GENERALIZED MAX-CUT AND THE APPROXIMATION RATIO

by

Junsi Zhang

Bachelor of Science, University of Toronto, 2017

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Applied Mathematics

Toronto, Ontario, Canada, 2019

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Abstract

Generalized Max-Cut and the Approximation Ratio

Master of Science 2019

Junsi Zhang

Applied Mathematics

Ryerson University

In this thesis, we formulate a new problem based on Max-Cut called *Generalized Max-Cut.* This problem requires a graph as input and two real numbers (a, b) where $a > 0$ and $-a < b < a$ and outputs a number. The restriction on the pair (a, b) is to avoid trivializing the problem. We formulate a quadratic program for Generalized Max-Cut and relax it to a semi-definite program. Most algorithms in this thesis will require solving this semi-definite program.

The main algorithm in this thesis is the 2-Dimensional Rounding algorithm, designed by Avidor and Zwick, with the restriction that the semi-definite program of the input graph must have 2-Dimensional solutions. This algorithm uses a factor of randomness, $\beta \in [0, 1]$, that is dependent on the integer input to Generalized Max-Cut. We improve the performance of this algorithm by numerically finding better β.

Table of Contents

List of Figures

Chapter 1

Introduction

Our world can be structured using graphs. From obvious examples such as maps with roads that connect places, to more subtle examples such as relationships between people. Naturally, this structure gave rise to optimization problems, such as which route will take me point A to point B the fastest? Or where should hospitals be built to minimize the response time? In this thesis, we will demonstrate some applications of graphs, solving optimization problems, and approximating optimization problems.

Consider a company that decides to distribute free samples as an advertisement strategy. This company has access to a person (call him Carlos for simplicity) and a list of people he knows (for example, the people Carlos graduated high school with). This company wants to minimize the amount of free samples given out while maximizing the amount of people that are reached by this advertisement directly and indirectly. This group of people can be structured as a graph in which each person is a vertex and if two people know each other, then we connect them by an edge. We assume that people in this group talk to each other, so word of mouth is an indirect way to advertise this product. For example, if Carlos receives a free sample, then he will tell people he knows about it meaning that this product has directly reached Carlos and indirectly reached the people he is incident to.

We formulate the above real life problem as a mathematical problem using graphs. Let each person on the list of people Carlos knows, including Carlos, be a vertex. For each pair of people, we connect them with an edge if they know each other. There are two successful ways to advertise, the direct way is to receive a sample, and the indirect way is to be the immediate friend of someone who has received a sample (word of mouth). The goal is to maximize the number of people we successfully advertised to. We can solve this problem by separating the group of people into two groups where the number of edges connecting between the two groups are maximized. This problem is the same as the optimization version of Max-Cut. We target the smaller group with samples, while assuring the maximum number of people are indirectly advertised to.

An alternative version of this problem, suppose each edge is given a "weight" w as an integer. Realistically, not everyone in this group will be friends with each other. We assume some relationships are stronger than others and assign weights to these relationships accordingly. For instance, if two people are very close friends, then we give their relationship a weight of 10. If two people are only familiar with each other, then their weight is 5 and if they are enemies, then we give them weight of -5 and so on. This weight distribution is based on the assumption that if two people are very close, then they are more likely to believe the other's opinions on the product, and if they are enemies, then they will be more skeptical towards the other's opinions. In this scenario, it makes sense to want to separate two people whose relationships are stronger, namely has a higher weight, to increase the effects of indirect advertising. Hence, we want to separate the group into two groups where the sum of the weights of the relationships between the two groups are maximized.

In this thesis, we formulate a new problem based on Max-Cut that we call Generalized Max-Cut. We take into account of repercussions for the edges that are not cut. Consider the following scenario. For each person that is not reached by word of mouth or the free sample, the company will lose money. Furthermore, if two people who know each other both get the free sample, then the company will still lose money. We assign an arbitrary $a > 0$ for each connection between the two groups, and b, where $-a < b < a$, for each connection not between the two groups. This is a problem that we will analyze in great detail throughout this thesis.

1.1 Literature Overview

Max-Cut is the problem of finding the maximum number of edges "cut" by a partition of the vertex set of a graph into two sets. Equivalently, the problem can be rephrased as maximizing the summation of a function that adds 1 for each edge cut, and subtract 0 for every edge that is uncut. Weighted Max-Cut is a more general variation of the original problem. In Weighted Max-Cut, each edge has a weight attached to it. Instead of maximizing the number of edges cut, we aim to maximize the sum of the weights of edges that are cut. Various attempts to solve Max-Cut exactly have been shown to be inefficient in terms of time complexity. In 1971, Stephen Cook showed that all problems in the set NP can be reduced to the problem $3SAT$ in polynomial-time, proving that $3SAT$ is NP-hard. Furthermore, as $3SAT$ is also a problem in NP, it is NP-complete [4]. In the following year, Richard Karp used the Cook-Levin theorem from Cook's paper to show a polynomial reduction from $3SAT$ to 21 other problems, including Max-Cut, which was proven to be NP-complete [12].

It is possible that no polynomial-time algorithms exist that can solve Max-Cut exactly, unless $P = NP$ [9]. Hence, it makes sense to design polynomial-time algorithms to approximate Max-Cut as close to the optimal value as possible. We test out several existing approximation algorithms on Generalized Max-Cut in this thesis, as well as improving upon an existing approximation algorithm to achieve the optimal result for our problem.

Since the introduction of Max-Cut, the best algorithm, introduced by Sahni and Gonzales, is a basic algorithm that was only able to give a $\frac{1}{2}$-approximation of the optimal value with no dependency on the input. On given any graph, the algorithm operates only on the set of vertices with uniform randomness [20]. This algorithm uses uniform randomness to determine the two partitioned sets of V. For every vertex $v \in V$, v is put into into one of the two sets with probability $\frac{1}{2}$ [20].

In 1995, a breakthrough was made by Goemans and Williamson, who provided a much better approximation of roughly 0.878, by solving a semi-definite formulation of Max-Cut [9]. The Hyperplane Rounding algorithm, presented by Goemans and Williamson is the first to use the technique of solving a semi-definite program for Max-Cut. This algorithm then uses a random hyperplane to determine the two partitions of vertices. The first step in the algorithm solves the semi-definite formulation of Max-Cut to obtain a set of unit vectors. These n-dimensional unit vectors, where $n \in \mathbb{Z}$, represent the vertices. The goal is to round these vectors to -1 or 1 where -1 and 1 represent the two partitions. The second step of this algorithm is to pick a unit vector r uniformly at randomly on the unit circle. Then they round the unit vectors obtained from solving the semi-definite program using the hyperplane perpendicular to r [9]. We run this algorithm and the trivial algorithm on Generalized Max-Cut as a reference to see how much better other algorithms perform on our problem.

A natural question that arises from the Goemans and Williamson lower bound ratio, is whether we can find a polynomial-time algorithm that performs better than the ~ 0.878 ratio for Max-Cut on general graphs [1][11]. The answer to this question has been studied and analyzed by many scientists and it comes down to two important conjectures: $P \neq NP$ [4][22] and the Unique Games Conjecture [13].

The conjecture $P \neq NP$ was officially and formally introduced in 1971 by Stephen Cook [4]. This conjecture implies that there are problems that cannot be solved exactly in polynomial-time. However, we can still approximate solutions for such problems using polynomial-time algorithms. We determine how good an algorithm performs on a given problem using the ratio $\inf_I \dfrac{OPT(I)}{ALGO(I)}$, where $OPT(I)$ is the global optimal value for a problem on instance I and $ALGO(I)$ is the optimal value of the problem obtained from the algorithm on instance I. This ratio also provides an insight to the integrality gap of problems, since the integrality gap, $\inf_I \dfrac{OPT(I)}{SDP(I)}$ where the $OPT(I)$ and $SDP(I)$ are the optimal value of a problem on instance I and the semi-definite formulation of a problem on instance I respectively, is never greater than the performance ratio of an algorithm on the same problem.

Assuming $P \neq NP$, the best ratio that can be obtained for Max-Cut in polynomial-time is $\dfrac{16}{17}$ [10][14]. Goemans and Williamson provided a polynomial-time algorithm that achieves a ratio of roughly 0.878. However, there are papers that improve upon this result for specific cases of Max-Cut. Feige and Schechtman showed in 2001 that no algorithm that requires solving a semi-definite program will have a better approximate ratio than 0.891 by adding triangle inequalities to the semi-definite formulate of Max-Cut [6]. In dense graphs, using the simple greedy algorithm has been proven to achieve near optimal value. The algorithm simply runs a greedy algorithm until we are satisfied with the cut it presents. This algorithm is guaranteed to converge to near optimal value in polynomial time $OPT - \epsilon n^2$, for some $\epsilon > 0$ and $n \in \mathbb{N}$ [16].

A variant of Max-Cut, known as Max-Bisection is the problem of solving Max-Cut where the two partitioned sets have an equal amount of vertices (assume even order graphs). Since Max-Bisection is as hard as Max-Cut, the Unique Games Conjecture implies that it is hard to approximate Max-Bisection to within a factor of $0.878 + \epsilon$[14]. A new algorithm is presented in the paper 'Better Balance by Being Biased',this algorithm reaches a ratio of 0.87765366 [2]. This paper leaves an open question of whether the bisection constraint makes Max-Cut harder.

Another variant of Max-Cut is to analyze graphs with bounded degree, meaning graphs with degree no more than $\Delta \in \mathbb{Z}$ number of vertices. This variation admits an algorithm that achieves a ratio of at least $0.878 + \epsilon$, where $\epsilon > 0$ and depends only on Δ [5].

Now instead of considering special graphs, consider cases where solving the semi-definite formulation of Max-Cut returns nearly exact solutions, that is, vectors that are only 2 or 3 dimensional [3]. Would this improve the Goemans and Williamson ratio? The answer to this question is yes. Using Gengenbauer polynomials, we can improve upon this ratio [3]. This algorithm uses similar techniques to the Hyperplane rounding algorithm used by Goemans and Williamson [3]. We first solve the semi-definite formulation, and obtain a set of vectors that are only 2-dimensional. Then we obtain another set of unit vectors by mapping the original solutions using a special Gengenbauer polynomial. Now we run the Hyperplane algorithm on the original vectors using probability β and the new vectors with probability $1 - \beta$. We improve upon this algorithm in our thesis using details unique to our problem. The contribution made by us in this thesis will be discussed in detail in the next section.

A *Unique Game* is a constraint satisfaction problem, where the goal is to satisfy as many constraints as possible [7]. In a constraint satisfaction problem, we have a set of variables, a set of values that the variables are allowed to assume, and a set of constraints. Max-Cut is also an example of a constraint satisfaction problem.

The Unique Games Conjecture uses Unique Games in its definition: for all $\epsilon, \delta > 0$, and a list of constraints, we are given an instance with two possible guarantees: either this instance is highly unsatisfiable, meaning only δ percentage of the constraints can be satisfied, or this instance is almost satisfiable, meaning $1 - \epsilon$ percentage of the constraints can be satisfied [13]. The Unique Games Conjecture states that the problem of checking to see which case the instance belongs to is NP-hard.

The Unique Games Conjecture is believed to be independent of the conjecture $P \neq NP$. If we assume the Unique Games Conjecture instead of $P \neq NP$, then the ratio 0.878, obtained by Goemans and Williamson using hyperplane rounding is optimal in polynomial-time [14].

1.2 Our Contributions

We formulate a generalized version of Max-Cut by changing the objective function. Max-Cut counts the number of edges cut which is equivalent to a function that adds one for each edge cut and 0 for each edge uncut. In our generalized version, we consider the function that adds $a \in \mathbb{R}$ for each edge cut and subtracts b, where $-a < b < a$ for each edge uncut.

We follow closely the Hyperplane algorithm presented by Goemans and Williamson and run the Hyperplane Rounding algorithm on the Generalized Max-Cut to see how well the algorithm performs. Since we have more inputs than Max-Cut, the results we obtain use numerical analysis and rely on graphs to illustrate the significance of it.

In addition, we improve the approximate ratio and difference comparisons by considering the case where vectors obtained from solving the semi-definite formulation of Generalized Max-Cut are 2-dimensional [3]. We follow Zwick and Avidor's algorithm that runs the Goemans Williamson Hyperplane Rounding Algorithm with probability β and a Gengenbauer Polynomial Rounding algorithm with probability $1 - \beta$. We improve upon this algorithm to find the best β for the fixed a and b in each instance.

This thesis is structured as follows. In Chapter 2, we discuss theorems, definitions and lemmas in Linear Algebra, Graph Theory and Combinatorial Optimization. In Chapter 3, we discuss definitions of various mathematical programming, and some properties of mathematical programs. In Chapter 4, we define our problem Generalized Max-Cut and its formulation. In Chapter 5, we explore the motivation approximation algorithms and the conjecture $P \neq NP$. In chapter 6, we run the trivial algorithm on Generalized Max-Cut to see how well it performs. In Chapter 7, we run the Hyperplane Rounding algorithm by Goemans and Williamson and evaluate numerically the approximate ratio and difference of the expected value of the algorithm on Generalized Max-Cut and its semi-definite formulation. In Chapter 8, we follow Zwick and Avidor's Gengenbauer algorithm and improve it for Generalized Max-Cut.

Chapter 2

Preliminaries

In this Chapter, we introduce some results from linear algebra and graph theory, as well as some background information. Although most theorems, definitions and facts mentioned in this section are well known, it is convenient to gather these results together and specify the notations that will be used throughout this thesis. More detailed explanations and proofs can be found in the linear algebra textbook by Nicholson, W. Keith [17]. We begin with linear algebra, followed by an introduction to graph theory.

2.1 Linear Algebra

Definition 2.1.1. *(Inner Product) Let* $A, B \in \mathbb{R}^{n \times n}$ *be two matrices. We define the* ***inner product*** *of* A *and* B *to be* $\langle A, B\rangle = \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}B_{ij}$.

Lemma 2.1.1. *(Properties of Inner Product)* $\forall A, B, C \in \mathbb{R}^{n \times n}$*, for all* $a \in \mathbb{R}$*, we have that:*

1. $\langle A, B\rangle = \langle B, A\rangle$

2. $\langle aA + C, B\rangle = a\langle A, B\rangle + \langle C, B\rangle$.

3. $\langle A, A\rangle \geq 0$. *Equality holds when* $A = 0$.

The proof of these facts can be found in any linear algebra textbook and thus will be omitted from this thesis [17].

We define the notion of a n variable quadratic function and represent it as the inner product of two matrices.

Lemma 2.1.2. *Let* $x = (x_1, x_2, \cdots, x_n)^T$. *Any quadratic function* $f : \mathbb{R}^n \to \mathbb{R}$ *such that* $f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_ix_j + \sum_{i=1}^{n} b_ix_i + c$, *where* $a_{ij}, b_i, c \in \mathbb{R}$ *can be written as a sum of inner products in the form:* $\langle A, X\rangle + \langle b, x\rangle + c$.

Proof. Define X by the following:

$$X = xx^T = \begin{bmatrix} x_1^2 & x_1x_2 & \dots & x_1x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_nx_1 & x_nx_2 & \dots & x_n^2 \end{bmatrix}.$$

Define the symmetric coefficient matrix $A = \dfrac{\bar{A}^T + \bar{A}}{2}$ where:

$$\bar{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix},$$

$$A = \begin{bmatrix} a_{11} & \dfrac{a_{12}+a_{21}}{2} & \dots & \dfrac{a_{1n}+a_{n1}}{2} \\ \dfrac{a_{21}+a_{12}}{2} & a_{22} & \dots & \dfrac{a_{2n}+a_{n2}}{2} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{a_{n1}+a_{1n}}{2} & \dfrac{a_{n2}+a_{2n}}{2} & \dots & a_{nn} \end{bmatrix}.$$

By definition, the inner product of these two matrices is: $\langle A, X\rangle = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_ix_j$. Now let $b \in \mathbb{R}^n$, and $c \in \mathbb{R}$. It is clear to see that $\sum_{i=1}^{n} b_ix_i + c = \langle b, x\rangle + c$ using Definition 2.1.1, and we have shown that $f(x) = \langle A, X\rangle + \langle b, x\rangle + c$ as needed. □

This matrix definition of quadratic functions will be used to formulate Max-Cut and Generalized Max-Cut as a quadratic optimization problem in later Chapters.

Definition 2.1.2. *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix, then it has real eigenvalues λ such that $Av = \lambda v$ for all non-zero $v \in \mathbb{R}^n$. We call v the* ***eigenvector*** *corresponding to the* ***eigenvalue*** *λ.*

Definition 2.1.3. *A symmetric matrix A is called* ***positive semi-definite****, denoted by $A \succeq 0$, if every eigenvalue of A is nonnegative.*

Positive semidefinite matrices have important properties that will be used in later Chapters. In this Chapter, we focus on elaborating and proving these properties.

Recall that a *symmetric matrix* is a matrix A such that $A = A^T$. The eigenvalues of A are roots of the polynomial $det(A - \lambda I)$, where I is the identity matrix of appropriate dimensions. Here are some Lemmas regarding symmetric matrices.

Lemma 2.1.3. *If A is symmetric, then all eigenvalues of A are real. [17]*

Lemma 2.1.4. *If A is symmetric, then there exist an orthogonal matrix Q such that $A = QDQ^T$ where D is the diagonal matrix of eigenvalues. [17]*

Theorem 2.1.1. *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric, positive semidefinite matrix. There exist some $B \in \mathbb{R}^{n\times n}$ such that $A = BB^T$.*

Proof. Let λ_i, $i = 1, \cdots, n$ be eigenvalues of A. Let D be the diagonal matrix of these eigenvalues, that is $D = \text{diag}(\lambda_1, \cdots, \lambda_n)$. By Lemma 2.1.4, we write $A = QDQ^T$. From Lemma 2.1.3, we know that $\sqrt{\lambda_i} \in \mathbb{R}$ is well defined, $\forall \text{i} = 1, \cdots, n$. Let $S = \text{diag}(\sqrt{\lambda_1}, \cdots, \sqrt{\lambda_n})$ such that $SS^T = D$. Now we have $A = QSS^TQ^T = QS(QS)^T$. Let $B = QS$, and we derive that $A = BB^T$ as needed. ☐

Proposition 2.1.1. *The following are equivalent for symmetric $A \in \mathbb{R}^{n\times n}$:*

1. *A is positive semi-definite.*
2. *There exist $B \in \mathbb{R}^{n\times n}$ such that $A = BB^T$.*
3. *There exist vectors $v_i \in \mathbb{R}^n, i = 1, 2, \ldots, n$ such that $A_{ij} = \langle v_i, v_j\rangle$.*
4. *$x^TAx \geq 0$ for all $x \in \mathbb{R}^n$.*

Proof. We show that these propositions are equivalent by showing that $1 \Rightarrow 2$, $2 \Leftrightarrow 3$, $2 \Rightarrow 4$, and $4 \Rightarrow 1$.

1. $(1 \Rightarrow 2)$. This is Theorem 2.1.1.

2. $(2 \Rightarrow 3)$. Suppose we have have a matrix A that can be written as BB^T. Let v_i be the i^{th} row of the matrix B such that $B = \begin{pmatrix} v_1^T \cdots v_n^T \end{pmatrix}$, it is clear to see that $A_{ij} = \langle v_i, v_j \rangle$.
   $(3 \Rightarrow 2)$ Similarly, let v_i be the rows of the matrix B.

3. $(2 \Rightarrow 4)$. Since there exist matrix B such that $A = BB^T$, we can write x^TAx as x^TBB^Tx, which is equal to $(x^TB)(x^TB)^T = \|x^TB\|^2 \geq 0$.

4. $(4 \Rightarrow 1)$. Let λ be any eigenvalue of A, and let x be the corresponding eigenvector of A. Then $x^TAx = x^T\lambda x = \lambda x^Tx = \lambda\|x\|^2 \geq 0$, hence $\lambda \geq 0$ as needed.

We have shown that these propositions are equivalent. ☐

Definition 2.1.4. *The* ***rank*** *of a matrix* $A \in \mathbb{R}^{n\times n}$ *is the number of linearly independent rows.*

Theorem 2.1.2. *The rank of a* ***symmetric matrix*** $A \in \mathbb{R}^{n\times n}$ *is* $k \leq n$ *if and only if* $n-k$ *eigenvalues of* A *are 0.*

Proof. Recall that the dimension of null(A) is the number of linearly independent eigenvectors x such that $Ax = 0x$, namely the number of eigenvalues that are 0. Since we have null(A) = dim(A) - rank(A) by the rank-nullity theorem, we have that null(A) $= n-k$ as needed. ☐

Corollary 2.1.2.1. *A matrix* $A \in \mathbb{R}^{n\times n}$ *is a rank* k *PSD matrix if and only if there exist vectors* $v_i \in \mathbb{R}^k$ *such that* $A_{ij} = \langle v_i, v_j \rangle$.

Proof. Let $\lambda_1, \cdots, \lambda_n$ be eigenvalues of A. By Lemma 2.1.4 we have $A = QDQ^T$ for some orthogonal Q and $D = diag(\lambda_1, \cdots, \lambda_n)$. By Lemma 2.1.3 and Theorem 2.1.2, we know that $\lambda_i \geq 0$ and $\lambda_i \in \mathbb{R}$, and that $n-k$ eigenvalues are 0. Thus, $D = diag(\lambda_1, \cdots, \lambda_k, 0, \cdots, 0)$. Let $S = diag(\sqrt{\lambda_1}, \cdots, \sqrt{\lambda_k}, 0, \cdots, 0)$ such that $A = (QS)(QS)^T$. Let v_i be the i^{th} row of QS without the 0 entries, and we have $A_{ij} = \langle v_i, v_j \rangle$ as needed. ☐

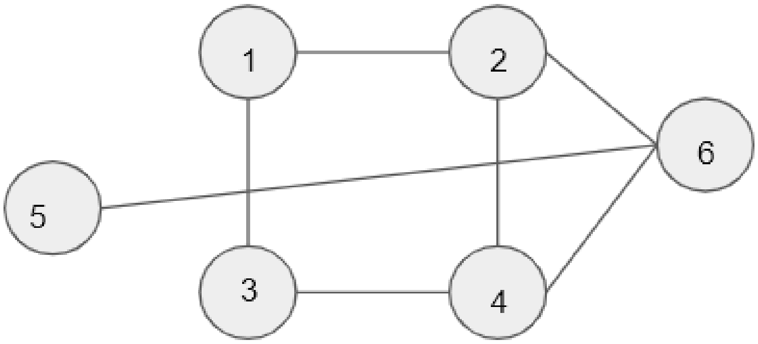
2.2 Graph Theory

Graphs are data structures that consist of two sets: a set of elements and a set of pairwise connections of these elements. In more technical terms, a graph $G = (V, E)$ is of a pair of sets: V, the nonempty set of *vertices*; and E, the set of *edges* where each edge $e = \{i, j\} \in E$ is a subset of V.

In this thesis, we focus on *finite simple graphs.* In a finite simple graph, every pair of vertices has *at most* one edge connecting them, and we have a finite number of vertices and edges .

Consider the following example of a graph $G = (V, E)$:

Figure 2.1: Graph G



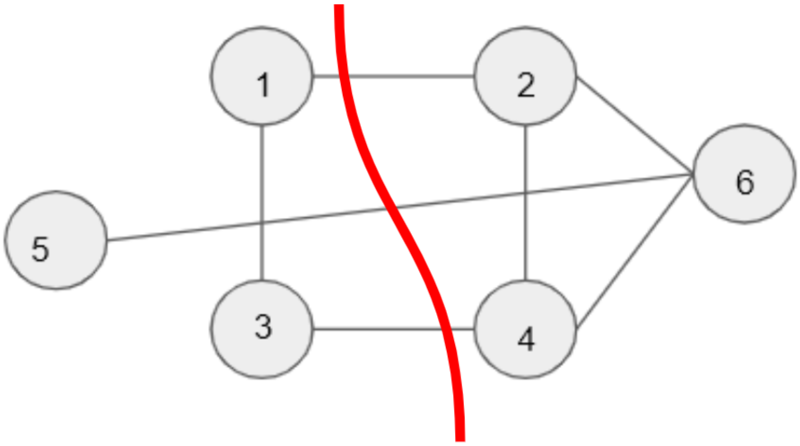
The graph in Figure 2.1 has vertices $V = \{1, 2, 3, 4, 5, 6\}$, and edges $E = \{a, b, c, d, e, f, g\}$ as follows:

a	b	c	d	e	f	g
$\{1, 2\}$	$\{1, 3\}$	$\{2, 4\}$	$\{3, 4\}$	$\{2, 6\}$	$\{4, 6\}$	$\{5, 6\}$

Definition 2.2.1. *Let $G = (V, E)$ be a graph, a* ***cut*** *in G is a partition of V into two sets: $S \subseteq V$ and $V \backslash S$. We say an edge $e = \{i, j\} \in E$ is* ***cut*** *if i, j are not both in the same partitioned set. We denote the set of cut edges induced by S as $\delta(S)$.*

Definition 2.2.2. *The* ***size*** *of a cut S is the number of edges that are* ***cut****, denoted by $|\delta(S)|$.*

Figure 2.2: A cut in G

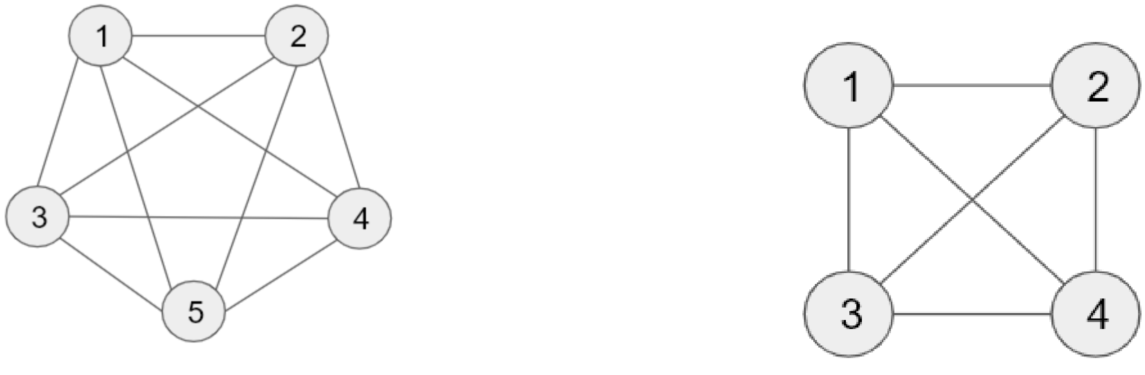


Example 2.2.1 (A cut in G)**.** *Consider Figure 2.2, one potential cut for this graph is the following partition:* $S = \{1,3,5\}$ *and* $V \backslash S = \{2,4,6\}$*. The size of this cut is 3 since 3 edges,* $\{1,2\}$*,* $\{5,6\}$ *and* $\{3,4\}$ *are cut. The edge cut set is given by* $\delta(S) = \big\{\{1,2\},\{5,6\},\{3,4\}\big\}$.

Definition 2.2.3. *Let* $G = (V,E)$ *be a graph. We say that the vertices* i, j ***adjacent*** *if* $\{i,j\}$ *is an edge.*

Definition 2.2.4. *A* ***clique*** *is a special type of graph where all vertices are adjacent. We denote a* ***clique*** *with* n *vertices by* K_n.

In other words, every vertex in a clique is connected to every other vertex by an edge. Observe that since there exist an edge for every distinct pair of vertices, a clique with n vertices has $\binom{n}{2}$ many edges. We illustrate two examples, K_5 and K_4 in Figure 2.3.



(a) A graph K_5 with 5 vertices and $\binom{5}{2} = 10$ edges

(b) A graph K_4 with 4 vertices and $\binom{4}{2} = 6$ edges

Figure 2.3: Clique graph examples

2.3 Combinatorial Optimization Problems

In this section, we define explicitly what it means to solve an optimization problem, and what categorizes a problem as combinatorial.

An **optimization problem** is a problem that involves finding the best solution in a set of all feasible solutions. Typically, optimization involves maximizing or minimizing a function. For example, a company would want to minimize production time, or maximize profit.

Definition 2.3.1. *A* ***combinatorial optimization problem*** *is an optimization problem where the set of possible solutions is finite or discrete.*

An example of a combinatorial optimization problem is *Max-Cut*. Informally, Max-Cut is an optimization problem that aims to maximize the size of the edge cut set of a given graph. Formally, we have the following definition:

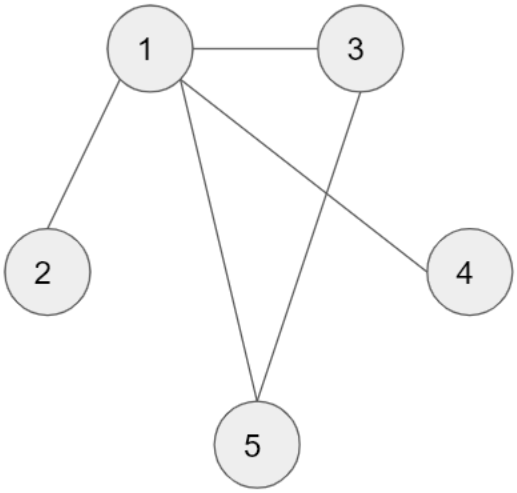
Definition 2.3.2. *Given a graph $G = (V, E)$,* ***Max-Cut*** *is the problem of finding the maximum size of the cuts in G.*

One way to find a Max-Cut in a graph is to find a set $S \subseteq V$ such that for any other $S' \subseteq V$, $|\delta(S)| \geq |\delta(S')|$. Any subset $S \subseteq V$ is a feasible solution since it provides us with $\delta(S)$. Here is an example of a maximum cut in a graph:

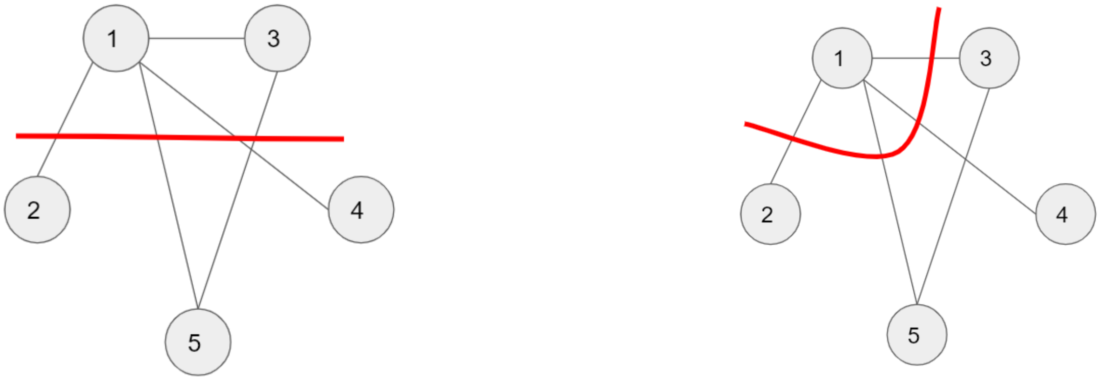
Example 2.3.1 (Max-Cut for G)**.** *Consider Figure 2.4. We have $G = (V, E)$, where $V = \{1, 2, 3, 4, 5\}$ and $E = \{a, b, c, d, e\}$ as follows:*

a	b	c	d	e
$\{1,2\}$	$\{1,3\}$	$\{1,4\}$	$\{1,5\}$	$\{3,5\}$

Figure 2.4: Graph - Table Format



The maximum cut of G can be induced by two different sets S and S' as shown below.



(a) g1cut

(b) g1cut2

Figure 2.5: Cut example

As shown in Figure 2.5a, the edge cut set induced by $S = \{1, 3\}$ and $V \backslash S = \{2, 4, 5\}$ is

$$\delta(S) = \big\{\{1, 2\}, \{1, 4\}, \{1, 5\}, \{3, 5\}\big\}.$$

The size of this set $|\delta(S)| = 4$, which is the maximum value attainable by this function.

In Figure 2.5b, the edge cut set induced by the partition $S' = \{1\}$ and $V \backslash S' = \{2, 3, 4, 5\}$ is

$$\delta(S') = \big\{\{1, 2\}, \{1, 4\}, \{1, 5\}, \{1, 3\}\big\}.$$

The size of the set $|\delta(S')|$ is also 4.

This example emphasizes the difference between optimal solutions and the value given by those optimal solutions. Note that we have two distinct solutions that produce the same optimal value. We only care to find the optimal *size* of the edge cut set, the actual solution does not matter to us. In general, finding the *set* that is maximum cut of a graph is difficult, and finding the size of this set is also difficult. In this thesis, we aim to approximate the size of a maximum cut on a given graph, rather than to find the actual cut.

We discussed Max-Cut as a combinatorial problem in this section. In the following section, we formulate this optimization problem formally and mathematically using *mathematical programming*.

Chapter 3

Mathematical Programming

Mathematical programs are optimization problems that involve maximizing or minimizing a function with constraints. These programs are often used to formulate real life problems mathematically, making the real life problems easier to understand. In this section, we discuss four types of mathematical programs: integer programs, linear programs, quadratic programs, and semi-definite programs. We also discuss some properties of these programs and how they relate to Max-Cut.

3.1 Integer Programming

An *integer program* is the problem of optimizing a linear function with $n \in \mathbb{N}$ variables subject to linear constraints, and a mandatory constraint that all feasible solutions must be integers. For example, suppose we want to maximize the function $f(x_1, x_2) = x_1 + x_2$ subject to the constraint $x_1 - x_2 \leq 5$, $x_i \in \mathbb{Z}^n$. Typically, an integer program has the following form:

$$\begin{aligned} \text{maximize} \quad & c^T x \\ \text{subject to} \quad & a_1^T x \leq b_1, \\ & a_2^T x \leq b_2, \\ & \quad \vdots \\ & a_m^T x \leq b_m \\ & \quad x \in \mathbb{Z}^n, \end{aligned} \tag{3.1}$$

where $a_1, a_2, \ldots, a_m \in \mathbb{R}^n$, $b_1, b_2, \ldots, b_m \in \mathbb{R}$, and $x = (x_1, \ldots, x_n)$ is a vector of n variables.

Definition 3.1.1. *(Feasibility) We say $x \in \mathbb{R}^n$ is a* ***feasible*** *solution for an integer program if x satisfies* *all the constraints of the program.*

The set of all feasible solutions to a mathematical program is called the *feasible region.* If this region is nonempty, then the mathematical program is *feasible,* else, it is infeasible.

We use Max-Cut as an example throughout the thesis. In this section, we define and prove the integer formulation of Max-Cut.

Recall that **Max-Cut** is a combinatorial optimization problem that takes an instance of a graph as input. Any partition $S \subseteq V$ is a feasible solution and the goal is to maximize the size of the edge cut.

Lemma 3.1.1. *The integer program formulation of Max-Cut is as follows:*

$$\begin{aligned}
\textit{maximize} \quad & \sum_{e \in E} z_e & \\
\textit{subject to} \quad & z_e \leq x_i + x_j, \forall e = ij \in E & \textit{(constraint 1)} \\
& z_e \leq 2 - (x_i + x_j), \forall e = ij \in E & \textit{(constraint 2)} \\
& z_e \in \{0,1\}, \forall e \in E & \textit{(constraint 3)} \\
& x_i \in \{0,1\}, \forall i \in V & \textit{(constraint 4)}
\end{aligned} \tag{3.2}$$

Proof. Let S be a random partition of V, that is a subset of V where the elements are randomly chosen from V. We define the indicator variables $x_i \in \{0,1\}$ and $z_e \in \{0,1\}$ to be:

$$x_i = \begin{cases} 0 & \text{if } i \notin S \\ 1 & \text{if } i \in S \end{cases} \qquad z_e = \begin{cases} 0 & \text{if } e \notin \delta(S) \\ 1 & \text{if } e \in \delta(S) \end{cases}$$

($\Rightarrow$) We want to show that every feasible solution for Max-Cut is also feasible for the integer formulation. Suppose we have an instance of a graph $G = (V, E)$ and a partition of vertices S and correspondingly, $V \backslash S$. Let $x_i = 1$ if $i \in S$ and 0, otherwise.

Suppose we have an edge e such that $e = ij \in \delta(S)$. In other words, the vertices i, j are not both in S or not both in $V \backslash S$. Then we have $x_i + x_j = 1$ and $2 - (x_i + x_j) = 1$, meaning that $z_e = 1$ or $z_e = 0$ by constraints 1 and 2. Since we aim to maximize the size of this cut, we take the large value of $z_e = 1$. This effectively adds 1 to the objective function for ever edge $e \in \delta(S)$.

Now consider $e = ij \notin \delta(S)$. One of constraints 1 and 2 will be 0, hence z_e will be 0. This adds 0 to the objective function for every edge that is not cut as needed.

($\Leftarrow$) We begin with an optimal solution to the integer program, and show that it maps to an edge cut where the size of the edge cut is the value of the objective function.

Suppose we are given $x_i's$ and $z_e's$ that are optimal to the integer program. Define $S = \{i \in V | x_i = 1\}$. Now all we need to show is $\sum\limits_{e \in E} z_e = |\delta(S)|$.

Let $e = ij \in \delta(S)$. Without loss of generality, we have $i \in S$, $j \notin S$ and $x_i \neq x_j$. Then $x_i + x_j = 2 - (x_i + x_j) = 1$, and $z_e = 1$, since this is the optimal solution. Now consider $ij \notin \delta(S)$, then $i, j \in S$ or $i, j \in V \backslash S$. In either case, one of constraints 1, 2 is equal to 0, meaning that $z_e = 0$. Thus, we have $\sum\limits_{e \in E} z_e = |\delta(S)|$. □

3.2 Relaxations and Integrality Gap

Solving integer programs exactly can be difficult. Practically, it makes more sense to find good approximation to solutions fast rather than to spend lots of time finding exact solutions. Relaxations of mathematical programs allow for more solutions by "relaxing" some of the constraints of the mathematical program.

Definition 3.2.1. *(Relaxations) Let M be any mathematical program. A* ***relaxation*** *of M, denoted M_R, is any mathematical program where all feasible solutions to M are also feasible to M_R.*

In general, the feasible region of a relaxation is a superset of the feasible region of the original program.

Lemma 3.2.1. *If minimization or maximization problem M_R is a relaxation to minimization or maximization problem M, and I is any instance to the problem M, then $OPT_M(I) \geq OPT_{M_R}(I)$ or $OPT_M(I) \leq OPT_{M_R}(I)$, respectively. Where $OPT_{M_R}(I)$ and $OPT_M(I)$ are optimal values for M_R and M respectively [21].*

Proof. Since the feasible region for M_R contains the feasible region of M, M_R contains **at least** the optimal solution for M and possibly more. Hence we have $OPT(M) \geq OPT(M_R)$ for minimization problems, and $OPT(M) \leq OPT(M_R)$ for maximization problems. □

As relaxations allow more feasible solutions, the optimal solution of a relaxation might not be feasible for the original. A natural question is how close is the relaxed optimal solution to the original optimal solution? The answer to this is what is known as an integrality gap. Since we only work with combinatorial problems in this thesis, we define what integrality gap means for a formulation of a combinatorial optimization problems.

Definition 3.2.2. *(Integrality Gap) Let M be a formulation for a combinatorial optimization problem, and M_R be any relaxation to M. Denote $OPT_M(I)$ to be the optimal value of M on instance I, and $OPT_{M_R}(I)$*

the optimal value of M_R *on instance* I*. The integrality gap (IG) of* M_R *is defined as*

$$IG := \begin{cases} \sup\limits_I \dfrac{OPT_M(I)}{OPT_{M_R}(I)}, & \textit{for minimization problems} \\ \\ \inf\limits_I \dfrac{OPT_M(I)}{OPT_{M_R}(I)}, & \textit{for maximization problems} \end{cases}$$

We use supremum and infinmum because the maximum and the minimum might not exist.

3.2.1 Linear Programming

A linear program consists of optimizing a linear objective function subject to linear constraints. Notice that a linear program is similar to an integer program except it does not have the constraint that requires solutions to be integers. This allows feasible solutions to be any real number.

Remark. *Every integer program can be relaxed into a linear program by removing the integrality constraint.*

Here is a brief justification. Let M be an integer program, let M_R be the relaxation of M where the integer constraint is removed. We want to show that every feasible solution to M is also feasible to M_R. Let x be any arbitrary solution to M, x is also a solution to M_R as it satisfies all the same constraints of M_R.

For example, we have the linear program relaxation of Max-Cut.

$$\begin{array}{lll} \text{maximize} & \sum\limits_{e \in E} z_e & \\ \text{subject to} & z_e \leq x_i + x_j, \forall e = ij \in E & \text{(constraint 1)} \\ & z_e \leq 2 - (x_i + x_j), \forall e = ij \in E & \text{(constraint 2)} \\ & z_e \in [0,1], \forall e \in E & \text{(constraint 3)} \\ & x_i \in [0,1], \forall i \in V & \text{(constraint 4)} \end{array} \tag{3.3}$$

This linear program is similar to the integer formulation except for the integer contraints. Now we have solutions that can be all real numbers between 0 and 1.

Lemma 3.2.2. *The size of the edge cut* $|\delta(S)|$ *induced by any partition* $S \subseteq V$ *of a clique is* $|S| \cdot |V \backslash S|$*.*

Proof. Suppose we have any arbitrary partition $S \subseteq V$ of a clique K_n. For each vertex i in S, we know that it is connected to all other vertices. Moreover, it is connected to all vertices in $V \backslash S$. Hence the number of edges cut for vertex i is how many vertices are in $V \backslash S$, namely the size, $|V \backslash S|$. This is true for all vertices in S, thus we have that the size of the edge cut produced by any S is $|S| \cdot |V \backslash S|$. ☐

Lemma 3.2.3. *The maximum cut of a clique with n vertices, K_n has value at most* $\dfrac{n^2}{4}$.

Proof. Given a clique K_n, we want to show that if $\delta(S)$ is a maximum cut for K_n induced by S, then $|\delta(S)| \leq \dfrac{n^2}{4}$.

Let $s = |S|$, then $n - s = |V \backslash S|$. From Lemma 3.2.2., we can define a function for the size of the edge cut with respect to the size of the partition S. Let $f(s) = s(n-s)$, where $f(s)$ is the size of the edge cut induced by S. Now we can find the maximum of this function with respect to s. $f'(s) = n - 2s$, we have critical points at $s = \frac{n}{2}$. Since $f(s)$ is a concave parabola, $f(\frac{n}{2}) = \frac{n}{2}(n - \frac{n}{2}) = \frac{n^2}{4}$ must be the maximum. □

Theorem 3.2.1. *The linear program relaxation of Max-Cut has integrality gap less or equal to* $\frac{1}{2}$.

Proof. Since the claim states the integrality gap is less or equal to $\frac{1}{2}$, it suffices to find one instance of a graph where this holds true.

Consider a clique K_n. We know from Lemma 3.2.3. that it has optimal value less or equal to $\frac{n^2}{4}$. Consider the feasible solution $x_i = \frac{1}{2}, z_e = 1$, this clearly satisfies all the constraints of the linear program relaxation of Max-Cut. Moreover, this produces a value of $\binom{n}{2} = \frac{n(n-1)}{2}$. Now we use the definition of the integrality gap, define I to be an instance of a graph. Let M denote the integer program of Max-cut, and M_R denote the linear relaxation of Max-Cut with the integer constraint removed.

$$\begin{aligned} IG &= \inf_I \frac{OPT_M(I)}{OPT_{M_R}(I)} \\ &\leq \inf_n \frac{OPT_M(K_n)}{OPT_{M_R}(K_n)} \\ &\leq \inf_n \frac{\frac{n^2}{4}}{\frac{n(n-1)}{2}} \\ &\leq \frac{1}{2}. \end{aligned}$$

So the integrality gap of the Max-Cut integer formulation and its linear program relaxation if at most $\frac{1}{2}$. □

One important reason to relax integer programs into linear programs is that linear programs can be solved fast due to their convexity property.

3.3 Convexity of Mathematical Programs

Convex mathematical programs have important properties that we discuss in this section. We first define what it means for sets, and vectors in $\mathbb{R}^n$ to be convex.

Definition 3.3.1. *(Set Convexity) A set $A \in \mathbb{R}^n$ is called* ***convex*** *if for every $x_1, x_2 \in A$, $\theta x_1 + (1-\theta)x_2 \in A$ for all $\theta \in [0,1]$. In other words, every convex combination of elements in A is also an element in A.*

Definition 3.3.2. *$x \in \mathbb{R}^n$ is a convex combination of vectors $\{w^{(i)}\}$; if there exist non negative $\{\lambda_i\}_i$, with $\sum_i \lambda_i = 1$ and $x = \sum_i \lambda_i w^{(i)}$.*

We define half-space $H := \{x \in \mathbb{R}^n : a^T x + b \leq 0\}$, for $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Note that this is the feasible region for a mathematical program with constraints $a^T x + b \leq 0$.

Lemma 3.3.1. *All half-spaces are convex.*

Proof. Let $H = \{x \in \mathbb{R}^n : a^T x + b \leq 0\}$ be a half-space. Let $x_1, x_2 \in H$, then we have $a \in \mathbb{R}^n, b \in \mathbb{R}$ such that $a^T x_1 + b \leq 0$, and $a^T x_2 + b \leq 0$. Consider $\theta x_1 + (1-\theta)x_2$ for some $\theta \in [0,1]$. We want to show that $a^T\left[\theta x_1 + (1-\theta)x_2\right] + b \leq 0$:

$$\begin{aligned}
& a^T\left[\theta x_1 + (1-\theta)x_2\right] + b \\
=& \theta a^T x_1 + (1-\theta)a^T x_2 + b \\
\leq& \theta(-b) + (1-\theta)(-b) + b \\
=& -\theta b - b + \theta b + b \\
=& 0
\end{aligned}$$

□

Lemma 3.3.2. *For every convex sets $A, B \subseteq \mathbb{R}^n$, the set $A \cap B$ is convex.*

Proof. Let A, B be convex sets, let $x_1, x_2 \in A \cap B$. We want to show that for all $\theta \in [0,1]$, $\theta x_1 + (1-\theta)x_2 \in A \cap B$.

Since $x_1, x_2 \in A \cap B$, we have $x_1, x_2 \in A$ and $x_1, x_2 \in B$. By definition, $\theta x_1 + (1-\theta)x_2 \in A$ and $\theta x_1 + (1-\theta)x_2 \in B$ for all $\theta \in [0,1]$. Hence we have $\theta x_1 + (1-\theta)x_2 \in A \cap B$ as needed. □

Corollary 3.3.0.1. *The feasible region of linear programs are convex sets.*

Proof. Constraints of linear programs are in the form $a_i x \leq b_i$, where $a_i \in \mathbb{R}^n$, and $b \in \mathbb{R}$. The feasible x's form a half-space by definition, hence, the feasible region of linear programs are intersections of half-spaces. By Lemma 3.3.2 and Lemma 3.3.1, the intersections of half-spaces are convex, and thus the feasible region of linear programs are convex. ☐

Definition 3.3.3. *(Function Convexity) A function* $f : \mathbb{R}^n \to \mathbb{R}$ *is called convex if for every* $x_1, x_2 \in \mathbb{R}^n$, *we have* $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$ *for all* $\lambda \in [0,1]$.

Here is an important definition regarding mathematical programs:

Definition 3.3.4. *(Mathematical Program Convexity) A mathematical program is called* ***convex*** *if its objective function is convex, and its feasible region is convex.*

Lemma 3.3.3. *Linear and affine functions are convex.*

Proof. Let $f(x) = ax + b$, where $a, b \in \mathbb{R}$ and $x \in \mathbb{R}^n$. We want to show that for all $\lambda \in [0,1]$, and any $x_1, x_2 \in \mathbb{R}^n$: $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$.

$$\begin{aligned} f(\lambda x_1 + +(1-\lambda)x_2) &= a(\lambda x_1 + (1-\lambda)x_2) + b \\ &= \lambda a x_1 + (1-\lambda) a x_2 + b \\ &= \lambda a x_1 + (1-\lambda) a x_2 + b + \lambda b - \lambda b \\ &= \lambda a x_1 + \lambda b + (1-\lambda) a x_2 + b - \lambda b \\ &= \lambda f(x_1) + (1-\lambda) f(x_2) \end{aligned}$$

☐

Theorem 3.3.1. *All linear programs are convex.*

Proof. We want to show that for any linear program, the set of feasible solutions and the objective function are convex. By Lemma 3.3.3, and Corollary 3.3.0.1, we have that all linear programs are convex. ☐

3.4 Quadratic Programs

Definition 3.4.1. *(Quadratic Program) A* ***quadratic program*** *is mathematical program that has constraints constant, linear or quadratic, and a quadratic objective function.*

Definition 3.4.2. *(Quadratic Formulation) Let* $f, g_i : \mathbb{R}^n \to \mathbb{R}$ *be quadratic functions. Let* $x^T = (x_1, x_2, \cdots, x_n)$*. A quadratic program can be written as:*

$$\begin{aligned} &\textit{maximize} && f(x) \\ &\textit{subject to} && g_i(x) = b_i, i = 1, \cdots, m \end{aligned} \tag{3.4}$$

Theorem 3.4.1. *Any quadratic program can be written in the following form, let* F, G_i *be symmetric* $n \times n$ *matrices:*

$$\begin{aligned} &\textit{maximize} && \langle F, X \rangle \\ &\textit{subject to} && \langle G_i, X \rangle = b_i, i = 1, \cdots, m \\ & && X = xx^T, x^T = (1, x_1, \cdots, x_n) \end{aligned} \tag{3.5}$$

Proof. This is true by Lemma 2.1.2 □

The linear relaxation of Max-Cut does not give the best integrality gap. The relaxation of the quadratic formulation of Max-Cut provides a much better integrality gap as we will see in later Chapters.

Corollary 3.4.1.1. *Any quadratic program with equality constraints on* n *variables can be written in the following form:*

$$\begin{aligned} &\textit{maximize} && \langle F, X \rangle \\ &\textit{subject to} && \langle G_i, X \rangle = b_i, i = 1, \cdots, m \\ & && X \succeq 0, X \in \mathbb{R}^{n \times n} \\ & && rank(X) = 1 \end{aligned} \tag{3.6}$$

Proof. See Corollary 2.1.2.1. □

Using Theorem 3.4.1, we can formulate the quadratic program of Max-Cut by the following:

Let $x = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$. Let $X = xx^t \begin{bmatrix} x_1^2 & \cdots & x_1x_n \\ x_2x_1 & \cdots & x_2x_n \\ \vdots & \cdots & \vdots \\ x_nx_1 & \cdots & x_n^2 \end{bmatrix}$.

Define the coefficient matrix F as the following: For every edge $ij \in E$, let the ij^{th} and ji^{th} entries of F be $-\frac{1}{2}$, and the ii^{th}, jj^{th} entires be $\frac{1}{4}$.

Let G_i be an $n \times n$ matrix where all entries are 0, except $(G_i)_{ii} = 1$.

Finally, b is an $n \times 1$ vector of all 1's.

We write the quadratic formulation of Max-Cut in function form for simplicity by expanding the inner products, $\langle F, X\rangle$ and $\langle G_i, X\rangle$.

$$\begin{aligned} \text{maximize} \quad & \sum_{ij\in E} \frac{(x_i - x_j)^2}{4} \\ \text{subject to} \quad & x_i^2 = 1, \forall i \in V \end{aligned} \tag{3.7}$$

We show that this is indeed the formulation of Max-Cut by showing that a solution is feasible and optimal for Max-Cut if and only if it is also feasible for this formulation.

Proof. Let $x_i = 1$ if $i \in S$, and -1 if $i \in V\backslash S$, then this satisfies the constraint $x_i^2 = 1$. Consider an arbitrary edge $\{i, j\} \in E$, if $x_i \neq x_j$, then the edge $\{i, j\}$ is cut. We want to check if the objective function counts one whenever $x_i \neq x_j$ and 0 otherwise.

Clearly, $\frac{(x_i - x_j)^2}{4} = 1$ whenever $x_i \neq x_j$, and 0 otherwise.

Now suppose we have a solution of x_i's to the quadratic program. Let $S := \{i \in V | x_i = 1\}$. Since x_i's are an optimal solution, clearly it provides us with an optimal S. □

3.5 Semi-Definite Programming

Definition 3.5.1. *(Semi-Definite Programs) A* ***semi-definite program*** *is any optimization problem of the following form:*

$$\begin{aligned} \inf_X \quad & \langle F, X\rangle \\ \textit{subject to} \quad & \langle G_i, X\rangle = b_i, i = 1, \cdots, m \\ & X \succeq 0, X \in \mathbb{R}^{n\times n}, \end{aligned} \tag{3.8}$$

where $F, G_i \in \mathbb{R}^{n\times n}$ *are given symmetric, constant matrices.*

Theorem 3.5.1. *All quadratic programs can be relaxed to a semi-definite program.*

Proof. Let Q be a quadratic program. Consider the same mathematical program with the constraint $rank(X) = 1$ removed, call this program Q_R. We want to show that Q_R is the semi-definite relaxation of Q. Clearly, Q_R allows more solutions including X's of rank 1. Hence Q_R is a semi-definite relaxation of Q. □

An alternative way to write a semi-definite program is by replacing the variables with inner products of vectors. Max-Cut is one example where its semi-definite relaxation is actually in the form of a *vector program.*

Definition 3.5.2. *(Vector Program) A* ***vector program*** *is a mathematical program where variables are inner products of vectors.*

Corollary 3.5.1.1. *Every semi-definite program can be written as a vector program.*

Proof. Since $X \succeq 0$, we know there exist vectors x_i, x_j such that $X = \langle x_i, x_j \rangle$. Thus we can write the variables as an inner product of vectors. □

Theorem 3.5.2. *The following is a relaxation of the quadratic program for Max-Cut into a semi-definite program:*

$$\begin{aligned} \textit{maximize} \quad & \sum_{ij \in E} \frac{\|v_i - v_j\|^2}{4} \\ \textit{subject to} \quad & \|v_i\|^2 = 1, \forall i \in V \\ & v_i \in \mathbb{R}^{|V|}, \forall i \in V \end{aligned} \tag{3.9}$$

Proof. We see how to get this objective function from the quadratic program. We begin with $(x_i - x_j)^2 = x_i^2 - 2x_i x_j - x_j^2 = X_{ii} - 2X_{ij} - X_{jj}$. Since $X \succeq 0$, there exist vectors v_i, v_j such that this is same as: $\langle v_i, v_i \rangle - 2\langle v_i, v_j \rangle - \langle v_j, v_j \rangle$, using the definition of linear product we have $\langle v_i - v_j, v_i - v_j \rangle = \|v_i - v_j\|^2$. By relaxing the integer constraint to n-dimensions, we allow feasible solutions to be n-dimensional vectors as well as one dimensional integers. □

Chapter 4

Generalization of Max-Cut

In this Chapter, we define a variation of Max-Cut we call Generalized Max-Cut. We also formulate a quadratic program for Generalize Max-Cut and relax it into a semi-definite program.

Definition 4.0.1. *(Generalized Max-Cut) Fix two real numbers $a, b \in \mathbb{R}$ such that*

$$\begin{cases} a > 0 \\ -a < b < a \end{cases}$$

Let $G = (V, E)$ be a graph, and $f(S) : S \subseteq V \to \mathbb{R}$ be defined as $f(S) = a|\delta(S)| - b|E - \delta(S)|$. ***Generalized Max-Cut*** *is the problem of finding the maximum value of the function $f(S)$.*

4.1 Formulation of Generalized Max-Cut

We formulate Generalized Max-Cut as follows:

$$\begin{aligned} \text{maximize} \quad & a \sum_{ij \in E} \frac{(y_i - y_j)^2}{4} - b \sum_{ij \in E} \frac{(y_i + y_j)^2}{4} \\ \text{subject to} \quad & y_i^2 = 1, \ \forall i \in \{1, \dots, n\} = V \end{aligned} \tag{4.1}$$

We prove that this is indeed a formulation for Generalized Max-Cut by showing that any feasible solution for the definition is also feasible for the formulation, and vice versa. We also show that the value of the solutions are equal to each other.

Suppose we have a graph $G = (V, E)$, and a $S \subseteq V$ such that for any other $S' \subseteq V$, $|\delta(S)| \geq |\delta(S')|$.

Define

$$y_i = \begin{cases} 1, & \text{if } i \in S \\ -1, & \text{if } i \notin S. \end{cases}$$

What we want to show now is that $\sum_{ij\in E} \frac{(y_i - y_j)^2}{4} = |\delta(S)|$, and $\sum_{ij\in E} \frac{(y_i + y_j)^2}{4} = |E - \delta(S)|$.

Let ij be an arbitrary edge. If it is cut, or it does not have both vertices in the same set, then $y_i \neq y_j$. In this case, we have that $\frac{(y_i - y_j)^2}{4} = 1$, and $\frac{(y_i + y_j)^2}{4} = 0$.

Now suppose ij is not cut. Then $y_i = y_j$, and we have that $\frac{(y_i - y_j)^2}{4} = 0$, and $\frac{(y_i + y_j)^2}{4} = 1$.

Since we are taking the summation over all the edges, it is clear that the objective function will give the total number of edges that are cut multiplied by a, minus the number of uncut edges multiplied by b. Thus we have shown that $\sum_{ij\in E} \frac{(y_i - y_j)^2}{4} = |\delta(S)|$, and $\sum_{ij\in E} \frac{(y_i + y_j)^2}{4} = |E - \delta(S)|$.

4.2 Simplification-Objective Function

We simplify the objective function of the above formulation for easier analysis in later Chapters.

$$\begin{aligned} & a\sum_{ij\in E} \frac{(y_i - y_j)^2}{4} - b\sum_{ij\in \mathbb{E}} \frac{(y_i + y_j)^2}{4} \\ =& a\sum_{ij\in E} \frac{y_i^2 - 2y_iy_j + y_j^2}{4} - b\sum_{ij\in E} \frac{y_i^2 + 2y_iy_j + y_j^2}{4} \end{aligned}$$

Since our constraint in the formulation above states that $y_i^2 = 1, \forall i \in V$

$$\begin{aligned} &= \sum_{ij\in E} \frac{a(1 - 2y_iy_j + 1)}{4} - \sum_{ij\in E} \frac{b(1 + 2y_iy_j + 1)}{4} \\ &= \sum_{ij\in E} \left[\frac{a}{2} - \frac{ay_iy_j}{2}\right] - \sum_{ij\in E} \left[\frac{b}{2} + \frac{by_iy_j}{2}\right] \end{aligned}$$

Now combining the summation gives us:

$$\begin{aligned} &= \sum_{ij\in E} \left[\frac{a}{2} - \frac{ay_iy_j}{2} - \frac{b}{2} - \frac{by_iy_j}{2}\right] \\ &= \sum_{ij\in E} \left[\frac{1}{2}(a - b) - \frac{1}{2}(a + b)y_iy_j\right] \end{aligned}$$

The formulation we will be using from here on will be the following quadratic program of Generalized Max-Cut:

$$\begin{aligned} \text{maximize} \quad & \sum_{ij \in E} \left[\frac{1}{2}(a-b) - \frac{1}{2}(a+b) y_i y_j \right] \\ \text{subject to} \quad & y_i^2 = 1, \ \forall i \in \{1, \dots, n\} = V \end{aligned} \tag{4.2}$$

4.3 Relaxation

In this section we relax the quadratic formulation of Generalized Max-Cut into a vector program. We use the same techniques as in Chapter 3 when we relaxed Max-Cut into a vector program. We state the vector program of Generalized Max-Cut without a rigorous proof. We briefly explain that it is indeed a relaxation of the quadratic formulation of Generalized Max-Cut.

Theorem 4.3.1. *The vector relaxation of Generalized Max-Cut is as follows:*

$$\begin{aligned} \textit{maximize} \quad & \sum_{ij \in E} \left[\frac{1}{2}(a-b) - \frac{1}{2}(a+b) \langle v_i, v_j \rangle \right] \\ \textit{subject to} \quad & \|v_i\| = 1, \ i = 1, \dots, n = |V| \\ & v_i \in \mathbb{R}^n, i = 1, \dots, n. \end{aligned} \tag{4.3}$$

Proof. Since feasible solutions are now in $\mathbb{R}^n$, it will contain the integer solutions as well. Hence this is a relaxation of the quadratic problem. □

Theorem 4.3.2. *The semi-definite relaxation of Generalized Max-Cut is a convex program.*

Proof. The semi-definite program of Generalized Max-Cut is a linear program with the added constraint $X \succeq 0$. Thus we only need to show that the set $C = \{X \in \mathbb{R}^{n \times n} | X \succeq 0\}$ is convex.

Let $\lambda \in [0,1]$, we want to show that for any $X_i, X_j \in C$, we have $\lambda X_i + (1-\lambda) X_j \in C$. Since each $X_i \succeq 0$, we know that $v X_i v^T \geq 0$ for all $v \in \mathbb{R}^n$, so it is enough to show that $v[\lambda X_i + (1-\lambda) X_j] v^T \geq 0$.

$$\begin{aligned} v[\lambda X_i + (1-\lambda) X_j] v^T &= v \lambda X_i x^T + x(1-\lambda) X_j v^T \\ &= \lambda v X_i v^T + (1-\lambda) v X_j v^T \\ &\geq 0 \end{aligned}$$

□

Chapter 5

Algorithms

An algorithm is a step-by-step method for solving a problem. Many types of algorithms are designed to not only solve problems, but to solve them as efficiently and exactly as possible [7]. However, many problems do not have an efficient algorithm that outputs an exact solution [7]. We therefore aim to design algorithms that output approximate solutions. In this section, we discuss various approximation algorithms and their run-time.

5.1 Approximation Algorithms

Suppose M is an optimization problem, let I be an instance to M. We denote the optimal value of this problem on instance I to be $\mathrm{OPT}_M(I)$. Let A be any algorithm, we denote $\mathrm{SOL}_M^A(I)$ to be the value achieved by A on instance I.

Definition 5.1.1. *(Approximate Algorithm) For* ***maximization*** *problems: An algorithm is called an* ***approximation algorithm*** *if there is a real number* $\alpha \in [0, 1]$ *such that for all instances* I*, then* $\mathrm{SOL}_M^A(I)$ *is at least* $\alpha \cdot \mathrm{OPT}_M(I)$*.*

Note that when $\alpha = 1$, then $\mathrm{SOL}_M^A(I) = \mathrm{OPT}_M(I)$ meaning we have found the exact optimal solution. Hence, we want to design algorithms where α is as close to 1 as possible.

All algorithms must terminate at some step and return with an output. The *run-time* of an algorithm is the number of steps an algorithm takes to terminate. We call an algorithm A *efficient* if it terminates within a polynomial number of steps with respect to the size of the input.

The inputs we work with in this thesis are all graphs, so it is useful to define explicitly what this means for graphs. We define the *size* of a graph $G = (V, E)$ to be the number of vertices, denoted by $|V|$. A graph has vertices and edges, hence the storage required is larger than $|V|$. However, the number of edges is at most $\binom{n}{2}$, which is polynomial with respect to $|V|$, meaning the algorithms used still terminate in a polynomial number of steps with respect to the size of the input.

5.2 Expected Value of Generalized Max-Cut

All algorithms presented in this thesis use randomness. The solutions given by these algorithms on a fixed instance vary each time due to randomness, so we replace the value of the solution of an algorithm on a given problem with its expected value.

Suppose we are given a $G = (V, E)$, and a randomly chosen $S \subseteq V$. Define random variables, Y_i, where $i \in V$ such that

$$\begin{cases} Y_i = 1 & \text{if } i \in S \\ Y_i = -1 & \text{otherwise} \end{cases}$$

Lemma 5.2.1. *The expected performance of any randomized algorithm on Generalized Max-Cut can be calculated using:*

$$\sum_{ij \in E} \left[\frac{1}{2}(a-b) - \frac{1}{2}(a+b)\mathbb{E}\big[Y_i Y_j\big] \right]$$

Proof. We use the simplified version of the objective function as presented in Section 4.2.

If X is the value of any feasible solution to the formulation of Generalized Max-Cut, then we have that:

$$X = \sum_{ij \in E} \left[\frac{1}{2}(a-b) - \frac{1}{2}(a+b)Y_i Y_j \right]$$

We want to find $\mathbb{E}[X]$:

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{ij \in E} \frac{1}{2}(a-b) - \frac{1}{2}(a+b)Y_i Y_j \right]$$

Using linearity of expectation function, we have that:

$$\begin{aligned} &= \sum_{ij \in E} \mathbb{E}\left[\frac{1}{2}(a-b) - \frac{1}{2}(a+b)Y_i Y_j \right] \\ &= \sum_{ij \in E} \left[\frac{1}{2}(a-b) - \frac{1}{2}(a+b)\mathbb{E}\big[Y_i Y_j\big] \right] \end{aligned}$$

□

The only variation between different algorithms is the expected value $\mathbb{E}\big[Y_iY_j\big]$, which will be calculated separately in each analysis.

5.3 $P \neq NP$

$P \neq NP$ is a famous conjecture that holds relevance in many fields of research. In this section, we focus on the importance of $P \neq NP$ in computational complexity theory. We begin with an introduction to computational complexity theory, defining what the sets P and NP are, and their implications on Max-Cut.

Computational complexity classifies problems by the difficulty of their solvability [18]. In most cases, the difficulty of a problem is quantified by how much time or space is required to solve the problem.

We say that a problem can be solved *efficiently* or *fast* if it can be solved in a polynomial number of steps with respect to the size of the input to the problem. Every problem we have encountered so far is a combinatorial optimization problem, and in most cases can be reduced to a *decision problem.*

Definition 5.3.1. *(Decision Problem) A* ***decision problem*** *is a problem to which the answer is either yes or no.*

P is defined as the set of *decision problems* that can be answered fast [18]. NP is the set of decision problems whose solutions can be *verified* fast. It is important to note that the solutions for the problems in NP are not required to be found fast.

The decision version of Max-Cut is a problem in NP. Given a graph $G = (V, E)$ and an integer $k \in \mathbb{Z}$, is there a partition of V that induces an edge cut set of at least size k?

Theorem 5.3.1. *The optimization version of Max-Cut can be solved fast if and only if the decision version of Max-Cut can be solved in polynomial time with respect to the size of the input.*

Proof. We show that if the optimization version of Max-Cut can be solved fast, then so can be decision version, and vice versa. Suppose we can find the size of the largest edge cut on a given graph fast. Then on input $G = (V, E)$ and k, return yes if k is less or equal to the largest size, and no, otherwise.

Now suppose we have an algorithm D that runs in polynomial time with respect to the size of the input: $G = (V, E)$, k and outputs yes if there is a an edge cut in G of size at least k, and no, otherwise. Consider the following algorithm:

```
for $i = 1, \cdots, |V|$, do:
    run $D$ on $(G, i)$
    if $D$ returns no:
        return $i - 1$
end
```

Clearly, this algorithm runs in polynomial time with respect to $|V|$. This algorithm checks to see if there is a cut of size i and increase i by 1 each time until there is no cut of size i, meaning the previous size is the largest cut. □

Max-Cut is a problem in NP because we can verify any proposed solutions fast. Suppose we are given a graph $G = (V, E)$, an integer k and a partition of vertices $S \subseteq V$. It is easy to count how many edges are cut by the partition S, and to check if that number is greater or less than the integer k. What makes Max-Cut interesting is that moreover, Max-Cut is *NP-hard*, meaning that all problems in NP can be reduced to Max-Cut in polynomial time. This NP-hard property indicates that if we can solve Max-Cut in polynomial time, then we can solve every problem in NP in polynomial time, effectively proving that $P = NP$.

We assume the conjecture $P \neq NP$ in this thesis, hence the integer formulation of Max-Cut cannot be solved fast. However, the relaxations of Max-Cut can be solved fast due its convexity property. Algorithms such as the Ellipsoid Method are capable of solving problems fast precisely due to the fact that they are convex . Hence relaxation are a useful tool for our analysis.

5.4 Algorithm Analysis

We analyze the values of solutions returned by running approximation algorithms using the definition of approximation algorithms. Recall that for any approximation algorithm A, there exist some $\alpha \in [0, 1]$ such that $SOL_A^M \geq \alpha OPT_M$ for some maximization problem M. We rearrange this inequality to $\frac{SOL_A^M}{OPT_M} \geq \alpha$. If this inequality holds for all instances to M, then α can be referred to as the α approximation ratio of algorithm A.

We redefine the above notations and definitions to be specific to Generalized Max-Cut as it is the only problem we will be analyzing. Recall that Generalized Max-Cut is a maximization problem whose instances are graphs.

Denote $OPT(G)$ to be the optimal value of Generalized Max-Cut for any graph G, and $\mathbb{E}[SOL_A(G)]$ to be the expected value of the solution of Generalized Max-Cut obtained by running randomized algorithm A on an instance G. If we can show that $\frac{\mathbb{E}[SOL_A(G)]}{OPT(G)} \geq \frac{\mathbb{E}[SOL_A(G)]}{REL(G)}$, then we can claim that A is an α-approximation algorithm, where $\frac{\mathbb{E}[SOL_A(G)]}{REL(G)} \geq \alpha$ for some $\alpha \in [0, 1]$. We can also claim that the relaxation of Generalized Max-Cut has integrality gap at least α. Some interesting observations for maximization problems:

1. Linear relaxations, and semi-definite programs can be solved fast, in other words in polynomial-time with respect to the size of the input. Methods such as the Ellipsoid method can solve such programs fast due to the convexity property [7].

2. On any given graph G, we have that $REL(G) \geq OPT(G)$.

3. On any given graph G and approximation algorithm A, we have that $\mathbb{E}[SOL_A(G)] \leq OPT(G)$

Figure 5.1: Ratio Scale

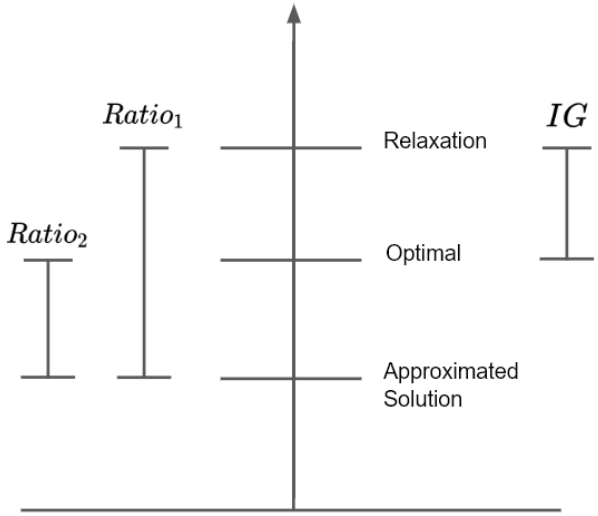


Figure 5.1 roughly (not to scale) shows the relationship between an optimal solution, a relaxed optimal solution and an approximated solution for a maximization problem. The relaxation allows more feasible solutions, hence the value of its optimal solution is at least the the value of the optimal one. The approximated value is more than the optimal value. We claim that $\frac{\mathbb{E}[SOL_A(G)]}{OPT(G)} \geq \frac{\mathbb{E}[SOL_A(G)]}{REL(G)}$ for some graph G. By showing that $Ratio1$ is not too big, we also show that the $Ratio2$ is not too big. Similarly for the IG , which is at most equal to $Ratio1$. From Theorem 3.2.1, we can claim that the linear program relaxation of Max-Cut is bound to emit an approximation algorithm that cannot perform better than $\frac{1}{2}$ times the optimal value.

In addition to the ratio comparison, we also analyze the difference comparison between the relaxed optimal value and the approximated value of Generalized Max-Cut. We will see in later Chapters that the ratios obtained are not always valid, hence it makes sense to use alternative ways such as difference comparison to analyze the algorithms used. We use the formula $\mathbb{E}[SOL_A(G)] - OPT(G)$ to calculate the difference for the algorithms presented.

Chapter 6

Sahni and Gonzales

Since the introduction of Max-Cut, the best approximation algorithm has been a trivial algorithm that had no dependency on input. This algorithm was introduced in 1976 by Sartaj Sahni and Teofilo Gonzalez in their paper P-Complete Approximation Problems [20]. We run this algorithm on Generalized Max-Cut for a reference on how much of an improvement other algorithms are in later Chapters.

6.1 $\frac{1}{2}$-Approximation Algorithm

In this section, we find the approximate ratio of Generalized Max-Cut and the Coin Flip algorithm to be $\dfrac{1+k}{2}$. We label this algorithm to be *CF (Coin Flip)*. On input graph $G = (V, E)$ and (a, b):

Algorithm 1: CF

```
let $S = \varnothing$
for each $i \in V$, do:
    add $i$ to $S$ with $\frac{1}{2}$ probability
    leave $i$ as it is in $V$ with $\frac{1}{2}$ probability
end
```

We calculate the expected performance of this algorithm on our problem, Generalized Max-Cut as follows:

Lemma 6.1.1. *The expected performance of the Coin Flip algorithm on Generalized Max-Cut is* $\sum_{ij\in E}\left[\frac{1}{2}(a-b)\right]$.

Proof. Using Theorem 2.1, we calculate $\mathbb{E}[Y_iY_j]$:

$$\begin{aligned}\mathbb{E}_{SG}[Y_iY_j] &= (1)Pr\big[y_i = y_j\big] - Pr\big[y_i \neq y_j\big]\\ &= (1)Pr\big[y_i = y_j\big] - (1 - Pr\big[y_i = y_j\big])\\ &= \frac{1}{2} - (1 - \frac{1}{2})\\ &= 0\end{aligned}$$

By Lemma 3.1, the expected performance of the Coin Flip algorithm on Generalized Max-Cut is:

$$\begin{aligned}&\sum_{ij\in E}\left[\frac{1}{2}(a-b) - \frac{1}{2}(a+b)\mathbb{E}\big[Y_iY_j\big]\right]\\ =&\sum_{ij\in E}\left[\frac{1}{2}(a-b) - \frac{1}{2}(a+b)(0)\right]\\ =&\sum_{ij\in E}\left[\frac{1}{2}(a-b)\right]\end{aligned}$$

□

We calculate the comparative ratio of Generalized Max-Cut and the approximated value by the Coin Flip algorithm. As $\sum_{ij\in E}\left[\frac{1}{2}(a-b)\right]$ has no dependency on the edges ij, we write it as $\left[\frac{1}{2}(a-b)\right]|E|$, where $|E|$ is the number of edges in the input graph G. Recall that the value of Generalized Max-Cut is $\sum_{ij\in E}\left[\frac{1}{2}(a-b) - \frac{1}{2}(a+b)y_iy_j\right]$, the optimal value is at most when every edge is cut. When $y_iy_j = -1$ for all $ij \in E$, we have $\left[\frac{1}{2}(a-b) + \frac{1}{2}(a+b)\right]|E|$. On input $G = (V, E)$ and (a, b), we have the relation $OPT(G) \leq \left[\frac{1}{2}(a-b) + \frac{1}{2}(a+b)\right]|E|$, where $OPT(G)$ is the optimal value of Generalized Max-Cut on input graph G and (a, b).

We scale (a, b) to one variable $k = -\frac{b}{a}$ for simplicity. The expected performance of the Coin Flip algorithm on Generalized Max-Cut, $\frac{1}{2}(a-b)|E|$ is rewritten as $\frac{a}{2}(1+k)|E|$, and the optimal value is at most $\left[\frac{a}{2}(1+k) + \frac{a}{2}(1-k)\right]|E| = a|E|$.

We calculate approximate ratio using the expected performance of the Coin Flip algorithm on Generalized Max-Cut, and the value obtained from solving the semi-definite program of Generalized Max-Cut.

Theorem 6.1.1. *The approximate ratio of Generalized Max-Cut and the Coin Flip algorithm is* $\frac{1+k}{2}$.

Proof. Let $\mathbb{E}[SOL_{CF}(G)]$ be the expected performance of the Coin Flip algorithm of Generalized Max-Cut on input $G = (V, E)$, and (a, b). Let $OPT(G)$ be the optimal value of Generalized Max-Cut on the same input. Let $k = -\frac{b}{a}$, we have:

$$\begin{aligned}
\frac{\mathbb{E}[SOL_{CF}(G)]}{OPT(G)} &\leq \frac{\mathbb{E}[SOL_{CF}(G)]}{a|E|} \\
&= \frac{\frac{a}{2}(1+k)|E|}{a|E|} \\
&= \frac{1+k}{2}
\end{aligned}$$

□

Chapter 7

Goemans and Williamson

In 1995, a breakthrough was made by Michel X. Goemans and his student David P. Williamson [9]. A 0.878 approximation algorithm for Max-Cut was published, this algorithm involves solving a relaxation of Max-Cut and randomized rounding. In this chapter, we closely follow their paper, and run their algorithm and analysis on Generalized Max-Cut.

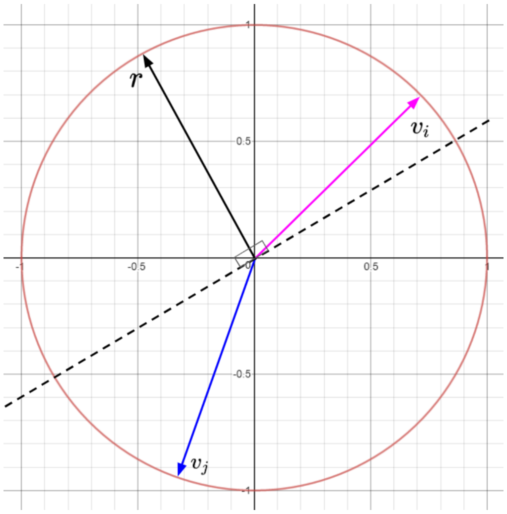
7.1 Hyperplane Rounding Algorithm

The logistics of the Hyperplane Rounding (HR) algorithm involves solving the relaxation of Generalized Max-Cut to get a set of feasible vectors to the vector program. Let $G = (V, E)$ be the input, and $\{v_i : i = 1, \cdots, n = |V|\}$ be a feasible solution to the relaxation of Generalized Max-Cut on input G. We map these vectors to an n-dimensional unit sphere, and pick a random vector r on the n-dimesional sphere. We round each vector v_i to 1 if $v_i \cdot r \geq 0$, and -1, otherwise. This algorithm is referred to as the Hyperplane Algorithm because it produces a partition of the vertices induced by a random hyperplane whose normal is r[9]. The actual algorithm is as follows, on input $G = (V, E)$:

Algorithm 2: HR

```
let $S = \varnothing$
solve the relaxation of Generalized Max-Cut to get a set of vectors $v_i$.
pick a vector $r$ uniformly at random on the $n$-dimensional unit sphere.
for each $v_i$, do:
    add $i$ to $S$ if $v_i \cdot r \geq 0$
    leave $i$ as it is in $V$ if $v_i \cdot r < 0$
end
```

Figure 7.1: HR Algorithm



Although this sphere is n-dimensional, we can project each pair of vectors down onto a two dimensional circle. Our illustration will demonstrate this in two dimensions. The graph on the right shows a pair of vectors v_i, v_j, the randomly picked vector r, and the hyperplane induced by r. In this specific example, v_i and v_j are on different sides of the hyperplane, so the edge ij is cut. The mathematical way of looking at this is for any pair of vectors v_i, v_j, if not both $\langle v_i, r\rangle \geq 0$ and $\langle v_j, r\rangle \geq 0$ or $\langle v_i, r\rangle < 0$ and $\langle v_j, r\rangle < 0$, then the edge ij is cut. The probability of an edge ij being cut is equal to the probability that the randomly picked r lies in the angle between v_i and v_j. By symmetry, we only need to consider half of the unit circle. Thus $Pr[ij \text{ is cut}] = Pr[y_i \neq y_j] = \frac{\theta_{ij}}{\pi}$, where θ_{ij} is the angle between the vectors v_i and v_j. We calculate the expected performance of this algorithm on Generalized Max-Cut as follows:

Theorem 7.1.1. *The probability of an edge ij being cut using the Hyperplane Rounding algorithm is* $\dfrac{\arccos\langle v_i, v_j\rangle}{\pi}$.

Proof. We use the definition of inner products here to justify our theorem.

$$\begin{aligned}
\langle v_i, v_j\rangle &= \|v_i\|\|v_j\|\cos\theta_{ij} \\
\langle v_i, v_j\rangle &= \cos\theta_{ij} && \text{since } \|v_i\| = 1 \quad \forall i \in V \\
\theta_{ij} &= \arccos\langle v_i, v_j\rangle
\end{aligned}$$

□

Lemma 7.1.1. *The expected value of the Hyperplane Rounding algorithm on Generalized Max-Cut is*

$$\sum_{ij\in E}\left[\frac{1}{2}(a-b) - \frac{1}{2}(a+b)\left[1 - 2\frac{\arccos\langle v_i, v_j\rangle}{\pi}\right]\right].$$

Proof.

$$\begin{aligned}\mathbb{E}_{HR}[Y_iY_j] &= (1)Pr[Y_i = Y_j] + (-1)Pr[Y_i \neq Y_j]\\ &= (1 - Pr[Y_i \neq Y_j]) - Pr[Y_i \neq Y_j]\\ &= 1 - 2Pr[Y_i \neq Y_j]\\ &= 1 - 2\frac{\arccos \langle v_i, v_j\rangle}{\pi}\end{aligned}$$

Thus we have the expected value $\sum_{ij\in E}\left[\frac{1}{2}(a-b) - \frac{1}{2}(a+b)\left[1 - 2\frac{\arccos\langle v_i, v_j\rangle}{\pi}\right]\right]$ as needed. ☐

We simplify the expected value for later analysis.

$$\begin{aligned}&\sum_{ij\in E}\left[\frac{1}{2}(a-b) - \frac{1}{2}(a+b)\left[1 - 2\frac{\arccos\langle v_i, v_j\rangle}{\pi}\right]\right]\\ &= \sum_{ij\in E}\left[\frac{(a-b)}{2} - \frac{(a+b) - 2(a+b)\frac{\arccos\langle v_i, v_j\rangle}{\pi}}{2}\right]\\ &= \sum_{ij\in E}\left[\frac{(a-b)}{2} - \frac{(a+b)}{2} + \frac{2(a+b)\frac{\arccos\langle v_i, v_j\rangle}{\pi}}{2}\right]\\ &= \sum_{ij\in E}\left[\frac{a-a+2b}{2} + \frac{(a+b)\arccos\langle v_i, v_j\rangle}{\pi}\right]\\ &= \sum_{ij\in E}\left[-b + (a+b)\frac{\arccos\langle v_i, v_j\rangle}{\pi}\right].\end{aligned}$$

7.2 Hyperplane Rounding Analysis

In this section we analyze the ratio comparison and the difference comparison. We replace the integers a and b with $k = -\frac{b}{a}$ for simplicity. By the restrictions on a and b, it is easy to see that $-1 \leq k \leq 1$. We rewrite both the relaxation and the expected value of the Hyperplane Rounding algorithm in terms of k.

The relaxation of Generalized Max-Cut is as follows:

$$\begin{aligned}
&\sum_{ij\in E}\left[\frac{1}{2}(a-b)-\frac{1}{2}(a+b)\langle v_i, v_j\rangle\right]\\
=&\sum_{ij\in E}\left[\frac{1}{2}(a+ak)-\frac{1}{2}(a+(-ak))\langle v_i, v_j\rangle\right]\\
=&\,a\sum_{ij\in E}\left[\frac{1}{2}(1+k)-\frac{1}{2}(1-k)\langle v_i, v_j\rangle\right]
\end{aligned}$$

Now we rewrite the expected value:

$$\begin{aligned}
&\sum_{ij\in E}\left[-b+(a+b)\frac{\arccos\langle v_i, v_j\rangle}{\pi}\right]\\
=&\sum_{ij\in E}\left[ak+(a+(-ak))\frac{\arccos\langle v_i, v_j\rangle}{\pi}\right]\\
=&\,a\sum_{ij\in E}\left[k+(1-k)\frac{\arccos\langle v_i, v_j\rangle}{\pi}\right]
\end{aligned}$$

7.2.1 Approximation Ratio

We calculate the approximation ratio between the expected performance of the Hyperplane Rounding algorithm on Generalized Max-Cut, and the value obtained from solving the semi-definite program of Generalized Max-Cut.

Theorem 7.2.1. *On input $G = (V, E)$, and $k = -\frac{b}{a}$, where $k \in [0, 1]$, the Hyperplane Rounding Algorithm admits an approximate ratio on Generalized Max-Cut of at least $f(k) = (1-g)k + g$, where g is the Goemans-Williamson's 0.878 constant.*

We find the function $f(k) = (1-g)k+g$, where g is the Goemans-Williamson's 0.878 ratio and $k \in [0, 1]$, to be the lower bound function for our approximation ratio.

We compare the ratio of the expected value and the relaxation. Recall that the optimal value of the relaxation is at least the exact optimal value.

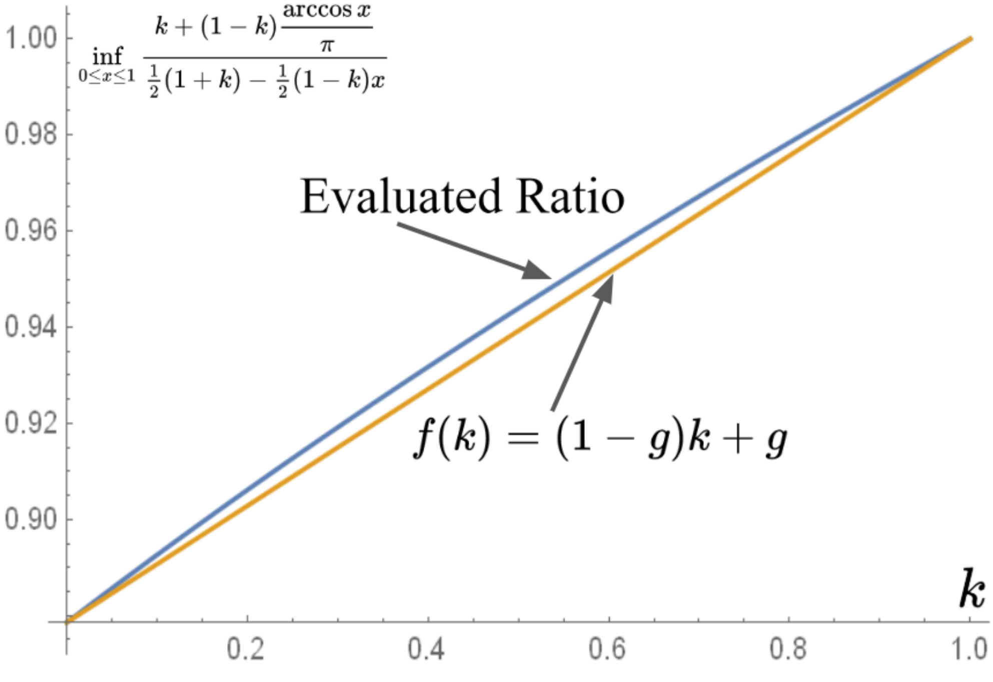
$$\frac{\mathbb{E}[SOL_{HR}(G)]}{OPT(G)} \geq \frac{\mathbb{E}[SOL_{HR}(G)]}{REL(G)} \tag{1}$$

$$= \frac{a\sum\limits_{ij\in E}\left[\left(k+(1-k)\dfrac{\arccos\langle v_i, v_j\rangle}{\pi}\right)\right]}{a\sum\limits_{ij\in E}\left[(\dfrac{1}{2}(1+k)-\dfrac{1}{2}(1-k)\langle v_i, v_j\rangle)\right]} \tag{2}$$

$$\geq \inf_{0\leq x\leq 1}\frac{\left(k+(1-k)\dfrac{\arccos x}{\pi}\right)}{\left(\frac{1}{2}(1+k)-\frac{1}{2}(1-k)x\right)} \tag{3}$$

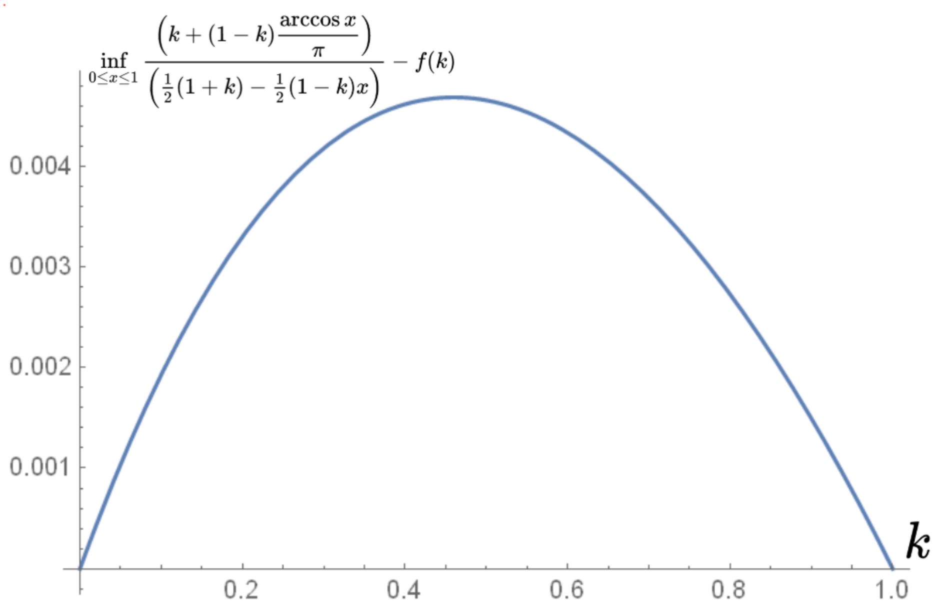
1. Using the fact that $OPT(G) \leq REL(G)$.
2. We substitute the expected value of the Hyperplane Rounding algorithm on Generalized Max-Cut, given input G, through some simplification, we see that the constant a can be cancelled out.
3. Since we know that $\langle v_i, v_j\rangle$ is between 0 and 1, we aim to find the smallest value this ratio can obtain to get a lower bound for our analysis.

Figure 7.2: Approximation Ratio



The values of the above ratio are evaluated numerically. We present this result along with a lower bound function visually in Figure 7.2. Note when $a = 1$ and $b = 0$, Generalized Max-Cut becomes Max-Cut and we obtain the Goemans-Williamson's 0.878 ratio. Another note is that k is bounded between 0 and 1 instead its full domain between -1 and 1. This is due to the reason that when $k < 0$, this ratio may become negative, it makes no sense to analyze a negative ratio as we require $\alpha \in [0,1]$. Thus, in this ratio analysis section, we restrict $k \geq 0$, and consider the case $k < 0$ in the next section using comparative difference.

Figure 7.3: Approximation Ratio Lower Bound Difference



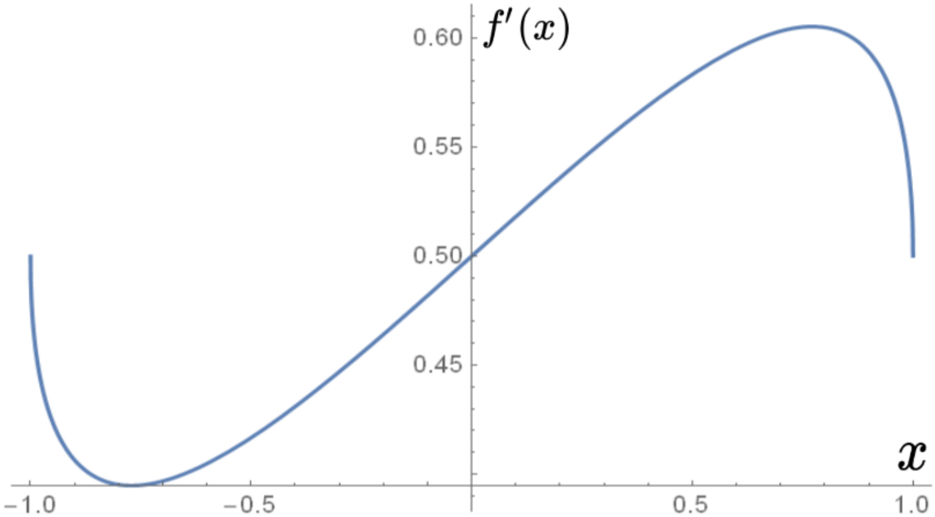
We show that $f(k) = (1-g)k + g$ is a lower bound by graphing the difference between $f(k)$ and the evaluated ratio in Figure 7.3. This graph shows the difference between the lower bound function $f(k)$ and the evaluated ratio is all nonnegative, hence we can claim $f(k)$ to be a fair lower bound for the ratio comparison equation.

7.2.2 Approximation Difference

When $k < 0$, the ratio becomes negative, and cannot be analyzed. We will instead calculate and analyze the comparative difference. In this section, we find the approximate difference to be $a|E|(0.1053)(1-k)$, where $|E|$ is the number of edges in E.

Theorem 7.2.2. *On input $G = (V, E)$, and $k = -\frac{b}{a}$, where $k \in [-1, 1]$. The expected performance of the Hyperplane Rounding algorithm on Generalized Max-Cut is at most $a|E|(-0.1053)(1-k)$ away from the optimal value.*

Figure 7.4: Derivative of $f(x) = \dfrac{\arccos x}{\pi} + \dfrac{x}{2}$



$$\begin{aligned}
\mathbb{E}[SOL_{HR}(G)] - OPT(G) &\geq \mathbb{E}[SOL_{HR}(G)] - REL(G) \\
&= \sum_{ij\in E}\left[a\Big(k + (1-k)\frac{\arccos\langle v_i, v_j\rangle}{\pi}\Big)\right] - \sum_{ij\in E}\left[a(\frac{1}{2}(1+k) - \frac{1}{2}(1-k)\langle v_i, v_j\rangle)\right] \\
&\geq \inf_{-1\leq x\leq 1}\left[a\Big(k + (1-k)\frac{\arccos x}{\pi}\Big) - a\Big(\frac{1}{2}(1+k) - \frac{1}{2}(1-k)x\Big)\right]|E| \\
&= a|E| \inf_{-1\leq x\leq 1}\left[\Big(k + (1-k)\frac{\arccos x}{\pi}\Big) - \Big(\frac{1}{2}(1+k) - \frac{1}{2}(1-k)x\Big)\right] \\
&= a|E| \inf_{-1\leq x\leq 1}\left[k - \frac{1+k}{2} + (1-k)\Big(\frac{\arccos x}{\pi} + \frac{x}{2}\Big)\right] \\
&= a|E|\left[k - \frac{1+k}{2} + (1-k)\inf_{-1\leq x\leq 1}\Big(\frac{\arccos x}{\pi} + \frac{x}{2}\Big)\right]
\end{aligned}$$

Lemma 7.2.1. $\displaystyle\inf_{-1\leq x\leq 1}\left(\frac{\arccos x}{\pi} + \frac{x}{2}\right) \approx 0.394743$

Proof. First we find the derivative of the function $\dfrac{\arccos x}{\pi} + \dfrac{x}{2}$. As Figure 7.4 shows, there are four extrema points, so the derivative gave us four values. We tested each of the x-values, and saw that the minimum occurs when $x = -\dfrac{\sqrt{-4+\pi^2}}{\pi}$, which gives us the minimum y-value of 0.394743. ☐

We simplify $\left[k - \frac{1+k}{2} + (1-k)0.3947\right]$ to $-0.1053 + 0.1053k$, thus giving us a final difference of $a|E|(-0.1053)(1-k)$.

There are specific cases where the Goemans and Williamson constant can be improved upon. In the following section, we attempt to improve this ratio for a specific instance of Generalized Max-Cut.

Chapter 8

2-Dimensional Rounding

Avidor and Zwick showed that the 0.878 ratio can be improved if the relaxation of Max-Cut gave an optimal solution in low dimensions, especially if the solutions are 2-dimensional[3]. Avidor and Zwick use an algorithm that runs one of two possible algorithms with probability β. In this section, we describe the algorithm in detail, and analyze its approximate ratio and difference on Generalized Max-Cut. Moreover, we improve their algorithm accordingly with the inputs of Generalized Max-Cut.

8.1 Gegenbaur Polynomial Rounding

The Gegenbaur Polynomial Rounding algorithm uses a function $f(x)$ to map the unit vectors of the solved relaxation to another set of unit vectors and round these new vectors using the Hyperplane Rounding Algorithm [3]. This is one of the sub-algorithms that will be used in the final algorithm. The Gegenbaur Polynomial we use is the same one used in the paper by Avidor and Zwick, we call this function $f(x)$, and define it in the algorithm [3].

Algorithm 3: Gegenbaur Polynomial Rounding (GP)

```
Let $f(x) = 8x^4 - 8x^2 + 1$
    Solve the relaxation of Generalized Max-Cut to get a set of vectors $v_i$.
    Find vectors $u_1, \ldots u_n$ unit vectors such that $u_i \cdot u_j = f(\langle v_i, v_j \rangle)$.
    Pick a vector $r$ uniformly at random on the $n$-dimensional unit sphere.
    Round each $u_i$ to 1 if $r \cdot u_i \geq 0$, and $-1$ if $r \cdot u_i \leq 0$.
```

Here is a rough proof of the existence of the unit vectors u_i's. Recall that the constraints of a semi-definite program is a positive semi-definite matrix. In this section, solving the semi-definition program of Generalized

Max-Cut will give us a positive semi-definite matrix $M_{ij} = \langle v_i, v_j \rangle$ that has rank 2, the mapping of M with f will give us another positive semi-definite matrix $N_{ij} = \langle u_i, u_j \rangle$, where $\langle u_i, u_j \rangle = f(\langle v_i, v_j \rangle)$. Thus guaranteeing the existence of u_i's. Avidor and Zwick used a convex combination Gegenbauer polynomials and the fact that $f(1) = 1$ in their paper to prove the existence of these vectors [3]. Rujuta .S. Joshi proves in her paper Polynomial Optimization and Discrete Geometry that if $X \in \mathbb{R}^{n \times n}$ such that $X \succeq 0$ with rank at most n, then $X' \succeq 0$, where $X'_{ij} = f(\langle x_i, x_j \rangle)$ and f is a Gegebauer polynomial [19]. We use this result without proof in our thesis. We continue with our analysis of the Gegenbauer Polynomial algorithm by calculate the probability of an edge being cut using this algorithm.

Lemma 8.1.1. *The probability of an edge ij being cut using the Gengenbaur Polynomial Rounding algorithm is* $\dfrac{\arccos f(\langle v_i, v_j \rangle)}{\pi}$.

Lemma 8.1.2. *The expected value of the Gegenbaur Polynomial Rounding algorithm on Generalized Max-Cut is* $\displaystyle\sum_{ij \in E} \left[a\Big(k + (1-k)\frac{\arccos f(\langle v_i, v_j \rangle)}{\pi}\Big) \right]$.

Proof.

$$\begin{aligned}
\mathbb{E}_{GP}[Y_i Y_j] &= (1)Pr[Y_i = Y_j] + (-1)Pr[Y_i \neq Y_j] \\
&= (1 - Pr[Y_i \neq Y_j]) - Pr[Y_i \neq Y_j] \\
&= 1 - 2Pr[Y_i \neq Y_j] \\
&= 1 - 2\frac{\arccos f(\langle v_i, v_j \rangle)}{\pi}
\end{aligned}$$

This gives us that

$$\begin{aligned}
&\sum_{ij \in E} \left[\frac{1}{2}(a-b) - \frac{1}{2}(a+b)\left[1 - 2\frac{\arccos f(\langle v_i, v_j \rangle)}{\pi}\right]\right] \\
&= \sum_{ij \in E} \left[\frac{(a-b)}{2} - \frac{(a+b) - 2(a+b)\frac{\arccos f(\langle v_i, v_j \rangle)}{\pi}}{2} \right] \\
&= \sum_{ij \in E} \left[\frac{(a-b)}{2} - \frac{(a+b)}{2} + \frac{2(a+b)\frac{\arccos f(\langle v_i, v_j \rangle)}{\pi}}{2} \right] \\
&= \sum_{ij \in E} \left[\frac{a - a + 2b}{2} + \frac{(a+b)\arccos f(\langle v_i, v_j \rangle)}{\pi} \right] \\
&= \sum_{ij \in E} \left[-b + (a+b)\frac{\arccos f(\langle v_i, v_j \rangle)}{\pi} \right]
\end{aligned}$$

Substituting $k = -\dfrac{b}{a}$, we have $a \displaystyle\sum_{ij \in E} \left[\Big(k + (1-k)\frac{\arccos f(\langle v_i, v_j \rangle)}{\pi}\Big) \right]$ as the expected value of the Gegenbauer Polynomial Rounding Algorithm on Generalized Max-Cut . □

8.2 2-Dimensional Rounding Algorithm

We combine the above two algorithms using a probability β obtained from Adivor and Zwick's paper.

```
Algorithm 4: 2-Dimensinal Rounding (2D)
1 Let $\beta = 0.963322$, and $1 - \beta = 0.036678$
2     Solve the relaxation of Generalized Max-Cut to get a set of vectors $v_i$.
3     Run the HR algorithm with probability $\beta$, and the GP algorithm with probability $1 - \beta$.
```

Lemma 8.2.1. *The expected value of the 2-Dimensional Rounding Algorithm is*

$$a \sum_{ij \in E} \left[k + (1-k) \left[\beta \left(1 - 2 \frac{\arccos \langle v_i, v_j \rangle}{\pi} \right) + (1-\beta) \left(1 - 2 \frac{\arccos f(\langle v_i, v_j \rangle)}{\pi} \right) \right] \right].$$

Proof.

$$\begin{aligned} \mathbb{E}[Y_i Y_j] &= \beta \mathbb{E}_{HR} + (1-\beta)\mathbb{E}_{GP} \\ &= \beta \arccos \langle v_i, v_j \rangle + (1-\beta) \arccos f(\langle v_i, v_j \rangle) \\ &= \beta \left(1 - 2 \frac{\arccos \langle v_i, v_j \rangle}{\pi} \right) + (1-\beta) \left(1 - 2 \frac{\arccos f(\langle v_i, v_j \rangle)}{\pi} \right) \end{aligned}$$

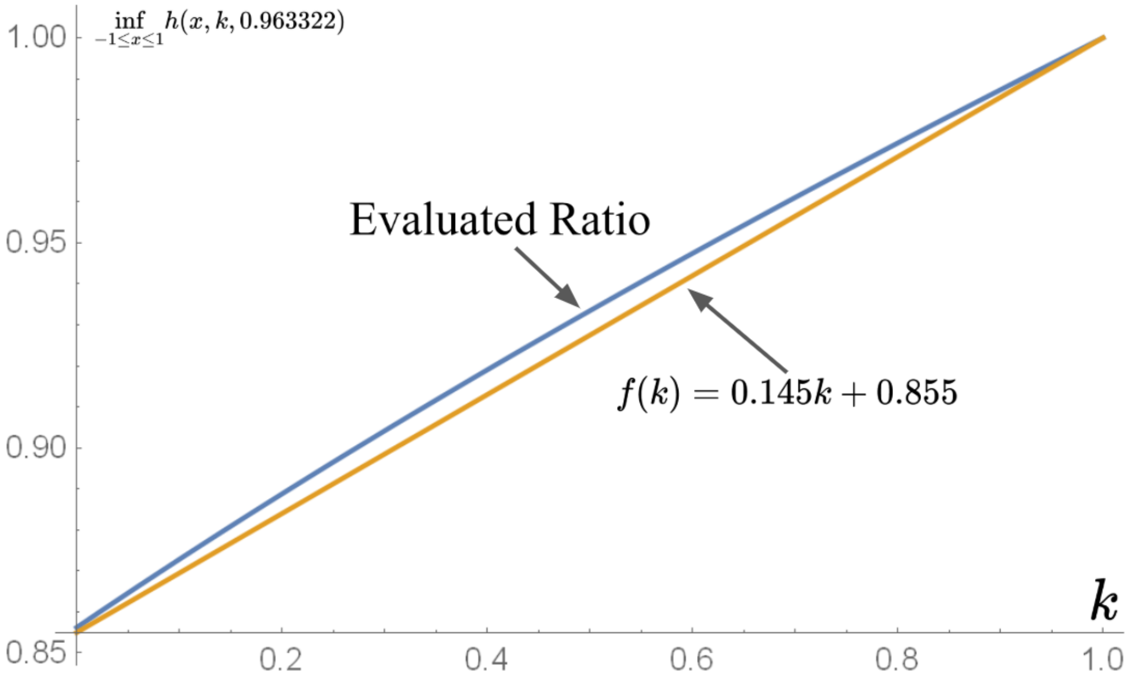
Substituting the above expected value into the full equation, we get the following:

$$a \sum_{ij \in E} \left[k + (1-k) \left[\beta \left(1 - 2 \frac{\arccos \langle v_i, v_j \rangle}{\pi} \right) + (1-\beta) \left(1 - 2 \frac{\arccos f(\langle v_i, v_j \rangle)}{\pi} \right) \right] \right] \text{ as needed.} \qquad \square$$

8.3 Approximation Ratio

In our numerical evaluation of the comparative ratio, we separate parts of the above expected value to understand it better. Let $f(x) = 8x^4 - 8x^2 + 1$, this is the fixed Gegenbauer function obtained from the algorithm. Let $g(x, k) = k + (1-k)\frac{\arccos x}{\pi}$, where $x \in [-1, 1]$. This is the expected value of running the Hyperplane Rounding algorithm on Generalized Max-Cut, if we replace x with $f(x)$ as input we get $g(f(x), k)$, which is the expected value of running the Gegenbaur Polynomial algorithm on Generalized Max-Cut. Finally, let $h(x, k, \beta) = \frac{\beta g(x,k) + (1-\beta) g(f(x), k)}{\frac{1}{2}(1+k) - \frac{1}{2}(1-k)x}$ and we see that finding the infimum of $h(x, k, \beta)$ with $-1 \le x \le 1$ and $\beta = 0.963322$ is the same as finding the infimum of the ratio of the 2-Dimensional Rounding Algorithm. We evaluate $\inf_{-1 \le x \le 1} h(x, k, 0.963322)$ with $k \in [0, 1]$ and plot the results in Figure 8.1.

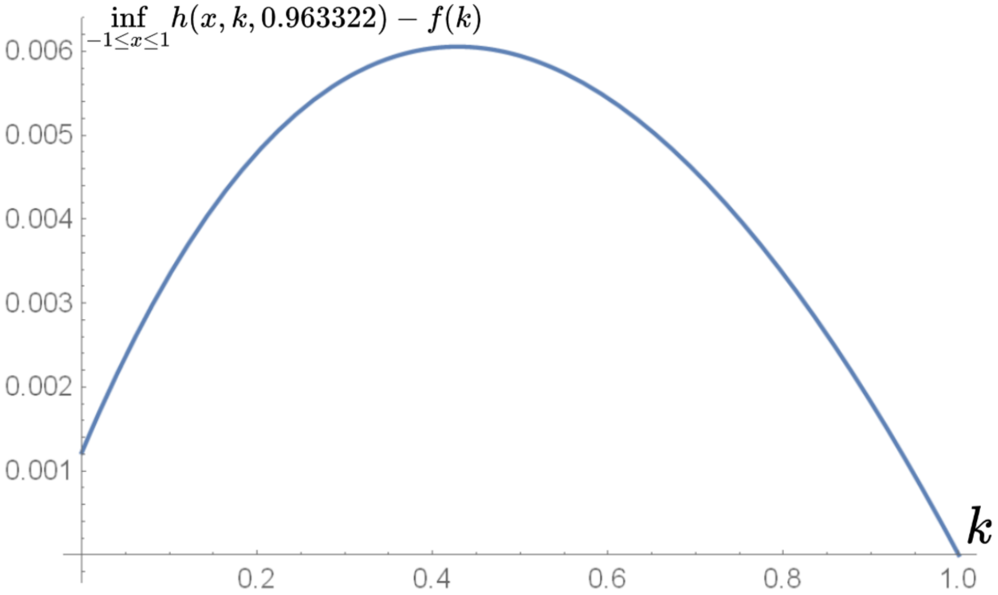
Figure 8.1: 2-Dimensional Rounding Ratio



Theorem 8.3.1. *When the semi-definite program of Generalized Max-Cut has only 2-dimensional optimal solutions, the 2-Dimensional Rounding algorithm on Generalized Max-Cut admits an approximate ratio of at least* $f(k) = 0.145k + 0.855$.

When $k = 0$, the problem becomes Max-Cut and we get $\inf_{-1\le x\le 1} h(x,k,0.963322) = \frac{32}{25+5\sqrt{5}} \approx 0.8844$, the same ratio Adivor and Zwick achieved in their paper for Max-Cut.

Figure 8.2: 2-Dimensional Rounding: Approximated Ratio and $f(k)$ Difference

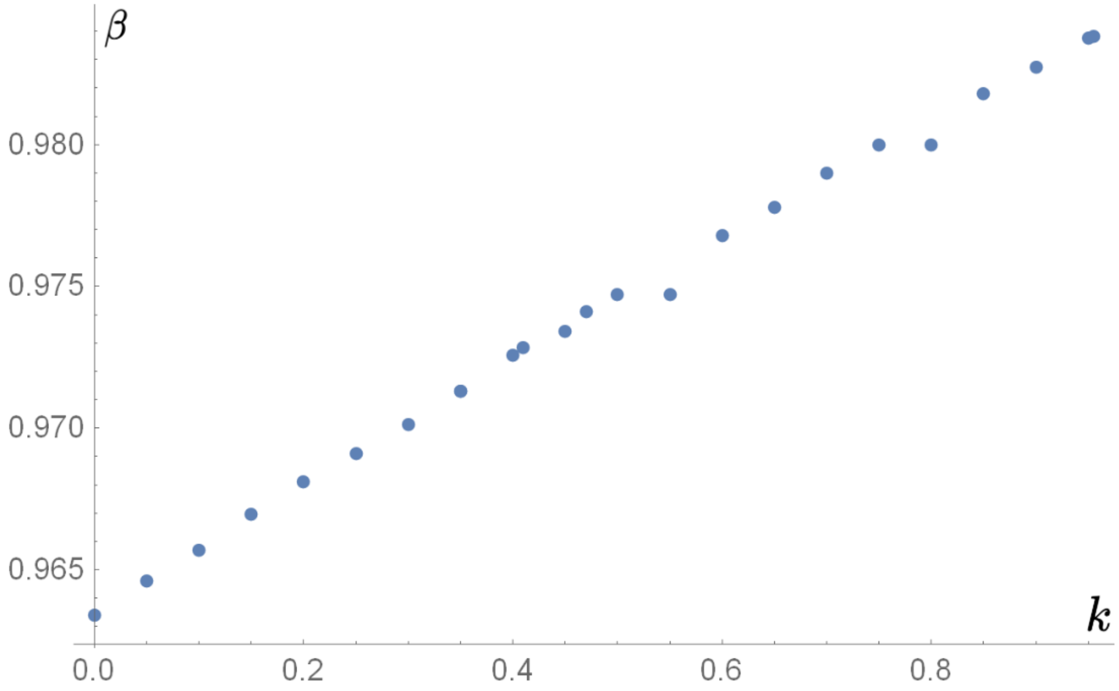


As shown in Figure 8.2, the difference between the lower bound function $f(k)$ and the approximated ratio of the 2-Dimensional Rounding algorithm on Generalized Max-Cut is positive for all $k \in [0,1]$, and are all very close to 0. Hence $f(k)$ is a good estimated lower bound for this approximated ratio.

Note that the lower bound function of the 2-Dimensional algorithm is slightly greater than the lower bound function for the Hyperplane Rounding algorithm, this indicates that the 2-Dimensional algorithm performs better on Generalized Max-Cut.

The above result is based on a fixed β obtained from Adivor and Zwick's paper. We ask the question: is this the β that will give us the optimal value for Generalized Max-Cut? The answer is no. In fact, we can find different β's for each k input. We tested many k values to numerically obtain a list of points (k, β) and plot them to get a line of best fit (shown in Figure 8.2).

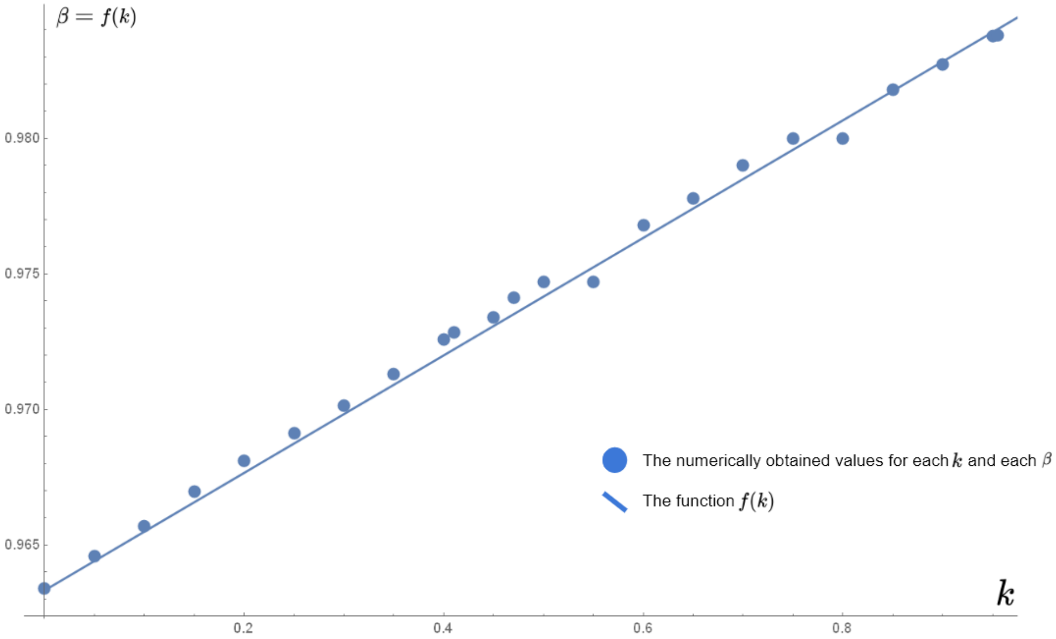
Figure 8.3: Optimal β on Input k



We observe from Figure 8.3 that the best β's are near linear with respect to k. Thus we define the function, $B(k) = 0.963322 + 0.021679k$. This function gives the β that optimizes the ratio on any given $k \in [0, 1]$.

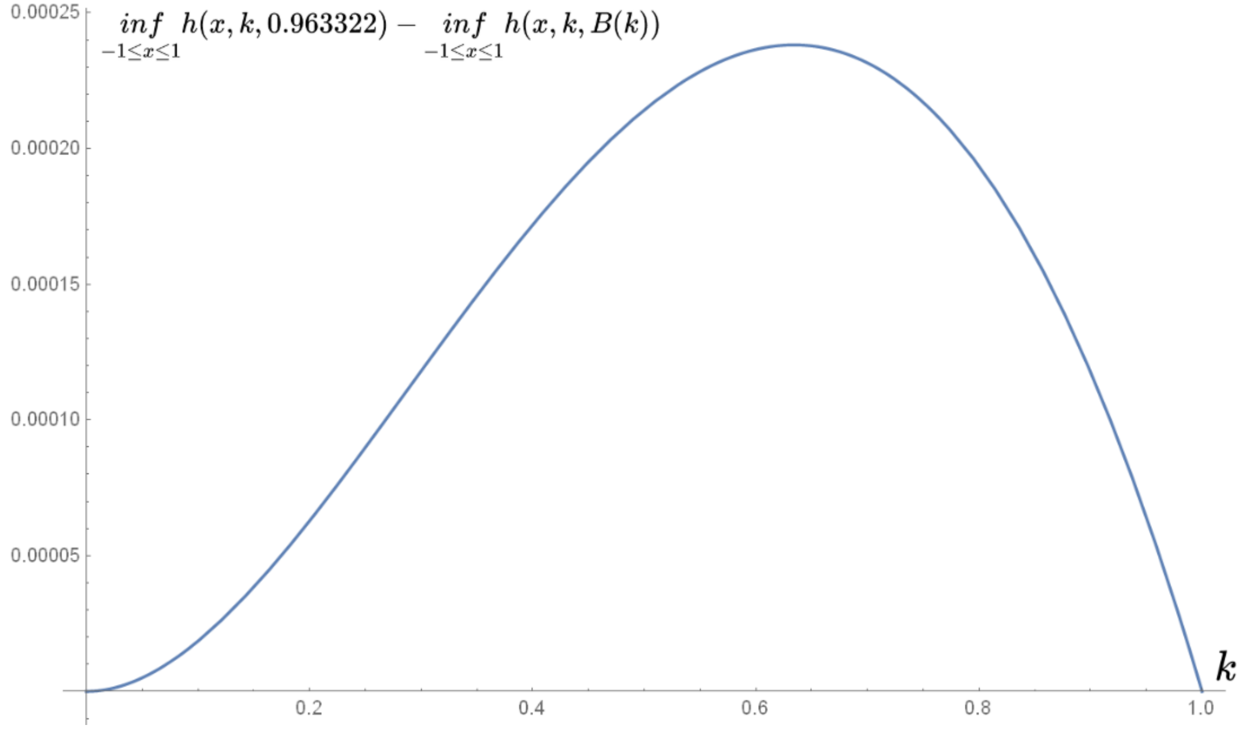
Theorem 8.3.2. *There exist a β for each input k, $\beta = B(k) = 0.963322 + 0.021679k$, such that the 2-Dimensional Rounding algorithm performs better on Generalized Max-Cut with $B(k)$ than with the fixed $\beta = 0.963322$.*

Figure 8.4: Numerical β vs $B(k)$



Not all the numerically obtained values fit on this line, however we show that the function $B(k)$ does indeed output a β that performs better than the constant $\beta = 0.963322$ by graphing the difference between the functions $\underset{-1\leq x\leq 1}{inf} h(x, k, 0.963322)$ and $\inf_{-1\leq x\leq 1} h(x, k, B(k))$ with $k \in [0, 1]$. At $k = 0$ and $k = 1$ we have the difference equal to 0 as we used the same β's to make the $B(k)$ function.

Figure 8.5: Difference of $\inf_{-1\le x\le 1} h(x,k,0.963322)$ and $\inf_{-1\le x\le 1} h(x,k,B(k))$



8.4 Approximate Difference

The above section focuses on $k \in [0,1]$ since the ratio may become negative with $k \in [-1,0]$. In this section, we compare the difference between the expected value of running the Gegenbaur Polynomial algorithm on Generalized Max-Cut and the relaxation.

Theorem 8.4.1. *When the semi-definite program of Generalized Max-Cut has only 2-dimensional optimal solutions, the expected value of 2-Dimensional Rounding algorithm on Generalized Max-Cut is at most* $\frac{|E|(1-k)}{2\pi}(0.656646)$ *away from the optimal value.*

Using the same functions $f(x)$ and $g(x,k)$ as the previous section, we want to evaluate:

$$a \sum_{ij \in E} \Big[\beta g\big(\langle u_i, u_j \rangle, k\big) + (1-\beta) g\big(f(\langle u_i, u_j \rangle), k\big) \Big] - a \sum_{ij \in E} \Big[\frac{1}{2}(1+k) - \frac{1}{2}(1-k)\langle u_i, u_j \rangle \Big]$$
$$\geq \inf_{-1 \leq x \leq 1} \Big[|E| \Big[\beta g(x,k) + (1-\beta) g\big(f(x), k\big) \Big] - \Big[\frac{1}{2}(1+k) - \frac{1}{2}(1-k)x \Big] \Big]$$

Using Mathematica, we simplify the above to the following

$$= \inf_{-1 \leq x \leq 1} |E| \left[\frac{(-1+k)(\pi - \pi x - 2\beta \arccos(x) + 2(-1+\beta)\arccos(1 - 8x^2 + 8x^4)}{2\pi} \right]$$

We see from the simplified version that the constant $\frac{|E|(-1+k)}{2\pi}$ can be factored out, where $-1+k$ is at most 0 since $-1 \leq k \leq 1$. We conclude that the simplified version can be written as the function $t(\beta, x) = -(\pi - \pi x - 2\beta \arccos(x) + 2(-1+\beta)\arccos(1 - 8x^2 + 8x^4)$, which has no dependency on k.

We want to find the β that maximizes the quantity $\inf_{-1 \leq x \leq 1} t(\beta, x)$. We find the derivative of $t(\beta, x)$ with respect to x and found 4 roots. We numerically evaluated each root and found the best β to be 0.9846. This gives us the result:

$$\inf_{-1 \leq x \leq 1} |E| \left[\frac{(-1+k)(\pi - \pi x - 2\beta \arccos(x) + 2(-1+\beta)\arccos(1 - 8x^2 + 8x^4)}{2\pi} \right] = \frac{|E|(k-1)}{2\pi}(-0.656646).$$

Chapter 9

Conclusion

In this thesis we present a new problem we developed based on Max-Cut called Generalized Max-Cut. Our problem generalized Max-Cut by transforming the objective function, we add to the input two integers (a, b) such that $a > 0$ and b, where $-a < b < a$. Instead of counting one for each edge cut and zero for each edge uncut, we add a for each edge cut and subtract b for each edge uncut.

We analyze the performances on various algorithms. The first algorithm is the simple Coin Flip algorithm that has an approximate ratio $\frac{1+k}{2}$, where $k = -\frac{b}{a}$.

The second algorithm is the famous Goemans-Williamson's Hyperplane Algorithm, without restriction of types of graphs as input, we graphed the approximate ratio delivered by this algorithm and find $f(k) = (1-g)k + g$, where g is the 0.878 ratio and $k \in [0, 1]$, to be the lower bound of approximate ratio. The ratio obtained is only significant when $k \geq 0$, thus, we also calculated the approximate difference for when $k < 0$. We found the lower bound of the approximate difference is $a|E|(0.1053)(1-k)$, where $|E|$ is the number of edges of the input graph.

The third algorithm is the 2-Dimensional algorithm that requires the restriction of the input to be such that the semi-definite program of the input graph must have 2-Dimensional solutions. This algorithm uses Gegenbaur polynomials to map these vectors to new unit vectors, then it runs the Hyperplane algorithm on the original vectors with probability $\beta = 0.963322$, and the new vectors with probability $1 - \beta$. We evaluated Generalized Max-Cut using the β presented in the original algorithm, and tested numerically which β's would give the optimal solution for Generalized Max-Cut. We found the function $B(k) = 0.963322 + 0.021679k$, where $k \in [0, 1]$ to be function that provides the best β for each input k. We also found the approximate difference for when $k \in [0, 1]$, we numerically calculated the difference between the optimal value and the expected performance of the algorithm to be $\frac{|E|(1-k)}{2\pi}(0.656646)$.

We used three existing algorithms to test how well they perform on Generalized Max-Cut. There are numerous other algorithms such as the greedy algorithm on dense graphs, or algorithms that require the solutions to the semi-definite program of the input graphs to be 3-dimensional. A natural question is to ask whether other algorithms for Max-Cut can be improved upon on Generalized Max-Cut.

Appendices

Appendix A

Mathematica Code

A.1 Approximation Ratio for Hyperplane Rounding

The following is the Mathematica code used for this thesis. (Not included: brute force testings, and various test cases).

```
f[k_, x_] := 2*(k + (1 - k)*ArcCos[x]/Pi)/((1 + k) - (1 - k)*x)
(*This is the function of the ratio*)

minimizer[k_] := x /. NMinimize[{f[k, x], -1 <= x <= 1}, x][[2]]
(*This gives us the tupple (kvalue, x) that minimizes k*)
approxratio[k_] := f[k, minimizer[k]];
mc = approxratio[0];
(*check to see if ratio makes sense*)
Plot[{approxratio[k], (1 - 0.87856)*k + 0.87856}, {k, 0, 1}]
Plot[{approxratio[k] - ((1 - 0.87856)*k + 0.87856)}, {k, 0, 1}]
```

A.2 Approximation Difference for Hyperplane Rounding

```
f[k_, x_] := 2*(k + (1 - k)*ArcCos[x]/Pi) - ((1 + k) - (1 - k)*x)
(*This is the function of the ratio*)

minimizer[k_] := x /. NMinimize[{f[k, x], -1 <= x <= 1}, x][[2]]
(*This gives us the tupple (kvalue, x) that minimizes k*)
```

```
approxratio[k_] := f[k, minimizer[k]];
mc = approxratio[0];
mc
(*check to see if ratio makes sense*)
Plot[{approxratio[k]}, {k, 0, 1}]
```

A.3 Approximate Ratio for 2-Dimensional Rounding

```
f[x_] := 8*x^4 - 8*x^2 + 1;
gw[x_, k_] := k + (1 - k)*ArcCos[x]/Pi;

test[x_, k_, \[Theta]_] := (\[Theta]*gw[x, k] + (1 - \[Theta])*
      gw[f[x], k])/(1/2*(1 + k) - 1/2*(1 - k)*x);

newcompratio[k_, \[Theta]_] :=
NMinimize[ {test[x, k, \[Theta]], -1 <= x <= 1}, {x}][[1]]
```

A.4 Approximate Difference for 2-Dimensional Rounding

```
f[x_] := 8 x^4 - 8*x^2 + 1;

g[x_, k_] := k + (1 - k)*ArcCos[x]/Pi;

j[x_, k_, \[Beta]_] := (\[Beta]*g[x, k] + (1 - \[Beta])*
      g[f[x], k]) - (1/2*(1 + k) - 1/2*(1 - k)*x);

bb = 0.963322;
b2 = 0;
oldalg[k_] := NMinimize[ {j[x, k, bb], -1 <= x <= 1}, x][[1]];
oldalg2[k_] := NMinimize[ {j[x, k, b2], -1 <= x <= 1}, x][[1]];
Plot[{oldalg[k], oldalg2[k]}, {k, -1, 1}]
```

References

[1] Noga Alon, Benny Sudakov, and Uri Zwick. Constructing Worst Case Instances for Semidefinite Programming Based Approximation Algorithms. *SIAM Journal on Discrete Mathematics*, 15(1):58–72, 2001.

[2] Per Austrin, Siavosh Benabbas, and Konstantinos Georgiou. Better balance by being biased: A 0.8776-approximation for max bisection. *ACM Trans. Algorithms*, 13(1):2:1–2:27, October 2016.

[3] Adi Avidor and Uri Zwick. Rounding two and three dimensional solutions of the sdp relaxation of maxcut. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 14–25, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[4] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[5] Uriel Feige, Marek Karpinski, and Michael Langberg. Improved approximation of max-cut on graphs of bounded degree. *Journal of Algorithms*, 43(2):201 – 219, 2002.

[6] Uriel Feige and Gideon Schechtman. On the integrality ratio of semidefinite relaxations of max cut. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 433–442, New York, NY, USA, 2001. ACM.

[7] Bernd Gärtner and Jirí Matousek. *Approximation Algorithms and Semidefinite Programming.* Springer Publishing Company, Incorporated, 2014.

[8] Bernd Gärtner and Jirí Matousek. *Approximation Algorithms and Semidefinite Programming.* Springer Publishing Company, Incorporated, 2014.

[9] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, November 1995.

[10] Johan Haastad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, July 2001.

[11] Howard Karloff. How good is the goemans–williamson max cut algorithm? *SIAM Journal on Computing*, 29(1):336–350, 1999.

[12] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[13] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 767–775, New York, NY, USA, 2002. ACM.

[14] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM J. Comput.*, 37(1):319–357, April 2007.

[15] Lszl Lovsz. Semidefinite programs and combinatorial optimization, 1995.

[16] Claire Mathieu and Warren Schudy. Yet another algorithm for dense max cut: Go greedy. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 176–182, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[17] W. Keith Nicholson. *Linear algebra with applications.* Toronto : McGraw-Hill Ryerson, 4th ed edition, 2002. First-2nd eds. published under title: Elementary linear algebra, with applications.

[18] Christos H. Papadimitriou. Computational complexity. In *Encyclopedia of Computer Science*, pages 260–265. John Wiley and Sons Ltd., Chichester, UK.

[19] Joshi Rujuta. Polynomial Optimization and Discrete Geometry. pages 25–26, 2015.

[20] Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, July 1976.

[21] Vijay V. Vazirani. *Approximation Algorithms.* Springer-Verlag, Berlin, Heidelberg, 2001.

[22] Wikipedia contributors. Leonid levin — Wikipedia, the free encyclopedia, 2019. [Online; accessed 9-June-2019].

[23] Robin J Wilson. *Introduction to Graph Theory.* John Wiley & Sons, Inc., New York, NY, USA, 1986.