 APPROXIMABILITY OF MODULARITY CLUSTERING AND RELATED RESULTS 

 By 

 DEVENDRA J. DESAI 

 A dissertation submitted to the 
 Graduate School—New Brunswick 
 Rutgers, The State University of New Jersey 
 in partial fulfillment of the requirements 
 for the degree of 
 Doctor of Philosophy 
 Graduate Program in Computer Science 

 Written under the direction of 
 Mario Szegedy 

 and approved by 

 New Brunswick, New Jersey 
 October, 2013
ABSTRACT OF THE DISSERTATION

Approximability of Modularity Clustering and related results

by DEVENDRA J. DESAI

Dissertation Director: Mario Szegedy

For the problem of Modularity Clustering, first introduced by Newman and Girvan in 2004, we are given a graph and the goal is to partition the vertex set into an unknown number of clusters such that we maximize a certain objective function which evaluates the fitness of the clusters. This fitness function measures the statistically surprising distribution of edges between different clusters and in the clusters themselves. Despite having found widespread popularity in the fields of biology and social sciences, this problem is known to be \textit{NP}-hard and up till the work in this thesis, only heuristics were known.

In this thesis, we initiate a study of the approximability of modularity clustering. We give the first approximation algorithms and the first hardness of approximation results for the problem. In doing so, we employ various techniques like semidefinite programming and the regularity lemma. Our main results include a factor 1.0009 inapproximability for dense graphs and a logarithmic (in the
degree) approximation for sparse graphs. We also extend some of these results to directed and weighted graphs and the more general problem of MAXCUT where negative edge weights are allowed.
Preface

Portions of this dissertation are based on work previously published or in preparation by the author [DasGupta and Desai, 2013, 2011; Desai et al., 2013].
Acknowledgements

A thesis is seldom a single person’s effort. This thesis was made possible by a
great many number of people, whom I would like to thank profusely.

First of all, I thank my amazing advisor, Mario Szegedy, for his unwavering
support and guidance that he has provided to me for all these years. His insights
into hard problems are deeply helpful and his love for the subject is infectious.
I am also grateful to my excellent thesis committee, Michael Saks, Shubhangi
Saraf, and Viswanath Nagarajan for their expert advice and support.

I have learned a lot from my co-authors and collaborators, Bhaskar DasGupta,
Special thanks to Bhaskar DasGupta, who first introduced me to the problem of
Modularity, and with whom I have written most of the results in this thesis.

I have had a ton of medical problems over the years, especially at the start of
my doctorate studies. The support and encouragement that the Rutgers Com-
puter Science department has provided me over the years has been invaluable.
I am grateful to my colleagues and professors, especially Martin Farach-Colton,
Gerard Richter, Eric Allender, William Steiger, Rajat Mittal, Pravin Shankar,
Carol DiFrancesco, and Shirley Hinds for helping me cope through the years. I
also owe a great deal to the staff and doctors at the Hemophilia Treatment Center
at Robert Wood Johnson Hospital for taking care of me all those times when I
was ill.

Finally, my gratitude goes out to my family and friends, both here and back
home in India. Their support and encouragement is the reason I could come this far. Thank you.
Dedication

To Aai and Baba.
# Table of Contents

Abstract ................................................................. ii  
Preface ................................................................. iv  
Acknowledgements ...................................................... v  
Dedication ............................................................... vii  
List of Figures .......................................................... xi  

1. Introduction ......................................................... 1  
   1.1. Graphs and Clustering ........................................... 2  
   1.1.1. Model-based Clustering ....................................... 3  
   1.2. The Modularity Parameter ....................................... 4  
      Drawbacks .......................................................... 5  
   1.3. Approximation and Hardness ..................................... 6  
   1.4. Our Results ..................................................... 7  
   1.5. Outline ......................................................... 9  

2. Definitions and Preliminaries ........................................ 10  
   2.1. Definition revisited .............................................. 10  
      2.1.1. Weighted graphs ............................................ 11  
   2.2. Alternative Formulations ....................................... 11  
      2.2.1. Intracluster form .......................................... 12  
      2.2.2. Intercluster form .......................................... 12
List of Figures

2.1. LP relaxation for modularity clustering. ........................................ 18
2.2. SDP relaxation for bimodularity. .................................................... 20
4.1. Greedy algorithm MODALG for modularity clustering in unweighted graphs. ................................................................. 40
4.2. Greedy algorithm WMODALG for modularity clustering in weighted graphs. ................................................................. 42
4.3. Rounding Algorithm for SDP. ......................................................... 45
Chapter 1
Introduction

In this age of analyzing big data, it is sometimes essential to partition the data into groups based on the similarity of various attributes of the data. This is often called clustering of data. Such groupings of data have several advantages, ranging from ease in analyzing groups of data to understanding interactions within groups and between dissimilar groups.

Data can be clustered in various ways, and depending on the application (what information you want to extract from the data), the clustering can either be good or bad. Hence, to measure the quality of the clustering, scientists have come up with various metrics over the years. A recent metric, first introduced by Newman and Girvan [2004], called modularity, has found widespread popularity in the fields of biology and social sciences (Albert [2005]; Albert et al. [1999]). It measures how "statistically surprising" interactions between data inside the clusters look in relation to a global null model which shares some properties with the data.

Given a metric like modularity, it is natural to try to optimize the clusters to generate the best result, but this problem turns out to be $\mathcal{NP}$-hard Brandes et al. [2008]. In spite of this, various heuristics are employed to detect clusterings having large modularity Newman and Girvan [2004]; Newman [2006]; Leicht and Newman [2008]. Prior to the work in this thesis, no approximation algorithms were known that had provable quality guarantees for the optimization version of modularity. Neither were there any hardness results proving the impossibility of
such guarantees.

In this thesis, we initiate a study of the approximability of modularity clustering. We give the first approximation algorithms and the first hardness of approximation results for the problem. In doing so, we employ various techniques like semidefinite programming and the regularity lemma.

We also extend some of these results to the more general problem of MAXCUT with negative edge weights.

1.1 Graphs and Clustering

Many systems of interaction in biology and social science are modeled as a graph (or network) of pairwise interaction of entities Albert [2005]; Albert et al. [1999]. In the simplest setting, a graph can be thought of as a set of vertices and a set of edges, where an edge is a connection between two vertices. One can then add weights to each edge, signifying the robustness or importance of such a connection. Sometimes, edges could be directed from one vertex to the other, signifying a one-way relation between the two.

An important problem for these types of graphs is to partition the nodes into so-called ‘communities’ or ‘modules’ of statistically significant interactions. Such partitions facilitate studying interesting properties of these graph in their applications, such as studying the behavioral patterns of an individual in a society, and serve as important components in the computational analysis of these graph. In this thesis we consider the static model of interaction in which the network interconnections do not change over time.
1.1.1 Model-based Clustering

Simplistic definitions of modules, such as cliques, unfortunately do not apply well in the context of biological and social networks and therefore alternative definitions are most often used. In the model-based community finding approach, one first starts with an appropriate global null model $\mathcal{N}$ of a background random graph and then attempts to place nodes in the same module if their interaction patterns are significantly stronger than that inferred from the null model. The null model $\mathcal{N}$ may provide, implicitly or explicitly, the probability $p_{u,v}$ of an edge between two nodes $u$ and $v$.

As an illustration, suppose that our input is an edge-weighted graph with all weights being positive and normalized between 0 and 1. Then, if $p_{u,v}$ differs significantly from $w_{u,v}$, the weight of the edge between nodes $u$ and $v$, the edge may be considered to be statistically significant; thus, if $p_{u,v} \ll w_{u,v}$ then it is preferable that $u$ and $v$ should be placed in the same module whereas if $p_{u,v} \gg w_{u,v}$ then it is preferable that $u$ and $v$ should be placed in different modules.

As another example, the standard $\{+,-\}$-correlation clustering that appears in the computer science literature extensively Bansal et al. [2004]; Charikar et al. [2005]; Swamy [2004] can be placed in the above model-based clustering framework in the following manner: given the input graph $G$ with each edge labeled as $+$ or $-$, let $H$ be the graph consisting of all edges labeled $+$ in $G$, $p_{u,v} = 0$ (resp. $p_{u,v} = 1$) if the edge $(u,v)$ was labeled $+$ or missing (resp., labeled $-$), the ‘contribution’ of an edge is $A_{u,v} - p_{u,v}$ where $A_{u,v}$ is the $(u,v)^{th}$ entry in the adjacency matrix of $H$ and the total contribution is a function of individual contributions of edges as induced by the clustering.
1.2 The Modularity Parameter

In this thesis, we investigate a model-based clustering approach originally introduced by Newman and subsequently studied by Newman and others in several papers Newman and Girvan [2004]; Newman [2006]; Leicht and Newman [2008].

The null model in this approach is dependent on the degree distribution of the given graph. The intuition is that if one only knew the degrees of the respective nodes of the graph, then the presence of an edge between two low degree nodes is ‘surprising’ (and therefore significant). This should prompt one to put the two nodes in the same cluster. On the other hand, an edge between two very high degree nodes is less surprising and should largely be ignored for clustering purposes. However, the absence of an edge between two very high degree nodes is also surprising and should prompt one to put the two nodes into different clusters.

For any pair \( \{u, v\} \) of vertices in the graph, the following contribution of that pair captures the gist of the previous paragraph:

\[
A_{u,v} - p_{u,v},
\]

where \( A_{u,v} \) is the graph’s adjacency matrix and \( p_{u,v} \propto \text{degree}(u) \cdot \text{degree}(v) \). For a cluster \( C \), the modularity \( Q(C) \) of that cluster is then given by the sum total of the contributions of each pair in the cluster:

\[
Q(C) = \sum_{u,v \in C} A_{u,v} - p_{u,v},
\]

and the total modularity for a clustering \( C \) is the sum total of the modularities of the individual clusters:

\[
Q(C) = \sum_{C \in C} Q(C).
\]

Since we will be dealing with undirected networks (except in Chapter 5), we can assume that \( p_{u,v} = p_{v,u} \). Moreover, if we introduce the (natural) constraint that
the modularity of the clustering where all the nodes are placed into one cluster is 0, then this gives the following choice for $p_{u,v}$:

$$p_{u,v} = \frac{\text{degree}(u) \cdot \text{degree}(v)}{2 \cdot \text{number of edges in the graph}}.$$ 

The above definition of $p_{u,v}$ makes the null model similar to the configuration model (first introduced by Bender and Canfield [1978]) that generates random graphs dependent on a given degree distribution. Thus, one could also think of modularity as measuring the dissimilarity of the given graph with the random configuration model graph that uses the same degree sequence.

**Drawbacks.**

Although the modularity parameter has found widespread use in the analysis of various real-world networks, it suffers from some inherent drawbacks.

Concisely, the modularity measure of a graph charts its dissimilarity to a global null model. Any clustering measure that relies on such global null models, which usually are probabilistic in nature, suffers from the drawback that each node can get attached to any other node of the graph. This behavior does not quite mimic real world scenarios, where other factors such as geographical locality (in the case of social networks) might influence the connections among nodes.

Another important drawback called the resolution limit was observed by Fortunato and Barthélemy [2007]. The cause for this is related to the above objection. When the size of the network becomes very large, the optimal modularity clustering could merge different communities that are sparsely connected to each other into a single community. This is unavoidable due to the use of a global null model. However, one could do a multiresolution analysis of the formed communities, essentially running a local version of modularity clustering inside each community.
1.3 Approximation and Hardness

Research in computer science almost always deals with tackling what appear to be hard problems. Such problems are hard in the computational complexity theoretic sense, and usually they are categorized as \( \mathcal{NP} \)-hard. In the simplest terms, this means that goodness of solutions to such problems can easily (in polynomial time) be checked, but no one knows whether coming up with such solutions is easy (again, in polynomial time). If one did succeed in doing this, one would have proven the famous mathematical conjecture that \( \mathcal{P} \neq \mathcal{NP} \).

To get around this seemingly impossible task, researchers have devised several strategies. One strategy is to come up with efficient heuristics to find good solutions for the hard problems. Another way is to devise polynomial time algorithms that output solutions that are provably close to the optimal solution. These algorithms are called approximation algorithms and the quality of the solution is measured in terms of the approximation factor.

The best approximation factors (corresponding to approximation algorithms) vary greatly between different \( \mathcal{NP} \)-hard problems. Whereas a problem like Knapsack admits efficient algorithms that get arbitrarily close to the optimal solution\(^1\), there are problems like 3-Coloring that only have polynomial approximation factors.

The question of whether it is even possible to come up with efficient algorithms achieving a good approximation factor for a particular problem is called hardness of approximation. For instance, the famous result of Hästad [2001] states that unless \( \mathcal{P} = \mathcal{NP} \), there is no polynomial time approximation algorithm achieving a ratio better than \( \frac{7}{8} \) for the Max-3Sat problem.

---

\(^1\) Such algorithms are called Polynomial Time Approximation Schemes (PTAS).
1.4 Our Results

The purpose of this thesis is to explore the approximability of the modularity parameter. There are two aspects to this, impossibility (or hardness) results and approximation algorithms. In this section, we highlight our main results.

On the hardness front, we show that optimizing modularity over two clusters (a problem we call bimodularity) is \( \text{NP} \)-hard for sparse graphs (Theorem 9). Previously, no hardness result for modularity was known for sparse graphs. This result requires the construction of well-connected sparse graphs, which we provide.

We show that it is \( \text{NP} \)-hard to approximate modularity within a multiplicative factor of 1.0009 (Theorem 12) by starting with a hard instance of maximum independent set on 3-regular graphs. The hardness for bimodularity is straightforward, but extending the result to an arbitrary number of clusters requires a detailed case analysis.

We also show that the standard LP relaxation for modularity (Figure 2.1) has a large integrality gap (Theorem 8), particularly for constant degree expanders. This precludes the use of linear programs to obtain good approximations to modularity. The proof uses the mixing lemma to upper bound the worst case modularity of \( d \)-regular graphs.

Moving on to algorithmic results, we start by showing that calculating optimal bimodularity is a 2 approximation to optimal modularity (Lemma 3). The proof is short and uses the probabilistic method. Moreover, such a result is extremely useful from an approximation standpoint, since we can sometimes concentrate on only approximating bimodularity.

We show an additive approximation scheme for dense graphs (Theorem 20). If we know a constant lower bound for the modularity of a dense graph, this
result translates to a PTAS. The proof is simple and works by reformulating the modularity problem as a max-$k$-cut problem and using the weak regularity lemma.

For sparse graphs, we start by showing a number of lower bounds for modularity in terms of the maximum degree $\Delta$ in the graph (Section 4.2). Some of these lower bounds are constructive, and since the optimal modularity is always upper bounded by 1, translate into efficient approximation algorithms.

But the main reason to prove these lower bounds is to use them in constructing an $O(\log \Delta)$ approximation factor algorithm for bimodularity (Theorem 28 and Theorem 29). This algorithm uses semidefinite programming and randomized rounding to obtain good clusters.

Some of the results for modularity can be generalized to directed graphs (Theorem 30). Both the hardness results, the lower bounds, and the SDP rounding analysis carry over to the directed case, but we have to be more careful while doing the analysis.

The modularity parameter has its quirks. An optimal (or close to optimal) clustering can have a lot of trivial clusters. To combat this problem, we propose two alternatives. The first alternative definition to modularity is to maximize (across clusterings) over the minimum modularity cluster in that clustering. We show that optimizing over this objective is not much different than optimizing bimodularity (Theorem 32). We also consider a simpler alternative to modularity, which uses the Erdos-Renyi null model.

The problem of finding the max cut in a graph having both positive and negative edge weights can be considered as an interesting generalization to optimizing modularity. In this case, even certifying the existence of a non-negative cut in the graph becomes $\mathcal{NP}$-hard. Nevertheless, we show that if a certain bound holds, this
problem becomes easy (Theorem 36). The proof uses the probabilistic method and a modification of the Goemans-Williamson rounding analysis.

1.5 Outline

We first introduce the formal definition of modularity and some notation in Chapter 2. We also prove some preliminary results in this chapter. This lays the groundwork for the results in the subsequent chapters.

The two main hardness results for modularity are proved in Chapter 3. Approximation algorithms and lower bounds for modularity are dealt with in Chapter 4.

We collect some related results and generalizations in Chapter 5. Namely, we look at how the results in the previous chapters apply to directed graphs and we also investigate the approximability of different clustering measures that are related to modularity. The chapter ends with an algorithm for solving special cases of generalized max cut.

We end with a number of open questions in Chapter 6 with a discussion of some interesting open problems related to the study of approximating modularity.
Chapter 2
Definitions and Preliminaries

In this chapter we will formalize the definition and notation for modularity. We will prove some basic but useful facts about the optimization problem and state some known hardness results and some popular heuristics. We will also define useful concepts like expanders, linear programming, and semidefinite programming.

2.1 Definition revisited

Let \( G = (V, E) \) be the given (for now) undirected and unweighted graph with \( n = |V| \) vertices and \( m = |E| \) edges. Let \( A \) be the adjacency matrix of graph \( G \), i.e., \( A_{u,v} = 1 \) if \( u, v \in E \) and \( A_{u,v} = 0 \) otherwise. Let \( d_v \) denote the degree of vertex \( v \in V \), that is, \( d_v = \sum_{u \neq v} A_{u,v} \).

The intuition behind the fitness function, as originally suggested by Newman and Girvan [2004]; Newman [2006], is as follows. Conditional to the degree sequence of the given graph, the probability that an edge \( \{u, v\} \) appears in a random graph is \( \approx \frac{d_u d_v}{2m} \). Thus, if vertices \( u \) and \( v \) belong to the same community, it should contribute the term \( A_{u,v} - \frac{d_u d_v}{2m} \) to the fitness function. So the modularity of a community/cluster \( C \subseteq V \) is defined as

\[
Q(C) = \frac{1}{2m} \sum_{u,v \in C} \left( A_{u,v} - \frac{d_u d_v}{2m} \right).
\]

Note that each pair of vertices \( u \) and \( v \) contribute twice to the above sum. Also
note that $u$ and $v$ are allowed to be equal in the summation, thereby effectively providing a *negative weight* to each vertex.

Now, a clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ has a modularity value of

$$Q(\mathcal{C}) = \sum_{C_i \in \mathcal{C}} Q(C_i) = \sum_{C_i \in \mathcal{C}} \sum_{u,v \in C_i} \left( A_{u,v} - \frac{d_u d_v}{2m} \right).$$

(2.1)

### 2.1.1 Weighted graphs

The above definition can be easily extended to weighted graphs, as was observed by Newman [2004]. We can think of weighted graphs as complete graphs with *non-negative* weights assigned to every edge. The adjacency matrix $A$ thus has the weight of edge $\{u, v\}$ in the entry $A_{u,v}$. \footnote{Note that the diagonal entries of $A$ still remain 0.} For consistency with the unweighted case, we scale all the weights such that any $A_{u,v} \leq 1$. This can be achieved by dividing all the weights by $(A_{u,v})_{\text{max}}$. Scaling does not affect the optimal modularity clusterings.

Finally, we can redefine the parameter $m = \frac{1}{2} \sum_u d_u$. With these new definitions, the modularity of a clustering $\mathcal{C}$ is given by the same formula, i.e. Equation (2.1).

### 2.2 Alternative Formulations

Given a cluster $C \subseteq V$, let

$$\mu(C) = \frac{1}{2m} \sum_{v \in C} d_v$$

denote the fraction of total degree in cluster $C$. In related literature, this quantity is sometimes called the *volume* or the *degree measure* of cluster $C$.

The objective function $Q(\cdot)$ can be equivalently formulated (via simple algebraic manipulation) in two important ways.
2.2.1 Intracluster form

Let \( i(C) \) denote the fraction of total edges inside cluster \( C \subseteq V \).

With our new notation, we can now write the modularity value of a cluster \( C \) as

\[
Q(C) = i(C) - \mu(C)^2. \tag{2.2}
\]

Let us say we are given a clustering \( \mathcal{C} = \{C_1, C_2, \ldots, C_k\} \). Then the modularity of the clustering can be written as

\[
Q(\mathcal{C}) = \sum_{C_i \in \mathcal{C}} Q(C_i) = \sum_{C_i \in \mathcal{C}} (i(C_i) - \mu(C_i)^2). \tag{2.3}
\]

We can write the fraction of total edges that are inside all the clusters as

\[
i(\mathcal{C}) = \sum_{C_i \in \mathcal{C}} i(C_i).
\]

Then in Definition 2.3, we can add all the terms counting the intracluster edges and write the modularity simply as

\[
Q(\mathcal{C}) = i(\mathcal{C}) - \sum_{C_i \in \mathcal{C}} \mu(C_i)^2. \tag{2.4}
\]

2.2.2 Intercluster form

In the previous form, we focused on edges that are inside the clusters. Now given a pair of clusters \( C, C' \subseteq V \), let \( e(C, C') \) denote the fraction of total edges with one endpoint in \( C \) and the other endpoint in \( C' \).

Let us denote the modularity between clusters \( C \) and \( C' \) as \( Q(C, C') \) and define it as

\[
Q(C, C') = 2\mu(C)\mu(C') - e(C, C').
\]

Then overall for a clustering \( \mathcal{C} = \{C_1, C_2, \ldots, C_k\} \),

\[
Q(\mathcal{C}) = \sum_{C_i, C_j \in \mathcal{C}, i < j} Q(C_i, C_j) = \sum_{C_i, C_j \in \mathcal{C}, i < j} (2\mu(C_i)\mu(C_j) - e(C_i, C_j)). \tag{2.5}
\]
We can again simplify the above formula by aggregating the number of outside edges. Let

\[ e(C) = \sum_{C_i, C_j; i < j} e(C_i, C_j) \]

be the fraction of total edges going across all clusters. Then we can add all the terms counting the \textit{intercluster} edges and write

\[ Q(C) = \left( \sum_{C_i, C_j; i < j} 2\mu(C_i)\mu(C_j) \right) - e(C). \tag{2.6} \]

\subsection*{2.3 Basic Results}

Just from the definitions in the previous sections, we can infer the following easy result.

\textbf{Fact 1.} \textit{For any cluster} \( C \subseteq V \),

\[ 2\mu(C) = e(C, \overline{C}) + 2i(C). \]

In other words, a cluster’s total degree is split into edges that are inside the cluster and edges that go to other clusters.

Given an input graph \( G \), the goal of the modularity clustering problem is to find a partition/clustering \( C \) (with unspecified number of clusters) to maximize \( Q(C) \). Let us call this maximum value \( Q(G) \). In other words,

\[ Q(G) = \max_C Q(C). \]

The following facts are easy to check.

\textbf{Fact 2.} \textit{For any clustering} \( C \) \textit{of graph} \( G \),

\[ -0.5 \leq Q(C) \leq 1. \]

Moreover,

\[ 0 \leq Q(G) \leq 1. \]
For a graph $G$, let $Q_k(G)$ be the optimal clustering of $G$ when one is allowed at most $k$ number of clusters. In other words,

$$Q_k(G) = \max_{C: |C| \leq k} Q(C).$$

The following lemma provides the (tight) connection between the two values.

**Lemma 3.** Given a graph $G$, for any $k \geq 1$,

$$1 - \frac{1}{k} \geq Q_k(G) \geq \left(1 - \frac{1}{k}\right) Q(G).$$

**Proof.** We will first prove the upper bound. For any clustering $C$, since all edges could be inside clusters, $i(C) \leq 1$. When $|C| \leq k$, by the Cauchy-Schwarz inequality we have

$$\sum_{C_i \in C} \mu(C_i)^2 \geq \frac{1}{k} \left(\sum_i \mu(C_i)\right)^2 = \frac{1}{k}.$$ 

Now using Definition (2.4) of modularity,

$$Q(C) = i(C) - \sum_{C_i \in C} \mu(C_i)^2 \leq 1 - \frac{1}{k}.$$ 

Since this is true for any $k$-clustering, it is also true for the optimal $k$-clustering.

To prove the lower bound, observe that for $k = 1$, the statement is trivially true. Now consider $k > 1$. We shall use Definition (2.5) of Modularity. Suppose that our optimal clustering $C$ has more than $k$ clusters. We have

$$Q(G) = Q(C) = \sum_{C_i, C_j : i < j} Q(C_i, C_j).$$

We can randomly assign each of the clusters to one of $k$ superclusters. Let $I_{i,j}$ be the indicator random variable of the event $C_i$ and $C_j$ are in different clusters and let $C_k$ denote the random $k$-clustering. It is easy to see that any pair $C_i$ and $C_j$ will contribute $Q(C_i, C_j)$ to the final clustering value if and only if they are not in the same supercluster. Therefore,

$$Q(C_k) = \sum_{i < j} Q(C_i, C_j) I_{i,j}.$$
Thus we get that $Q_k(G)$ is at least
\[
\mathbb{E} Q(C_k) = \sum_{i<j} Q(C_i, C_j) \mathbb{E} I_{i,j} \\
= \left(1 - \frac{1}{k}\right) \sum_{i<j} Q(C_i, C_j) \\
= \left(1 - \frac{1}{k}\right) Q(G). \tag*{\Box}
\]

**Fact 4** (Brandes et al. [2008]). *For a graph $G$, calculating the exact values of $Q(G)$ and $Q_2(G)$ is $\mathcal{NP}$-hard.*

The next lemma shows that the 2-clustering problem can also be viewed as a special kind of *subgraph selection* problem.

**Lemma 5.** For any bipartition $\mathcal{C} = \{C, \overline{C}\}$, $Q(C) = Q(\overline{C})$.

*Proof.* Since the clusters $C$ and $\overline{C}$ have the same outgoing edges, by Fact 1 we get $i(C) - i(\overline{C}) = \mu(C) - \mu(\overline{C})$. Hence using Definition (2.3) of Modularity,
\[
Q(C) - Q(\overline{C}) = i(C) - \mu(C)^2 - (i(\overline{C}) - \mu(\overline{C})^2) \\
= \mu(C) - \mu(\overline{C}) - (\mu(C)^2 - \mu(\overline{C})^2) \\
= 0,
\]
where we have also used the fact that $\mu(C) + \mu(\overline{C}) = 1$. \(\Box\)

### 2.4 Bimodularity, Cuts and Expanders

By Lemma 3 in the previous section, we have that
\[
\frac{1}{2} \geq Q_2(G) \geq \frac{Q(G)}{2}. \tag{2.7}
\]

This crucial fact leads us to consider the approximability of 2-clustering as an interesting special case. Any 2-clustering can be viewed as a cut, and we are
optimizing for a certain cut metric. We will henceforth refer to this cut problem as bimodularity. Thus the bimodularity of a cut \( \{C, \overline{C}\} \) is given by

\[
Q(C, \overline{C}) = 2\mu(C)(1 - \mu(C)) - e(C, \overline{C}).
\]

(2.8)

This means that in order to maximize bimodularity, we need to look for balanced cuts with very few cut edges, in other words, the opposite of graphs known as expanders.

Expander graphs are sparse graphs with excellent connectivity properties. These graphs are of fundamental importance to a number of areas in computer science (see for example the excellent survey of Hoory et al. [2006]). Cuts in expander graphs have a relatively large number of edges going across the cut and therefore give small modularity. Thus, expanders will serve as interesting extremal cases while studying bimodularity, specially on sparse graphs.

**Definition 6.** A graph \( G \) is said to be an \((n, d, \lambda)\)-expander if \( G \) has \( n \) edges, is \( d \)-regular, and \( \lambda \) is the second largest (in absolute value) eigenvalue of the adjacency matrix of \( G \).

When \( \lambda = o(d) \), the graph has good connectivity properties and behaves like a random graph. This is exemplified in a famous result known as the expander mixing lemma, which states that if we take any two subsets of the vertices of an expander graph, the number of edges between them is about the same as the number of edges between them if the edges were chosen randomly.

**Lemma 7** (Expander Mixing Lemma). Let \( G \) be an \((n, d, \lambda)\)-expander. For any two subsets \( A, B \subseteq V(G) \), the number of edges between them, \( \#edges(A, B) \), satisfies

\[
\left| \#edges(A, B) - \frac{d|A||B|}{n} \right| \leq \lambda \sqrt{|A||B|}.
\]
Dividing all the terms by \( m = \frac{nd}{2} \), the result translates to
\[
|e(A, B) - 2\mu(A)\mu(B)| \leq \frac{2\lambda}{d} \sqrt{\mu(A)\mu(B)},
\]
in the notation that we have defined in the previous sections. If we consider the subsets \( A, B \) as forming a partition of the vertices of \( G \), then the mixing lemma gives us an upper bound on the modularity of \( G \). For any 2-clustering \( \{C, \overline{C}\} \),
\[
Q(C, \overline{C}) = 2\mu(C)(1 - \mu(C)) - e(C, \overline{C}) \leq \frac{2\lambda}{d} \sqrt{\mu(C)(1 - \mu(C))}.
\]
This kind of bound will be useful later in the next section when we look at integrality gaps for the modularity linear program.

### 2.5 Convex optimization

Convex optimization deals with maximizing (or minimizing) a convex function over a convex set. The two most popular kinds of convex optimization, Linear Programming (LP) and Semidefinite Programming (SDP), have found wide uses in approximation algorithms. Both are known to be solvable in polynomial time.\(^2\) In this section, we will study the standard LP and SDP formulations of modularity.

It will be useful to define a *modularity matrix* \( M \) having the following entries:
\[
M_{u,v} = \frac{1}{2m} \left( A_{u,v} - \frac{d_ud_v}{2m} \right).
\]

#### 2.5.1 LP relaxation

Many papers like Brandes et al. [2008]; Agarwal and Kempe [2008] have stated and analyzed the LP formulation of modularity. We first start with an Integer

\(^2\) Within an arbitrarily small error.
Linear Programming (ILP) formulation which allows arbitrarily many clusters. For every pair of vertices $u$ and $v$ we define $\{0, 1\}$ variables $x_{u,v}$ with the intention that $x_{u,v} = 0$ if $u$ and $v$ belong to the same cluster and 1 otherwise. Triangle inequality constraints $x_{u,v} \leq x_{u,w} + x_{v,w}$ ensure that if $\{u, w\}$ and $\{v, w\}$ belong to the same cluster then $\{u, v\}$ also belong to the same cluster.

For the LP, we relax the constraint of $\{0, 1\}$ variables to require the variables to lie between 0 and 1. This formulation is shown in Figure 2.1.

\[
\begin{align*}
\text{maximize} & \quad \sum_{u,v} M_{u,v} (1 - x_{u,v}) \\
\text{subject to} & \quad x_{u,u} = 0 \quad \forall u, \\
& \quad x_{u,v} \leq x_{u,w} + x_{v,w} \quad \forall u, v, w, \\
& \quad 0 \leq x_{u,v} \leq 1 \quad \forall u, v.
\end{align*}
\]

**Figure 2.1:** LP relaxation for modularity clustering.

The values assigned to the variables in the optimal LP solution might be fractions, and therefore have to be rounded to 0 or 1. The paper Agarwal and Kempe [2008] used such an LP relaxation with several rounding schemes for empirical evaluations.

The result below states that the worst case integrality gap (defined as the ratio between the value of the LP and the integer optimum) of the LP relaxation is about the square root of the degree of the graph, thus ruling out logarithmic (in the degree) approximations via rounding such LP relaxations. For a graph $G$, we let $Q_{LP}(G)$ denote the optimal objective value of its LP.

**Theorem 8.** For any $d \geq 3$, there exist $d$-regular graphs $G$ satisfying

\[
\frac{Q(G)}{Q_{LP}(G)} = O \left( \frac{1}{\sqrt{d}} \right).
\]
Proof. Let $G = (V, E)$ be an $(n, d, \lambda = O(\sqrt{d}))$-expander. Most $d$-regular graphs, for sufficiently large $n$, satisfy this property (for a precise statement, see Friedman [2003]). For any cut $\{C, \overline{C}\}$ in $G$, Equation (2.9) (which is a restatement of Lemma 7) gives

$$Q(C, \overline{C}) \leq \frac{2\lambda}{d} \sqrt{\mu C(1 - \mu(C))} \leq \frac{\lambda}{d} = O\left(\frac{1}{\sqrt{d}}\right),$$

where we have used the fact that $\mu(1 - \mu) \leq \frac{1}{4}$ when $\mu \in [0, 1]$. Then by Lemma 3,

$$Q(G) \leq 2Q_2(G) = O\left(\frac{1}{\sqrt{d}}\right). \hspace{1cm} (2.10)$$

Now let us analyze the LP solution. For any graph $G = (V, E)$, a valid fractional solution of the LP relaxation is as follows: set $x_{u,v} = \frac{1}{2}$ for all $\{u, v\} \in E$ and set $x_{u,v} = 1$ otherwise. The terms in the objective corresponding to non-edges now disappear. Further, when the graph is $d$-regular, the objective function simplifies to

$$Q_{LP}(G) \geq \frac{1}{2m} \sum_{u,v; (u,v) \in E} \left(1 - \frac{d^2}{2m}\right)\left(1 - \frac{1}{2}\right) = \frac{1}{2} - \frac{d}{2n} \geq \frac{1}{4}, \hspace{1cm} (2.11)$$

where the last inequality holds for sufficiently large $n$. Dividing Equation (2.10) by Equation (2.11) gives the desired result.

2.5.2 SDP relaxation

For the SDP relaxation, we will only be concerned with optimizing bimodularity. We first start with a Quadratic Programming (QP) formulation. Let there be a $\{\pm 1\}$ variable $x_v$ for each vertex $v$ denoting which of the two clusters $v$ belongs to. Then the modularity of a cut $\mathcal{C}$ can be written as $Q(\mathcal{C}) = \sum_{u,v} M_{u,v} \frac{1}{2}x_u x_v = \frac{1}{2} \sum_{u,v} M_{u,v} x_u x_v$, where we have used the property that any modularity matrix satisfies, $\sum_{u,v} M_{u,v} = 0$. 

\qed
To form the SDP relaxation, we will use vector variables $\vec{x}_v$ for each vertex $v$ and impose the condition that they be real unit vectors. This SDP is shown in Figure 2.2.

$$\begin{align*}
\text{maximize} \quad & \frac{1}{2} \sum_{u,v} M_{u,v} \vec{x}_u \cdot \vec{x}_v \\
\text{subject to} \quad & \vec{x}_v \cdot \vec{x}_v = 1 \quad \forall v, \\
& \vec{x}_v \in \mathbb{R}^n \quad \forall v.
\end{align*}$$

Figure 2.2: SDP relaxation for bimodularity.

The paper Agarwal and Kempe [2008] also empirically analyzes this SDP with respect to various types of rounding schemes. The simplest of these is the Goemans-Williamson rounding Goemans and Williamson [1995], where the vertices are assigned clusters depending on which side of a random hyperplane their solution vector falls on. Unfortunately, this rounding guarantees an approximation factor only when each term in the matrix $M$ is nonnegative. This is not true for our case.

A more interesting rounding scheme considered in Agarwal and Kempe [2008] is the one by Charikar and Wirth [2004], which we will discuss in detail in Chapter 4. This rounding gives an approximation factor of $O\left(\frac{1}{\log n}\right)$ when the matrix $M$ has nonnegative diagonal entries. This is not true in our case, but we will see in Chapter 4 how to tweak their rounding to get good guarantees for sparse graphs.

Convex optimization techniques are not well suited for approximating modularity in dense graphs, which tend to have small optimums. Particularly, there exist even constant sized graphs $G$ for which $Q(G) = 0$ but the SDP solution is positive.
2.6 Spectral clustering and other heuristics

There are some interesting heuristics for finding a good modular clustering that have become popular. Unfortunately, there is no guarantee on the quality of the output clustering of these algorithms.

A popular clustering method is the Louvian method, first introduced in the paper Blondel et al. [2008]. This iterative greedy optimization method finds locally optimal communities that can withstand perturbations. These communities are then collapsed into nodes and a new graph is built and the process is repeated.

Newman, in the papers Newman [2006] and Newman [2006] suggested a spectral clustering approach. To understand this approach, we go back to the QP formulation for bimodularity. We want to maximize the quantity $\vec{x}^T M \vec{x}$ over all $x \in \{\pm 1\}^n$ vectors. If we were maximizing over real vectors, this would be achieved for the first eigenvector $e_1$ (the eigenvector corresponding to the largest eigenvalue) of the matrix $M$. So to get a good $\{\pm 1\}^n$ solution, we do the next best thing, which is to take the values in $e_1$ and round them to +1 or −1. We can repeat this partitioning procedure on each cluster to get more clusters.
Chapter 3
Hardness Results

In this chapter we will look at two impossibility results, namely, the \( \mathcal{NP} \)-hardness of calculating optimal bimodularity for sparse graphs, and a 1.0009-factor hardness for calculating optimal modularity for dense graphs.

Lemma 3 from the previous chapter shows that a small number of clusters well-approximate the optimal modularity value. In particular, partitioning into just two clusters already achieves at least half of the optimum. Thus, it becomes important to look at the complexity of the bimodularity problem.

Theorem 9 in Section 3.1 proves the \( \mathcal{NP} \)-completeness of the bimodularity problem for sparse graphs, specifically for \( d \)-regular graphs (\( d \geq 9 \)). The previous \( \mathcal{NP} \)-completeness result for this case (Brandes et al. [2008]) required the degree of every node to be large, roughly \( \Omega(\sqrt{n}) \). Our reduction is from the graph bisection problem for 4-regular graphs.

For dense graphs, Theorem 12 in Section 3.2 provides a 1.0009 factor inapproximability of the modularity clustering problem. This result holds irrespective of whether the number of clusters is specified in the input or the algorithm is allowed to select the best number of clusters. The required approximation gap in our reduction is derived from the approximation gap of the maximum independent set problem for 3-regular graphs in Chlebik and Chlebikova [2006].

The intuition behind our inapproximability result is that, for the type of dense graphs that are considered in our reduction, large-size cliques must be completely
contained within the clusters. However, the gap preservation calculations need to be done extremely accurately to avoid shrinking the inapproximability gap\(^1\).

### 3.1 \(\mathcal{NP}\)-hardness of sparse bimodularity

Brandes et al. Brandes et al. [2008] proved \(\mathcal{NP}\)-hardness of the bimodularity problem provided nodes with very large degrees are allowed in the input graph. Thus it is not a priori clear whether calculating modularity on very sparse graphs becomes easy and admits an exact polynomial-time algorithm. However, we rule out this possibility of exact solution.

For our construction, notice that we cannot anymore use the idea of hiding a large-size clique since the resulting graph should not have any cliques of size more than \(d\) and, for fixed \(d\), one can indeed enumerate all these cliques in polynomial time.

As stated in the introduction, our reduction is from the graph bisection problem for 4-regular graphs. Our construction is similar to that in Brandes et al. [2008], but carefully replaces dense graphs with nicely behaving sparse graphs. Intuitively, now an optimal solution for 2-clustering is constrained to have exactly the same number of nodes in each community to avoid any local improvement. We have to do a more careful analysis of the properties of an optimal 2-clustering so as to get the following result.

**Theorem 9.** Computing \(Q_2(G)\) is \(\mathcal{NP}\)-complete for \(d\)-regular graphs \(G\) for any constant \(d \geq 9\).

**Proof.** The decision version \(d\text{-BIMODULARITY}\) of our problem is as follows:

\(^1\) For example, the inapproximability gap of Berman and Karpinski [1999] does not suffice for our purposes.
Given a $d$-regular graph $G$ and a number $\eta$, is there a clustering $C$ of $G$ into at most two clusters for which $Q(C) \geq \eta$?

Our reduction is from the minimum graph bisection problem for 4-regular graphs (4-MinBisection):

Given a 4-regular graph $G$ with $n$ nodes (with even $n$) and an integer $c$, is there a partition of the vertices into two clusters each containing $\frac{n}{2}$ nodes such that at most $c$ edges are “cut”, i.e., at most $c$ edges have their two end-points in different clusters?

4-MinBisection is known to be $\mathcal{NP}$-complete (Kefeng et al. [1999]). We reduce an instance $G$ of 4-MinBisection to an instance $G'$ of $d$-Bimodularity in a manner similar to that in Brandes et al. [2008]. Every node in $G$ is replaced by a copy of an $n$-node $d$-regular graph $H$ which has the property that its minimum cut (minimum number of edges in any cut) is of size $d$. A constructive existence proof of such graphs follows from the claim below.

Claim 1. For every $n$ and $d < n$, there exist $d$-regular graphs having $n$ vertices and having their minimum cuts of size $d$.

Proof. Such a family of graphs can be constructed in the following recursive manner: for $d = 2$, the 2-regular graph, namely a simple cycle consisting of $n$ nodes, has a minimum cut of 2 edges.

For $d = 3$, consider two simple cycles $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$, each consisting of $\frac{n}{2}$ nodes. Consider an arbitrary matching between the nodes of $H_1$ and $H_2$ and add the edges corresponding to this matching to obtain a 3-regular graph $H = (V, E)$. Consider an arbitrary subset of nodes $C \subset V$ of $H$. Then, if $C \cap V_1 \neq \emptyset$ and $C \cap V_2 \neq \emptyset$, then the number of cut edges is at least 4. Otherwise, assume that $C \cap V_1 = \emptyset$ (the other case is symmetric) and thus $\emptyset \subset C \subseteq V_2$. If
\( C = V_2 \) then the number of cut edges is exactly \( \frac{n}{2} > 2 \). Otherwise, the number of cut edges is at least 2 (corresponding to two edges of the cycle in \( H_2 \)) plus 1 (corresponding to one of the matching edges added).

For \( d > 3 \), a recursive construction of such graphs follows in a similar manner: take such a \((d-2)\)-regular graph \( H \) on \( n \) nodes for which the inductive hypothesis applies and add a simple cycle to \( H \) all of whose edges are different from those in \( H \). Consider a cut in this graph. By the induction hypothesis the cut contains at least \( d-2 \) edges of \( H \) and at least 2 additional edges of the new cycle added to \( H \).

Let \( H_v \) denote the copy of \( H \) corresponding to the node \( v \in G \). Delete two independent edges (\( i.e. \), edges without any common end-points) in \( H_v \). The four edges connected to \( v \) are now connected to the four endpoints of these deleted edges. This is done in order to make the final graph \( G' \) \( d \)-regular\(^2\). Note that the number of nodes in the transformed graph \( G' \) is \( n^2 \), whereas the number of edges is \( m = \frac{dn^2}{2} \). Since two edges are removed from \( H \) in the construction, the minimum cut in each modified copy of \( H \) is at least \( d - 2 \).

Let \( C^* \) be an optimal clustering of \( G' \). The correctness of the reduction follows by showing that \( 4\text{-MinBisection} \) has a solution with at most \( c \) cut edges if and only if \( Q(C^*) \geq \frac{1}{2} - \frac{c}{m} \).

**Lemma 10.** \( C^* \) has exactly two clusters and \( Q(C^*) > 0 \).

*Proof.* It suffices to show a clustering \( C = \{C, \overline{C}\} \) such that \( Q(C) > 0 \). To this end, let \( C = \{H_v\} \) for some \( v \). In this case, there are exactly 4 intercluster edges.

\(^2\) This is one step that is different from the reduction in Brandes et al. [2008], where every node in \( G \) is replaced by a copy of \( K_n \) producing the final graph with non-constant degrees. Since \( G \) is 4-regular, we need \( d > 8 \).
Using Equation (2.8) and the fact that \(d(n - 1) > 4\), we get

\[
\begin{align*}
Q(C) &= 2\mu(C)(1 - \mu(C)) - e(C, \overline{C}) \\
&= \frac{2}{n} \left( 1 - \frac{1}{n} \right) - \frac{4}{dn^2/2} \\
&= \frac{2}{dn^2} (d(n - 1) - 4) \\
&> 0 .
\end{align*}
\]

The next lemma shows how to normalize a solution without decreasing the modularity value. Part (a) of the lemma states that \(C^*\) cannot have any copy of \(H\) split across clusters, whereas part (b) implies that any optimal clustering has to be a bisection of the graph.

**Lemma 11.** It is possible to normalize an optimal solution \(C^*\) without decreasing the modularity value such that the following two conditions hold:

(a) For every \(v \in G\), there exists a cluster \(C \in C^*\) such that \(H_v \subseteq C\).

(b) Each cluster in \(C^*\) contains exactly \(\frac{n}{2}\) copies of \(H\).

**Proof.** Suppose the set of nodes of \(G'\) is partitioned into three subsets \(A\), \(B\) and \(C\). Let \(C = \{A \cup C, B\}\), and we want to transfer the nodes in \(C\) to the other cluster to form the clustering \(C' = \{A, B \cup C\}\). For any two disjoint subsets \(X\) and \(Y\) of nodes of \(G'\), let \(m_{XY}\) denote the number of edges one of whose endpoints is in \(X\) and the other in \(Y\). Then, using Equation (2.3) or Equation (2.5), the gain in modularity between clusterings \(C\) and \(C'\) is \(Q(C) - Q(C')\) and it can be simplified (using the fact that \(G'\) is \(d\)-regular) and written as \(\frac{2}{dn^4} (d|C| (|A| - |B|) + n^2 (m_{BC} - m_{AC}))\). Let us define

\[
\text{Gain}(C, C') = \frac{dn^4}{2} (Q(C) - Q(C')) = d|C| (|A| - |B|) + n^2 (m_{BC} - m_{AC}) .
\]
Thus our aim in proving the lemma will be to show that we can modify an optimal clustering $C$ into a clustering $C'$ that respects (a) and (b) and results in $\text{Gain}(C, C') > 0$.

To prove Part (a), let us assume that there exists a $v \in G$ such that $H_v$ is split across clusters in the optimal clustering $C = \{C_1, C_2\}$. Without loss of generality, we can assume that $|C_1 \setminus H_v| \geq |C_2 \setminus H_v|$. To form clustering $C'$, we will transfer the part of $H_v$ in $C_1$ from $C_1$ to $C_2$. Let $A = C_1 \setminus H_v$, $B = C_2$, $C = H_v \setminus C_2$, and $|C| = k$. Then the part of $H_v$ in $C_2$ has a size of $n - k$. By our assumption,

$$|A| - |B| = |C_1 \setminus H_v| - |C_2| = |C_1 \setminus H_v| - |C_2 \setminus H_v| - |H_v \setminus C_2| \geq -(n - k).$$

Substituting this in Equation (3.1), we get

$$\text{Gain}(C, C') \geq d \left[ -k(n - k) \right] + n^2(m_{BC} - m_{AC}).$$

Now, since the original graph $G$ was 4-regular, at most 4 extra inter-cluster edges will appear after the transfer. Thus, $m_{AC} \leq 4$. The term $m_{BC}$ represents the number of edges between $C_2$ and $H_v \setminus C_1$, which is at least the number of edges between the two parts of $H_v$. Thus, $m_{BC}$ is at least the number of edges in a minimum cut of $H_v$ which is at least $d - 2$. This gives

$$\text{Gain}(C, C') \geq -dk(n - k) + n^2(d - 2 - 4)$$

$$\geq -d \frac{n^2}{4} + (d - 6)n^2$$

$$= \frac{(3d - 24)n^2}{4}$$

$$> 0,$$

where the second inequality is due to the fact that $k(n - k)$ is maximized when $k = n/2$, and the last inequality is satisfied when $d \geq 9$. Hence the modularity can be strictly improved by putting each copy of $H$ completely in a cluster.
Let us now prove Part (b). Because of Part (a), each $H_v$ is contained completely in one cluster of an optimal clustering $\mathcal{C} = \{C_1, C_2\}$. Now assume that $C_1$ has more copies of $H$ than $C_2$. Since $n$ is even, this implies that $C_1$ has at least two more copies of $H$ than $C_2$. We will create a new clustering $\mathcal{C}'$ by transferring a copy of $H$ from $C_1$ to $C_2$. Then the gain in modularity after this transfer is given by Equation (3.1), where $C$ denotes the transferred copy of $H$, $B = C_2$ and $A = C_1 \setminus C$. By our assumption, $|A| - |B| \geq |C|$. Therefore we can simplify the first term and get

$$\text{Gain}(\mathcal{C}, \mathcal{C}') \geq d |C|^2 + n^2 (m_{BC} - m_{AC}).$$

Also, since the original graph $G$ was 4-regular, at most 4 extra inter-cluster edges will appear after the transfer. Simplifying and substituting values,

$$\text{Gain}(\mathcal{C}, \mathcal{C}') \geq dn^2 - 4n^2 > 0.$$

Hence, the modularity can be strictly improved by balancing out the copies of $H$ in both clusters. \qed

Armed with the above lemma, one can now prove the $\mathcal{NP}$-completeness of our problem. We will use the above construction to reduce an instance $\langle G, c \rangle$ of 4-MinBisection to an instance $\langle G', K \rangle$ of $d$-Bimodularity with $K = \frac{1}{2} - \frac{c}{m}$. Now suppose $\mathcal{C}^* = \{A, B\}$ is an optimal 2-clustering of $G'$. Then, $Q(\mathcal{C}^*) = 2\mu(A)\mu(B) - e(A, B)$. By Lemma 11(b), $\mu(A) = \mu(B) = \frac{1}{2}$. Also, because of Lemma 11(a), $e(A, B)$ only has edges that were originally present in $G$, thus representing a bisection of $G$. Therefore, $\#edges(A, B) \leq c$ if and only if

$$Q(\mathcal{C}^*) \geq \frac{1}{2} - \frac{c}{m} = K. \quad \square$$
3.2 The inapproximability result

In this section, we will show that we cannot hope to approximate modularity on dense graphs in polynomial time within a factor of 1.0009. The hardness result holds for the general modularity problem, that is, when the number of clusters is unspecified.

This hardness result may be contrasted with the results in Section 4.1 where we show that the modularity value of any dense graph can be approximated to within any constant additive error using the regularity lemma. However, the \textit{APX}-hard instances here have modularity values that tend to 0 (more precisely, these instances have modularity values of $O\left(\frac{1}{n}\right)$), thus the constant additive error provides no guarantee on the approximation ratio.

**Theorem 12.** It is \textit{NP}-hard to approximate $Q(G)$ within a factor of $\frac{8835}{8827}$.

In the rest of this section, we will prove the above theorem.

We reduce the maximum-cardinality independent set problem for 3-regular graphs (3-MIS) to our problem. An instance of 3-MIS consists of a 3-regular graph $H = (V, E)$, and the goal is to find a maximum cardinality subset of nodes $C \subset V$ such that every pair of nodes $u$ and $v$ in $C$ is independent, i.e., $\{u, v\} \notin E$.

The following inapproximability result is known for 3-MIS.

**Theorem 13.** Chlebik and Chlebikova [2006] Given a graph $H$, it is \textit{NP}-hard to distinguish between the following two cases in polynomial time:

- The graph $H$ has an independent set of size at least $\frac{95}{194} n$.
- Every independent set in $H$ has size at most $\frac{94}{194} n$.

For notational convenience, let $\gamma = \frac{95}{194}$ and let $\sigma = \frac{94}{194}$. 

Let us start with a hard instance \( H = (V, F) \) of 3-MIS (as guaranteed by Theorem 13) with \(|V| = n\) nodes and \(|F| = \frac{3n}{2}\) edges. Consider the complement \( \overline{H} = (V, \overline{F}) \) of \( H \), i.e., the graph with edges \( \overline{F} = \{ \{u, v\} | u, v \in V, u \neq v \} \setminus F \). Since \( H \) is 3-regular, \( \overline{H} \) is \((n - 4)\)-regular. The input to our clustering problem is this graph \( \overline{H} \), which we will denote by \( G = (V, E) \) with \( E = \overline{F} \).

Note that \( C \subset V \) is an independent set of \( H \) if and only if \( C \) is a clique in \( G \). Let \( \alpha(H) \) denote the size of a maximum independent set of \( H \).

We show our theorem by proving the following two parts:

**Completeness:** If \( \alpha(H) \geq \gamma n \) then \( Q(G) \geq \frac{2(4\gamma^2 - \gamma)}{n-4} \).

**Soundness:** If \( \alpha(H) \leq \sigma n \) then \( Q(G) \leq \frac{4\sigma - 1}{n-4} \).

Dividing the inequalities in the completeness and soundness cases gives the theorem.

We will start by proving completeness.

### 3.2.1 Completeness

Recall that in this case, the input graph \( H \) satisfies \( \alpha(H) \geq \gamma n \). Thus, the graph \( G \) has a clique of size at least \( \gamma n \).

**Lemma 14.** If \( \alpha(H) \geq \gamma n \) then \( Q(G) \geq \frac{2(4\gamma^2 - \gamma)}{n-4} \).

**Proof.** Suppose \( G \) has a has a clique \( C \) of size \(|C| = tn\) for some \( t \geq \gamma \). Since \( C \) is a clique, it follows that there are \( \frac{tn(tn-1)}{2} \) edges inside \( C \). Using Lemma 5 and
Equation (2.2) we get that the modularity of the cut \( \{C, \bar{C}\} \) is

\[
Q(\{C, \bar{C}\}) = 2(i(C) - \mu(C)^2)
\]

\[
= 2 \left( \frac{t n (t n - 1)}{2 n (n - 4)} - \left( \frac{t n (n - 4)}{n (n - 4)} \right)^2 \right)
\]

\[
= 2 \left( \frac{4 t^2 - t}{n - 4} \right)
\]

\[
\geq \frac{2 (4 \gamma^2 - \gamma)}{n - 4}.
\]

Now we move on to soundness.

### 3.2.2 Soundness with two clusters

Recall that in the soundness case, the input graph \( H \) satisfies \( \alpha(H) \leq \sigma n \). Thus, the graph \( G \) has a clique of size at most \( \sigma n \). We will first consider the simpler scenario where an optimal solution has exactly two clusters.

Suppose that the optimal solution is \( \mathcal{C} = \{C, \bar{C}\} \) of 2-clustering on \( G \) with \( |C| = t n \) and \( 0 < t \leq \frac{1}{2} \).

**Lemma 15.** Let \( \delta n \) be the size of the largest clique in the subgraph induced by \( C \). Then, \( Q(C) \leq \frac{4 t^2 + 2 \delta - 3 t}{n - 4} \).

**Proof.** Since the size of the largest clique in the subgraph induced by \( C \) is \( \delta n \), for each of the remaining \((t - \delta)n\) nodes, they will not be connected to at least one node inside the clique. Hence, using Equation (2.2), we get

\[
Q(C) = i(C) - \mu(C)^2
\]

\[
\leq \frac{t n (t n - 1)}{2 n (n - 4)} - (t - \delta) n - t^2
\]

\[
= \frac{4 t^2 + 2 \delta - 3 t}{n - 4}.
\]
Lemma 16. $Q(C) \leq \frac{2\sigma - \frac{1}{2}}{n-4}$.

Proof. Using the previous lemma and the facts that $\delta \leq \min \{t, \sigma\}$ and $t \leq \frac{1}{2}$, we have two cases:

$t > \sigma$: Then $Q(C) \leq \frac{4t^2 + 2t - 3t}{n-4}$. The function $f(t) = 4t^2 - 3t$ is increasing in the range $[\sigma, \frac{1}{2}]$ since $\sigma > \frac{3}{8}$ and $\frac{\partial f}{\partial t} = 8t - 3 > 0$ if $t > \frac{3}{8}$. Thus, $\max_{\sigma \leq t \leq \frac{1}{2}} f(t) = f\left(\frac{1}{2}\right) = -\frac{1}{2}$, and thus $Q(C) \leq \frac{2\sigma - \frac{1}{2}}{n-4} \leq \frac{2\sigma - \frac{1}{2}}{n-4}$.

$t \leq \sigma$: Since $\delta \leq t$ and $4t^2 + 2\delta - 3t$ is an increasing function of $\delta$, we have $Q(C) \leq \frac{4t^2 + 2t - 3t}{n-4} = \frac{4t^2 - t}{n-4}$. The function $f(t) = 4t^2 - t$ satisfies $f(0) = 0$ and $\frac{\partial f}{\partial t} = 8t - 1 \begin{cases} < 0 & \text{if } t < 1/8 \\ > 0 & \text{if } 1/8 < t \leq \sigma \end{cases}$

Thus, $\max_{0 < t \leq \sigma} f(t) = f(\sigma)$ and we have

$$Q(C) \leq \frac{4\sigma^2 - \sigma}{n-4} \leq \frac{2\sigma - \frac{1}{2}}{n-4}. \quad \square$$

Finally, using Lemma 5, $Q(C) = 2Q(C) \leq \frac{4\sigma - 1}{n-4}$, completing the soundness proof for when the optimal solution has only two clusters.

### 3.2.3 Soundness with more than two clusters

For legibility of calculations, we would like to drop the denominator scaling term from Equation (2.2). To this end, we define $Q'(C) = n(n-4)Q(C)$. Let $\mathcal{C} = \{V_1, V_2, \ldots, V_{m+1}\}$ be an optimal solution of modularity clustering that uses a minimum $m > 1$ number of clusters. Let $|V_i| = t_i n$, and suppose that $\emptyset \subset V_i' \subseteq V_i$ is a largest clique of size $\delta_i n$ in the graph $(V_i, (V_i \times V_i) \cap E)$. Note that $0 < \delta_i \leq \min \{t_i, \sigma\}$ for all $1 \leq i \leq m + 1$, $\sum_{i=1}^{m+1} t_i = 1$ and we need to show that $Q'(C) \leq (4\sigma - 1) n$. Let $\widehat{V}_i$ denote $V \setminus V_i$. 


Lemma 17. $Q'(V_i) \leq (4t_i^2 - t_i)n$.

Proof. $Q'(V_i)$ is maximized when the nodes in $V_i$ form a clique. Thus,

$$Q'(V_i) \leq \left(\frac{4}{n} - 1\right)(t_in) + \left(\frac{4}{n}\right)(t_in - 1)(t_in) = (4t_i^2 - t_i)n \quad \square$$

Corollary 18. If $|V_i| \leq \frac{n}{4}$ then $Q'(V_i) \leq 0$. If $|V_i| = (\frac{1}{4} + \epsilon)n > \frac{n}{4}$ then $Q'(V_i) \leq (4\epsilon^2 + \epsilon)n$.

Lemma 19. Suppose that $t_i = \frac{1}{2} + \epsilon > \frac{1}{2}$ for some $0 < \epsilon < \frac{1}{2}$ and $\hat{\delta}_i$ is the size of a largest clique in $(\hat{V}_i, (\hat{V}_i \times \hat{V}_i) \cap E)$. Then,

$$Q'(V_i) \leq \left(4\epsilon^2 - \epsilon - \frac{1}{2} + 2\hat{\delta}_i\right)n \leq \left(2\sigma - \frac{1}{2}\right)n$$

Proof. Note that $|\hat{V}_i| = \frac{1}{2} - \epsilon < \frac{1}{2}$. Then by Lemma 5,

$$Q'(V_i) = Q'(\hat{V}_i) \leq \left(4\left(\frac{1}{2} - \epsilon\right)^2 + 2\hat{\delta}_i - 3\left(\frac{1}{2} - \epsilon\right)\right)n$$

$$= \left(4\epsilon^2 - \epsilon - \frac{1}{2} + 2\hat{\delta}_i\right)n$$

where the inequality follows from Lemma 15 if we replace $V_i$ by $\hat{V}_i$. Since $t_i \geq \frac{1}{2}$, we have

$$4\epsilon^2 - \epsilon - \frac{1}{2} + 2\hat{\delta}_i = 4t_i^2 - 5t_i + 1 - 2\hat{\delta}_i \leq 4t_i^2 + 2\hat{\delta}_i - 3t_i$$

Since $\hat{\delta}_i \leq \sigma < t_i$, the arguments in Lemma 16 can be directly applied on $4t_i^2 + 2\hat{\delta}_i - 3t_i$ to show that $\left(4\epsilon^2 - \epsilon - \frac{1}{2} + 2\hat{\delta}_i\right)n \leq (2\sigma - \frac{1}{2})n$. \quad \square

Let us call a cluster $V_i$ a giant component if $t_i > \sigma$. Note that since $3\sigma > 1$, we can have at most two giant components. We have therefore three cases depending on the number of giant components.

Case (i): $\mathcal{C}$ has no giant components Note that $\mathcal{C}$ can have at most three clusters containing strictly more than $\frac{n}{4}$ nodes.
If \( C \) contains no such cluster then by Corollary 18 \( Q'(C) \leq 0 \).

If \( C \) contains exactly one such cluster, say \( V_1 \), then \( Q'(C) \leq Q'(V_1) \leq (2\sigma - \frac{1}{2}) n < (4\sigma - 1) n \) by Lemma 16 (if \( t_i \leq \frac{1}{2} \)) or Lemma 19 (if \( t_i > \frac{1}{2} \)).

If \( C \) contains exactly two such clusters, say \( V_1 \) and \( V_2 \), then again \( Q'(C) \leq Q(V_1) + Q'(V_2) \leq 2 \left(2\sigma - \frac{1}{2}\right) n = (4\sigma - 1) n \) by Lemma 16 and Lemma 19.

Otherwise, suppose that \( C \) contains exactly three such clusters, say \( V_1, V_2 \) and \( V_3 \). Let \( t_i = \frac{1}{4} + \varepsilon_i \) for \( i = 1, 2, 3 \). Then, \( 0 < \varepsilon_1 + \varepsilon_2 + \varepsilon_3 < \frac{1}{4} \). Using Corollary 18 we have:

\[
\begin{align*}
\sum_{i=1}^{3} Q'(V_i) &= \left(4 \sum_{i=1}^{3} \varepsilon_i^2 + \sum_{i=1}^{3} \varepsilon_i \right) n < \left(4 \left( \sum_{i=1}^{3} \varepsilon_i \right)^2 + \frac{1}{4} \right) n \\
&< \left(4 \left( \frac{1}{4} \right)^2 + \frac{1}{4} \right) n = \frac{n}{2} < (4\sigma - 1) n
\end{align*}
\]

Case (ii): \( C \) has one giant component Let \( V_1 \) be the giant component. Since \( 1 - t_1 < 1 - \sigma < \frac{3}{4} \), there are at most two other clusters with strictly more than \( \frac{n}{4} \) nodes.

Subcase (ii-a): there is one other cluster with strictly more than \( \frac{n}{4} \) nodes Let this cluster be \( V_2 \). By Corollary 18, \( \sum_{j=3}^{m+1} Q'(V_j) \leq 0 \). Note that \( t_2 \leq \sigma \). Now, by reusing the calculations of Lemma 16 and using Lemma 19 we get

\[
Q'(C) = Q'(V_1) + Q'(V_2) + \sum_{j=3}^{m+1} Q'(V_j) \leq Q'(V_1) + Q'(V_2)
\]

\[
\leq \left(2\sigma - \frac{1}{2}\right) n + \left(2\sigma - \frac{1}{2}\right) n = \left(4\sigma - 1\right) n
\]

by Lemma 19 if \( t_1 > \frac{1}{2} \) by Lemma 16 since \( t_2 \leq \sigma \)

by Lemma 16 if \( t_1 \leq \frac{1}{2} \)

Subcase (ii-b): there are two other clusters with strictly more than \( \frac{n}{4} \) nodes Let these clusters be \( V_2 \) and \( V_3 \). Then, \( \sigma n < |V_1| < \frac{n}{2} \). By Corollary 18,
\[ \sum_{j=4}^{m+1} Q'(V_j) \leq 0. \] Let \( t_2 = \frac{1}{4} + \varepsilon_2 \) and \( t_3 = \frac{1}{4} + \varepsilon_3 \) with \( 0 < \varepsilon_2 \leq \varepsilon_3 < \frac{1}{2} - \sigma < \frac{2}{100}. \)

Thus,

\[
Q'(\mathcal{C}) \leq Q'(V_1) + Q'(V_2) + Q'(V_3)
\]

\[
\leq \left(2\sigma - \frac{1}{2}\right) n + \left(4\varepsilon_2^2 + \varepsilon_2\right) n + \left(4\varepsilon_3^2 + \varepsilon_3\right) n
\]

by Lemma 16 since \( t_1 < \frac{1}{2} \)

by Corollary 18

by Corollary 18

Since \( 4\varepsilon_2^2 + \varepsilon_2 + 4\varepsilon_3^2 + \varepsilon_3 < 8\left(\frac{2}{100}\right)^2 + 2\left(\frac{2}{100}\right) < 2\sigma - \frac{1}{2} \), we have \( Q'(\mathcal{C}) \leq (4\sigma - 1) n. \)

**Case (iii): \( \mathcal{C} \) has two giant components** Let \( V_1 \) and \( V_2 \) be the two giant components with \( t_1 = \sigma + \nu_1 \) and \( t_2 = \sigma + \nu_2 \) for some \( 0 < \nu_1 \leq \nu_2 < 1 - 2\sigma. \) Since

\[
|\bigcup_{j=3}^{m+1} V_i| = (1 - t_1 - t_2) n \leq (1 - 2\sigma) n < \frac{n}{4}, \]

by Corollary 18 \( \sum_{j=3}^{m+1} Q'(V_j) \leq 0. \)

Now, by reusing the calculations in the proof of the case of \( t > \sigma \) of Lemma 16 and using Lemma 19 we get

\[
Q'(\mathcal{C}) = Q'(V_1) + Q'(V_2) + \sum_{j=3}^{m+1} Q'(V_j)
\]

\[
\leq \left(2\sigma - \frac{1}{2}\right) n + \left(2\sigma - \frac{1}{2}\right) n = (4\sigma - 1) n
\]

by Lemma 19 if \( t_1 > \frac{1}{2} \)

by Lemma 19 if \( t_2 > \frac{1}{2} \)

by Lemma 16 if \( t_1 \leq \frac{1}{2} \)

by Lemma 16 if \( t_2 \leq \frac{1}{2} \)

Thus we find that in all cases, \( Q'(\mathcal{C}) \leq (4\sigma - 1)n, \) which means that \( Q(\mathcal{C}) \leq \frac{4\sigma - 1}{n-4}. \)
Chapter 4

Approximation Algorithms

In this chapter, we will be concerned with how close we can get to the optimal modularity of a graph in polynomial time. We will analyze a number of approximation algorithms and lower bounds in this chapter.

We start with the case when the input graph is dense. Here, we can use the famous result of Frieze and Kannan [1996] for approximating the $\ell$-way cut problem to get an additive approximation scheme for dense graphs. This is shown in Section 4.1.

In Section 4.2, we will prove some lower bounds for the modularity of sparse graphs. Since optimal modularity is bounded between 0 and 1, an efficient constructive proof of a lower bound for a class of graphs yields an approximation algorithm for that class of graphs.

In Section 4.3, we will use the lower bounds of Section 4.2 and semidefinite programming to come up with algorithms that have approximation factors logarithmic in the average degree of the graph.

4.1 Dense Graphs and the Regularity Lemma

Using the algorithmic version of the regularity lemma in Frieze and Kannan [1996] we can show that if the given graph $G$ is dense then, for any given constant $\tau > 0$, there is a polynomial-time algorithm that returns a solution of modularity value
at least $Q(G) - \tau$.

**Theorem 20.** Suppose that the given graph $G = (V, E)$ is dense, i.e., $m = |E| = \delta n^2$ for some constant $0 < \delta < \frac{1}{2}$. Then, for any given constant $0 < \tau < 1$, there is a polynomial-time algorithm that returns a solution of value at least $Q(G) - \tau$.

**Proof.** The $\ell$-way cut problem is defined as follows. We are given a weighted graph $G = (V, E)$ with $w(u, v) \in \mathbb{R}$ being the weight of the edge $\{u, v\} \in E$. A valid solution is a partition of $V$ to $\ell$ subsets $\mathcal{C} = \{S_1, S_2, \ldots, S_\ell\}$, and the goal is to maximize the sum of weights of those edges whose end-points are in different subsets, i.e., maximize $w(\mathcal{C}) = \sum_{(u,v) \in E(\mathcal{C})} w(u, v)$, where $E(\mathcal{C})$ is the set of all pairs of vertices that belong to different partitions, i.e., $E(\mathcal{C})$ is the set of all “inter-partition” edges.

The following result was proved in Frieze and Kannan [1996].

**Theorem 21 (Frieze and Kannan [1996]).** Given an weighted graph $G = (V, E)$ of $n$ nodes and any constant $0 < \varepsilon < 1$ there is a polynomial-time algorithm $A_{\varepsilon}$ which, computes a partition $\mathcal{C}_{\varepsilon}$ of $V$ such that

$$w(\mathcal{C}_{\varepsilon}) \geq w(\mathcal{C}^*) - \varepsilon n^2$$

where $\mathcal{C}^*$ is an optimal (maximum weight) partition.

Equation (2.5) can be used to assign edge weights to cast our modularity clustering problem as an $\ell$-way cut problem in the following manner. Consider the complete graph on $n$ nodes ($K_n$) and let $w_{u,v} = 2 \delta \left( \frac{d_u d_v}{2m} - a_{u,v} \right)$ for the edge $\{u, v\}$ of $K_n$. Then, for a partition $\mathcal{C} = \{S_1, S_2, \ldots, S_\ell\}$ of the nodes of $K_n$,

$$w(\mathcal{C}) = \sum_{\{u,v\} \in E(\mathcal{C})} 2 \delta \left( \frac{d_u d_v}{2m} - a_{u,v} \right) = 2m \delta Q(\mathcal{C}) = 2 \delta^2 n^2 Q(\mathcal{C})$$
Let $A_\varepsilon$ be the objective value of an approximate solution of the modularity clustering problem on the given graph obtained by using the $\ell$-way partitioning of Theorem 21 with $\varepsilon = 2 \tau \delta^2$. Then,

$$2\delta^2 n^2 A_\varepsilon \geq 2\delta^2 n^2 Q(G) - \varepsilon n^2$$

$$A_\varepsilon \geq Q(G) - \tau.$$ 

\[\Box\]

4.2 Lower Bounds

The first two lower bounds for regular graphs are non-constructive, whereas the rest of the lower bounds are constructive. Since any modularity value is upper bounded by 1, the constructive lower bounds constitute efficient approximation algorithms for the problem.

4.2.1 Unweighted Graphs

The first non-constructive lower bound follows from a result of Alon [1997] that upper bounds the bisection width of extremely sparse regular graphs.

**Lemma 22.** Given a $d$-regular $n$ vertex graph $G$, if $n > 40d^9$, then $Q(G) > \frac{1}{4\sqrt{d}}$.

**Proof.** For simplicity, we will assume that $n$ is even. The same proof holds for the odd case for large enough $n$.

It was shown in Alon [1997] that every $d$-regular graph with $n$ nodes having $n > 40d^9$ has a bisection (a cut with equal number of vertices on each side) having at most $\frac{n}{2} \left( \frac{d}{2} - \frac{\sqrt{d}}{4} \right)$ cut edges. Let $\mathcal{C} = \{C, \overline{C}\}$ be such a bisection. Then
by Equation (2.8), we get

\[ Q(C, \mathcal{C}) = 2\mu(C)(1 - \mu(C)) - e(C, \mathcal{C}) \]
\[ \geq \frac{1}{2} - \left( \frac{1}{2} - \frac{1}{4\sqrt{d}} \right) \]
\[ = \frac{1}{4\sqrt{d}}. \]

The above lemma is tight in the sense that there exist \(d\)-regular graphs for which \(Q(G) = O\left(\frac{1}{\sqrt{d}}\right)\) (the proof of Theorem 8 shows that \(d\)-regular expanders are one such class of graphs).

The proof of the next non-constructive lower bound uses a result in Henning and Yeo [2007] on the size of a maximum-cardinality matching of a regular graph.

**Lemma 23.** For a \(d\)-regular \(n\) vertex graph \(G\), \(Q(G) > \frac{0.86}{d} - \frac{1.86}{n}\).

**Proof.** Consider a maximum-cardinality matching \(\{u_1, v_1\}, \ldots, \{u_k, v_k\}\) of \(G\) of size \(k\). It is known Henning and Yeo [2007] that for any \(d > 2\),

\[ k \geq \begin{cases} \min \left\{ \frac{n(d^2 + 4)}{2d^2 + 2d + 4}, \frac{n - 1}{2} \right\}, & \text{if } d \text{ is odd} \\ \frac{d^3 - d^2 - 2)n - 2d + 2}{2(d^3 - 3d)}, & \text{otherwise} \end{cases} \]

which gives \(k > 0.43n\) for any \(d\). We create \(k\) clusters \(\{C_1, C_2, \ldots, C_k\}\) where \(C_i = \{u_i, v_i\}\) and for the remaining \(n - 2k\) nodes \(u\) not covered by the matching, we create singular clusters \(\{u\}\). Let the resulting clustering be \(\mathcal{C}\). Using Equation (2.4), we have

\[ Q(C) = i(C) - \sum_{C_i \in \mathcal{C}} \mu(C_i)^2 \]
\[ = \frac{k}{nd/2} - k \left( \frac{2}{n} \right)^2 - (n - 2k) \left( \frac{1}{n} \right)^2 \]
\[ \geq \frac{0.86}{d} - \frac{1.86}{n}, \]

where the last inequality follows by substituting for \(k\). \(\square\)
Let us now consider unweighted graphs where the maximum degree is small. The next lower bound is constructive and uses the algorithm of Figure 4.1. We will call this algorithm MODALG.

1. Initially, $G'$ is set to the input graph and $C$ is the empty clustering.
2. Form a cluster $C \subseteq V(G')$ out of the node with the highest degree in $G'$ and its neighbors. Add this cluster to the clustering, i.e., $C = C \cup C$.
3. Remove the vertices (and the adjacent edges) in $C$ from the graph $G'$.
4. Repeat Step 2 until there are no edges left in $G'$.
5. Form singleton clusters out of the remaining vertices.

Figure 4.1: Greedy algorithm MODALG for modularity clustering in unweighted graphs.

Lemma 24. Given an unweighted graph $G$ with $n$ vertices and a maximum degree $\Delta$ satisfying $\Delta \leq \frac{\sqrt{n}}{2}$, the clustering $C$ returned by algorithm MODALG satisfies

$$Q(C) \geq \frac{1}{2\Delta}.$$  

Proof. Let the result of MODALG be the clustering $C$. Without loss of generality, we can assume that the graph is connected.

We will first analyze the total fraction of internal edges, $i(C)$. The number of edges going out of any cluster in $C$ is at most $\Delta^2$. This means that for any cluster, the number of internal edges of that cluster are at least a $\frac{1}{\Delta}$ fraction of the number of outgoing edges from that cluster. Combining over all the clusters, this gives

$$i(C) \geq \frac{1}{\Delta + 1}. \quad (4.1)$$

Let the total number of edges in $G$ be denoted by $m$. Now the second term
in Equation (2.4) can be upper bounded as follows:
\[
\sum_{C_i \in \mathcal{C}} \mu(C_i)^2 \leq (\mu(C_i))_{\text{max}} \sum_{C_i \in \mathcal{C}} \mu(C_i) \\
= (\mu(C_i))_{\text{max}} \leq \frac{\Delta^2}{2m} \\
\leq \frac{\Delta^2}{n} \leq \frac{1}{8\Delta},
\]
where the first equality follows from the fact that the sum of the fractional degrees of all the clusters is 1, the third inequality follows from the fact that $G$ is connected, and the final inequality uses our upper bound on $\Delta$.

Finally, substituting the bounds (4.1) and (4.2) in Equation (2.4) we get
\[
Q(\mathcal{C}) = i(\mathcal{C}) - \sum_{C_i \in \mathcal{C}} \mu(C_i)^2 \\
\geq \frac{1}{\Delta + 1} - \frac{1}{8\Delta} \\
\geq \frac{1}{2\Delta},
\]
where the final inequality follows from the fact that $\Delta \geq 2$. \hfill \Box

The above lemma shows that MODALG achieves an approximation factor of $2\Delta$ for unweighted graphs $G$ that have maximum degree $\Delta$ satisfying $\Delta \leq \frac{3\sqrt{n}}{2}$. This easily follows from the fact that optimal modularity always satisfies $Q(G) < 1$.

### 4.2.2 Weighted Graphs

Recall our definition of weighted modularity from Section 2.1.1. We can think of a weighted graph as a complete graph with non-negative weights (between 0 and 1) on the edges. In this case, we will show that a simple tweak to the greedy procedure from the unweighted case will give us good clusters.
Note that we cannot use the MODALG procedure directly, since the entire graph could be put into a single cluster. Therefore, we need to eliminate very small weights from the graph before forming the clusters. The new algorithm is called WMODALG and it is outlined in Figure 4.2.

1. Initialize graph $H$ from input graph $G$ by removing the edges that have weights less than $\frac{1}{2}$.
2. Run Algorithm MODALG on the graph $H$.

Figure 4.2: Greedy algorithm WMODALG for modularity clustering in weighted graphs.

Before we proceed with the analysis of WMODALG, we will need our weighted graph to be nicely behaving. The weights have to satisfy some general uniformity conditions so as to avoid degenerate cases (like a weight distribution where 99% of the weights are extremely small). Note that we did something similar in the unweighted case when we avoided disconnected graphs. Such cases are easy from a modularity maximization standpoint.

To this end, we will call a weighted graph with $n$ vertices and total weight $m$ uniform if $m \geq n$ and if the sum of the weights larger than $\frac{1}{2}$ is at least $\frac{m}{2}$.¹

Lemma 25. Given a uniform weighted graph $G$ with $n$ vertices and a maximum degree $\Delta$ satisfying $\Delta \leq \frac{2\sqrt{n}}{3}$, the clustering $\mathcal{C}$ returned by algorithm WMODALG satisfies

$$Q(\mathcal{C}) \geq \frac{1}{8\Delta}.$$ 

Proof. Let the result of WMODALG be the clustering $\mathcal{C}$. Note that the maximum degree in the reduced graph $H$ is still $\Delta$.

¹ Note that there is nothing special about the fraction $\frac{1}{2}$ in this definition. Any constant fraction will give us a lower bound for the greedy algorithm in terms of the maximum degree.
We will first analyze the total fraction of internal edges, $i(C)$. Let $i_H(C)$ be the total fraction of internal edges relative to the graph $H$. Since all the edge weights in $H$ are bigger than $\frac{1}{2}$, any node has a maximum of $2\Delta$ neighbors. In round $r$, let $\Delta_r$ be the maximum degree in the unclustered part of $H$. Then, the total weight of all the outgoing edges of any cluster in $C$ is at most $2\Delta_r^2$. This means that for any cluster, the number of internal edges of that cluster are at least a $\frac{1}{2\Delta_r}$ fraction of the number of adjacent edges on that cluster. Combining over all the clusters, using our uniformity condition, and noting that $\Delta \geq \Delta_r$ for any round $r$, this gives

$$i(C) \geq \frac{i_H(C)}{2} \geq \frac{1}{2(\Delta + 2)}. \quad (4.3)$$

Let the total weight of the edges in $G$ be denoted by $m$. The analysis of the second term in Equation (2.4) is very similar to the unweighted case:

$$\sum_{C_i \in C} \mu(C_i)^2 \leq (\mu(C_i))_{\max} \sum_{C_i \in C} \mu(C_i) = (\mu(C_i))_{\max} \leq \frac{2\Delta^2}{2m} \leq \frac{\Delta^2}{n} \leq \frac{1}{27\Delta}, \quad (4.4)$$

where again the first equality follows from the fact that the sum of the fractional degrees of all the clusters is 1, the third inequality follows from the fact that $G$ is uniform, and the final inequality uses our upper bound on $\Delta$.

Finally, substituting the bounds (4.3) and (4.4) in Equation (2.4) we get

$$Q(C) = i(C) - \sum_{C_i \in C} \mu(C_i)^2$$

$$\geq \frac{1}{2(\Delta + 2)} - \frac{1}{27\Delta}$$

$$\geq \frac{1}{8\Delta},$$

where the final inequality follows from the fact that $\Delta \geq 1$. \qed
4.3 SDP based algorithm

In this section we detail the logarithmic (in the maximum degree) approximation algorithm for sparse graphs, both unweighted and weighted.

4.3.1 Charikar-Wirth Rounding

Recall the vector program for bimodularity (Figure 2.2):

$$\text{Maximize } \frac{1}{2} \sum_{u,v} M_{u,v} \bar{x}_u \cdot \bar{x}_v \quad \text{subject to } \|\bar{x}_v\| = 1 \quad \forall v. \quad (4.5)$$

where $M$ stood for the modularity matrix,

$$M_{u,v} = \frac{1}{2m} \left( A_{u,v} - \frac{d_u d_v}{2m} \right).$$

The output of the vector program (6.2) for a given graph $G$ is the maximum value, which we will call $Q_{VP}(G)$, and a set $\{\bar{x}_v\}_{v \in V(G)}$ of vectors (one for each vertex). Note that the solution to the SDP always satisfies

$$Q_{VP}(G) \geq Q_2(G) \geq \frac{Q(G)}{2}.$$ 

We need to convert the SDP’s solution vectors to $\pm 1$ values signifying a cut. We will use the Charikar-Wirth Rounding algorithm from Charikar and Wirth [2004]. This is a special case of the $RPR^2$ rounding (randomized projection, randomized rounding) from Feige and Langberg [2006] and is outlined below in Figure 4.3.

4.3.2 Analysis

In this section we will analyze the $RPR^2$ procedure of Figure 4.3. The main theorem for this section is given below.
Given vector program solutions \( \{ \vec{x}_v \}_{v \in V(G)} \) to input graph \( G \), and a parameter \( T > 0 \):

1. Pick a random vector \( \vec{r} \in \mathbb{R}^n \), where each coordinate is sampled from an independent standard normal distribution.

2. **Randomized Projection:** For every vertex \( v \in V(G) \), obtain new variables,

\[
y_v = \begin{cases} 
-1 & \text{if } \frac{\vec{x}_v \cdot \vec{r}}{T} < -1 \\
+1 & \text{if } \frac{\vec{x}_v \cdot \vec{r}}{T} > 1 \\
\frac{\vec{x}_v \cdot \vec{r}}{T} & \text{otherwise.}
\end{cases}
\]

3. **Randomized Rounding:** For every vertex \( v \in V(G) \), obtain final \( \pm 1 \) solution:

\[
x_v = \begin{cases} 
-1 & \text{with probability } \frac{1 - y_v}{2} \\
+1 & \text{with probability } \frac{1 + y_v}{2}.
\end{cases}
\]

**Figure 4.3:** Rounding Algorithm for SDP.

**Theorem 26.** If the input graph \( G \) satisfies \( Q_2(G) \geq \eta \) and its modularity matrix \( M \) satisfies \( \text{Tr}(-M) \leq \frac{\eta}{44 \ln(1/\eta)} \), then the output of the Charikar-Wirth rounding on the solution of the modularity SDP satisfies

\[
\mathbb{E} \left[ \frac{1}{2} \sum_{u,v} M_{u,v} x_u x_v \right] \geq \frac{Q_2(G)}{44 \ln(1/\eta)}.
\]

**Proof.** We will set \( T = \sqrt{22 \ln(1/\eta)} \) in the rounding of Figure 4.3.

For the purpose of separating the error caused by truncating the quantity \( \frac{\vec{x}_v \cdot \vec{r}}{T} \) to between \( +1 \) and \( -1 \), let us define the truncation error term for a pair of vertices \( u, v \) as:

\[
\delta_{uv} = \frac{(\vec{x}_u \cdot \vec{r})(\vec{x}_v \cdot \vec{r})}{T^2} - y_u y_v.
\]
Now for any pair of vertices $u$ and $v$, it is easy to see that when $u \neq v$,

$$
\mathbb{E}[x_u x_v] = \mathbb{E}_r[y_u y_v]
= \mathbb{E}_r \left[ \frac{(\vec{x}_u \cdot \vec{r})(\vec{x}_v \cdot \vec{r})}{T^2} \right] - \mathbb{E}_r[\delta_{uv}]
= \frac{\vec{x}_u \cdot \vec{x}_v}{T^2} - \mathbb{E}_r[\delta_{uv}],
$$

where the final equality follows from the fact that the coordinates of $\vec{r}$ are independent identically distributed standard normals. When $u = v$, it is easy to see that

$$
\mathbb{E}[x_u x_v] = 1.
$$

We shall use the following result of Charikar and Wirth [2004] that says that the expected truncation error is exponentially small in $T^2$.

**Lemma 27 (Charikar and Wirth [2004]).** For any pair of vertices $u \neq v$,

$$
|\mathbb{E}_r[\delta_{uv}]| \leq 8e^{-(T^2/2)}.
$$

Thus, summing over all pairs, we get

$$
\mathbb{E} \left[ \frac{1}{2} \sum_{u,v} M_{u,v} x_u x_v \right]
\geq \frac{Q_{VP}(G)}{T^2} - \frac{1}{2} \sum_{u \neq v} M_{u,v} \mathbb{E}_r[\delta_{uv}] - \frac{\text{Tr}(-M)}{2}, \quad (4.6)
$$

where the inequality for the first term follows from the fact that the diagonal entries of the matrix $M$ are all negative. The middle term can be simplified as
follows:

\[
\frac{1}{2} \sum_{u \neq v} M_{u,v} \mathbb{E}[\delta_{uv}] \leq \frac{1}{2} \left| \sum_{u \neq v} M_{u,v} \mathbb{E}[\delta_{uv}] \right|
\]

\[
\leq \frac{1}{2} \sum_{u \neq v} |M_{u,v}| \left| \mathbb{E}[\delta_{uv}] \right|
\]

\[
\leq \frac{8e^{-(T^2/2)}}{2} \sum_{u \neq v} |M_{u,v}|
\]

\[
\leq \frac{8e^{-(T^2/2)}}{2} \left( \sum_{u,v} A_{u,v} + \sum_{u,v} \frac{d_u d_v}{4m^2} \right)
\]

\[
= 8e^{-(T^2/2)},
\]

where we have used Lemma 27 to bound the error term. Substituting this back in Equation (4.6) we get

\[
\mathbb{E} \left[ \frac{1}{2} \sum_{u,v} M_{u,v} x_u x_v \right] \geq \frac{Q_{VP}(G)}{T^2} - 8e^{-(T^2/2)} - \frac{\text{Tr}(-M)}{2}
\]

\[
\geq \frac{Q_2(G)}{T^2} \left( \frac{1}{T^2} - 8e^{-(T^2/2)/\eta} - \frac{\text{Tr}(-M)}{2\eta} \right)
\]

\[
\geq \frac{Q_2(G)}{44 \ln(1/\eta)},
\]

where the second inequality follows from \( Q_{VP}(G) \geq Q_2(G) \geq \eta \), and the final inequality follows from the upper bound on \( \text{Tr}(-M) \), our choice of the parameter \( T \), and the fact that \( \eta \leq \frac{1}{2} \).

The main results of this section now follow as simple consequences of the above result and the lower bounds we listed in Section 4.2.

**Theorem 28.** Given an unweighted graph \( G \) with \( n \) vertices and maximum degree \( \Delta \leq \frac{3 \sqrt{n}}{2} \), there exists a randomized polynomial time \( O(\log \Delta) \) approximation for \( Q(G) \).
Proof. Instantiate Theorem 26 with \( \eta = \frac{1}{4\Delta} \) and use the lower bound of Lemma 24 (along with the fact that \( Q_2(G) \geq \frac{Q(G)}{2} \)). It is easy to see that since \( \Delta \) is much smaller than \( n \), \( \text{Tr}(-M) \ll \frac{1}{200\Delta \log \Delta} \). Thus, we are guaranteed a cut \( C \) (with high probability) with modularity at least \( \Omega\left(\frac{1}{\log \Delta}\right)Q(G) \). \( \square \)

The following theorem applies to sparse weighted graphs. Its proof is similar to the unweighted case.

**Theorem 29.** Given a uniform weighted graph \( G \) with \( n \) vertices and maximum degree \( \Delta \leq \frac{\sqrt{n}}{3} \), there exists a randomized polynomial time \( O(\log \Delta) \) approximation for \( Q(G) \).
Chapter 5
Generalizations and Alternatives

In this chapter we will look at some results related to the approximability of modularity clustering. The first section will focus on extending the results in the previous chapters to directed weighted graphs.

The second section will explore alternatives to the modularity measure and extend the results in the previous chapters to these changed objectives.

Finally, we consider the problem of approximating the max-cut in graphs where both positive and negative weights are allowed. This is a generalization of the modularity measure, and it is $\mathcal{NP}$-hard to even tell whether the max-cut is non-negative. We show that under certain conditions, it is possible to efficiently establish the positivity of the max-cut.

5.1 Directed Graphs

Leicht and Newman [2008] generalized the modularity measure to weighted directed graphs in the following manner. Let $G = (V, E, \ell)$ be the input directed graph with $\ell: E \mapsto \mathbb{R}^+$ being the function mapping edges to non-negative weights. For a node $v \in V$, let $d^\text{in}_v$ and $d^\text{out}_v$ denote the weighted in-degree and the weighted out-degree of $v$, respectively. Let $m = \sum_{v \in V} d^\text{in}_v + \sum_{v \in V} d^\text{out}_v$ and let $A = [a_{u,v}]$ denote the weighted adjacency matrix of $G$, i.e., $a_{u,v} = \ell(u,v)$.
if \((u, v) \in E\) and \(a_{u,v} = 0\) otherwise. Note that the matrix \(A\) is not necessarily symmetric now. Then, Equation (2.1) computing the modularity value of a cluster \(C \subseteq V\) needs to be modified as

\[
Q(C) = \frac{1}{m} \left( \sum_{u,v \in C} \left( a_{u,v} - \frac{d_{u}^{\text{out}} d_{v}^{\text{in}}}{m} \right) \right)
\]

With some effort, we show that we can extend all our complexity results for undirected networks to directed networks. Let \(\Delta = \sum_{v \in V} d_{v}^{\text{in}} = \sum_{v \in V} d_{v}^{\text{out}}\) denote the average weighted degree of nodes of \(G\), and let \(d_{\text{max}}^{\text{in}} = \max_{v \in V} d_{v}^{\text{in}}\) and \(d_{\text{max}}^{\text{out}} = \max_{v \in V} d_{v}^{\text{out}}\) denote the maximum weighted in-degree and maximum weighted out-degree, respectively, of nodes in \(G\). For convenience, we normalize all the weights such that \(\sum_{v \in V} d_{v}^{\text{in}} + \sum_{v \in V} d_{v}^{\text{out}}\) is exactly twice the number of directed edges of \(G\). Since the given graph can be assumed to be weakly-connected, \(\Delta \geq 1 - \frac{1}{n}\).

**Theorem 30.**

(a) For any directed graph \(G\), computing \(Q_2(G)\) is \(\mathcal{NP}\)-complete even if every node \(v\) has \(d_{v}^{\text{in}} = d_{v}^{\text{out}} = d\), for any fixed \(d \geq 9\).

(b) It is \(\mathcal{NP}\)-hard to approximate \(Q(G)\) within a factor of 1.0009.

(c) There is an \(O(\log d)\) approximation algorithm for unweighted directed graphs if the in-degree and out-degree of all nodes is exactly the same, say \(d\), and \(d \leq \frac{n}{100}\).

(d) There is an \(O\left(\log \left(\frac{d_{\text{max}}^{\text{in}} + d_{\text{max}}^{\text{out}}}{\sqrt{n}}\right)\right)\)-approximation algorithm for weighted graphs provided \(\max \left\{ d_{\text{max}}^{\text{in}}, d_{\text{max}}^{\text{out}} \right\} \leq \frac{3\sqrt{n}}{64}\).

**Proof.** Remember that

\[
Q(C) = \frac{1}{m} \left( \sum_{u,v \in C} \left( a_{u,v} - \frac{d_{u}^{\text{out}} d_{v}^{\text{in}}}{m} \right) \right)
\]

The corresponding modification in Equation (2.3) is

\[
Q(C) = \sum_{C_i \in \mathcal{C}} \left( \frac{m_i}{m} - \left( \frac{D_{i}^{\text{in}} \times D_{i}^{\text{out}}}{m^2} \right) \right)
\]
where \( D_{i}^{\text{in}} = \sum_{v \in C_{i}} d_{v}^{\text{in}} \), \( D_{i}^{\text{out}} = \sum_{v \in C_{i}} d_{v}^{\text{out}} \) and \( m_{i} \) as the total weight of edges whose both endpoints are in the cluster \( C_{i} \). Finally, since \( \sum_{v \in V} \left( a_{u,v} - \frac{d_{v}^{\text{out}} d_{v}^{\text{in}}}{m} \right) = 0 \) for any \( u \in V \), we can alternatively express \( Q(C) \) as

\[
Q(C) = \frac{1}{m} \left( \sum_{u \in C, v \not\in C} \left( \frac{d_{v}^{\text{out}} d_{v}^{\text{in}}}{m} - a_{u,v} \right) \right).
\]

Thus, Equation (2.5) now becomes

\[
Q(C) = \sum_{C_{i},C_{j}} \left( \frac{D_{i}^{\text{out}} D_{j}^{\text{in}}}{m^2} - \frac{m_{ij}}{m} \right)
\] (5.3)

where \( m_{ij} \) as the total weight of the edges directed from \( C_{i} \) to \( C_{j} \).

(a) & (b) These two results follow by the following easy observation. Consider a given undirected unweighted graph \( G \) with \( n \) nodes and \( m \) edges, and let \( \tilde{G} \) be the directed graph obtained by replacing each edge \( \{u,v\} \) of \( G \) by two directed edges \( (u,v) \) and \( (v,u) \), each of weight 1; thus \( \tilde{m} = \sum_{v \in V} d_{v}^{\text{in}} + \sum_{v \in V} d_{v}^{\text{out}} = 4m \). Let \( \tilde{A} = [\tilde{a}_{u,v}] \) be the adjacency matrix of \( \tilde{G} \), and \( \tilde{d}_{v}^{\text{in}} \) and \( \tilde{d}_{v}^{\text{out}} \) be the in-degree and out-degree of the node \( v \) in \( \tilde{G} \). Then, it is easy to see that every clustering of \( G \) of modularity value \( x \) translates to a corresponding clustering of \( \tilde{G} \) of the same modularity value and vice versa.

(c) & (d) It is easy to see that the proof of Lemma 3 works for directed networks as well by using Equation (5.3) instead of Equation (2.5) in the proof. Thus again it suffices to approximate \( Q_{2}(G) \).

Let \( W = [w_{u,v}] \in \mathbb{R}^{n \times n} \) be the matrix whose entries are defined by \( w_{u,v} = a_{u,v} - \frac{d_{v}^{\text{out}} d_{v}^{\text{in}}}{2m} \). Then, letting \( x_{u} \in \{-1, 1\} \) be the indicator variable denoting in which partition the node \( u \in V \) belongs, Equation (5.1) can be rewritten for a 2-clustering of directed networks as

\[
Q(C) = \sum_{u,v \in V} w_{u,v} (1 + x_{u} x_{v}) = \sum_{u,v \in V} w_{u,v} x_{u} x_{v}
\]

\[
= x^{T} W x = x^{T} \left( \frac{W + W^{T}}{2} \right) x = x^{T} W' x
\]
where \( W' = \frac{W + W^T}{2} = [w'_{u,v}] \) is a symmetric matrix. Note that
\[
w'_{u,v} = \frac{\delta_{u,v} - \frac{d_{in}^u d_{in}^v + d_{out}^u d_{out}^v}{2m}}{2m}
\]
where \( \delta_{u,v} \) is given by:
\[
\delta_{u,v} = \delta_{v,u} = \begin{cases} 
1, & \text{if both } (u,v) \in E \text{ and } (v,u) \in E \\
0, & \text{if both } (u,v) \notin E \text{ and } (v,u) \notin E \\
\frac{1}{2}, & \text{otherwise.}
\end{cases}
\]

Let \( \hat{W} = [\hat{w}_{u,v}] \) be the real symmetric matrix defined by
\[
\hat{w}_{u,v} = \begin{cases} 
0, & \text{if } u = v \\
w'_{u,v}, & \text{otherwise.}
\end{cases}
\]

As in the proof of Theorem 26, it follows that \( \sum_{u,v \in V} \hat{w}_{u,v} < 2 \). For notational convenience, define \( D = \text{Tr} \left( \hat{W} - W' \right) = \sum_{u \in V} w'_{u,u} \) and \( Q'_2(G) = \max_{x \in \{0,1\}^n} x^T \hat{W} x \).

\( (c) \) \( G \) is an unweighted directed graph with \( d_{in}^v = d_{out}^v = d \) for every node \( v \), and \( d \leq \frac{n}{100} \).

The proof of Theorem 26 on the quadratic form \( \max_{x \in \{0,1\}^n} x^T \hat{W} x \) gives an approximation factor of \( \gamma \ln d \), for some constant \( \gamma > 0 \), for our directed network provided we can show that
\[ \frac{Q'_2(G)}{\gamma \ln d} - D = \Omega \left( \frac{Q'_2(G)}{\gamma \ln d} \right) \], and
\[ Q_2(G) = \Omega \left( d^{-c} \right) \text{ for some constant } c > 0. \]

Let \( H \) be the undirected graph obtained from the given graph \( G \) by ignoring the direction of the edges and removing parallel edges (if any); every node in \( H \) has a degree between \( d \) and \( 2d \). Greedily pick a maximal matching in \( H \), each time selecting an edge and deleting all (at most \( 4d - 1 \)) edges that have a common endpoint with the picked edge. Such a matching contains at least \( \frac{(nd)/2}{4d} = \frac{n}{8} \) edges,
each of weight at least \( \frac{1}{4m} - \frac{8d^2}{4m^2} = \frac{1}{8dn} - \frac{1}{2n^2} \) in \( G \). Consider the clustering of \( G \) where each edge in the matching is a separate cluster of two nodes, and each of the remaining nodes is a separate cluster of one node. The modularity value of this solution is at least

\[
\left( \frac{1}{8dn} - \frac{1}{2n^2} \right) \frac{n}{8} - \text{Tr} \left( W' - \hat{W} \right) \geq \frac{1}{64d} - \frac{1}{16n} - \frac{1}{2n}
\]

Thus, \( Q'_2(G) \geq \frac{1}{128d} - \frac{9}{32n} = \Omega(d^{-1}) \). Moreover, since \( d \leq \frac{n}{100} \) we have

\[
\frac{Q'_2(G)}{\ln d} - D = \frac{Q'_2(G)}{\ln d} - \frac{1}{2n} = \Omega \left( \frac{Q'_2(G)}{\ln d} \right)
\]

(d) \( \max \{ d_{\text{max}}^{\text{in}}, d_{\text{max}}^{\text{out}} \} < \frac{3\sqrt{n}}{64} \).

Let \( G'' = (V, E'') \) be the undirected weighted graph obtained from \( G \) whose adjacency matrix is \( W'' = [w''_{u,v}] \) with

\[
w''_{u,v} = \begin{cases} 
    w'_{u,v} - \frac{1}{2}, & \text{if } \delta_{u,v} = 1 \\
    w'_{u,v}, & \text{otherwise}.
\end{cases}
\]

Since \( w''_{u,v} \geq w'_{u,v} \), it suffices to show an approximation for \( \max_{x \in \{0,1\}^n} x^TW''x \).

The algorithm in the proof of Theorem 26 with \( W = W'' \) can now be appropriately modified to obtain the desired approximation if one identified the quantity \( d_{\text{max}} \) in that proof with \( d_{\text{max}}^{\text{in}} + d_{\text{max}}^{\text{out}} \). \qed

### 5.2 Alternatives to Modularity

#### 5.2.1 The max-min Objective

Exact or approximate solutions to the modularity measure may produce many trivial clusters of single nodes. For example, the following proposition shows that for a large class of graphs there exists a clustering in which every cluster except
one consists of a single node gives a modularity value that has a modularity value of at least 25% of the optimal.

**Proposition 31.** There exists a clustering for a graph $G$ in which every cluster except one consists of a single node and whose modularity value is at least 25% of the optimal if

- $G$ is $d$-regular with $d < \frac{n}{8}$, or
- $G$ is an uniform weighted graph with $d_{\text{max}} < \frac{3\sqrt{n}}{32}$.

**Proof.** Let $\{V', V \setminus V'\}$ be an optimal 2-clustering of $G$. By Lemma 3 and Lemma 5 $Q(V') = \frac{Q(G)}{2} \geq \frac{Q(G)}{4}$. Suppose that we replace the cluster $V \setminus V'$ by $|V \setminus V'|$ trivial clusters each of a single node, and let $C$ be this new clustering. If $G$ is $d$-regular, then $Q(C) = Q(V') - D = \frac{Q(G)}{4} - \frac{1}{n}$. By Lemma 23, $Q(G) > \frac{0.86}{d} - \frac{4}{n}$, and thus $Q(C) = \frac{Q(G)}{4} - o(1)$.

Similarly, for the case when $G$ is a uniform weighted with $d_{\text{max}} < \frac{3\sqrt{n}}{32}$, the proof of Theorem 26 shows that $D \leq o(1)$, and thus $Q(C) = Q(V') - D \geq \frac{Q(G)}{4} - o(1)$. By Lemma 25 $Q(G) > \frac{1}{8d_{\text{max}}}$, and thus again $Q(C) = \frac{Q(G)}{4} - o(1)$.

We investigate one alternative to overcome such a shortcoming: define the modularity of the network as the minimum of the modularities of individual clusters. The max-min modularity of a clustering $C$ now becomes

$$Q^{\text{max-min}}(C) = \min_{C_i \in C} Q(C_i)$$

We will add the superscript “max-min” to differentiate the relevant quantities for this objective from the usual summation objective discussed before, e.g., we will use $Q(G)^{\text{max-min}}$ instead of $Q(G)$. In a nutshell, our results in the following lemma show that the max-min objective indeed avoids generating trivial clusters.
(Lemma 32(a)), and the optimal objective value for max-min objective is precisely scaled by a factor of 2 from that of the SUM objective, thereby keeping the overall quantitative measure the same (Lemma 32(b)).

**Lemma 32.** Let $G$ be a weighted undirected graph with $m$ edges and maximum degree $d_{\text{max}}$. Then, the following claims hold:

(a) No optimal solution for max-min objective has a cluster with fewer than $\frac{4mQ(G)_{\text{max-min}}}{d_{\text{max}}}$ nodes.

(b) $Q(G)_{\text{max-min}} = \frac{Q_2(G)}{2}$.

**Proof.**

(a) Since only an edge with positive weight can increase the modularity of a cluster, it is easy to check that a cluster with $y$ nodes can have a modularity value of at most $yd_{\text{max}}/4m$.

(b) Consider an optimal clustering $\mathcal{C} = \{V_1, V_2, \ldots, V_k\}$ with a minimum number $k$ of clusters such that $Q(G)_{\text{max-min}} = Q_{\text{max-min}}(\mathcal{C}) = \min_{1 \leq i \leq k} \{Q(V_i)\} > 0$. First, consider the case when $k > 3$. We will show that for some non-empty subset $T$ of $\{V_1, V_2, \ldots, V_k\}$ we must have $Q(\cup_{V_j \in T}V_j) \geq Q_{\text{max-min}}^\mathcal{C}$; this contradicts the minimality of $k$ in our choice of the optimal cluster. Note that $Q(\mathcal{C}) = \sum_{i=1}^{k} Q(V_i) \geq k \cdot Q_{\text{max-min}}^\mathcal{C}$. We will make use of Equation (2.1) of modularity of a cluster. Let $Q(\tilde{\mathcal{C}}) = \frac{1}{2m} \left( \sum_{u \in V_i, v \in V_j, i \neq j} (a_{u,v} - \frac{d_u d_v}{2m}) \right)$. Then, $Q(\tilde{\mathcal{C}}) = -Q(\mathcal{C})$.

Consider a subset $T$ obtained by randomly and uniformly selecting each $V_i$ with a probability of $\frac{1}{2}$. Note that each pair of nodes $u$ and $v$ belonging to the same cluster is selected with a probability of $\frac{1}{2}$, whereas each pair of nodes belonging
to different clusters is selected with a probability of $\frac{1}{4}$. Thus,
\[
\mathbb{E} \left[ Q(\bigcup_{V_j \in T} V_j) \right] = \frac{Q(C)}{2} + \frac{Q(\tilde{C})}{4} = \frac{Q(C)}{4}
\geq \left( \frac{k}{4} \right) Q^{\text{max-min}}(C) \geq Q^{\text{max-min}}(C)
\]
and therefore there exists such a subset $T$ with the properties as claimed.

Otherwise, consider the case when $k = 3$. Let $Q_{i,j} = \sum_{u \in V_i} (a_{u,v} - d_{u,v}^2) / 2m$ for $i < j$. Without loss of generality, let $Q(V_1) = a$, $Q(V_2) = a + b$ and $Q(V_3) = a + c$ for some $a > 0$ and $b \geq c \geq 0$; thus, $Q^{\text{max-min}}(C) = a$. Consider the three 2-clusterings of $G$: $C_1 = (V_1 \cup V_2, V_3)$, $C_2 = (V_2 \cup V_3, V_1)$ and $C_3 = (V_1 \cup V_3, V_2)$. Since none of these three 2-clusterings should be an optimal solution, we must have
\[
Q^{\text{max-min}}(C_1) - Q^{\text{max-min}}(C) < 0
\]
\[
\equiv \min \{ 2a + b + Q_{1,2}, a + c \} < a \equiv Q_{1,2} < -(a + b)
\]
\[
Q^{\text{max-min}}(C_2) - Q^{\text{max-min}}(C) < 0
\]
\[
\equiv \min \{ 2a + b + c + Q_{2,3}, a \} < a \equiv M_{2,3} < -(a + b + c)
\]
\[
Q^{\text{max-min}}(C_3) - Q^{\text{max-min}}(C) < 0
\]
\[
\equiv \min \{ 2a + c + Q_{1,3}, a \} < a \equiv M_{1,3} < -(a + c)
\]
Thus, we have $Q(V_1) + Q(V_2) + Q(V_3) = 3a + b + c = -Q_{1,2} - Q_{2,3} - Q_{1,3} > 3a + 2b + 2c$ which implies $b + c < 0$, contradicting $b \geq c \geq 0$.

Thus, we have shown there is an optimal solution for our max-min objective with no more than two clusters. Obviously, if $Q(G)^{\text{max-min}} > 0$ then an optimal solution cannot consist of a single cluster. Let $V_1, V_2$ be the two clusters in this case. By Lemma 5, we have $Q(V_1) = Q(V_2)$ which implies $Q(G)^{\text{max-min}} = \frac{Q_2(G)}{2}$.

\[\square\]
5.2.2 The Erdös-Rényi null model

A theoretically appealing choice for alternative null models is the classical Erdös-Rényi random graph model $G(n, p)$, namely each possible edge $\{u, v\}$ is selected in $G$ uniformly and randomly with a probability of $p$ for some fixed $0 < p < 1$. To summarize, our results in this section show that the new modularity measure is precisely Newman’s modularity measure on an appropriately defined regular graph, and thus our previous results on regular graphs can be applied to this case.

We will add the superscript “ER” to differentiate the relevant quantities for this objective from the usual summation objective discussed before, e.g., we will use $Q(G)^{ER}$ instead of $Q(G)$. For simplicity, we consider the case of unweighted graphs only. Let $G = (V, E)$ be the given unweighted input graph with $m = n\Delta$ number of edges. Select $p = \frac{2\Delta}{n-1}$ such that the null model has the same number of edges in expectation as the given graph $G$. The modularity of cluster $C$ then becomes

$$Q^{ER}(C) = \frac{\sum_{u,v\in C}(a_{u,v} - p)}{2m}$$

Let $n$ be sufficiently large such that $p \approx (2\Delta)/n$. It can then be seen that $Q^{ER}(C)$ is precisely the same as $Q(C)$ on a $(2\Delta)$-regular graph. Thus, our previous results on regular graphs can be generalized to this case in the following manner:

- Computing $Q(G)^{ER}$ is $\mathcal{NP}$-complete for graphs with $\Delta \geq 18$.
- If $\Delta < \frac{n}{16}$ then the problem admits a $O(\log \Delta)$-approximation.

5.3 Max-Cut with General Weights

If we allow edge weights for a given graph to be both negative and non-negative, then, intuitively, a maximum cut on this graph should be one in which positive
edges cross the cut and negative edges do not cross the cut. However, can we even tell if a cut exists such that the total weight crossing it is positive? Although several special cases of MAX-CUT\(\pm\) have been studied McCormick et al. [2003], to the best of our knowledge, the complexity of determining if a graph with both negative and non-negative edge weights has a positive cut has not been studied. Note that when all edges are non-positive, then MAX-CUT\(\pm\) is equivalent to the minimum cut problem in the graph where all edge weights are replaced by their absolute values. Since this extreme case (all non-positive weights) is well-known to be polynomial time solvable, and the other extreme case (all non-negative weights) can be approximated to a large positive constant, a natural question is the approximability of the problem in between these two extreme cases, i.e. both negative and non-negative edge weights allowed.

It turns out that this problem is quite hard in the worst case. In Section 5.3.1, we show that it is \(\text{NP}\)-hard to determine if such a graph has any cut with positive value. In particular, this implies that MAX-CUT\(\pm\) has no multiplicative approximation guarantee unless \(\mathcal{P} = \mathcal{NP}\), since any such guarantee would allow one to determine if there exists a cut with positive value.

### 5.3.1 Hardness of MAXCUT\(\geq 0\)

The decision version of SPARSEST-CUT is stated as follows: Given a weighted graph \(G = (V, E)\) and a value \(t\), does there exist a normalized cut \((S, \overline{S})\) such that the value of this cut is at most \(t\)? In other words, a cut such that:

\[
\frac{w(S, \overline{S})}{|S| \cdot |\overline{S}|} < t.
\]

We also consider the decision version of MAX-CUT with positive and negative edge weights. Given a weighted graph, where weights may be any real number,
we ask, is there a cut with positive weight? We refer to this decision problem as \( \text{MAXCUT}_{\geq 0} \). Now we give a reduction from the decision version of \text{SPARSEST-CUT} to \( \text{MAXCUT}_{\geq 0} \).

**Lemma 33.** \( \text{MAXCUT}_{\geq 0} \) is \( \mathcal{NP} \)-complete.

**Proof.** The proof is via a reduction from \text{SPARSEST-CUT}. Let \((G = (V, E), t)\) be an instance of extscSparsest-Cut, with weight function \( w : E \to \mathbb{Z}^+ \). Let \( G' = (V, E') \) be a complete graph with weight function \( w' : E' \to \mathbb{Z} \), such that \( w'(i, j) = t - w(i, j) \). Thus, if an edge \((i, j)\) does not appear in \( E \), then \( w'(i, j) = t \), since \( w(i, j) = 0 \). If there exists a cut \((S, \overline{S})\) such that \( w'(S, \overline{S}) \geq 0 \), then it follows that:

\[
\begin{align*}
t \cdot |S| \cdot |\overline{S}| - w(S, \overline{S}) & \geq 0 \\
\therefore t & \geq \frac{w(S, \overline{S})}{|S| \cdot |\overline{S}|}.
\end{align*}
\]

Thus, we conclude that it is \( \mathcal{NP} \)-hard to decide if a graph with arbitrary real edge weights has a maximum cut with positive weight. \( \square \)

**Corollary 34.** The Max-Cut problem with both positive an negative edge weights does not have any multiplicative approximation guarantee.

This corollary follows from the fact that any multiplicative approximation guarantee would result in an algorithm for the decision problem Max-Cut0.

### 5.3.2 An algorithm for \( \text{MAXCUT}_{\geq 0} \)

We investigate the conditions under which the value of the maximum cut in a graph with general weights is positive.

Let \( G = (V, E) \) be a weighted graph, and let \( \{w_{ij}\} \) be a set of edge weights containing both positive and negative values.
Suppose we have a set of vectors, \( I = \{v_i\} \), corresponding to the vertices in \( V \). We define the function \( f \) as follows:

\[
f(I) = \max \sum_{i<j} w_{ij} \left( \frac{1 - v_i \cdot v_j}{2} \right)
\]

We consider a random hyperplane which results in a partition of the vertices \((S, \overline{S})\) according to the side of the hyperplane on which each vertex falls. Let \( C(S, \overline{S}) \) denote the weight of such a cut. Let \( \theta_{ij} = \arccos(v_i \cdot v_j) \). The expected value of such a cut is:

\[
\mathbb{E}[C(S, \overline{S})] = \sum_{i<j} w_{ij} \left( \frac{\theta_{ij}}{\pi} \right).
\]

If the value \( f(I) \) is sufficiently large, then we can prove that \( \mathbb{E}[C(S, \overline{S})] \) is non-negative. If we can compute an upper bound on the righthand side of the following equation,

\[
f(I) - \mathbb{E}[C(S, \overline{S})] = \sum_{i<j} w_{ij} \left( \frac{1 - v_i \cdot v_j}{2} - \frac{\theta_{ij}}{\pi} \right)
\leq \sum_{i<j} \left| w_{ij} \right| \left| \frac{1 - v_i \cdot v_j}{2} - \frac{\theta_{ij}}{\pi} \right|,
\]

then we can use this to lower bound the value of \( \mathbb{E}[C(S, \overline{S})] \). Define the function \( g(\theta) \) as follows:

\[
g(\theta) = 1 - \frac{\cos(\theta)}{2} - \frac{\theta}{\pi}.
\]

Since we want to upper bound the value \( f(I) - \mathbb{E}[C(S, \overline{S})] \), we want to upper bound the value of \( g(\theta) \). The following lemma can be proved by simple calculus.

**Lemma 35.** For \( \theta : 0 \leq \theta \leq \pi \), we have the following inequality:

\[
\left| \frac{1 - \cos(\theta)}{2} - \frac{\theta}{\pi} \right| \leq .1053.
\]

This leads us to our main result in this section.
Theorem 36. If there is an SDP solution for MAXCUT, $I = \{v_i\}$ such that $f(I) > (.1053)|W|$, then the optimal cut has a positive value.

Proof. Using Lemma 35, we have:

\[
\begin{align*}
    f(I) - \mathbb{E}[W(S, \overline{S})] & \leq \sum_{i < j} |w_{ij}|(.1053) \Rightarrow \quad (5.4) \\
    \mathbb{E}[C(S, \overline{S})] & \geq f(I) - (.1053)|W|. \quad (5.5)
\end{align*}
\]

Thus, if $f(I) > (.1053)|W|$, then (5.5) is positive, and the expected hyperplane rounding yields a positive cut. \qed
Chapter 6
Other results and Open Problems

In the previous chapters, we undertook a detailed study of the approximability of the modularity problem. We proved some hardness results and analyzed some approximation algorithms for the problem and even generalized some of them. There still remain a plethora of open problems in this area. In this concluding chapter, we will look at some of them.

In the first section, we will tackle the interesting open problem of achieving a constant factor approximation for bimodularity. We will show a sub-exponential time algorithm for this problem and an example calculation which might point to a future result.

In the next section, we study the effectiveness of the simple Goemans-Williamson rounding on the modularity SDP. It is still an open question as to what the approximation factor actually is in this case. However, we show an approximation factor that depends on the most negative value that the SDP can achieve.

In Section 3, we discuss the relation between the density of the graph and its modularity value. Finally, in Section 4, we investigate a possible connection between sparsest cut and bimodularity.
6.1 Sub-exponential time algorithm

In this section, we describe a sub-exponential time constant factor approximation for bimodularity. We also show a rough analysis for a specific value of modularity. It is an open question whether the algorithm indeed gives a constant factor approximation. The algorithm uses results from the breakthrough paper of Arora et al. [2010].

All graphs considered in this section are undirected and unweighted. Let $G = (V, E)$ denote the given input graph with $n = |V|$ nodes and $m = |E|$ edges, recall that $d_v$ denote the degree of a node $v \in V$, and let $A(G) = [a_{u,v}(G)]$ denote the adjacency matrix of $G$.

We will need the following definitions from Arora et al. [2010] for our calculation:

- If $G$ is $d$-regular for some given $d$, then its symmetric stochastic walk matrix is denoted by $\widehat{A}(G)$, i.e., $\widehat{A}(G) = \frac{a_{u,v}(G)}{d_v}$.

- For a $\tau \in [0, 1)$, the $\tau$-threshold rank of $G$, denoted by $\text{rank}_\tau(G)$, is the number of eigenvalues $\lambda$ of $\widehat{A}(G)$ satisfying $|\lambda| > \tau$.

- For a subset $\emptyset \subset S \subset V$ of nodes, the following quantities are defined:
  - The (normalized) measure of $S$ is $\mu(S) = \frac{|S|}{n}$. This is the same as the degree measure of $S$ since the graph is regular.
  - The (normalized) expansion of $S$ is given by
    \[ \Phi(S) = \frac{|\{ (u, v) \mid u \in S, v \notin S, \{u, v\} \in E \}|}{\sum_{v \in S} d_v} . \]
  - The (normalized) density of $S$ is $D(S) = 1 - \Phi(S)$.

- For a function $f(n)$, $\exp(f(n))$ denotes $2^{cf(n)}$ for some fixed constant $c > 0$. 
6.1.1 Small Set Expansion

The following results are from Arora et al. [2010], restated in our terminologies after instantiation of parameters with specific values and trivial algebraic simplification.

**Theorem 37.** Arora et al. [2010]

Assuming \( \text{rank}_{1-10^{-5}}(G) < n^{0.1} \), there is an \((\exp(n^{0.1})\text{poly}(n))\)-time algorithm that outputs a subset \( \emptyset \subset S \subset V \) such that \( 0.92 |S^*| \leq |S| \leq 1.08 |S^*| \) and \( \Phi(S) \leq \Phi(S^*) + 0.08 \).

**Theorem 38.** Arora et al. [2010]

Let \( H \) be a regular graph with \( r \) vertices. Assuming \( \text{rank}_{1-10^{-5}}(H) \geq r^{0.1} \), one can find in \( \text{poly}(r) \) time a subset \( S \) of vertices of \( H \) such that \( |S| \leq r^{1-10^{-3}} \) and \( \Phi(S) \leq 10^{-2} \).

6.1.2 The Remark

The following remark is a conjecture, and we will show the calculations for the specific value of \( \varepsilon = 10^{-6} \).

**Remark 39.** Let \( G \) be a \( d \)-regular graph. Then, Theorems 37 and 38 imply that there is an algorithm \( A_\varepsilon \) for \( \varepsilon > 0 \) with the following properties:

(a) \( A_\varepsilon \) runs in sub-exponential time, i.e., in time \( \exp(\delta n) \) for some constant \( 0 < \delta = \delta(\varepsilon) < 1 \) that depends on \( \varepsilon \) only.

(b) \( A_\varepsilon \) distinguishes instances with \( Q(G) \geq 1 - \varepsilon \) from instances with \( Q(G) \leq \varepsilon \).

**Proof.** Set \( \varepsilon = 10^{-6} \). We assume that \( G \) is \( d \)-regular, and either \( Q(G) \geq 1 - 10^{-6} \) or \( Q(G) \leq 10^{-6} \).

---

1 Instantiate Theorem 2.2 in Arora et al. [2010] with \( \eta = 10^{-4} \) and \( \varepsilon = 10^{-6} \).

2 Instantiate Theorem 2.2 in Arora et al. [2010] with \( \eta = 10^{-4} \) and \( \gamma = 0.1 \).
We will provide an approximation for $Q_2(G)$ and then use the fact that $Q_2(G) \geq \frac{Q(G)}{2}$. Note that if $Q(G) \leq 10^{-6}$ then obviously $Q_2(G) \leq 10^{-6}$, whereas if $Q(G) \geq 1 - 10^{-6}$ then $Q_2(G) \geq 0.5 - \frac{10^{-6}}{2}$.

Consider a partition $\mathcal{C}$ of $V$ into exactly two sets, say $S$ and $\overline{S} = V \setminus S$ with $0 < \mu(S) \leq \frac{1}{2}$. By Lemma 5, $Q(S) = Q(\overline{S})$, and thus

$$Q(\mathcal{C}) = 2 \left( i(S) - \left( \frac{|S|}{n} \right)^2 \right) = 2 (D(S)\mu(S) - \mu(S)^2)$$

Thus, letting $D = D(S), \mu = \mu(S)$ and $\Phi = \Phi(S) = 1 - D$, our goal is to maximize the following function over all possible valid choices of $D$ and $\mu$:

$$f(\mu, D) = 2 (\mu D - \mu^2) = 2 (\mu(1 - \Phi) - \mu^2)$$

Let $\mathcal{C}^* = \{ S^*, \overline{S}^* \}$ be an optimal solution for the problem of partitioning into 2 communities, with $D = D^*, \mu = \mu^*, \Phi = \Phi^*$ (and thus $Q_2(G) = f(\mu^*, D^*)$). Obviously,

$$\left| \mu^* - \frac{D^*}{2} \right| < \frac{D^*}{2}$$

$$f \left( \frac{D^*}{2} + \delta, D^* \right) = f \left( \frac{D^*}{2} - \delta, D^* \right) \text{ for any positive } \delta > 0$$

Note that we need to show that, if $Q_2(G) = f(\mu^*, D^*) > 0.5 - \frac{10^{-6}}{2}$, then there is an algorithm $A$ as described in Remark 39 that outputs a valid choice of $\mu$ and $D$, say $\mu'$ and $D'$, such that $f(\mu', D') > 10^{-6}$.

**Guessing $D^*$**

Note that there are at most $O(d n^2)$ choices for $D^*$ since $D^*$ is of the form $\frac{i}{(j \cdot d)}$ for $j \in \{1, 2, \ldots, \frac{n}{2}\}$ and $i \in \{1, 2, \ldots, j \cdot d\}$. In the sequel, we will run our algorithm for each choice of $D^*$ and take the best of these solutions. Thus, it will suffice to prove our approximation bound assuming we have guessed $D^*$ exactly.
In the remainder of the proof, we will make use of Theorems 37 and 38. The description is self-contained, and the reader will not need any prior knowledge of expansion properties of graphs. Remember that we assume that $f(\mu^*, D^*) > 0.5 - \frac{10^{-6}}{2}$ and thus $\mu^* > 0.5 - \frac{10^{-6}}{2}$. Since $\mu^* \leq \frac{1}{2}$, this implies $D^* = \frac{1 - 10^{-6}}{4\mu} + \mu > \sqrt{1 - 10^{-6}} > 1 - 10^{-6}$, and thus $\Phi^* = 1 - D^* < 10^{-6}$.

**Case I: Small Threshold Rank of $G$, i.e., $\text{rank}_{1-10^{-6}}(G) < n^{0.1}$**

We run the algorithm as outlined in Theorem 37, and return $\{S, S\}$ as our solution. Note that:

\[
\Phi(S) \leq \Phi^* + 0.08 < 0.080001 \implies D(S) > 1 - 0.080001 = 0.919999
\]

\[
0.92\mu^* \leq \mu(S) \leq 1.08\mu^* \implies 0.4599 \leq \mu(S) \leq 0.54
\]

and thus

\[
f(\mu(S), D(S)) = 2\mu(S) (D(S) - \mu(S)) > 2 \times 0.4599 \times (0.919999 - 0.54) > 10^{-6}
\]

**Case II: Remaining Case, i.e., $\text{rank}_{1-10^{-6}}(G) \geq n^{0.1}$**

Our strategy is to use the algorithm in Theorem 38 to repeatedly extract high-rank parts from the given graph until we cannot do so anymore\(^3\). Namely, we compute in polynomial time an ordered partition of nodes $(T_1, T_2, \ldots, T_k, V \setminus \bigcup_{i=1}^k T_i)$ such that each $T_i$ is obtained by using the algorithm in Theorem 38 on graph $G_i$ induced by the set of nodes $V \setminus \bigcup_{j=1}^{i-1} T_j$, and the last (possibly empty) graph $G''$ induced by the set of nodes $V'' = V \setminus \bigcup_{i=1}^k T_i$ satisfy $\text{rank}_{1-10^{-6}}(G'') < |V''|^{0.1}$. Let $G'$ be the graph induced by the set of nodes $V' = \bigcup_{i=1}^k T_i$. The following cases

---

\(^3\) Arora et al. [2010] points out how to regularize the remaining graph each time a set of nodes have been extracted by adding appropriate number of self-loops of weight $\frac{1}{2}$. 
arise.

**Case II(a) |** \( S^* \cap V'' \geq \frac{|S^*|}{2} \).

Let \( S^*_1 \) be the set containing an arbitrary \( \frac{|S^*|}{2} \) elements from the set \( S^* \cap V'' \). Note that \( \mu(S^*_1) = \frac{\mu^*}{2} \) and \( \Phi(S^*_1) \leq 2 \Phi^* \). Using Theorem 37 on the graph \( G'' \) with \( |S^*| \) replaced by \( \frac{|S^*|}{2} \) outputs a set \( S \) of nodes from \( V'' \) such that

\[
\Phi(S) \leq 2 \Phi^* + 0.08 < 0.080002 \implies D(S) > 1 - 0.080002 = 0.919998
\]

\[
0.46 \mu^* \leq \mu(S) \leq 0.54 \mu^* \implies 0.229 < \mu(S) \leq 0.27
\]

and thus

\[
f(\mu(S), D(S)) = 2 \mu(S) \left( D(S) - \mu(S) \right) > 2 \times 0.229 \times (0.919998 - 0.27) > 10^{-6}
\]

**Case II(b) |** \( S^* \cap V'' < \frac{|S^*|}{2} \).

Since \( |T_i| \leq \left| V \setminus \bigcup_{j=1}^{i-1} T_j \right| \left(1 - 10^{-3}\right) < n^{1 - 10^{-3}} \) for any \( i \) and \( |S^*| \geq \left(0.5 - \frac{10^{-6}}{2}\right) n \), there exists an index \( i \) such that \( \frac{|S^*|}{2} - n^{1 - 10^{-3}} < \left| \bigcup_{j=1}^{i} T_j \right| < \frac{|S^*|}{2} + n^{1 - 10^{-3}} \). Notice that the graph induced by the set of nodes \( S = \bigcup_{j=1}^{i} T_j \) satisfy \( \Phi(S) \leq 0.01 \) and, as a consequence of the previous observation, \( 0.24 < \mu(S) < 0.51 \). Thus,

\[
f(\mu(S), D(S)) = 2 \mu(S) \left( D(S) - \mu(S) \right) > 2 \times 0.24 \times (0.99 - 0.51) > 10^{-6}
\]

An interesting open question is whether it is possible to prove the converse of Remark 39, i.e., can an appropriate sub-exponential approximation algorithm for modularity maximization be used to design a sub-exponential algorithm for small-set expansions?

### 6.2 When is Goemans-Williamson rounding good?

In this section, we analyze the Goemans-Williamson rounding applied to the solution vectors of the modularity SDP. The goal is to come up with a multiplicative
guarantee for approximating $Q(G)$. Because of Lemma 3, it suffices to approximate bimodularity.

Recall that $M$ stood for the modularity matrix of a graph. Let $B = \frac{M}{2}$ be the matrix given by

$$b_{ij} = \frac{1}{4m} \left( a_{ij} - \frac{d_id_j}{2m} \right)$$

Note again that all rows and columns of $B$ sum up to 0. Thus, $Q_2(G)$ is given by the quadratic integer program

$$\text{Maximize} \quad x^T B x \quad \text{subject to} \quad x_i \in \{-1, 1\} \quad \forall i \quad (6.1)$$

which has the following SDP relaxation

$$\text{Maximize} \quad \sum_{ij} b_{ij} u_i \cdot u_j \quad \text{subject to} \quad \|u_i\| = 1 \quad \forall i \quad (6.2)$$

Let us use the Goemans-Williamson rounding for this SDP. According to Agarwal and Kempe [2008], the GW rounding method provides experimentally better results than the Charikar and Wirth [2004] rounding method. We will use the analysis from Section 4.1 in the paper by Alon and Naor [2006]. The following lemma is useful, and we reproduce its proof from Alon and Naor [2006] below for completeness.

**Lemma 40.** Let $r = (r_1, \ldots, r_n)$ be a random Gaussian vector where each $r_i$ is an independent normal Gaussian random variable. Then for any two unit vectors $u, v \in \mathbb{R}^n$,

$$\frac{\pi}{2} \mathbb{E}[\text{sign}(u \cdot r) \text{sign}(v \cdot r)] =$$

$$u \cdot v + \mathbb{E} \left[ \left( u \cdot r - \sqrt{\frac{\pi}{2}} \text{sign}(u \cdot r) \right) \left( v \cdot r - \sqrt{\frac{\pi}{2}} \text{sign}(v \cdot r) \right) \right].$$
Proof. By linearity of expectation, the R.H.S. is

\[ u \cdot v + \mathbb{E}[(u \cdot r)(v \cdot r)] + \frac{\pi}{2} \mathbb{E}[\text{sign}(u \cdot r)\text{sign}(v \cdot r)] \]
\[ - \sqrt{\frac{\pi}{2}} \mathbb{E}[(u \cdot r)\text{sign}(v \cdot r)] - \sqrt{\frac{\pi}{2}} \mathbb{E}[(v \cdot r)\text{sign}(u \cdot r)]. \] (6.3)

Since the \( r_i \)'s are independent, \( \mathbb{E}[(u \cdot r)(v \cdot r)] = u \cdot v \). Using the fact that the distribution of \( r \) is spherically symmetric, we may assume that \( v = (1,0,\ldots,0) \) and \( u = (a,b,0,\ldots,0) \). Hence, \( u \cdot v = a \) and

\[ \mathbb{E}[(u \cdot r)\text{sign}(v \cdot r)] = \mathbb{E}[(ar_1 + br_2)\text{sign}(r_1)] \]
\[ = a \mathbb{E}[r_1\text{sign}(r_1)] + b \mathbb{E}[r_2] \mathbb{E}[\text{sign}(r_1)] \]
\[ = a \mathbb{E}[r_1\text{sign}(r_1)] \]
\[ = 2a \int_0^\infty \frac{1}{\sqrt{2\pi}} xe^{-x^2/2} dx = \sqrt{\frac{2}{\pi}} a = \sqrt{\frac{2}{\pi}} u \cdot v. \]

Substituting the above values in (6.3) gives us the lemma. \( \square \)

Let \( Q_{VP}(G) \) denote the value of SDP (6.2) and let \( u_1,\ldots,u_n \) denote the optimal solution vectors. Then with the aid of Lemma 40, we can apply GW rounding to get

\[ \frac{\pi}{2} \sum_{i,j} b_{ij} \mathbb{E}[\text{sign}(u_i \cdot r)\text{sign}(u_j \cdot r)] = Q_{VP}(G) \] (6.4)
\[ + \sum_{i,j} b_{ij} \mathbb{E} \left[ \left( u_i \cdot r - \sqrt{\frac{\pi}{2}} \text{sign}(u_i \cdot r) \right) \left( u_j \cdot r - \sqrt{\frac{\pi}{2}} \text{sign}(u_j \cdot r) \right) \right]. \]

Note that each term of the form

\[ \mathbb{E} \left[ \left( u_i \cdot r - \sqrt{\frac{\pi}{2}} \text{sign}(u_i \cdot r) \right) \left( u_j \cdot r - \sqrt{\frac{\pi}{2}} \text{sign}(u_j \cdot r) \right) \right] \]

is the inner product of two vectors, whose norm is \( \sqrt{(\pi/2) - 1} \) (to see this, substitute \( u = v \) in Lemma 40). Thus we get

\[ \frac{\pi}{2} \sum_{i,j} b_{ij} \mathbb{E}[\text{sign}(u_i \cdot r)\text{sign}(u_j \cdot r)] \geq Q_{VP}(G) - \left( \frac{\pi}{2} - 1 \right) \max_{\|v\|=1} \left| \sum_{i,j} b_{ij} v_i \cdot v_j \right| \] (6.5)
Let
\[ \beta = \max_{\|u_i\| = 1} \sum_{i,j} b_{ij} u_i \cdot u_j \]
\[ \max_{\|u_i\| = 1} \left| \sum_{i,j} b_{ij} u_i \cdot u_j \right| \]

Then Equation (6.5) becomes
\[ \sum_{i,j} b_{ij} \mathbb{E}[\text{sign}(u_i \cdot r) \text{sign}(u_j \cdot r)] \geq \left( \frac{2}{\pi} - \frac{\pi - 2}{\pi \beta} \right) Q_{VP}(G). \quad (6.6) \]

The bigger \( \beta \) is, the better approximation we get. Trivially, \( \beta \geq 0 \). We conjecture that for many graphs that have large modularity, \( \beta > 1 \).

**Remark 41.** The paper Alon and Naor [2006] deals with analyzing the quantity
\[ \max_{\|u_i\| = \|v_j\| = 1} \sum_{i,j} b_{ij} u_i \cdot v_j \]
and they use the same analysis as above. The main difference is that in their case, \( \beta \) (whose definition is appropriately changed) is exactly 1. This is because the most positive value of \( \sum_{i,j} b_{ij} u_i \cdot v_j \) is equal to the most negative value of the same sum. To see this, note that we can change the set \( \{v_j\} \) to its antipodes and get the same value of the sum, albeit negated.

### 6.3 Density of Graphs and Modularity

It is well known that the modularity of the complete graph is zero. Below we show that it remains zero when you delete a matching from the complete graph.

**Lemma 42.** Let \( G \) be an \((n - 2)\)-regular graph. Then \( Q(G) = 0 \).

**Proof.** It suffices to prove that \( Q_2(G) \leq 0 \). Let \( \mathcal{C} = \{C, \overline{C}\} \) be any cut in the graph with \(|C| = k\) being the smaller of the two sides. The subgraph induced by
$C$ can be a clique. Therefore,

$$Q(C) = i(C) - \mu(C)^2 \leq \frac{k(k - 1)}{n(n - 2)} - \frac{k^2}{n^2} = \frac{k}{n^2(n - 2)}(2k - n) \leq 0,$$

since we assumed that $k \leq \frac{n}{2}$. Using Lemma 5, $Q(C) \leq 0$. \hfill \Box

It is an open problem to identify all classes of dense graphs that exhibit zero modularity.

### 6.4 Relation to sparsest cut

Finally, we will see a relation between sparsest cut and modularity. Suppose we have a $d$-regular graph with $n$ vertices. Recall that the modularity of a cut $\{C, \overline{C}\}$ is given by

$$Q(C, \overline{C}) = 2\mu(C)\mu(\overline{C}) - e(C, \overline{C}).$$

Let $SC(G)$ denote the value of the sparsest cut for a graph $G$, i.e.

$$SC(G) = \min_{C \subseteq V} \frac{\text{edges}(C, \overline{C})}{|C||\overline{C}|}.$$

In terms of the modularity parameters, $SC(G)$ can be rewritten as

$$SC(G) = \frac{d}{n} \min_{C \subseteq V} \frac{e(C, \overline{C})}{2\mu(C)\mu(\overline{C})}.$$

These definitions now immediately give us the following lemma.

**Lemma 43.** Given an $n$-vertex, $d$-regular graph $G$, $SC(G) < \frac{d}{n}$ if and only if $Q(G) > 0$.

It is an open problem to characterize positive values of bimodularity in terms of sparsest cut, even approximately.
Bibliography


Vita

DEVENDRA J. DESAI

2013  Ph. D., Department of Computer Science, Rutgers University.

2005  M. Tech., Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India.

2003  B. E., Department of Computer Engineering, University of Pune, India.

1999  Graduated from Fergusson College, India.

2013–13  Teaching Assistant, Department of Computer Science, Rutgers University.

2011–13  Graduate Assistant, Department of Computer Science, Rutgers University.

2005–11  Teaching Assistant, Department of Computer Science, Rutgers University.

2004–05  Teaching Assistant, Department of Computer Science and Engineering, IIT Kharagpur, India.