have an improvement of 2 edges for each balanced pair.

Altering one such pair may affect up to $2(\Delta - 1)$ other pairs. Thus, performing the improvement will yield at least $2\omega n \cdot \frac{1}{2(\Delta - 1)} \geq \frac{Opt}{\Delta}$ additional cut edges. \qed
Chapter 4

Discussion

4.1 Conclusions

We were able to show that relatively small alterations to existing algorithms allow us to obtain an improved approximation ratio. Specifically, we have an expected approximation ratio of at least $\alpha_{GW} + \frac{1}{\Delta^3}$ for graphs of bounded degree $\Delta$. Compared to the original approximation ratio by Feige, Karpinski, and Langberg (2002) (where the additive term was $\frac{1}{\Delta^4}$), we have a better approximation ratio for any $\Delta > 2$. This improvement stems from our algorithm avoiding the interference between candidate vertices, which contributed a factor $\frac{1}{\Delta}$ in the previous result.

We were also able to show an improved approximation ratio for graphs of bounded degree $\Delta$ with at least a constant $\omega$ fraction of vertices with odd degree. Specifically, the additive term $\epsilon_\Delta = \frac{1}{\Delta^3}$ from the general case was replaced with a term $\epsilon^{\text{odd}}_\Delta = \frac{\omega}{\Delta^2}$. This is an improvement whenever $\omega > \frac{1}{\Delta^2}$, which is true for large groups of graphs (e.g. random graphs where the expected number of odd-degree vertices is reasonably large, and probably many graphs involved in real applications).

Assuming that Conjecture 3.4.1 holds, a similar result to the odd-degree one can be applied to the general bounded-degree case as well, where the additive term is $\Omega(\frac{1}{\Delta^2})$. Some reasons to believe this may be the case were presented, though no conclusive proof was found.

4.2 Future work

4.2.1 Probability of unbalanced neighbors

As mentioned before, proving Conjecture 3.4.1 could substantially improve the approximation ratio for bounded-degree graphs. Of course, it is not known whether
CHAPTER 4. DISCUSSION

this is actually possible. The experimental evidence in Section 3.4.2 suggests that there is at least a chance the conjecture is true.

The intuitive notion that adding more pairs of vectors should decrease the chance of them being partitioned equally also suggests an approach for proving the conjecture through induction.

The moment-based approach described in Section 3.4.3 may also be useful in proving the conjecture, though we were unable to do so during the work on this thesis.

4.2.2 Utilizing balanced pairs

There are a few obstacles remaining in order to be able to use the balanced pair handling described in Section 3.4.4.

One approach is to assume that there is a large number of edges belonging to balanced vertices (or vertices with a high probability of being balanced). If so, we should also have a relatively large number of balanced pairs. Otherwise, we have a large number of unbalanced vertices (or vertices with a high probability of being unbalanced). This sounds good - unbalanced vertices is what we wanted in the previous analysis. However, we also want the corresponding vector to lie close to the hyperplane. Otherwise, its neighbors will most likely be on the other side of the plane - and this could be the reason the vertex had a high probability of being unbalanced in the first place. Thus, further research is needed to determine whether this approach can yield any results.

Another way of looking at it is that the problematic cases are those where:

1. The neighbors $u_i$ are unbalanced with high probability
2. Their projections orthogonal to $v$ are balanced with high probability

The converse of the first would imply that balanced pairs are common, while the converse of the second would allow simple local improvements (essentially what is implied by Conjecture 3.4.1). Thus, finding a way to handle those cases could be another way to improve the approximation ratio.

4.2.3 Using Hierarchies

Semidefinite programs can be strengthened by adding additional constraints. Hierarchies give a structured approach for adding such local constraints. Chlamtac and
4.2. FUTURE WORK

Tulsiani (2012) describe several of these, but based on the comparison by Laurent (2003) the Lasserre hierarchy seems the most promising.

The Lasserre hierarchy works in levels, with increasing levels giving tighter relaxations. Each level adds additional variables and corresponding constraints for increasingly large subsets of the original problem’s variables. Solving the relaxation at a given level $t$ takes time $n^{O(t)}$ (i.e. a constant $t$ gives a solution in polynomial time).

It may be possible to obtain an improved approximation ratio by using these hierarchies. For instance, the constraints added at $t = 2$ are similar the triangle inequalities (2.2) and (2.3), which played an important role in the proof of the main theorem by Feige, Karpinski, and Langberg (2002).
Bibliography


