Algorithms for Minimum Graph Bisection and their Performance on Specific Graph Classes

Martin R. Schuster

From the Institute of Theoretical Computer Science
of the Universität zu Lübeck
Director: Prof. Dr. math. Rüdiger Reischuk

Algorithms for Minimum Graph Bisection and their Performance on Specific Graph Classes

Dissertation
for the Fulfillment of
Requirements for the
Doctoral Degree
of the Universität zu Lübeck

from the Department of Computer Science

Submitted by

Martin R. Schuster
from Leipzig

Lübeck 2018

First referee	Prof. Dr. Maciej Liśkiewicz
Second referee	Prof. Dr. Klaus Jansen
Date of oral examination	07. 06. 2018
Approved for printing	12. 06. 2018

Abstract

The Minimum Graph Bisection Problem is a well-studied NP-hard problem. It is defined to partition the vertices of an undirected graph into two equal sized sets, such that the number of edges between the sets is minimized. In this thesis we focus on polynomial-time heuristics that determine provably minimum graph bisections or output “fail” when the optimality cannot be certified. Boppana’s spectral based heuristic is one of the most prominent methods of this type. It is claimed to work well on the random *planted bisection model* – the standard class of graphs for the analysis of minimum bisection and related problems – but a complete proof was missing so far. We provide one now.

Since the behavior of Boppana’s algorithm on the *semirandom model* by Blum and Spencer remained unknown, Feige and Kilian proposed a new semidefinite programming (SDP) based approach and proved that it works on this model. The relationship to Boppana’s algorithm, however, was still left as an open problem. We solve this by proving that both algorithms obtain the same results. As a consequence, we get that Boppana’s algorithm achieves the optimal threshold for exact cluster recovery in the *stochastic block model.* On the other hand, we prove some limitations of Boppana’s approach: we show that, if the density difference on the parameters of the planted bisection model is too small, then the algorithm fails with high probability in the model.

The inherent weakness of Boppana’s approach is the huge computational complexity of the required high-dimensional convex optimization. We propose a simple modification of Boppana’s algorithm which avoids the extensive convex search. This leads to a substantial speedup: while Boppana’s algorithm works only for graphs with up to 150 vertices in a reasonable amount of time, our heuristic bisects graphs of up to millions of vertices in few seconds. Our heuristic also outperforms the SDP approach from Feige and Kilian, which reaches its limit at 2000 vertices.

In order to obtain graphs with largest bisection width and thus largest pathwidth, we provide an explicit construction for simple cubic Ramanujan graphs, which can be easily generated in linear time. Our construction allows, for the first time, to analyze experimentally a dense family of explicitly constructed cubic Ramanujan graphs. We estimate the asymptotic bisection width at approximately 0.12 times the number of vertices, leading to the conjecture that the lower bound of Kostochka and Melnikov on the worst case bisection width over *all* cubic graphs of n vertices is fulfilled already by the cubic Ramanujan graphs.

Zusammenfassung

Das Minimum-Graph-Bisektionsproblem ist ein gut untersuchtes NP-hartes Problem. Es ist definiert als Partitionierung der Knoten eines ungerichteten Graphen in zwei gleichgroße Teilmengen, sodass die Anzahl der Kanten zwischen den beiden Mengen minimiert wird. In dieser Arbeit konzentrieren wir uns auf Polynomialzeit-Heuristiken, die beweisbar minimale Bisektionen finden oder "fail" ausgeben, wenn die Optimalität nicht nachgewiesen werden kann. Boppanas Spektralansatz ist eine der bekanntesten Heuristiken dieser Art. Sie funktioniert gut für das *Planted-Bisection-Modell* – einem Standardmodell zur Analyse des Minimum-Bisektionsproblems und anderen relevanten Problemen – wobei bisher ein vollständiger Beweis fehlte. Wir liefern diesen.

Da das Verhalten von Boppanas Algorithmus auf dem *Semi-Zufalls-Modell* von Blum und Spencer offen war, entwickelten Feige und Kilian einen neuen Ansatz basierend auf semidefiniter Programmierung (SDP). Das Verhältnis zu Boppanas Algorithmus blieb ein offenes Problem. Wir lösen es, indem wir zeigen, dass beide Algorithmen die gleichen Ergebnisse liefern. Als direkte Konsequenz daraus können wir ableiten, dass Boppanas Algorithmus die optimale Grenze für das Exact-Cluster-Recovery-Problem im *Stochastic-Block-Modell* erreicht. Auf der anderen Seite zeigen wir einige Einschränkungen von Boppanas Ansatz: Wenn die Dichteunterschiede der Parameter des Planted-Bisection-Modells zu klein sind, dann gibt der Algorithmus mit hoher Wahrscheinlichkeit "fail" aus.

Eine grundsätzliche Schwäche von Boppanas Ansatz ist der hohe Berechnungsaufwand in der erforderlichen hochdimensionalen konvexen Optimierung. Wir schlagen eine einfache Anpassung von Boppanas Algorithmus vor, die die aufwendige konvexe Optimierung überflüssig macht. Dies führt zu einer erheblichen Beschleunigung: Während Boppanas Algorithmus nur für Graphen bis zu 150 Knoten in angemessener Zeit läuft, findet unsere neue Heuristik die optimale Bisektion für Graphen mit mehreren Millionen Knoten. Unsere Heuristik überzeugt auch im Vergleich mit dem SDP-Ansatz von Feige und Kilian, der seine Grenze bei 2000 Knoten erreicht.

Um Graphen mit größtmöglicher Bisektionsweite und folglich größtmöglicher Pfadweite zu erhalten, entwickeln wir eine explizite Konstruktion von einfachen kubischen Ramanujan-Graphen, die sich in Linearzeit generieren lassen. Unsere Konstruktion lässt es erstmalig zu, experimentell eine dichte Klasse von explizit konstruierten kubischen Ramanujan-Graphen zu analysieren. Wir schätzen die asymptotische Bisektionsweite mit ungefähr 0.12-facher Anzahl der Knoten ab und stellen die Vermutung auf, dass die untere Schranke von Kostochka und Melnikov für die Worst-Case-Bisektionsweite *aller* kubischer Graphen mit n Knoten bereits für kubische Ramanujan-Graphen gilt.

Contents

1 Introduction

Graph partitioning problems are notorious for their intractability, both in theory and in practice. This thesis studies a fundamental variant of this problem – the minimum graph bisection. It consists in partitioning the set of vertices $V = \{1, \ldots, n\}$ (n even) of a given undirected graph G into two equal sized sets, such that the number of cut edges, i. e. edges with endpoints in different bisection sides, is minimized. The bisection width of G, denoted by $\mathrm{bw}(G)$, is then the minimum number of cut edges in a bisection of G.

The research on graph bisection is motivated by computational challenges arising in many diverse applications such as VLSI design [50, 8], image processing and computer vision [75, 49], divide-and-conquer graph algorithms [51], scientific simulations [67], and many others. Since the Minimum Graph Bisection Problem is NP-hard [38], in theory its complexity has been a subject of intensive studies from an approximability point of view [66, 6, 33, 32] as well as from the parameterized [57, 73, 27] and average-case [15] perspectives. The research shows that the problem is even hard to approximate [32].

Promisingly, the problem is fixed parameter tractable – in their breakthrough paper [27] Cygan et al. provide an algorithm finding a minimum bisection in time $2^{\mathcal{O}(\mathrm{bw}^3(G))} n^3 \log^3 n$. However, this implies a polynomial-time exact algorithm only for graphs of bisection width $\mathcal{O}(\sqrt[3]{\log n})$.

In this thesis we study polynomial-time heuristics that, for any input graph, either determine an optimum bisection or output "fail" when the optimality of the solution cannot be certified by the algorithm. We will focus on Boppana's spectral-based algorithm [12] which belongs to one of the most prominent methods of this type, and we focus on particular random graph models used in the context of bisection problems. Boppana's algorithm works well in the random *planted bisection model* – the standard class of graphs for the analysis of minimum bisection and related problems. In the planted bisection model, denoted by $\mathcal{G}_n(p, q)$, a vertex set with an even number of vertices n is partitioned randomly into two equal sized sets V_1 and V_2, called the *planted bisection*. Then, for every pair of vertices do independently: if both vertices belong to the same part of the bisection, include an edge between them with probability p; if the two vertices belong to different parts, connect the vertices by an edge with probability q. Assuming the density difference

$$p - q \geq c\sqrt{p \log n}/\sqrt{n} \quad \text{for a certain constant } c > 0, \tag{1.1}$$

Boppana's algorithm finds and certifies the minimum bisection of $\mathcal{G}_n(p, q)$ with probability tending to 1 as the number of vertices $n \to \infty$. We will refer to this by w.h.p. (with high probability) and say the algorithm works well on this model. Boppana's algorithm also finds and certifies w.h.p. the minimum bisection of graphs from the *regular graph model* $\mathcal{R}_n(r, b)$, assuming that[1]

$$r \geq 6 \quad \text{and} \quad b \leq o(n^{1-1/\lfloor (r/2+1)/2 \rfloor}).$$

In this model, a graph is chosen at random from the set of all regular graphs with degree r and bisection width b.

In 2001, Feige and Kilian [31] posed the question if Boppana's algorithm works well in the semirandom model by Blum and Spencer [9]. In the semirandom model, initially a graph G is chosen at random according to the model $\mathcal{G}_n(p, q)$. Then a monotone adversary is allowed to remove any edge crossing a minimum bisection and is allowed to add any edge not crossing the bisection.

1.1 Main Contributions of this Thesis

In this thesis we answer the question from Feige and Kilian affirmatively: Boppana's algorithm works well in the semirandom model. Moreover, we introduce and analyze a natural generalization of the semirandom model. Instead of $\mathcal{G}_n(p, q)$, we start with an arbitrary initial graph model $\mathcal{G}_n$, and then the monotone adversary applies changes as in the semirandom model. We denote such a model by $\mathcal{A}(\mathcal{G}_n)$. One of our main positive results is that, if Boppana's algorithm outputs the minimum-size bisection for graphs in $\mathcal{G}_n$ w.h.p., then the algorithm finds a minimum bisection w.h.p. for the adversarial graph model $\mathcal{A}(\mathcal{G}_n)$, too. As a corollary, we get that Boppana's algorithm works well not only in the semirandom model $\mathcal{A}(\mathcal{G}_n(p, q))$ but also in the semirandom regular model $\mathcal{A}(\mathcal{R}_n(r, b))$. To the best of the author's knowledge, Boppana's algorithm is the only method known so far that finds (w.h.p.) provably optimum bisections on all of these mentioned random graph classes.

On the other hand, we prove some limitations of Boppana's approach. We show that the density difference in Eq. (1.1) is tight: we prove that $p - q \leq o(\sqrt{p \cdot \log n}/\sqrt{n})$ implies that the algorithm fails on $\mathcal{G}_n(p, q)$ w.h.p.

[1] In [12], one finds $r \geq 3$ and $b \leq o(n^{1-1/\lfloor (r+1)/2 \rfloor})$. Reviewing the reference [13], this seems to hold for $2r$-regular graphs.

Since the behavior of Boppana's algorithm on semirandom graphs remained unknown, Feige and Kilian proposed a new semidefinite programming (SDP) based approach and proved that it works on this model. The relationship between the performance of the SDP based algorithm and Boppana's approach was left as an open problem. In this thesis we solve the problem in a complete way by proving that the bisection algorithm of Feige and Kilian provides exactly the same results as Boppana's algorithm.

A problem related to Minimum Graph Bisection is the Cluster Recovery Problem, which asks to find the planted bisection from which the graph was generated. Hajek et al. [42] showed that an SDP for cluster recovery equivalent to the one of Feige and Kilian achieves the optimal threshold on the parameters where exact cluster recovery is possible. We use our new results and show that also Boppana's algorithm achieves the optimal threshold for cluster recovery.

The inherent weakness of Boppana's approach is the huge computational complexity of the required high-dimensional convex optimization. This severely limits the practical applicability of the algorithm. Therefore, we propose a simple modification of Boppana's algorithm which avoids the expensive convex search. This leads to a new heuristic with substantial speedup: while Boppana's algorithm works only for graphs with up to 150 vertices in a reasonable amount of time our heuristic bisects graphs of up to millions of vertices in few seconds. We provide experimental results which show that the heuristic works well on the graph models for which Boppana's algorithm theoretically guarantees provable solutions. The experiments also confirm that our heuristic achieves the optimum threshold for exact cluster recovery in the stochastic block model. Though, we leave as an open problem if this heuristic works provably well for the planted bisection model.

Besides the Minimum Bisection Problem, i. e. finding the minimum bisection in a given graph, we consider the task of constructing graphs with largest possibly bisection width. We investigate explicit constructions for cubic Ramanujan graphs as deterministic candidates to have asymptotically the largest bisection width and thus largest pathwidth. In 1992, Chiu [19] gave a construction for such graphs of sizes roughly q^3 for prime numbers q satisfying some additional conditions. Accordingly, only four such graphs exist with the number of nodes up to 10^6. In 2016, Cohen [20] obtained the first known polynomial time algorithm to find bipartite Ramanujan multigraphs of any degree and any size. However, so far it is not clear whether the algorithm is practically implementable. Based on the method of Chiu, we provide an explicit construction for simple cubic Ramanujan graphs of sizes $q + 1$ for primes q (satisfying the same conditions as above) such that the proposed graphs can be easily generated in linear time. Due to these properties, our construction allows, for the first time, to analyze experimentally a dense family of explicitly constructed cubic Ramanujan graphs. We estimate the asymptotic bisection width at approximately 0.12 times the number of vertices, leading to the conjecture that the lower bound of Kostochka and Melnikov [48] on the worst case bisection width over *all* cubic graphs of n nodes is fulfilled already by the cubic Ramanujan graphs. Our graphs can also serve as a benchmark with cubic graphs of (provably) large bisection width, and thus of large pathwidth, which seems to be useful for experimental analysis of parameterized algorithms.

1.2 Related Work

Several other algorithms have been proven to work on the planted bisection model as well. Condon and Karp [26] developed a linear time algorithm for the more general l-partitioning problem. Their algorithm finds the optimal partition with probability $1 - \exp(-n^{\Theta(\varepsilon)})$ in the planted bisection model with parameters satisfying $p - q = \Omega(1/n^{1/2-\varepsilon})$. Carson and Impagliazzo [16] show that a hill-climbing algorithm is able to find the planted bisection w.h.p. for parameters $p - q = \Omega((\log^3 n)/n^{1/4})$. Dyer and Frieze [30] provide a min-cut via degrees heuristic that, assuming $n(p - q) = \Omega(n)$, finds and certifies the minimum bisection w.h.p. Note that the density difference in Eq. (1.1) assumed by Boppana still outperforms the above ones. Moreover, a disadvantage of the methods against Boppana's algorithm, except for the last one, is that they do not certify the optimality of the solutions. In [58], McSherry describes a spectral based heuristic that, applied to $\mathcal{G}(p, q)$, finds a minimum bisection w.h.p. if p and q satisfy assumption in Eq. (1.1), but it does not certify the optimality. Importantly, the algorithms above, similarly as Boppana's method, solve the cluster recovery problem for the stochastic block model with two communities.

In [24], Coja-Oghlan developed a new spectral-based algorithm that, on the planted bisection model $\mathcal{G}_n(p, q)$, enables for a wider range of parameters than in Eq. (1.1), certifying the optimality of its solutions. The algorithm [24] assumes that $p - q \geq \Omega(\sqrt{p \log(np)}/\sqrt{n})$. If the parameters p and q describe non-sparse graphs, this condition is essentially the same as Boppana's assumption. For sparse graphs, however, Coja-Oghlan's constraint allows a larger subclass. For example, the algorithm works in $\mathcal{G}_n(p, q)$ for $q = \mathcal{O}(1)/n$ and $p = \sqrt{\log n}/n$. From our results we know that Boppana's algorithm fails w.h.p. for such graphs. Interestingly, the condition on the density difference by Coja-Oghlan allows graphs for which the minimum bisection width is w.h.p. strictly smaller than the width of the planted bisection. However, a drawback of Coja-Oghlan's algorithm is that, to work well in the planted bisection model with *unknown* parameters p and q, the algorithm has to learn the parameters since it is based on the knowledge of values p and q. Also the performance of the algorithm on other families, like e. g. semirandom graphs and regular random graphs, is unknown. Recent research by Coja-Oghlan et al. [25] contributes to a better understanding of the planted bisection model and average case behavior of a minimum bisection.

1.3 Organization of this Thesis

This thesis is organized as follows: we introduce Boppana's graph bisection algorithm [12] and the SDP algorithm by Feige and Kilian [31] in Chapter 2. There we also present the random graph models used in the analysis of the bisection algorithms. Since the proof on the performance of Boppana's algorithm is incomplete in his original work, we provide a completed and self-contained proof in Chapter 3. The reconstruction of an optimum bisection in Boppana's algorithm contains an issue which will be discussed in Chapter 4. We develop modifications of the algorithm to fix this problem. In Chapter 5 we explore the abilities of Boppana's approach and show the limitations of the algorithm. Then, in Chapter 6, we formulate Boppana's algorithm as an SDP and show that his approach is the same as the SDP provided by Feige and Kilian. Consequently, Boppana's algorithm proves to be optimal for certain regimes of the planted bisection model on the Cluster Recovery Problem.

A huge drawback of Boppana's algorithm and the SDP of Feige and Kilian is the computational complexity. In Chapter 7 we develop a fast heuristic, which allows to solve instances with up to 10^6 vertices while it is still able to give a certificate of optimality. We conduct a series of experiments to show the power of the new heuristic. Our experiments indicate that even this heuristic achieves the optimal threshold on the Cluster Recovery Problem (see Section 6.4 for a definition of this problem).

In Chapter 8 we move the focus on constructing graphs with largest possible bisection width. We develop new explicit constructions of dense classes of Ramanujan graphs and propose a new algorithm for finding a good estimate of the bisection width on these graphs. We relate the results to the known bounds on worst case bisection width. Finally, we have a discussion on the results of this thesis and show up open problems.

2 Preliminaries: Graph Models and Bisection Algorithms

In this chapter we present several random graph models commonly used to analyze the performance of bisection heuristics, among them the already mentioned planted bisection model and the semirandom model.

We further present Boppana's graph bisection algorithm from [12], which is known to work well on the planted bisection model and the random regular model. Along we provide some known properties of Boppana's algorithm.

Since the behavior of Boppana's algorithm on the semirandom model remained unknown so far, Feige and Kilian proposed in [31] a new semidefinite programming (SDP) based approach which works for semirandom graphs. In this chapter we also present their algorithm.

Let use note here that, although Boppana's algorithm and the algorithm from Feige and Kilian are different, they output exactly the same results, as we will show in this thesis.

2.1 Random Graph Models

In order to analyze heuristics for graph bisection, we investigate two well-studied graph models: the *planted bisection model* and its extension the *semirandom model*, which are widely used to analyze and benchmark graph partitioning algorithms. We refer to [15, 30, 12, 9, 26, 31, 16, 58, 11, 24, 53] to cite some of the relevant works. We also include the *stochastic block model* in our discussions, which is closely related to the planted bisection model. Moreover, we consider the *regular graph model* introduced by Bui et al. [15], and a new extension of the semirandom model.

In the *planted bisection model*, denoted as $\mathcal{G}_n(p, q)$ with parameters $1 > p = p(n) \geq q(n) = q > 0$ and n even, the vertex set $V = \{1, \dots, n\}$ is partitioned randomly into two equal sized sets V_1 and V_2, called the *planted bisection*. Then for every pair of vertices do independently: if both vertices belong to the same part of the bisection (either both belong to V_1 or both belong to V_2), then include an edge between them with probability p; if the two vertices belong to different parts, then connect the vertices by an edge with probability q.

The planted bisection model was first proposed in the sociology literature [44] under the name *stochastic block model* to study community detection problems in random graphs. In this setting, the planted bisection V_1, V_2 (as described above) models latent communities in a network and the goal here is to recover the communities from the observed graph. In the general case, the model allows some errors by recovering, allows multiple communities, and also allows that $p(n) < q(n)$. The community detection problem on the stochastic block model has been subject of a considerable amount of research in physics, statistics and computer science (see e. g. [1, 61] for current surveys). In particular, an intensive study has been carried out on providing lower bounds on $|p - q|$ to ensure recoverability of the planted bisection.

In the *semirandom model*, as used by Feige and Kilian [31], initially a graph G is chosen at random according to the model $\mathcal{G}_n(p, q)$. Then a monotone adversary is allowed to modify G by applying an arbitrary sequence of the following monotone transformations: (1) The adversary may remove from the graph any edge crossing a minimum bisection; (2) The adversary may add to the graph any edge not crossing the bisection.

We consider also the *regular graph model* introduced by Bui et al. [15] and denote it as $\mathcal{R}_n(r, b)$, with $r = r(n) < n$ and $b = b(n) \leq (n/2)^2$. In the model, we start with a random partition V_1 and V_2 of equal sizes, and add firstly b cut edges and then further edges within the sets, and this under the constraint that at the end every vertex has degree r.

For a (semi)random model we say that some property is satisfied w.h.p. if the probability that the property holds tends to 1, as the number of vertices $n \to \infty$.

2.2 Definitions, Notations, and Basic Properties

Before we are able to formulate Boppana's algorithm, we need to give some definitions and notations from linear algebra. Unless stated otherwise, we consider simple undirected graphs G with an even number of vertices n and vertex set $V = \{1, \dots, n\}$. For a pair of vertices u and v from G, the graph $G \oplus \{u, v\}$ is obtained from G by "toggling" the edge, i. e. we perform the following modification: if there is an edge between u and v in G, the edge is removed. Otherwise, we add the edge. A graph G can be represented by an adjacency matrix $A \in \{0, 1\}^{n \times n}$. By $I^{(n)}$ we denote the n-dimensional identity matrix and by J we denote an $n \times n$ matrix of ones. For a vector $x \in \mathbb{R}^n$ we denote $\|x\|^2 = x^T x$. When using the logarithm, by log we refer to the natural logarithm and by $\log_b$ to logarithm with base b.

A vector x is called a *bisection vector* if $x \in \{+1, -1\}^n$ and $\sum_i x_i = 0$. Such x determines a bisection of G of the cut width denoted as

$$\mathrm{cw}(x) = \sum_{\{i,j\} \in E} \frac{1 - x_i x_j}{2}. \tag{2.1}$$

In the context of the planted bisection model $\mathcal{G}_n(p, q)$ with planted bisection V_1 and V_2, the *planted bisection vector* y is a bisection vector with entries $y_u = +1$ for $u \in V_1$ and $y_u = -1$ for $u \in V_2$. A vector x is called a *relaxed bisection vector* if $x \in \mathbb{R}^n$ and $\sum_i x_i = 0$, i. e. we drop the $+1/-1$ constraint. We define the subspace $S \subset \mathbb{R}^n$ of all vectors $x \in \mathbb{R}^n$ with $\sum_i x_i = 0$, i. e. this subspace contains all relaxed bisection vectors.

The subspace S plays a central role in Boppana's algorithm. A projection from $\mathbb{R}^n$ to S can be realized by the projection matrix $P = I - \frac{1}{n}J$. The matrix P maps a vector $x \in \mathbb{R}^n$ to the projection Px of vector x into the subspace S. This projection simply subtracts the mean of the components of x from each component. However, using this mapping from an n-dimensional space to an $(n-1)$-dimensional subspace can be troublesome in computations.

Therefore, to be able to restrict computations to subspace S, we define a matrix Q that provides a bijective mapping $x \mapsto Qx$ from $\mathbb{R}^{n-1}$ to S: Let Q be an $n \times (n-1)$ matrix whose column vectors are an orthonormal basis of the subspace S, i. e. $\mathrm{span}(\{Q_{\cdot,1}, \ldots, Q_{\cdot,n-1}\}) = S$ and $Q^T Q = I^{(n-1)}$. Note that the inverse mapping from S to $\mathbb{R}^{n-1}$ is done by $x' \mapsto Q^T x'$, since $Q^T Q = I^{(n-1)}$. We point out that $QQ^T = P$ holds.

Now, when we need to perform computations in S, we can use $\mathbb{R}^{n-1}$ and the bijective mapping via Q and Q^T. When we use Q in our algorithms, we e. g. use the following explicit matrix $Q \in \mathbb{R}^{n \times (n-1)}$ which fulfills the properties described above:

$$Q_{ij} = \frac{1}{n + \sqrt{n}} \cdot \begin{cases} -1 + n + \sqrt{n} & \text{if } i = j \\ -1 - \sqrt{n} & \text{if } i = n \\ -1 & \text{otherwise.} \end{cases}$$

For $B \in \mathbb{R}^{n \times n}$ and $d \in \mathbb{R}^n$ we denote the sum of B's elements as $\mathrm{sum}(B) = \sum_{ij} B_{ij}$ and by $\mathrm{diag}(d)$ we denote the $n \times n$ diagonal matrix D with the entries of the vector d on the main diagonal, i. e. $D_{ii} = d_i$. Often we will have an adjacency matrix $A \in \{0, 1\}^{n \times n}$, a vector $d \in \mathbb{R}^n$ and then set $B := A + D = A + \mathrm{diag}(d)$.

Now assume $B \in \mathbb{R}^{n \times n}$ is symmetric. Then the matrix has n real eigenvalues $\lambda_1(B) \geq \ldots \geq \lambda_n(B)$. We denote by $\lambda(B) = \lambda_1(B)$ its largest eigenvalue. The largest eigenvalue can be computed using the Rayleigh quotient as follows:

$$\lambda(B) = \max_{x \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{x^T B x}{x^T x}. \tag{2.2}$$

Vectors x that attain the maximum are exactly the eigenvectors corresponding to the largest eigenvalue. In Boppana's algorithm, we will need to compute

$$\max_{x \in S \setminus \{\mathbf{0}\}} \frac{x^T B x}{x^T x}. \tag{2.3}$$

For this purpose, let subscript Q of B for some matrix B be defined as $B_Q = Q^T B Q$. Note that here B_Q is an $(n-1) \times (n-1)$ matrix and when B is symmetric, B_Q is symmetric as well. We obtain

$$\begin{aligned}\max_{x \in S \setminus \{\mathbf{0}\}} \frac{x^T B x}{x^T x} &= \max_{x \in \mathbb{R}^{n-1} \setminus \{\mathbf{0}\}} \frac{(Qx)^T B (Qx)}{(Qx)^T (Qx)} \\ &= \max_{x \in \mathbb{R}^{n-1} \setminus \{\mathbf{0}\}} \frac{x^T Q^T B Q x}{x^T Q^T Q x} = \max_{x \in \mathbb{R}^{n-1} \setminus \{\mathbf{0}\}} \frac{x^T B_Q x}{x^T x} = \lambda(B_Q),\end{aligned} \tag{2.4}$$

i. e. Eq. (2.3) can be computed as $\lambda(B_Q)$.

There is another way to compute Eq. (2.3) which will be useful in our analysis of SDP formulations. Instead of the matrix Q we will use the (quadratic) projection matrix P. The matrix $B_P = P^T B P$ projects a vector $x \in \mathbb{R}^n$ to S, then applies B and projects the result again into S. Denote by $\mathbb{R}^n_{\neq c\mathbf{1}}$ the real space $\mathbb{R}^n$ without the subspace spanned by the identity vector $\mathbf{1}$, i. e. $\mathbb{R}^n_{\neq c\mathbf{1}} = \mathbb{R}^n \setminus \{c\mathbf{1} : c \in \mathbb{R}\}$. While we have seen that Eq. (2.3) is the same as $\lambda(B_Q)$, we observe that it is almost the same as $\lambda(B_P)$:

$$\begin{aligned}\max_{x \in S \setminus \{\mathbf{0}\}} \left(\frac{x^T B x}{x^T x}, 0 \right) &\overset{\text{Lemma 2.2}}{=} \max_{x \in \mathbb{R}^n_{\neq c\mathbf{1}}} \left(\frac{x^T B_P x}{x^T x}, 0 \right) \\ &\overset{B_P(c\mathbf{1}) = \mathbf{0}}{=} \max_{x \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{x^T B_P x}{x^T x} = \lambda(B_P).\end{aligned}$$

Note that a constant vector becomes a null vector when projected into S. Now, if we want to use $\lambda(B_P)$ to compute Eq. (2.3), we have to ensure that Eq. (2.3)≥ 0 holds. This will be discussed in detail when needed. We conclude:

Fact 2.1. *Let $B \in \mathbb{R}^{n \times n}$ be symmetric. Then* $\max(\lambda(B_Q), 0) = \lambda(B_P)$.

This Lemma we have already used above:

Lemma 2.2. *Let $B \in \mathbb{R}^{n \times n}$ be symmetric. Then*

$$\max_{x \in S \setminus \{\mathbf{0}\}} \frac{x^T B x}{\|x\|^2} = \max_{x \in \mathbb{R}^n_{\neq c\mathbf{1}}} \frac{x^T B_P x}{\|x\|^2}.$$

Proof. Since P projects a vector into S, we have $Px' \in S \setminus \{\mathbf{0}\}$ for all $x' \in \mathbb{R}^n_{\neq c\mathbf{1}}$. Furthermore, $Px' = x'$ for every $x' \in (S \setminus \{\mathbf{0}\}) \subset \mathbb{R}^n_{\neq c\mathbf{1}}$. Thus, we replace $x = Px'$:

$$\begin{aligned} \max_{x \in S \setminus \{\mathbf{0}\}} \frac{x^T B x}{\|x\|^2} &= \max_{x' \in \mathbb{R}^n_{\neq c\mathbf{1}}} \frac{(Px')^T B (Px')}{\|Px'\|^2} && \Big| \, x = Px' \\ &= \max_{x' \in \mathbb{R}^n_{\neq c\mathbf{1}}} \frac{x'^T (P^T B P) x'}{\|Px'\|^2}. \end{aligned} \tag{2.5}$$

In the next step we use the Pythagorean theorem, since the projected vector Px' (which has mean zero) and the remaining component $x' - Px'$ (which is a stretched 1-vector) are orthogonal:

$$\begin{aligned} &= \max_{x' \in \mathbb{R}^n_{\neq c\mathbf{1}}} \frac{x'^T (P^T B P) x'}{\|x'\|^2 - \|x' - Px'\|^2} && \Big| \, \|x'\|^2 = \|Px'\|^2 + \|x' - Px'\|^2 \\ &\geq \max_{x' \in \mathbb{R}^n_{\neq c\mathbf{1}}} \frac{x'^T (P^T B P) x'}{\|x'\|^2} && \Big| \, \text{Increasing (or equal) denominator} \\ &\geq \max_{x' \in S \setminus \{\mathbf{0}\}} \frac{x'^T (P^T B P) x'}{\|x'\|^2} && \Big| \, \text{Restrict the set of vectors} \\ &= \max_{x' \in S \setminus \{\mathbf{0}\}} \frac{x'^T B x'}{\|x'\|^2}. \end{aligned} \tag{2.6}$$

This is the same term as the left term of Eq. (2.5) we started with. Hence, the $\geq$ has to be $=$, and we can use Eq. (2.6) to compute Eq. (2.5). □

We will also use the following basic properties in this thesis.

Fact 2.3. *For $A, B \in \mathbb{R}^{n \times n}$, it holds $\lambda_1(A + B) \leq \lambda_1(A) + \lambda_1(B)$.*

Proof.

$$\lambda_1(A + B) = \max_{x \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{x^T (A + B) x}{\|x\|^2} = \max_{x \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{x^T A x + x^T B x}{\|x\|^2}$$

$$\begin{aligned}&\leq \max_{x\in\mathbb{R}^n\setminus\{\mathbf{0}\}} \frac{x^T A x}{\|x\|^2} + \max_{x\in\mathbb{R}^n\setminus\{\mathbf{0}\}} \frac{x^T B x}{\|x\|^2} \\ &= \lambda_1(A) + \lambda_1(B)\end{aligned}$$

□

For a matrix $A \in \mathbb{R}^{n\times n}$, the trace $\operatorname{tr}(A)$ is defined as

$$\operatorname{tr}(A) = \sum_{i=1}^{n} A_{ii}.$$

Fact 2.4. *For a symmetric matrix $A \in \mathbb{R}^{n\times n}$ and for $k \in \mathbb{N}$, it holds*

$$\operatorname{tr}(A^k) = \sum_{i=1}^{n} (\lambda_i(A))^k.$$

Proof. Since A is symmetric, we can use an orthogonal transformation to obtain a diagonal matrix. Then, the eigenvalues are its diagonal entries. The trace is invariant under orthogonal transformations, such that the fact follows directly from the definition of the trace. □

2.3 Boppana's Spectral-Based Algorithm BB

For a given graph $G = (V, E)$, with $V = \{1, \ldots, n\}$, Boppana defines a function f for all real vectors $x, d \in \mathbb{R}^n$ as

$$f(G, d, x) = \sum_{\{i,j\}\in E} \frac{1 - x_i x_j}{2} + \sum_{i\in V} d_i(x_i^2 - 1). \tag{2.7}$$

Based on f, the function g' is defined as follows

$$g'(G, d) = \min_{\|x\|^2=n,\, x\in S} f(G, d, x). \tag{2.8}$$

Note as an interesting fact that g' is invariant under shifting d, i. e. $g'(G, d + \beta(1, \ldots, 1)^T) = g'(G, d)$ for every $\beta \in \mathbb{R}$ (see Lemma 2.7).

For a bisection vector x (recall: $x \in \{+1, -1\}^n$ and $\sum_i x_i = 0$) the function f takes the value Eq. (2.7) regardless of d, as Eq. (2.7) is essentially the same as the cutwidth defined in Eq. (2.1). Minimization over all such x would give the minimum bisection width. Since g' uses a relaxed constraint, we get $g'(G, d) \leq \operatorname{bw}(G)$, where, recall, $\operatorname{bw}(G)$ denotes the bisection width of G. To improve the bound, Boppana tries to find some d which leads to a minimal decrease of the function value of g' compared to the bisection width:

$$h(G) = \max_{d \in \mathbb{R}^n} g'(G, d). \tag{2.9}$$

It is easy to see that for every graph G we have $h(G) \leq \mathrm{bw}(G)$.

In order to compute g' efficiently, Boppana expresses the function in spectral terms. Let G be an undirected graph with n vertices and adjacency matrix A. Let further $d \in \mathbb{R}^n$ be some vector and let $B = A + \mathrm{diag}(d)$, then we define

$$g(G, d) = \frac{\mathrm{sum}(B) - n\lambda(B_Q)}{4}. \tag{2.10}$$

Recall that we defined subscript Q of B as $B_Q = Q^T BQ$.

Lemma 2.5 (In [12] without proof)**.** *Let G be an undirected graph with n vertices and $d \in \mathbb{R}^n$. Then*

$$g(G, d) = g'(G, -d/4).$$

Proof. We start by reformulating the function f:

$$\begin{aligned}
f(G, -d/4, x) &= \sum_{\{i,j\} \in E} \frac{1 - x_i x_j}{2} + \sum_{i \in V} \frac{-d_i}{4}(x_i^2 - 1) && \Big| \text{Def. } f \\
&= \frac{2|E| - x^T A x}{4} - \frac{1}{4}\left(\sum_{i \in V} d_i x_i^2 - \sum_{i \in V} d_i\right) \\
&= \frac{\mathrm{sum}(A) - x^T A x}{4} - \frac{1}{4}\left(x^T D x - \mathrm{sum}(D)\right) && \Big| D = \mathrm{diag}(d) \\
&= \frac{\mathrm{sum}(A + D) - x^T (A + D) x}{4} \\
&= \frac{\mathrm{sum}(B) - x^T B x}{4}. && \Big| B = A + D \quad (2.11)
\end{aligned}$$

Next, we take the definition of g' and insert f:

$$\begin{aligned}
g'(G, -d/4) &= \min_{\|x\|^2 = n, \sum_i x_i = 0} f(G, -d/4, x) && \Big| \text{Def. } g' \\
&= \min_{\|x\|^2 = n, \sum_i x_i = 0} \frac{\mathrm{sum}(B) - x^T B x}{4} && \Big| \text{Insert Eq. (2.11)} \\
&= \frac{\mathrm{sum}(B)}{4} - \frac{1}{4} \max_{\|x\|^2 = n,\, x \in S} x^T B x
\end{aligned}$$

$$= \frac{\mathrm{sum}(B)}{4} - \frac{n}{4} \max_{\|x\|^2=n,\, x\in S} \frac{x^T B x}{\|x\|^2}$$
$$= \frac{\mathrm{sum}(B)}{4} - \frac{n}{4} \max_{x\in S\setminus\{\mathbf{0}\}} \frac{x^T B x}{\|x\|^2}$$
$$= \frac{\mathrm{sum}(B) - n\lambda(B_Q)}{4} \qquad \Big| \text{cf. Eq. (2.4)}$$
$$= g(G,d). \qquad \Big| \text{Compare Eq. (2.10)}$$

□

Since in the definition of h in Eq. (2.9) we maximize over all d, we can conclude that

$$\begin{aligned} h(G) &= \max_{d\in\mathbb{R}^n} g(G,d) \\ &= \max_{d\in\mathbb{R}^n} \frac{\mathrm{sum}(A+\mathrm{diag}(d)) - n\lambda((A+\mathrm{diag}(d))_Q)}{4}. \end{aligned} \tag{2.12}$$

Next, Boppana observes that g is concave and hence, the maximum in function $h(G)$ can be computed in polynomial time with arbitrary precision [40]. Since the proof is left out in the original paper, we provide it here:

Lemma 2.6 (In [12] without proof)**.** *Function $g(G,d)$ is concave in d, i. e.*

$$g(G, \frac{d_1+d_2}{2}) \geq \frac{g(G,d_1)+g(G,d_2)}{2}.$$

Proof. The key to the proof of this lemma is simply the following property of the largest eigenvalues of real symmetric matrices A and B: $\lambda(A+B) \leq \lambda(A)+\lambda(B)$ (see Fact 2.3). We start from the definition of g found in Eq. (2.10):

$$g(G, \frac{d_1+d_2}{2}) = \frac{\mathrm{sum}(B) - n\lambda(B_Q)}{4}. \qquad \Big| \text{here: } B = A + (D_1+D_2)/2$$

Now we have

$$\begin{aligned} \mathrm{sum}(B) &= \mathrm{sum}(A + \frac{1}{2}(D_1+D_2)) \\ &= \mathrm{sum}(\frac{1}{2}(B_1+B_2)) = \frac{\mathrm{sum}(B_1)+\mathrm{sum}(B_2)}{2} \qquad \Big| B_1 = A + D_1,\ B_2 = A + D_2 \end{aligned}$$

and

$$\begin{aligned}
\lambda(B_Q) &= \lambda(Q^T(A + \frac{1}{2}(D_1 + D_2))Q) \\
&= \lambda(\frac{1}{2}(Q^T(B_1 + B_2)Q)) && \Big| B_1 = A + D_1,\ B_2 = A + D_2 \\
&= \frac{1}{2}\lambda(Q^T B_1 Q + Q^T B_2 Q) \\
&\leq \frac{\lambda((B_1)_Q) + \lambda((B_2)_Q)}{2}. && \Big| \text{Fact 2.3}
\end{aligned}$$

Together, we obtain

$$\begin{aligned}
\frac{\mathrm{sum}(B) - n\lambda(B_Q)}{4} &\geq \frac{1}{2}\left(\frac{\mathrm{sum}(B_1) + \mathrm{sum}(B_2) - n\lambda((B_1)_Q) - n\lambda((B_2)_Q)}{4}\right) \\
&= \frac{g(G, d_1) + g(G, d_2)}{2}.
\end{aligned}$$

□

Now we are ready to formulate Boppana's algorithm as Algorithm 2.1 (BB) that finds and certifies an optimum bisection.

Algorithm 2.1: Boppana's Algorithm BB (from [12])

```
  Input : Graph $G$ with adjacency matrix $A$.
1 Compute $h(G)$: numerically find a vector $d^{\mathrm{opt}}$ which maximizes $g(G, d)$;
2 Construct a bisection $\hat{x}$: Let $x'$ be an eigenvector of $(A + \mathrm{diag}(d^{\mathrm{opt}}))_Q$
    corresponding to the eigenvalue $\lambda((A + \mathrm{diag}(d^{\mathrm{opt}}))_Q)$ and let $x = Qx'$.
    Construct a bisection vector $\hat{x}$ by splitting at the median $\bar{x}$ of $x$, i. e.
    let $\hat{x}_i = +1$ if $x_i \geq \bar{x}$ and $\hat{x}_i = -1$ if $x_i < \bar{x}$. If $\sum_i \hat{x}_i > 0$, move
    (arbitrarily) $\frac{1}{2}\sum_i \hat{x}_i$ vertices $i$ with $x_i = \bar{x}$ to part $-1$ letting $\hat{x}_i = -1$;
3 Output $\hat{x}$; if $\mathrm{cw}(\hat{x}) = h(G)$, output "optimum bisection", else output
    "fail".
```

Last, we provide a useful property on the functions g' and g:

Lemma 2.7. *The function $g'(G, d)$ is invariant under shifting d, i. e. for every $\beta \in \mathbb{R}$, it holds*

$$g'(G, d + \beta(1, \ldots, 1)^T) = g'(G, d).$$

The same holds for the function $g(G, d)$.

Proof. Let $\mathbf{1} = (1, \dots, 1)^T$. Then

$$\begin{aligned}
&g'(G, d + \beta\mathbf{1}) \\
&= \min_{\|x\|^2=n,\, x\in S} f(G, d + \beta\mathbf{1}, x) \\
&= \min_{\|x\|^2=n,\, x\in S} \left(\sum_{\{i,j\}\in E} \frac{1 - x_i x_j}{2} + \sum_{i\in V} (d + \beta\mathbf{1})_i (x_i^2 - 1) \right) \\
&= \min_{\|x\|^2=n,\, x\in S} \left(\sum_{\{i,j\}\in E} \frac{1 - x_i x_j}{2} + \sum_{i\in V} d_i (x_i^2 - 1) + \sum_{i\in V} \beta\mathbf{1}_i (x_i^2 - 1) \right) \\
&= \min_{\|x\|^2=n,\, x\in S} \left(f(G, d, x) + \beta \left(\sum_{i\in V} x_i^2 - n \right) \right) \\
&= \min_{\|x\|^2=n,\, x\in S} f(G, d, x) = g'(G, d).
\end{aligned}$$

This proves the invariance of g'. The invariance of g follows directly with Lemma 2.5. □

Boppana claims that his algorithm, assuming the density difference

$$p - q \geq c\sqrt{p \log n}/\sqrt{n} \quad \text{for a certain constant } c > 0, \tag{2.13}$$

finds and certifies an optimum bisection for $\mathcal{G}_n(p, q)$ w.h.p. In this thesis we thoroughly examine the properties of Boppana's algorithm. We will learn that his algorithm indeed finds the optimum bisection width w.h.p., but for finding the corresponding bisection itself a modification of the algorithm will be needed. According to Boppana's original paper [12], his algorithm also works well on the regular graph model $\mathcal{R}_n(r, b)$, assuming that

$$r \geq 6 \quad \text{and} \quad b \leq o(n^{1-1/\lfloor (r/2+1)/2 \rfloor}). \tag{2.14}$$

Our experimental data presented in this thesis give evidence that this indeed is true, but, as for the planted bisection model, no complete proof has been published on this result. However, since we focus our theoretical analysis on the planted bisection model, we will not review the proof on the regular graph model here. Let us note again that in [12], one finds the bounds $r \geq 3$ and $b \leq o(n^{1-1/\lfloor (r+1)/2 \rfloor})$. Reviewing the reference [13], this seems to hold for $2r$-regular graphs. Hence, we modified the bounds accordingly and backed up this modification by our experimental evidence.

2.4 Semidefinite Programming (SDP)

Before we provide the bisection algorithm from Feige and Kilian, we need some more notations and the basics of semidefinite programs. For symmetric matrices $A, B \in \mathbb{R}^{n\times n}$, we denote by $A \bullet B$ the inner product of A and B defined as $A \bullet B = \mathrm{tr}(AB) = \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}B_{ij}$. A matrix $A \in \mathbb{R}^{n\times n}$ is called symmetric positive semidefinite if A is symmetric, i. e. $A^T = A$, and for all real vectors $v \in \mathbb{R}^n$ we have $v^T A v \geq 0$. This property is denoted by $A \succeq 0$. Note that the eigenvalues of a symmetric matrix are real.

For a given real vector $c \in \mathbb{R}^n$ and $m+1$ symmetric matrices $F_0, \ldots, F_m \in \mathbb{R}^{n\times n}$, a semidefinite program (SDP) over variables $x \in \mathbb{R}^n$ is defined as

$$\min_x c^T x \quad \text{subject to} \quad F_0 + \sum_{i=1}^{m} x_i F_i \succeq 0. \tag{2.15}$$

The dual program associated with the SDP (for details see e. g. [74]) is the program over the variable matrix $Y = Y^T \in \mathbb{R}^{n\times n}$:

$$\max_Y \; -F_0 \bullet Y \quad \text{subject to} \quad \forall i: \; F_i \bullet Y = c_i \quad \text{and} \quad Y \succeq 0. \tag{2.16}$$

It is known that the optimal value of the maximizing dual SDP is never larger than the optimal value of the minimizing primal counterpart. However, unlike in linear programming, for semidefinite programs there may be a duality gap, i. e. the primal and/or dual might not attain their respective optima.

2.5 Feige and Kilian's SDP Algorithm FK

In [31], Feige and Kilian studied the bisection problem on the semirandom graph model. In order to create an algorithm robust against the adversarial modifications, they express the Minimum Bisection Problem for a graph $G = (V, E)$ as an SDP over an $n \times n$ matrix Y as follows:

$$h_p(G) = \min_{Y\in\mathbb{R}^{n\times n}} h_Y(G) \quad \text{s.t.} \quad \forall i \; y_{ii} = 1, \; \sum_{i,j} y_{ij} = 0, \; \text{and } Y \succeq 0, \tag{2.17}$$

where $h_Y(G) = \sum_{\substack{\{i,j\}\in E \\ i<j}} \frac{1-y_{ij}}{2}$.

The algorithm then uses an SDP solver as core component, as seen in Algorithm 2.2 (FK). This way the function $h_p(G)$ can be computed in polynomial time with arbitrary precision.

Algorithm 2.2: Feige and Kilian's Algorithm FK (from [31])

```
  Input: Graph $G$.
1 Compute $h_p(G)$ using an SDP solver.
2 Construct a bisection $\hat{x}$: choose some vertex $u$ and label it 1. Then, for
    every vertex $v \neq u$ compute $h_p(G \oplus \{u, v\})$. If $h_p(G \oplus \{u, v\}) = h_p(G)$,
    label the vertex with 1, otherwise $-1$.
3 Output $\hat{x}$; if $\mathrm{cw}(\hat{x}) = h_p(G)$, output "optimum bisection", else output
    "fail".
```

Recall the notation $G \oplus \{u, v\}$, which means we "toggle" the edge $\{u, v\}$: if there is an edge between u and v in G, the edge is removed. Otherwise, we add the edge. Since SDPs can be solved in polynomial time, Algorithm FK runs in polynomial.

Feige and Kilian also provide an alternative way to reconstruct the bisection. That alternative requires solving their SDP only once to compute $h_p(G)$ and then derives the bisection from the matrix Y which minimizes $h_p(G)$, as seen in Algorithm 2.3 (FK*).

Algorithm 2.3: Feige and Kilian's Algorithm FK* (from [31])

```
  Input: Graph $G$.
1 Compute $h_p(G)$ using an SDP solver. Let $Y$ be the matrix which
    minimizes $h(G)$.
2 Construct a bisection vector $\hat{x}$ using the sign of the elements in the first
    row of $Y$, i. e. let $\hat{x}_i = +1$ if $Y_{1i} \geq 0$ and $\hat{x}_i = -1$ if $Y_{1i} < 0$. If
    $\sum_i \hat{x}_i > 0$, move (arbitrarily) $\frac{1}{2}\sum_i \hat{x}_i$ vertices $i$ with $x_i = \bar{x}$ to part $-1$
    letting $\hat{x}_i = -1$;
3 Output $\hat{x}$; if $\mathrm{cw}(\hat{x}) = h_p(G)$, output "optimum bisection", else output
    "fail".
```

The original paper [31] states that FK and FK* achieve the same performance on the planted bisection model as Boppana's Algorithm BB does, and additionally proves that FK and FK* work on the semirandom model. The performance on the regular graph model has not been discussed.

3 Boppana's Lower Bound: Completed Proofs

Let us recall the steps of Boppana's Algorithm BB: first, a lower bound $h(G)$ on the bisection width is computed with spectral techniques. Then, a bisection is derived and its width is compared with the lower bound. If they are equal, a certified optimum bisection is found. A key step in proving that the algorithm works well on the planted bisection model is to show that the lower bound $h(G)$ equals w.h.p. the bisection width $\operatorname{bw}(G)$ of graph G. Hence, for a sufficiently large constant $c > 0$, Boppana states the following:

Claim 3.1 (Boppana [12] without proof)**.** *Let G be a random graph from $\mathcal{G}_n(p,q)$, and let $p - q \geq c(\sqrt{p \log n}/\sqrt{n})$. Then with probability $1 - \mathcal{O}(1/n)$, the bisection width of G equals $h(G)$, i. e. $h(G) = \operatorname{bw}(G)$.*

The proof sketch for this claim found in [12] is incomplete, which has also been emphasized by other authors, e. g. [31, 63]. Especially, the proof sketch relies on a not further specified extension of a result due to Füredi and Komlós [37]. For analyzing a random matrix, the original results require the variance of each entry to be fixed, while Boppana's application wants them to tend to zero.

Unfortunately, the author of [12] has never provided a full journal version. Up to our best knowledge, there has been no proof for Boppana's lower bound published so far, so we are the first to provide one. Our upcoming proof follows in large parts the one of [31, Theorem 27] from Feige and Kilian, used to prove properties of their SDP based Algorithm FK for finding minimum bisections.

3.1 Proof of Boppana's Lower Bound

We prove the following Theorem 3.2 as refinement of Claim 3.1. In the proof, we use two auxiliary Propositions 3.5 and 3.6, which we provide in the subsequent sections.

Theorem 3.2. *Let G be a random graph from $\mathcal{G}_n(p,q)$ with probabilities fulfilling $p - q \geq 42\sqrt{p \log n}/\sqrt{n}$. Let A denote the adjacency matrix of G*

and y be the planted bisection vector. Set $d^{(y)} = -\operatorname{diag}(y)Ay$ and $B^{(y)} = A + \operatorname{diag}(d^{(y)})$. Then with probability at least $1 - 3n^{-5}$, it holds:

- *$g(G, d^{(y)}) = \operatorname{bw}(G) = h(G)$ and*
- *the largest eigenvalue of $B_Q^{(y)}$ is $\lambda(B_Q^{(y)}) = 0$ with multiplicity 1.*

Before we start with the proof, let us remark that the planted bisection vector y is not needed to compute $h(G)$. Note further that, with adjacency matrix A and planted bisection vector y, we set

$$d^{(y)} = -\operatorname{diag}(y)Ay. \tag{3.1}$$

An equivalent but more intuitive characterization of $d^{(y)}$ is the following: $d_i^{(y)}$ is the difference between the number of adjacent vertices in other part as vertex i and the number of adjacent vertices in same part as i.

In the following proof, this vector will turn out to maximize the function $g(G, d)$ w.h.p. Still we need to be aware of the fact that, although the numerical maximization used in Boppana's algorithm finds some vector d with $g(G, d) = h(G)$, the vector d might differ from $d^{(y)}$.

Proof of Theorem 3.2. We first show that the vector $Q^T y$ always is an eigenvector of $B_Q^{(y)}$ corresponding to the eigenvalue 0:

$$\begin{aligned} B_Q^{(y)}(Q^T y) &= Q^T B^{(y)} Q Q^T y = Q^T B^{(y)} P y = Q^T B^{(y)} y = Q^T (A + \operatorname{diag}(d^{(y)}))y \\ &= Q^T (Ay + \operatorname{diag}(d^{(y)})y) \overset{(*)}{=} Q^T (Ay - Ay) = \mathbf{0}. \end{aligned}$$

At (*), we use $\operatorname{diag}(d^{(y)})y = -Ay$. For this, consider component i of $\operatorname{diag}(d^{(y)})y$:

$$d_i^{(y)} y_i = (-y_i A_{i\cdot} y) y_i = -A_{i\cdot} y.$$

Next we show that with probability at least $1 - 3n^{-5}$ all other eigenvectors of $B_Q^{(y)}$ correspond to eigenvalues strictly smaller than 0. From this we conclude $\lambda(B_Q^{(y)}) = 0$ with multiplicity 1.

We use two auxiliary propositions: from Proposition 3.5, we have $\lambda(A - \mathbb{E}(A)) < 15\sqrt{pn \log n}$ with probability at least $1 - n^{-5}$. From Proposition 3.6, we have $\lambda(\operatorname{diag}(d^{(y)}) - \mathbb{E}(\operatorname{diag}(d^{(y)}))) \leq 6\sqrt{pn \log n}$ with probability at least $1 - 2n^{-5}$. Then, with probability at least $(1 - n^{-5})(1 - 2n^{-5}) \geq 1 - 3n^{-5}$, it holds

$$
\begin{aligned}
&\lambda(B^{(y)} - \mathbb{E}(B^{(y)})) \\
&\leq \lambda(A - \mathbb{E}(A)) + \lambda(\operatorname{diag}(d^{(y)}) - \mathbb{E}(\operatorname{diag}(d^{(y)}))) && \text{Fact 2.3} \\
&< 15\sqrt{pn\log n} + 6\sqrt{pn\log n} && \text{Prop. 3.5 and 3.6} \\
&\leq \frac{1}{2}(p-q)n. && \text{Condition from Thm.}
\end{aligned}
$$

For the remaining proof, we proceed with the case that $\lambda(B^{(y)} - \mathbb{E}(B^{(y)})) < \frac{1}{2}(p-q)n$ holds. We will derive $\lambda(B_Q^{(y)}) = 0$ with multiplicity 1 and $h(G) = \operatorname{bw}(G)$. Let us define the matrix $M^{(y)} \in \mathbb{R}^{n\times n}$ as follows:

$$
M_{ij}^{(y)} = \begin{cases} p & \text{if } y_i = y_j \\ q & \text{otherwise.} \end{cases}
$$

Then

$$
\begin{aligned}
&& \lambda(B^{(y)} - \mathbb{E}(B^{(y)})) &< \frac{1}{2}(p-q)n \\
\Rightarrow && \lambda(B^{(y)} - \mathbb{E}(B^{(y)}) - \frac{1}{2}(p-q)nI) &< 0 && \text{Shift all eigenvalues} \\
\Rightarrow && \lambda(B^{(y)} - M^{(y)}) &< 0. && M^{(y)} = \mathbb{E}(B^{(y)}) + \frac{1}{2}(p-q)nI
\end{aligned}
$$

The strictly smaller $<$ at this point will later give us the property that all eigenvectors orthogonal to $Q^T y$ correspond to eigenvalues strictly smaller than 0.

$$
\begin{aligned}
0 > \lambda(B^{(y)} - M^{(y)}) &= \max_{x\in\mathbb{R}^n\setminus\{\mathbf{0}\}} \frac{x^T(B^{(y)} - M^{(y)})x}{x^T x} \\
&\geq \max_{x\in S\setminus\{\mathbf{0}\}} \frac{x^T(B^{(y)} - M^{(y)})x}{x^T x} \\
&\geq \max_{x\in S\setminus\{\mathbf{0}\}, x\perp y} \frac{x^T(B^{(y)} - M^{(y)})x}{x^T x} \\
&= \max_{x\in S\setminus\{\mathbf{0}\}, x\perp y} \frac{x^T B^{(y)} x - x^T M^{(y)} x}{x^T x}
\end{aligned} \tag{3.2}
$$

With $x \perp y$, we have $0 = \langle x, y\rangle = \sum_{i:y_i=1} x_i - \sum_{i:y_i=-1} x_i$. With $x \in S$ we also have $\sum_{i=1}^n x_i = 0$ and thus we conclude $\sum_{i:y_i=1} x_i = \sum_{i:y_i=-1} x_i = 0$. Now, $(M^{(y)}x)_i = M_{i\cdot}^{(y)} x = \sum_j M_{ij}^{(y)} x_j = \sum_{j:y_j=1} M_{ij}^{(y)} x_j + \sum_{j:y_j=-1} M_{ij}^{(y)} x_j$. For each i with $y_i = 1$, this equals $\sum_{j:y_j=1} px_j + \sum_{j:y_j=-1} qx_j = p\sum_{j:y_j=1} x_j + q\sum_{j:y_j=-1} x_j$. From above we know this is zero. For each i with $y_i = -1$, only p and q are exchanged and we obtain zero as well. Hence, $M^{(y)}x = \mathbf{0}$. We can conclude:

$$\begin{aligned}
(3.2) \quad &= \max_{x\in S\setminus\{\mathbf{0}\}, x\perp y} \frac{x^T B^{(y)} x}{x^T x} \\
&= \max_{x\in\mathbb{R}^{n-1}\setminus\{\mathbf{0}\}, Qx\perp y} \frac{(Qx)^T B^{(y)} (Qx)}{(Qx)^T (Qx)} \\
&= \max_{x\in\mathbb{R}^{n-1}\setminus\{\mathbf{0}\}, Qx\perp y} \frac{x^T Q^T B^{(y)} Qx}{x^T Q^T Qx} \\
&= \max_{x\in\mathbb{R}^{n-1}\setminus\{\mathbf{0}\}, x\perp Q^T y} \frac{x^T B_Q^{(y)} x}{x^T x}. && \Big| \langle Qx, y\rangle = \langle x, Q^T y\rangle
\end{aligned}$$

Hence, we have

$$0 > \max_{x\in\mathbb{R}^{n-1}\setminus\{\mathbf{0}\}, x\perp Q^T y} \frac{x^T B_Q^{(y)} x}{x^T x}.$$

Recall from above that $Q^T y$ is eigenvector corresponding to eigenvalue 0. Now, all other eigenvectors orthogonal to $Q^T y$ have eigenvalue strictly smaller than 0. Hence, 0 is largest eigenvalue with multiplicity 1. Note that this property was derived from the specific choice of vector $d^{(y)}$ and the resulting specific matrix $B^{(y)}$. The convex search for d in the bisection algorithm itself might obtain the same function value with another vector.

Finally, we show that $\lambda(B_Q^{(y)}) = 0$ implies $h(G) = \mathrm{bw}(G)$. Recall, $h(G) \le \mathrm{bw}(G)$ and $h(G) = \max_{d\in\mathbb{R}^n} g(G, d)$. Now

$$\begin{aligned}
\mathrm{bw}(G) \ge g(G, d^{(y)}) &= \frac{\mathrm{sum}(B^{(y)}) - n\lambda(B_Q^{(y)})}{4} && \Big| \text{Def. in Eq. (2.10)} \\
&= \frac{1}{4}\,\mathrm{sum}(B^{(y)}) \\
&= \frac{1}{4}\,\mathrm{sum}(A) + \mathrm{sum}(-\operatorname{diag}(y) A y) \\
&= \frac{1}{4}\sum_{i,j}(A_{ij} - y_i y_j A_{ij}) = \mathrm{cw}(y).
\end{aligned}$$

Thus, we get $g(G, d^{(y)}) = \mathrm{bw}(G) = h(G)$, as claimed in the theorem. $\square$

To complete the proof, we provide the crucial Propositions 3.5 and 3.6 in the next sections.

3.2 Bounding Eigenvalues of $A - \mathbb{E}(A)$

In this section we want to bound the eigenvalues of $A - \mathbb{E}(A)$. We provide and prove Proposition 3.5, which is a crucial part of the proof of Theorem 3.2. We start with a definition and an auxiliary lemma.

Definition 3.3 (Closed walk). *A closed walk of length k in a graph G is a sequence of $k+1$ vertices, where two consecutive vertices are connected by an edge in G and the first and last vertex of the sequence are the same.*

Lemma 3.4 (Proposition 26 in [31]). *Let G be a graph with n vertices and m edges. Then the number of closed walks of length k in G is at most $n(2m)^{k/2}$.*

Proof. Let A be the adjacency matrix of G. Then the largest eigenvalue $\lambda_1(A)$ has also largest absolute value (Perron-Frobenius theorem). Obviously, $\operatorname{tr}(A^2) = 2m$ and further $2m = \operatorname{tr}(A^2) \overset{\text{Fact 2.4}}{=} \sum_{i=1}^{n}(\lambda_i(A))^2 \geq (\lambda_1(A))^2$. It follows $\lambda_1(A) \leq \sqrt{2m}$.

Now, $\operatorname{tr}(A^k) \overset{\text{Fact 2.4}}{=} \sum_{i=1}^{n}(\lambda_i(A))^k \leq n(\lambda_1(A))^k \leq n\sqrt{2m}^k$. Since $\operatorname{tr}(A^k)$ is the number of closed walks of length k, the proof is completed. □

Proposition 3.5. *Let G be a random graph from $\mathcal{G}_n(p,q)$ with $q \leq p$ and $p \geq 1/n$. Let A denote the adjacency matrix of G. Then with probability at least $1-n^{-5}$ over the choice of A, the eigenvalues of $C = A - \mathbb{E}(A)$ are bounded by $\lambda(C) < 15\sqrt{pn\log n}$.*

Note that in Theorem 3.2, we require $p - q \geq 42\sqrt{p\log n}/\sqrt{n}$. Since $p \geq p-q$, we have $p \geq 42^2 \log n/n \geq 1/n$, as required in this proposition.

Proof of Proposition 3.5. Let y denote the planted bisection vector of G. First note that

$$\mathbb{E}(A_{ij}) = \begin{cases} 0 & \text{if } i = j \\ q & \text{if } y_i \neq y_j \\ p & \text{otherwise.} \end{cases}$$

Moreover, A, $\mathbb{E}(A)$, and C are symmetric with pairwise independent entries (i,j) for all $i < j$.

We bound the expectation of $(\max_i(|\lambda_i(C)|))^k$ from above using the trace of matrix C^k for some even $k \geq 2$:

$$\mathbb{E}(\operatorname{tr}(C^k)) = \mathbb{E}\left(\sum_{i=1}^{n}(\lambda_i(C))^k\right) \qquad \Big| \text{ Fact 2.4}$$

$$\begin{aligned}
&\geq \mathbb{E}\left(\max_i((\lambda_i(C))^k)\right) \\
&= \mathbb{E}\left((\max_i |\lambda_i(C)|)^k\right). && \Big| \, k \text{ even} \qquad (3.3)
\end{aligned}$$

We will later choose $k = \Theta(\log n)$. The expectation of the trace of C^k can further be computed as follows: we consider all closed walks of length k in the complete graph to obtain an upper bound. Let each closed walk $\pi^{(k)}$ be described by a vertex sequence $\pi^{(k)} = (v_0, \dots, v_k)$, with $v_i \in \{1, \dots, n\}$ and $v_0 = v_k$. We define the weight $w(\pi^{(k)})$ of the walk $\pi^{(k)}$ as $w(\pi^{(k)}) = \prod_{j=0}^{k-1} C_{v_j, v_{j+1}}$. Then $\operatorname{tr}(C^k) = \sum_{\pi^{(k)}} w(\pi^{(k)})$. The expected weight of a single walk is $\mathbb{E}(w(\pi^{(k)})) = \mathbb{E}\left(\prod_{j=0}^{k-1} C_{v_j, v_{j+1}}\right)$. We can rewrite the product using the set of distinct edges $E(\pi^{(k)})$ on the walk and for each edge $e \in E(\pi^{(k)})$ its multiplicity $\mu(e)$. Since the edges are stochastically independent, we obtain

$$\mathbb{E}(w(\pi^{(k)})) = \mathbb{E}\left(\prod_{e \in E(\pi^{(k)})} C_e^{\mu(e)}\right) = \prod_{e \in E(\pi^{(k)})} \mathbb{E}(C_e^{\mu(e)}). \tag{3.4}$$

Since the expectation $\mathbb{E}(C_e) = 0$, each walk which contains some edge only once gets expectation 0. Thus, we restrict ourselves to walks which contain each edge at least twice. The expected weight contribution of an edge e with multiplicity $\mu(e) \geq 2$ and probability p_e can be bounded as follows, where p_e is either p or q:

$$\begin{aligned}
\mathbb{E}(C_e^{\mu(e)}) &= (1 - p_e)^{\mu(e)} p_e + (-p_e)^{\mu(e)} (1 - p_e) \\
&\leq (1 - p_e)^{\mu(e)} p_e + (p_e)^{\mu(e)} (1 - p_e) \\
&= p_e(1 - p_e)\left((1 - p_e)^{\mu(e)-1} + (p_e)^{\mu(e)-1}\right) \\
&\leq p_e(1 - p_e)(1 - p_e + p_e)^{\mu(e)-1} && \Big| \, a^z + b^z \leq (a+b)^z,\ a, b \geq 0 \\
&\leq \min(p_e, 1 - p_e) \leq p. && (3.5)
\end{aligned}$$

Let $\pi^{(k,l)}$ denote a walk with k vertices and $l \leq k/2$ distinct edges. From Eq. (3.4) and Eq. (3.5), we then obtain

$$\mathbb{E}(w(\pi^{(k,l)})) \leq p^l. \tag{3.6}$$

We want to bound the expected weight sum $\mathbb{E}(k, t, l)$ of all closed walks $\pi^{(k,t,l)}$

of length k with t distinct vertices and l distinct edges:

$$\begin{aligned}\mathbb{E}(k,t,l) &= \mathbb{E}\left(\sum_{\pi^{(k,t,l)}} w(\pi^{(k,t,l)})\right)\\ &\le \sum_{\pi^{(k,t,l)}} \mathbb{E}\left(w(\pi^{(k,t,l)})\right)\\ &\le \sum_{\pi^{(k,t,l)}} p^l. \qquad \Big| \text{Eq. (3.6)} \end{aligned} \tag{3.7}$$

To bound the number of walks, we consider subgraphs induced by walks. A walk induces a connected subgraph with $t \le l+1$ vertices. The number of induced subgraphs can be bounded as follows. There are $\binom{n}{t}$ ways to choose the t vertices and t^{t-2} possible spanning trees on these t vertices, each having $t-1$ edges. Next, there are at most $(t^2)^{l-t+1}$ ways to choose the remaining edges in order to complete the spanning tree to the subgraph with l edges induced by a walk. In total, we obtain the following upper bound on the number of induced subgraphs:

$$\binom{n}{t} t^{t-2} t^{2(l-t+1)} \le n^t t^{2(l-t)} \frac{t^t}{t!} \le n^t t^{2(l-t)} e^t.$$

From Lemma 3.4, we know that there are at most $t(2l)^{k/2}$ walks in each such subgraph. Thus, there are at most $n^t t^{2(l-t)} e^t \cdot t(2l)^{k/2}$ walks of length k with t distinct vertices and l distinct edges. Together with Eq. (3.7), we obtain

$$\mathbb{E}(k,t,l) \le n^t t^{2(l-t)} e^t t(2l)^{k/2} p^l = \left(\frac{n}{t^2}\right)^t (t^2)^l e^t t(2l)^{k/2} p^l. \tag{3.8}$$

Now the term $(n/t^2)^t$ is increasing with t for $t = \mathcal{O}(\log(n))$ and n large enough. Since for fixed k and l, the upper bound from Eq. (3.8) on $\mathbb{E}(k,t,l)$ increases exponentially in t, the sum $\mathbb{E}(k,l)$ over the expected weights of all closed walks of length k and with l distinct vertices can be bounded by

$$\begin{aligned}\mathbb{E}(k,l) &= \sum_{t=2}^{l+1} \mathbb{E}(k,t,l)\\ &\le 2\mathbb{E}(k,t=l+1,l)\\ &\le 2\left(\frac{n}{(l+1)^2}\right)^{l+1} ((l+1)^2)^l e^{l+1}(l+1)(2l)^{k/2} p^l\\ &\le 2n^{l+1}\frac{1}{l+1} e^{l+1}(2l)^{k/2} p^l\end{aligned}$$

$$\leq n^{l+1}e^{l+1}(2l)^{k/2}p^l.$$

For $p \geq 1/n$, this bound is monotone increasing in l. We can combine now:

$$\begin{aligned}\mathbb{E}(\mathrm{tr}(C^k)) &= \sum_{l=2}^{k/2}\mathbb{E}(k,l) \leq \frac{k}{2}\mathbb{E}(k, l=k/2)\\ &\leq \frac{k}{2}n^{k/2+1}e^{k/2+1}k^{k/2}p^{k/2} = \frac{k}{2}n^{k/2}e^{k/2}k^{k/2}p^{k/2}ne\\ &= \left(\sqrt{nekp}\right)^k \frac{nek}{2} \end{aligned} \tag{3.9}$$

and with $\bar{\lambda} = \max_i |\lambda_i(C)|$

$$\mathbb{E}(\bar{\lambda}^k) \overset{\text{Eq. (3.3)}}{\leq} \mathbb{E}(\mathrm{tr}(C^k)) \overset{\text{Eq. (3.9)}}{\leq} \left(\sqrt{nekp}\right)^k \frac{nek}{2}. \tag{3.10}$$

Now we use Markov's inequality $P(X \geq z) \leq \frac{\mathbb{E}(X)}{z}$ as follows. We want to bound the probability that $\bar{\lambda}$ exceeds the upper bound stated in the proposition. We have

$$\begin{aligned}\frac{1}{e^k} &\geq \Pr(\bar{\lambda}^k \geq e^k\mathbb{E}(\bar{\lambda}^k)) && \Big|\,\text{Markov's inequality}\\ &\geq \Pr(\bar{\lambda}^k \geq e^k\left(\sqrt{nekp}\right)^k \frac{nek}{2}) && \Big|\,\text{Eq. (3.10)}\\ &= \Pr(\bar{\lambda} \geq e^{3/2}\sqrt{nkp}(nek/2)^{1/k}).\end{aligned}$$

With $k = a\log n$ we have $n^{1/k} = n^{1/(a\log n)} = e^{1/a}$ and get

$$\frac{1}{e^k} = \frac{1}{n^a} \geq \Pr(\bar{\lambda} \geq e^{3/2+1/a+1/k}(k/2)^{1/k}\sqrt{a}\sqrt{np\log n}).$$

Assume $\log n \geq 4$ and choose $a \in [5, 5.2]$, such that k is even. We have $k \geq 20$ and

$$e^{3/2+1/a+1/k}(k/2)^{1/k}\sqrt{a} < 15$$

and obtain

$$\frac{1}{n^5} \geq \frac{1}{n^a} \geq \Pr(\bar{\lambda} > 15\sqrt{np\log n}) \geq \Pr(\lambda(C) > 15\sqrt{np\log n}).$$

□

3.3 Bounding Eigenvalues of $\operatorname{diag}(\boldsymbol{d}) - \mathbb{E}(\operatorname{diag}(\boldsymbol{d}))$

In this section we want to bound the eigenvalues of $\operatorname{diag}(d) - \mathbb{E}(\operatorname{diag}(d))$. We provide and prove Proposition 3.6, which is, besides Proposition 3.5, the key part of the proof of Theorem 3.2.

Proposition 3.6. *Let G be a random graph from $\mathcal{G}_n(p,q)$ with $p - q \geq 12\sqrt{p \log n/n}$. Let A denote the adjacency matrix of G and y be the planted bisection vector. Set $d^{(y)} = -\operatorname{diag}(y)Ay$. The eigenvalues of $C = \operatorname{diag}(d^{(y)}) - \mathbb{E}(\operatorname{diag}(d^{(y)}))$ are bounded by $\lambda(C) \leq 6\sqrt{pn \log n}$ with probability at least $1 - 2n^{-5}$.*

Proof. The eigenvalues of $\operatorname{diag}(d^{(y)}) - \mathbb{E}(\operatorname{diag}(d^{(y)}))$ are its diagonal entries. Recall that in the model $\mathcal{G}_n(p,q)$, each edge crossing the planted bisection is generated by probability q and each edge within a part with probability p. Since we defined $d^{(y)} = -\operatorname{diag}(y)Ay$, each entry $d_i^{(y)}$ can be modeled as the sum of $n/2$ Bernoulli experiments with probability q and the sum of $n/2 - 1$ Bernoulli experiments with probability p. We denote these sums by the random variables $X_i^{(q)}$ and $X_i^{(p)}$, i. e. $d_i^{(y)} = X_i^{(q)} - X_i^{(p)}$. The expectation values are $\mathbb{E}(X_i^{(q)}) = \frac{n}{2}q$ and $\mathbb{E}(X_i^{(p)}) = (\frac{n}{2} - 1)p$. We can note here that two different $d_i^{(y)}$ and $d_j^{(y)}$ are not independent but the Bernoulli experiments for each single $d_i^{(y)}$ are.

We want to show that the probability of any $d_i^{(y)} - \mathbb{E}(d_i^{(y)})$ exceeding the value

$$z = 6\sqrt{pn \log n} \tag{3.11}$$

tends to zero. For each i we have

$$\begin{aligned}
&\Pr(d_i^{(y)} - \mathbb{E}(d_i^{(y)}) \geq z) \\
&= \Pr\left(\left(X_i^{(q)} - X_i^{(p)}\right) - \mathbb{E}\left(X_i^{(q)} - X_i^{(p)}\right) \geq z\right) \\
&= \Pr\left(\left(X_i^{(q)} - \mathbb{E}(X_i^{(q)})\right) - \left(X_i^{(p)} - \mathbb{E}(X_i^{(p)})\right) \geq z\right) \\
&\leq \Pr\left(\left(X_i^{(q)} - \mathbb{E}(X_i^{(q)}) \geq \frac{z}{2}\right) \cup \left(X_i^{(p)} - \mathbb{E}(X_i^{(p)}) \leq -\frac{z}{2}\right)\right).
\end{aligned} \tag{3.12}$$

The union bound theorem for a finite set of events Z_i states

$$\Pr\left(\bigcup_i Z_i\right) \leq \sum_i \Pr(Z_i), \tag{3.13}$$

while the Z_i are not required to be independent. We obtain

$$\begin{aligned}
\Pr\left(\max_i(d_i^{(y)} - \mathbb{E}(d_i^{(y)})) \geq z\right) &= \Pr\left(\bigcup_i (d_i^{(y)} - \mathbb{E}(d_i^{(y)}) \geq z)\right) \\
&\overset{(3.13)}{\leq} \sum_i \Pr\left(d_i^{(y)} - \mathbb{E}(d_i^{(y)}) \geq z\right) \\
&\overset{(3.12)}{\leq} \sum_i \left(\Pr\left(X_i^{(q)} - \mathbb{E}(X_i^{(q)}) \geq \frac{z}{2}\right) + \Pr\left(X_i^{(p)} - \mathbb{E}(X_i^{(p)}) \leq -\frac{z}{2}\right)\right). \quad (3.14)
\end{aligned}$$

To estimate these probabilities, we use the Chernoff bound in the following form [59, Theorem 4.4]: for the sum X of n Bernoulli experiments with probability p and for $0 < \delta \leq 1$, [59, Theorem 4.4] gives

$$\Pr\left(X - \mathbb{E}(X) \geq \delta \cdot \mathbb{E}(X)\right) \leq e^{-\mathbb{E}(X)\delta^2/3}. \quad (3.15)$$

For the case $\delta > 1$, we start with the following form [59, Theorem 4.4]:

$$\Pr\left(X - \mathbb{E}(X) \geq \delta \cdot \mathbb{E}(X)\right) \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{\mathbb{E}(X)}. \quad (3.16)$$

For $\delta > 1$, we show that

$$\frac{e^\delta}{(1+\delta)^{1+\delta}} \leq e^{-\delta/3}. \quad (3.17)$$

Analogously to the proof of [59, Theorem 4.4], take the logarithm to obtain the equivalent condition

$$f(\delta) = \delta - (1+\delta)\log(1+\delta) + \delta/3 \leq 0$$

and its first derivative $f'(\delta) = -\log(1+\delta) + 1/3$. Now, $f(1) < 0$ and since $f'(\delta) < 0$ for $\delta > 1$, $f(\delta)$ is monotone decreasing. Hence, Eq. (3.17) holds.

From Eq. (3.16) and Eq. (3.17), we obtain the following for $\delta > 1$:

$$\Pr\left(X - \mathbb{E}(X) \geq \delta \cdot \mathbb{E}(X)\right) \leq e^{-\mathbb{E}(X)\delta/3}. \quad (3.18)$$

Now we combine Eq. (3.15) and Eq. (3.18) to the following convenient bound which holds for any $\delta > 0$:

$$\Pr\left(X - \mathbb{E}(X) \geq \delta \cdot \mathbb{E}(X)\right) \leq \exp\left(-\frac{\min(\delta, \delta^2)}{3}\mathbb{E}(X)\right). \quad (3.19)$$

Additionally, for $\delta \in [0,1]$, we will use the bound [59, Theorem 4.5]:

$$\Pr\left(X - \mathbb{E}(X) \leq -\delta \cdot \mathbb{E}(X)\right) \leq \exp\left(-\frac{\delta^2}{2}\mathbb{E}(X)\right). \tag{3.20}$$

We use the Chernoff bound from Eq. (3.19) with $X = X_i^{(q)}$, $\mathbb{E}(X) = \frac{qn}{2}$ and $\delta = 6 \cdot \frac{p}{q}\sqrt{\frac{\log n}{pn}}$. Then, with z as in Eq. (3.11), it holds $\delta \cdot \mathbb{E}(X) = \frac{z}{2}$ and hence

$$\Pr\left(X_i^{(q)} - \mathbb{E}(X_i^{(q)}) \geq \frac{z}{2}\right) \leq \exp\left(-\frac{\min(\delta, \delta^2)}{3}\mathbb{E}(X)\right).$$

With $\delta \cdot \mathbb{E}(X) = 3\sqrt{pn\log n} \geq 18\log n$, since $p \geq \frac{36\log n}{n}$, and $\delta^2 \cdot \mathbb{E}(X) = 18 \cdot \frac{p}{q}\log n \geq 18\log n$, since $p \geq q$, we obtain $\min(\delta, \delta^2) \cdot \mathbb{E}(X) \geq 18\log n$ and hence

$$\Pr\left(X_i^{(q)} - \mathbb{E}(X_i^{(q)}) \geq \frac{z}{2}\right) \leq \exp\left(-6\log n\right) = \frac{1}{n^6}. \tag{3.21}$$

Next, we use the Chernoff bound from Eq. (3.20) with $X = X_i^{(p)}$, $\mathbb{E}(X) = p\left(\frac{n}{2} - 1\right)$ and $\delta = 6\sqrt{\frac{\log n}{p(n-2)}}$. Since $p \geq \frac{72\log n}{n}$, we have $\delta < 1$. Furthermore, $\frac{z}{2} > \delta \cdot \mathbb{E}(X) = 3\sqrt{p(n-2)\log n}$ and $\delta^2 \cdot \mathbb{E}(X) = 18\log n$. Hence,

$$\begin{aligned}\Pr\left(X_i^{(p)} - \mathbb{E}(X_i^{(p)}) \leq -\frac{z}{2}\right) &\leq \Pr\left(X_i^{(p)} - \mathbb{E}(X_i^{(p)}) \leq -\delta \cdot \mathbb{E}(X_i^{(p)})\right) \\ &\leq \exp\left(-\frac{\delta^2}{2}\mathbb{E}(X)\right) \leq \exp\left(-9\log n\right) = \frac{1}{n^9}.\end{aligned} \tag{3.22}$$

We insert Eq. (3.21) and (3.22) into Eq. (3.14) and obtain

$$\Pr\left(\max_i(d_i^{(y)} - \mathbb{E}(d_i^{(y)})) \geq z\right) \leq \sum_i \left(\frac{1}{n^6} + \frac{1}{n^9}\right) \leq \frac{2}{n^5}$$

with $z = 6\sqrt{pn\log n}$. □

4 Boppana's Reconstruction: Issues and Solutions

From Theorem 3.2 one can conclude that the value $h(G)$ computed by Algorithm BB is, w.h.p., equal to the optimum bisection width of G. However, to guarantee that the algorithm works well one needs additionally to show that it also finds an optimum bisection. The Algorithm BB is indeed successful in the following case:

Proposition 4.1. *Let G be a graph with adjacency matrix A and $h(G) = \mathrm{bw}(G)$. Let d^{opt} be a vector computed in Step 1 of Algorithm BB. If the multiplicity of the largest eigenvalue of $(A + \mathrm{diag}(d^{\mathrm{opt}}))_Q$ is 1, then Boppana's Algorithm BB is able to certify the optimality of the resulting bisection.*

Unfortunately, we cannot assume that the multiplicity of the largest eigenvalue of matrix $B_Q = (A + \mathrm{diag}(d^{\mathrm{opt}}))_Q$ obtained by the algorithm is 1:

Proposition 4.2. *Let G be a graph with adjacency matrix A and $h(G) = \mathrm{bw}(G)$. Then there exists an optimum solution d^{opt}, with $g(G, d^{\mathrm{opt}}) = h(G)$, such that the largest eigenvalue of $(A + \mathrm{diag}(d^{\mathrm{opt}}))_Q$ has multiplicity at least two.*

In this case, finding a bisection vector with Algorithm BB cannot be guaranteed, even though $h(G) = \mathrm{bw}(G)$ holds. Note that [12, Theorem 4.1] (formulated as Claim 3.1 in this thesis) says nothing about the multiplicity of the largest eigenvalue and that Theorem 3.2, which is a reformulation of the main result of Boppana, does not give any guarantee on this.

In this chapter we prove Proposition 4.1 (in Section 4.1) and Proposition 4.2 (in Section 4.2). We further discuss the issues in the reconstruction process of a bisection in Boppana's algorithm arising from Proposition 4.2. We provide several modifications of Algorithm BB to find unique and non-unique bisections, where we say that a graph G has a unique optimum bisection if there exists a unique, up to the sign, bisection vector x such that $\mathrm{cw}(x) = \mathrm{cw}(-x) = \mathrm{bw}(G)$.

4.1 BB's Reconstruction Succeeds if Multiplicity is One

In this section we prove Proposition 4.1, i. e. show that Algorithm BB finds and certifies an optimum bisection if $h(G) = \mathrm{bw}(G)$ and the multiplicity of the largest eigenvalue of B_Q is 1. A key observation for reconstructing the optimum bisection is the following:

Lemma 4.3. *Let G be a graph with n vertices and $h(G) = \mathrm{bw}(G)$. Let $d^{\mathrm{opt}} \in \mathbb{R}^n$ s. t. $g(G, d^{\mathrm{opt}}) = \mathrm{bw}(G)$, and set $B^{\mathrm{opt}} = A + \mathrm{diag}(d^{\mathrm{opt}})$. Then, for every optimum bisection vector y, the vector $y' = Q^T y$ is an eigenvector of B_Q^{opt} corresponding to the largest eigenvalue $\lambda(B_Q^{\mathrm{opt}})$.*

Proof. From the assumptions, we have

$$g(G, d^{\mathrm{opt}}) = \mathrm{bw}(G) = \frac{\mathrm{sum}(B^{\mathrm{opt}}) - n\lambda(B_Q^{\mathrm{opt}})}{4} \qquad \Big| \; B^{\mathrm{opt}} = A + \mathrm{diag}(d^{\mathrm{opt}})$$

$$\Leftrightarrow \qquad \lambda(B_Q^{\mathrm{opt}}) = \frac{\mathrm{sum}(B^{\mathrm{opt}}) - 4\,\mathrm{bw}(G)}{n}. \tag{4.1}$$

We compute the value of the Rayleigh quotient of B_Q^{opt} and the vector $y' = Q^T y$:

$$\begin{aligned}
\frac{y'^T B_Q^{\mathrm{opt}} y'}{y'^T y'} &= \frac{(Q^T y)^T B_Q^{\mathrm{opt}} Q^T y}{(Q^T y)^T Q^T y} = \frac{y^T Q B_Q^{\mathrm{opt}} Q^T y}{y^T Q Q^T y} \\
&= \frac{y^T Q (Q^T B^{\mathrm{opt}} Q) Q^T y}{y^T Q Q^T y} \\
&= \frac{y^T P^T B^{\mathrm{opt}} P y}{y^T P y} = \frac{y^T B^{\mathrm{opt}} y}{y^T y} && \Big| \; QQ^T = P = P^T \text{ and } Py = y \\
&= \frac{y^T (A + \mathrm{diag}(d^{\mathrm{opt}})) y}{n} = \frac{y^T A y + \sum_i d_i^{\mathrm{opt}}}{n}. && \Big| \; y_i^2 = 1 \text{ on the diagonal}
\end{aligned}$$

We have $y^T A y = \sum_{i,j} A_{ij} y_i y_j$. According to the definition, $A_{ij} = 1$ if there is an edge $\{i, j\} \in E$. Edges with both vertices in the same part contribute (twice) by 1 to the sum. Cut edges on the other hand contribute (twice) by -1. There are $\mathrm{bw}(G)$ cut edges. Hence, $y^T A y = \mathrm{sum}(A) - 4\,\mathrm{bw}(G)$ and we get:

$$\begin{aligned}
\frac{y^T A y + \sum_i d_i^{\mathrm{opt}}}{n} &= \frac{\mathrm{sum}(A) - 4\,\mathrm{bw}(G) + \sum_i d_i^{\mathrm{opt}}}{n} \\
&= \frac{\mathrm{sum}(B^{\mathrm{opt}}) - 4\,\mathrm{bw}(G)}{n} \overset{(4.1)}{=} \lambda(B_Q^{\mathrm{opt}}).
\end{aligned}$$

Since the Rayleigh quotient of B_Q^{opt} and y' takes the value $\lambda(B_Q^{\text{opt}})$, we conclude that y' is an eigenvector of B_Q^{opt} corresponding to the eigenvalue $\lambda(B_Q^{\text{opt}})$. □

This Lemma shows us that the Algorithm BB w.h.p. obtains some B_Q^{opt} whose eigenspace of the largest eigenvalue contains vector $Q^T y$ for an optimum bisection vector y. If the largest eigenvalue has multiplicity 1, then the obtained bisection vector is equal to y or $-y$ and Proposition 4.1 follows directly. Note that Theorem 3.2 states that the largest eigenvalue of matrix $B_Q^{(y)} = A + \operatorname{diag}(d^{(y)})$ has multiplicity 1 w.h.p., but the Algorithm BB might compute some d^{opt} different than $d^{(y)}$ which in consequence affects that B_Q^{opt} differs from $B_Q^{(y)}$. The importance of the multiplicity of the largest eigenvalue for the reconstruction process has also been observed by Boppana and Blumofe in an unpublished work [10].

Now, on the one hand, the optimum bisection is unique w.h.p. under the assumptions made in Theorem 3.2 and the Algorithm BB might obtain an eigenspace of dimension 1. In experiments we have conducted on the algorithm, we observe that this actually is the case. On the other hand, the algorithm might also obtain an eigenspace which has dimension at least 2 for every input graph, letting the algorithm fail. We show this in the next section.

4.2 Issues with Multiplicity Two or More

In this section we prove Proposition 4.2, i. e. the largest eigenvalue in Algorithm BB can have multiplicity of 2 or more, even when the optimum bisection is unique. The following fact has been observed independently in [10]:

Lemma 4.4. *Let G be a graph with $h(G) = \operatorname{bw}(G)$ and let y be the bisection vector of an arbitrary optimum solution. Let A denote the adjacency matrix of G and set $d^{(y)} = -\operatorname{diag}(y)Ay$. Then for every d^{opt} which gives $g(G, d^{\text{opt}}) = \operatorname{bw}(G)$, there exists some $\alpha^{(y)} \in \mathbb{R}$ such that $d^{opt} = d^{(y)} + \alpha^{(y)} y + \lambda(B_Q^{\text{opt}})\mathbf{1}$ with $B^{\text{opt}} = A + \operatorname{diag}(d^{\text{opt}})$.*

Note that the function $g(G, d)$ is invariant under shifting of d (Lemma 2.7), which now reflects in the term $\lambda(B_Q^{\text{opt}})\mathbf{1}$. However, this shifting affects all eigenvalues of the matrix B_Q^{opt} equally, such that it has no further impact on the properties we are going to discuss.

Proof of Lemma 4.4. From Lemma 4.3 we know that $Q^T y$ is an eigenvector of B_Q^{opt} corresponding to the largest eigenvalue $\lambda(B_Q^{\text{opt}})$. In the following we shorthand $\lambda = \lambda(B_Q^{\text{opt}})$. We get the following sequence of equivalent conditions:

$$
\begin{aligned}
& & B_Q^{\text{opt}} Q^T y &= \lambda Q^T y & & \\
&\Leftrightarrow & Q^T B^{\text{opt}} Q Q^T y &= \lambda Q^T y & & \\
&\Leftrightarrow & Q^T B^{\text{opt}} y &= \lambda Q^T y & & \Big| \, y \text{ has mean zero} \\
&\Leftrightarrow & Q Q^T B^{\text{opt}} y &= \lambda Q Q^T y & & \\
&\Leftrightarrow & P B^{\text{opt}} y &= \lambda y. & & \Big| \, QQ^T y = Py = y
\end{aligned}
$$

Since P removes the mean of a vector, it holds for some $\alpha \in \mathbb{R}$ that

$$
\begin{aligned}
&\Leftrightarrow & B^{\text{opt}} y &= \lambda y + \alpha (1, \ldots, 1)^T & & \\
&\Leftrightarrow & (A + D^{\text{opt}}) y &= \lambda y + \alpha (1, \ldots, 1)^T & & \Big| \, D^{\text{opt}} = \text{diag}(d^{\text{opt}}) \\
&\Leftrightarrow & Ay + D^{\text{opt}} y &= \lambda y + \alpha (1, \ldots, 1)^T & & \\
&\Leftrightarrow & D^{\text{opt}} y &= \lambda y - Ay + \alpha (1, \ldots, 1)^T. & &
\end{aligned}
$$

In the next step, we multiply the vectors in the equation with the diagonal matrix $\text{diag}(y)$. Since the $y_i \in \{1, -1\}$, the multiplication is revertible and hence "$\Leftrightarrow$".

$$
\begin{aligned}
&\Leftrightarrow & \text{diag}(y) D^{\text{opt}} y &= \lambda \, \text{diag}(y) y - \text{diag}(y) A y + \alpha \, \text{diag}(y) (1, \ldots, 1)^T \\
&\Leftrightarrow & d^{\text{opt}} &= \lambda \mathbf{1} - \text{diag}(y) A y + \alpha y \\
&\Leftrightarrow & d^{\text{opt}} &= \lambda \mathbf{1} + d^{(y)} + \alpha y.
\end{aligned}
$$

This completes the proof. $\square$

From Lemma 4.4 we know two important things: first, every optimal vector d can be expressed in terms of an optimum bisection vector y and some parameter α. Second, if there are different optimal d and d', both can be expressed in terms of the same vector y, only with different α and α'. For illustration let us consider the example graph G_{ex} in Figure 4.1, which has a unique optimum bisection of width 6. The lower bound $h(G_{\text{ex}})$ is tight, i. e. $h(G_{\text{ex}}) = \text{bw}(G_{\text{ex}}) = 6$.

From the known bisection vector y, we compute $g(G_{\text{ex}}, d^{(y)} + \alpha y)$, as shown in Figure 4.2 left. Obviously, every $\alpha \in [-0.535, 1.53]$ results in an optimal vector

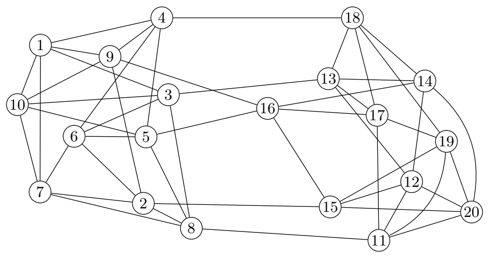


Figure 4.1: Example graph G_{ex} with optimum bisection $\{1, 2, \ldots, 10\}$ and $\{11, 12, \ldots, 20\}$. For this graph, $h(G_{\text{ex}}) = \text{bw}(G_{\text{ex}}) = 6$, i. e. the lower bound is tight.

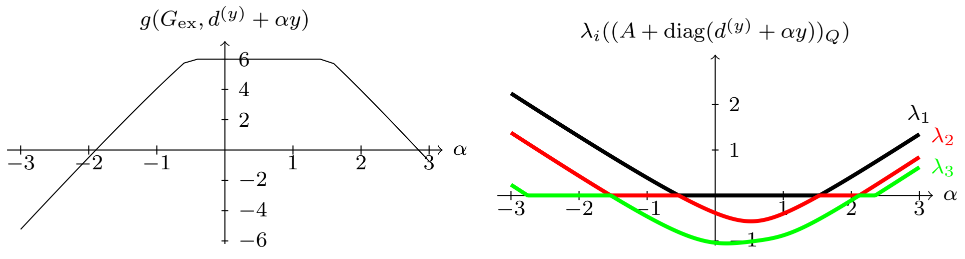


Figure 4.2: Function $g(G_{\text{ex}}, d^{(y)}+\alpha y)$ for graph G_{ex}. Every vector $d^{(y)}+\alpha y$ with $\alpha \in [-0.535, 1.53]$ is optimal (see left figure). The computation of $g(G_{\text{ex}}, d^{(y)}+\alpha y)$ is based on the largest eigenvalue of $(A+\text{diag}(d^{(y)}+\alpha y))_Q$. The largest three eigenvalues of this matrix are shown on the right. When λ_1 coincides with λ_2, the reconstruction of the bisection vector cannot be done by simply using the eigenvector corresponding to λ_1.

d maximizing $g(G_{\text{ex}}, d)$. In the proof of Theorem 3.2 that $h(G) = \text{bw}(G)$ w.h.p., we used the vector $d^{(y)}$ and have shown that w.h.p. the largest eigenvalue for the core problem is unique and leads to a tight bound. This corresponds to $\alpha = 0$, and we observe that for the example graph G_{ex}, the value of $g(G_{\text{ex}}, d^{(y)} + \alpha y)$ is maximal at $\alpha = 0$. Figure 4.2 right shows the three largest eigenvalues of the matrix $(A + \text{diag}(d^{(y)} + \alpha y))_Q$, where A denotes the adjacency matrix of G_{ex}. For $\alpha = 0$, we indeed have a unique largest eigenvalue. Due to Lemma 4.3, the eigenvector corresponding to the largest eigenvalue provides the optimum bisection in this case.

However, Algorithm BB finds any d^{opt}, which might not be the desired $\alpha = 0$ but might just be that $d^{(y)} + \alpha y$ where the largest and second largest eigenvalue coincide. Then the largest eigenvalue has multiplicity of at least two and the reconstruction cannot be done by simply taking some eigenvector from this eigenspace. This property is stated formally in the following lemma:

Lemma 4.5. *Let G be a graph with $h(G) = \text{bw}(G)$ and let y be the bisection vector of an arbitrary optimum bisection. Let further A denote the adjacency matrix of G. Then there exists some $\alpha \in \mathbb{R}$ such that $g(G, d^{\text{opt}}) = h(G)$ with $d^{\text{opt}} = d^{(y)} + \alpha y$ and the largest eigenvalue of $(A + \text{diag}(d^{\text{opt}}))_Q$ has multiplicity at least two.*

An intuition can be obtained from Figure 4.2 right: we start at some α with optimal value for g and increase it, until $\lambda_1 = \lambda_2$. The formal proof is as follows:

Proof. From the assumption $h(G) = \text{bw}(G)$, we know there exists some d with $g(G, d) = \text{bw}(G)$. Let $B = A + \text{diag}(d)$. From Lemma 4.3, we know that $Q^T y$ is an eigenvector of B_Q corresponding to the largest eigenvalue $\lambda(B_Q)$.

Consider $d' = d + \alpha' y$ and $B' = A + \text{diag}(d') = B + \text{diag}(\alpha' y)$ for any $\alpha' \in \mathbb{R}$. We show that $Q^T y$ is also eigenvector of B'_Q with the same eigenvalue. For this, we multiply vector $Q^T y$ from the right to B'_Q and obtain

$$\begin{aligned}
B'_Q Q^T y &= B_Q Q^T y + \text{diag}(\alpha' y)_Q Q^T y \\
&= B_Q Q^T y + Q^T \text{diag}(\alpha' y) Q Q^T y \\
&= B_Q Q^T y + Q^T \text{diag}(\alpha' y) y && \Big| \; QQ^T y = Py = y \\
&= B_Q Q^T y + Q^T \alpha' \mathbf{1} \\
&= B_Q Q^T y && \Big| \; \text{columns of } Q \text{ are basis of } S \\
&= \lambda(B_Q) Q^T y.
\end{aligned}$$

Note that $\lambda(B_Q)$ may not be the largest eigenvalue of B'_Q anymore. Next, we analyze the eigenvalues of B'_Q, as we increase α' starting from $\alpha' = 0$. At $\alpha' = 0$, the largest eigenvalue is $\lambda(B_Q)$. This value stays eigenvalue for any $\alpha' \geq 0$, so we only need to show that the second largest eigenvalue $\lambda_2(B'_Q)$ grows until it is equal to $\lambda(B_Q)$, while $d' = d + \alpha' y$ remains optimum solution. For this purpose, we recall the definition of function $g(G, d)$ as given in Eq. (2.10):

$$g(G, d) = \frac{\text{sum}(B) - n\lambda(B_Q)}{4}.$$

If we consider $g(G, d')$, we have $\mathrm{sum}(B') = \mathrm{sum}(B + \mathrm{diag}(\alpha' y)) = \mathrm{sum}(B)$, since $\mathrm{sum}(\alpha' y) = 0$. Thus, the value of $g(G, d')$ changes if and only if $\lambda(B'_Q)$ changes. We know from Lemma 2.6 that $g(G, d')$ is concave in d', hence $\lambda((A + \mathrm{diag}(d'))_Q)$ has to be convex in d'. Eigenvalues are continuous in the changes of d'. We increase α' until $g(G, d')$ is no longer optimal. Formally, we compute

$$\alpha'_{\max} = \sup_{\alpha' \geq 0}\{\alpha' : g(G, d') = g(G, d)\} = \sup_{\alpha' \geq 0}\{\alpha' : \lambda(B'_Q) = \lambda(B_Q)\}.$$

Necessarily, for $\alpha'_{\max}$, i. e. the largest α' with optimal value for g, the matrix B'_Q has equal largest and second largest eigenvalue. Note that such $\alpha'_{\max}$ exists, since there is some $\alpha' > 0$ with $\lambda(B'_Q) > \lambda(B_Q)$: for $\alpha' \to \infty$, the diagonal part $\alpha' y$ dominates the matrix B'. W.l.o.g., assume $y_1 = 1$. Consider the vector $x' \in \mathbb{R}^n$ with $x'_1 = n/2 - 1$ and $x'_i = -1$ for all $i \neq 1$ with $y_i = 1$. Set $x'_i = 0$ for all i with $y_i = -1$. Then, for the vector $x = Q^T x'$ we consider the Rayleigh quotient $\frac{x^T B'_Q x}{\|x\|^2}$ which provides a lower bound for the largest eigenvalue. With $QQ^T = P$, we obtain

$$\begin{aligned} \lambda(B'_Q) &\geq \frac{x^T B'_Q x}{\|x\|^2} = \frac{x'^T P B' P x'}{\|Q^T x'\|^2} = \frac{x'^T B' x'}{\|x'\|^2} \\ &= \frac{x'^T B x'}{\|x'\|^2} + \alpha' \frac{\|x'\|^2}{\|x'\|^2} = \frac{x'^T B x'}{\|x'\|^2} + \alpha'. \end{aligned}$$

Now, as $\alpha' \to \infty$, it follows that the largest eigenvalue of B'_Q tends to ∞.

As last step, in order to obtain α and d^{opt} as described in the lemma, we take $d' = d + \alpha'_{\max} y$ from the point where the two eigenvalues coincide. Then we apply Lemma 4.4 and decompose it into $d' = d^{(y)} + \alpha'' y + \lambda(B_Q)\mathbf{1}$. Now, α'' and $d^{\mathrm{opt}} = d^{(y)} + \alpha'' y$ are our solutions. The shifting part $\lambda(B_Q)\mathbf{1}$ can be omitted due to the invariance of function g (Lemma 2.7). □

Proposition 4.2 now follows directly from Lemma 4.5.

4.3 Finding a Unique Optimum Bisection BB-U

We have seen that the Algorithm BB might compute some B_Q^{opt} with multiplicity of 2 or more for the largest eigenvalue and hence causing difficulties in finding the optimum bisection. In this section we propose Algorithm BB-U, which is able to find a unique optimum bisection, regardless of the multiplicity of the largest eigenvalues in the computations.

For our modification of Boppana's algorithm, we lend an idea from Feige and Kilian [31] based on the following observation:

Lemma 4.6 (Similar to Lemma 20 in [31])**.** *Let G be a graph chosen from $\mathcal{G}_n(p,q)$ with probabilities fulfilling Eq.* (2.13)*. Then, with probability at least $1 - 1/n^{1-o(1)}$ over the choice of G, $h(G) = \mathrm{bw}(G)$ and for every $e \in V(G) \times V(G)$, it holds $h(G \oplus e) = \mathrm{bw}(G \oplus e)$.*

The proof can be found in a similar form in [31] and is provided here for completeness:

Proof. Due to Theorem 3.2, it holds $h(G) = \mathrm{bw}(G)$ with probability at least $1 - 3n^{-5}$. We estimate the a priori probability that we sample $G \oplus e$ instead of G. Assume $q > 1/n^{2+o(1)}$ and $p < 1 - 1/n^{2+o(1)}$. Then the a priori probabilities differ by a factor of at most $n^{2+o(1)}$. Hence, $h(G \oplus e) \neq \mathrm{bw}(G \oplus e)$ with probability at most $n^{2+o(1)}(3n^{-5}) = 3n^{-3+o(1)}$. Since there are $\binom{n}{2}$ possible choices of e, we obtain $h(G \oplus e) = \mathrm{bw}(G \oplus e)$ with probability at least $1 - 1/n^{1-o(1)}$. □

This enables us to reconstruct the optimum bisection with n calls of the function $h(G)$, since w.h.p. the function $h(G \oplus e)$ tells us if we modified a cut edge or an inner edge. The Algorithm BB-U still runs in polynomial time.

Algorithm 4.1: Boppana-based algorithm for finding unique optimum bisections BB-U

```
  Input : Graph $G$.
1 Compute $h(G)$: numerically find a vector $d^{\mathrm{opt}}$ which maximizes $g(G,d)$.
2 Construct a bisection $\hat{x}$: choose some vertex $u$ and label it 1. Then, for
    every vertex $v \neq u$ compute $h(G \oplus \{u,v\})$. If $h(G \oplus \{u,v\}) = h(G)$,
    label the vertex with 1, otherwise $-1$.
3 Output $\hat{x}$; if $\mathrm{cw}(\hat{x}) = h(G)$, output "optimum bisection", else output
    "fail".
```

Lemma 4.7. *Let G be a graph chosen from $\mathcal{G}_n(p,q)$ with probabilities fulfilling Eq.* (2.13)*. Then, w.h.p. Algorithm BB-U finds and certifies the optimum bisection in polynomial time.*

Proof. From Lemma 4.6, it holds w.h.p. $h(G) = \mathrm{bw}(G)$ and for every u, v considered by the algorithm holds $h(G \oplus \{u,v\}) = \mathrm{bw}(G \oplus \{u,v\})$. Now, assuming G has a unique bisection, we have $\mathrm{bw}(G \oplus \{u,v\}) = \mathrm{bw}(G)$ if and only if u and v are in the same part of an optimum bisection of G. □

However, making n calls to $h(G)$ seems to be high costs for finding the bisection, especially if we might have a largest eigenvalue of dimension two and thus only have to separate the information from two vectors. We will later propose our Algorithm BB-UI (see Section 5.2), which reduces the number of evaluations of h and needs at most as many evaluations as the multiplicity of the largest eigenvalue.

Under certain circumstances, there is another way of finding an optimum bisection vector. The idea is to find two optimal vectors d_1^{opt} and d_2^{opt}:

Corollary 4.8. *Let G be a graph with $h(G) = \text{bw}(G)$ and a unique optimum bisection with bisection vector y. Let d_1^{opt} and d_2^{opt} be two optimal vectors such that $g(G, d_1^{\text{opt}}) = g(G, d_2^{\text{opt}}) = \text{bw}(G)$ and $d_1^{\text{opt}} - \text{mean}(d_1^{\text{opt}})\mathbf{1} \neq d_2^{\text{opt}} - \text{mean}(d_2^{\text{opt}})\mathbf{1}$. Then $d_1^{\text{opt}} - d_2^{\text{opt}} = \beta y + \gamma\mathbf{1}$ for some $\beta \in \mathbb{R} \setminus 0$ and $\gamma \in \mathbb{R}$.*

Proof. We use Lemma 4.4. Since the bisection y is unique, we have $d_i^{\text{opt}} = d^{(y)} + \alpha_i y + \gamma_i \mathbf{1}$. The claim follows with $\beta = \alpha_1 - \alpha_2 \neq 0$ and $\gamma = \gamma_1 - \gamma_2$. □

In the unpublished work [10], Blumofe and Boppana propose to determine the set of all optimal vectors d and derive the bisection from this set, e. g. as we could do using two distinct vectors and apply Corollary 4.8. While it is known that for the concave function $g(G, d)$ an optimal d can be found in polynomial time with arbitrary precision, it is not clear to the author how to find two different optimal solutions. In practice choosing two random starting points does the job, but there usually is no guarantee that the search algorithms do not converge to the same optimal point.

4.4 Finding Non-Unique Optimum Bisections BB-NU

We are also able to provide a modification of Boppana's algorithm which handles cases where $h(G) = \text{bw}(G)$ but for which no unique bisection of minimum size exists. As we will see later, hypercubes satisfy these two conditions. We present our modification as Algorithm 4.2 (BB-NU). Note that, if the multiplicity of the largest eigenvalue of B_Q^{opt} is 1, then the algorithm outputs the same result as in the original Algorithm BB by Boppana.

Theorem 4.9. *If $h(G) = \text{bw}(G)$, then Algorithm BB-NU reconstructs all optimum bisections. Every achieved bisection vector corresponds to an optimum bisection.*

Proof. Due to Lemma 4.3, we know that for every optimum bisection vector y, the vector $Q^T y$ is eigenvector of the matrix $B_Q^{\text{opt}} = (A + \text{diag}(d^{\text{opt}}))_Q$

Algorithm 4.2: Boppana-based algorithm for finding non-unique optimum bisections BB-NU

```
Input: Graph $G$ with adjacency matrix $A$.
1 Perform Step 1 of Algorithm BB; Let $x$ be an eigenvector corresponding
    to the eigenvalue $\lambda((A + \mathrm{diag}(d^{\mathrm{opt}}))_Q)$ and let $k$ be the multiplicity of
    the largest eigenvalue of $(A + \mathrm{diag}(d^{\mathrm{opt}}))_Q$;
2 If $k = 1$ then construct a bisection vector $\hat{x}$ by splitting at the median $\bar{x}$
    of $Qx$ as in Step 2 of Algorithm BB; Next output $\hat{x}$ and if
    $\mathrm{cw}(\hat{x}) = h(G)$, output “optimum bisection”, else output “fail”; If $k > 1$
    then perform the steps below;
3 Let $M \in \mathbb{R}^{n\times k}$ be the matrix with the vectors $Qx$ obtained from the $k$
    linear independent eigenvectors $x$ corresponding to the largest
    eigenvalue of $(A + \mathrm{diag}(d^{\mathrm{opt}}))_Q$; Transform the matrix to the reduced
    column echelon form, i. e. there are $k$ rows which form an identity
    matrix, s. t. $M$ still spans the same subspace;
4 Brute force: for every combination of $k$ coefficients from $\{+1, -1\}$, take
    the linear combination of the $k$ column vectors of $M$ with the
    coefficients and verify if the resulting vector $x$ is a bisection vector, i. e.
    $x \in \{+1, -1\}^n$ with $\sum_i x_i = 0$. If yes and if $\mathrm{cw}(x) = h(G)$, then
    output $x$ and continue. This needs $2^k$ iterations;
5 If in Step 4 no bisection vector $x$ is found, then output “fail”.
```

and thus $QQ^Ty = Py = y$ is in the subspace spanned by the columns of M. We show even more, namely that non-optimum bisection vectors x are not in this subspace. For contradiction, let x be a bisection vector with $\mathrm{cw}_G(x) > \mathrm{bw}(G)$ and assume Q^Tx is an eigenvector of B_Q^{opt} corresponding to the largest eigenvalue. Then it holds

$$\begin{aligned}\lambda(B_Q^{\mathrm{opt}}) &= \frac{(Q^Tx)^T B_Q^{\mathrm{opt}}(Q^Tx)}{(Q^Tx)^T(Q^Tx)} = \frac{x^TB^{\mathrm{opt}}x}{x^Tx}\\ &= \frac{x^T(A+\mathrm{diag}(d^{\mathrm{opt}}))x}{x^Tx} = \frac{x^TAx + \sum_i d_i^{\mathrm{opt}}}{x^Tx}.\end{aligned}$$

For a bisection vector x, it holds $x^TAx = 2|E(G)| - 4\,\mathrm{cw}_G(x)$. Now, since $\mathrm{cw}_G(x) > \mathrm{bw}(G)$, we obtain for a minimum bisection vector y, that $x^TAx < y^TAy$. This is a contradiction to x maximizing the Rayleigh quotient above. Thus, no non-optimum bisection vector x can be contained in the subspace spanned by M. □

Corollary 4.10. *For the graph class of hypercubes H_n with n vertices, the bound $h(H_n)$ is tight and Algorithm BB-NU finds all optimum bisections.*

Proof. The eigenvalues for the family of hypercubes are explicitly known [43]. For a hypercube H_n with n vertices we have $h(H_n) = g(H_n, (2 - \log_2 n)\mathbf{1}) = n/2 = \mathrm{bw}(H_n)$. Due to Theorem 4.9, Algorithm BB-NU then finds all optimum bisections. $\square$

Since the hypercube with n vertices has $\log_2 n$ optimum bisections and the eigenspace of the largest eigenvalue of B_Q has multiplicity $\log_2 n$, the brute force part in our modification of Boppana's algorithm results in a linear factor of n for the overall running time. Thus, the Algorithm BB-NU runs in polynomial time for these graphs. With the results from Section 5.1, we will later extend this result and obtain that BB-NU works on adversarially modified hypercubes as well.

5 Boppana's Approach: New Frontiers of Applicability

In this chapter we systematically explore graph properties that guarantee $h(G) = \mathrm{bw}(G)$ and thus, that the Boppana-based Algorithms BB-U and BB-NU are able to certify the output bisections. We have already seen that random graphs from $\mathcal{G}_n(p, q)$ and $\mathcal{R}_n(r, b)$ satisfy such properties w.h.p. under the assumptions in Eq. (2.13) and Eq. (2.14) on p, q, r, and b.

However, the algorithms works also well for instances which deviate significantly from such random graphs: We introduce the adversarial graph model $\mathcal{A}(\mathcal{G}_n)$ as a natural generalization of the semirandom model of Feige and Kilian [31]. We will show that, if Algorithm BB-U or BB-NU outputs the minimum-size bisection for graphs in $\mathcal{G}_n$ w.h.p., then the algorithm finds a minimum bisection w.h.p. for the adversarial graph model $\mathcal{A}(\mathcal{G}_n)$, too. As a corollary, we get that, under the assumption in Eq. (2.13), the Algorithm BB-U works well in the semirandom model, denoted here as $\mathcal{A}(\mathcal{G}_n(p, q))$, and, assuming Eq. (2.14), in $\mathcal{A}(\mathcal{R}_n(r, b))$ – the semirandom regular model.

To analyze limitations of the spectral approach we provide structural properties of the space of feasible solutions searched by the algorithms. This allows us to prove that, if an optimum bisection contains some forbidden subgraphs, then Boppana's approach fails. Using these tools, we are able to show that, if the density difference $p - q$ is asymptotically smaller than $\sqrt{p \cdot \log n}/\sqrt{n}$, then Boppana's approach fails to determine a certified optimum bisection on $\mathcal{G}_n(p, q)$ w.h.p. Note that our impossibility result is not a direct consequence of the lower bound for the exact cluster recovery. For example, for $q = \mathcal{O}(1)/n$ and $p = \sqrt{\log n}/n$ from Mossel et al. [63], we know that for these parameters the exact recovery is impossible, but obviously this does not imply that determining of a certified optimum bisection is impossible either.

5.1 Bisections in Adversarial Models

We introduce the *adversarial model*, denoted by $\mathcal{A}(\mathcal{G}_n)$, as a generalization of the semirandom model in the following way: Let $\mathcal{G}_n$ be a graph model, i. e. a class of graphs with n nodes (n even) and some probability distribution over these graphs. In the model $\mathcal{A}(\mathcal{G}_n)$, initially a graph G is chosen at random according to $\mathcal{G}_n$. Let (Y_1, Y_2) be a fixed but arbitrary optimum bisection of G. Then, similarly as in [31], a monotone adversary is allowed to modify G by applying an arbitrary sequence of the following monotone transformations: The adversary may

1. remove from the graph any edge $\{u, v\}$ crossing the bisection ($u \in Y_1$ and $v \in Y_2$);
2. add to the graph any edge $\{u, v\}$ not crossing the bisection ($u, v \in Y_1$ or $u, v \in Y_2$).

For example, $\mathcal{A}(\mathcal{G}_n(p, q))$ is the semirandom model as defined in [31].

We will prove that Boppana's approach works well for graphs from the adversarial model $\mathcal{A}(\mathcal{G}_n)$ if it works well for $\mathcal{G}_n$. First we show that, if a Boppana-based algorithm is able to find an optimum bisection of a graph, we can add edges within the same part of an optimum bisection and we can remove cut edges, and the algorithm still works. This solves the open question of Feige and Kilian [31].

Note that the result follows alternatively from Corollary 6.5 (presented in Section 6) that the SDPs of [31] are equivalent to Boppana's optimization function and from the property proved in [31] that the objective function of the dual SDP of Feige and Kilian preserves the minimal bisection regardless of monotone transformations. The aim of this section is to give a direct proof of this property for Boppana's algorithm.

Theorem 5.1. *Let $G = (V, E)$ be a graph with $h(G) = \mathrm{bw}(G)$. Consider some optimum bisection Y_1, Y_2 of G.*

1. *Let u and v be two vertices within the same part, i. e. $u, v \in Y_1$ or $u, v \in Y_2$, and let $G' = (V, E \cup \{\{u, v\}\})$. Then $h(G') = \mathrm{bw}(G')$.*
2. *Let u and v be two vertices in different parts, i. e. $u \in Y_1$ and $v \in Y_2$, with $\{u, v\} \in E$ and let $G' = (V, E \setminus \{\{u, v\}\})$. Then $h(G') = \mathrm{bw}(G) - 1 = \mathrm{bw}(G')$.*

Proof. We start by proving the first part, i. e. when we add an edge $\{u, v\}$. Let A and A' denote the adjacency matrices of G and G', respectively. It holds $A' = A + A^{\Delta}$ with $A^{\Delta}_{uv} = A^{\Delta}_{vu} = 1$ and zero everywhere else. Since $h(G) = \mathrm{bw}(G)$, there exists a d^{opt} with $g(G, d^{\text{opt}}) = \mathrm{bw}(G)$. The main idea is now that we can derive a new optimal correction vector d' for G' based on the optimal correction vector d^{opt} for G. For G', we set $d' = d^{\text{opt}} + d^{\Delta}$ with

$$d^{\Delta}_i = \begin{cases} -1 & \text{if } i = u \text{ or } i = v, \\ 0 & \text{else.} \end{cases}$$

We set $B' = A + A^{\Delta} + \mathrm{diag}(d^{\mathrm{opt}} + d^{\Delta}) = B^{\mathrm{opt}} + A^{\Delta} + \mathrm{diag}(d^{\Delta})$ and get

$$\begin{aligned}
\lambda(B'_Q) &= \max_{x \in S\setminus\{\mathbf{0}\}} \frac{x^T(B^{\mathrm{opt}} + A^{\Delta} + \mathrm{diag}(d^{\Delta}))x}{\|x\|^2} \\
&= \max_{x \in S\setminus\{\mathbf{0}\}} \frac{x^T B^{\mathrm{opt}} x + x^T(A^{\Delta} + \mathrm{diag}(d^{\Delta}))x}{\|x\|^2} \\
&= \max_{x \in S\setminus\{\mathbf{0}\}} \frac{x^T B^{\mathrm{opt}} x + 2x_u x_v - x_u^2 - x_v^2}{\|x\|^2} \\
&= \max_{x \in S\setminus\{\mathbf{0}\}} \frac{x^T B^{\mathrm{opt}} x - (x_u - x_v)^2}{\|x\|^2} \\
&\leq \max_{x \in S\setminus\{\mathbf{0}\}} \frac{x^T B^{\mathrm{opt}} x}{\|x\|^2} = \lambda(B^{\mathrm{opt}}_Q).
\end{aligned}$$

For an optimum bisection vector, we have $x_u = x_v = 1$ or $x_u = x_v = -1$ and thus the last inequality is equality. Hence, $\lambda(B'_Q) = \lambda(B^{\mathrm{opt}}_Q)$. Note that by our construction $\mathrm{sum}(B') = \mathrm{sum}(B^{\mathrm{opt}})$. It follows $g(G', d') = g(G, d^{\mathrm{opt}}) = \mathrm{bw}(G) = \mathrm{bw}(G')$. This completes the proof for the first part.

The proof for the second part is similar to the first one. Let $\{u, v\}$, with $u \in Y_1$ and $v \in Y_2$, be an edge that gets removed from G. We define d^{Δ} as we have done above and we let $A' = A + A^{\Delta}$, with $A^{\Delta}_{uv} = A^{\Delta}_{vu} = -1$ and zero everywhere else. Note that this time $\mathrm{sum}(B') = \mathrm{sum}(B^{\mathrm{opt}}) - 4$.

$$\begin{aligned}
\lambda(B'_Q) &= \max_{x \in S\setminus\{\mathbf{0}\}} \frac{x^T B^{\mathrm{opt}} x + x^T(A^{\Delta} + \mathrm{diag}(d^{\Delta}))x}{\|x\|^2} \\
&= \max_{x \in S\setminus\{\mathbf{0}\}} \frac{x^T B^{\mathrm{opt}} x - 2x_u x_v - x_u^2 - x_v^2}{\|x\|^2} \\
&= \max_{x \in S\setminus\{\mathbf{0}\}} \frac{x^T B^{\mathrm{opt}} x - (x_u + x_v)^2}{\|x\|^2} \\
&\leq \max_{x \in S\setminus\{\mathbf{0}\}} \frac{x^T B^{\mathrm{opt}} x}{\|x\|^2} = \lambda(B^{\mathrm{opt}}_Q).
\end{aligned}$$

For an optimum bisection vector, we have $x_u = 1, x_v = -1$ or $x_u = -1, x_v = 1$ and hence the last inequality is equality. We can conclude

$$\begin{aligned}
g(G', d') &= \frac{\mathrm{sum}(B') - n\lambda(B'_Q)}{4} \\
&= \frac{\mathrm{sum}(B^{\mathrm{opt}}) - 4 - n\lambda(B_Q)}{4} = g(G, d^{\mathrm{opt}}) - 1 = \mathrm{bw}(G) - 1.
\end{aligned}$$

This completes the proof of the second part and of the theorem. □

The direct consequence for the algorithms BB-U and BB-NU is as follows:

Theorem 5.2. *If BB-U resp. BB-NU finds a minimum bisection for a graph model $\mathcal{G}_n$ w.h.p., then it finds a minimum bisection w.h.p. for the adversarial model $\mathcal{A}(\mathcal{G}_n)$, too.*

We obtain the following corollary regarding the semirandom graph model considered by Feige and Kilian:

Corollary 5.3. *Under assumption Eq.* (2.13) *on p and q, w.h.p. Boppana's lower bound is tight on $\mathcal{A}(\mathcal{G}_n(p,q))$, i. e. on the semirandom model, and Algorithm BB-U computes the minimum bisection.*

In [12], Boppana also considers random regular graphs $\mathcal{R}_n(r,b)$, where a graph is chosen uniformly over the set of all r-regular graphs with bisection width b. He shows that his algorithm works w.h.p. on this graph under the assumption that $b = o(n^{1-1/\lfloor (r+1)/2 \rfloor})$. We can now define the semirandom regular graph model as adversarial model $\mathcal{A}(\mathcal{R}_n(r,b))$. Applying Theorem 5.2, we obtain:

Corollary 5.4. *Under assumption Eq.* (2.14) *on r and b, w.h.p. Boppana's lower bound is tight on the semirandom regular model $\mathcal{A}(\mathcal{R}_n(r,b))$ and Algorithm BB-U computes the minimum bisection.*

Theorem 5.2 can also be applied on deterministic graph classes, e. g. the class of hypercubes. Extending Corollary 4.10, we obtain:

Corollary 5.5. *Algorithm BB-NU finds and certifies an optimum bisection on adversarial modified hypercubes.*

5.2 An Improved Algorithm for Unique Optimum Bisections BB-UI

We have already introduced Algorithm BB-U, which finds an optimum bisection w.h.p. in the case of an unique bisection in graph G and $h(G) = \mathrm{bw}(G)$. However, the algorithm always needs n evaluations of function h. Using the robustness of h against adversarial changes from Theorem 5.1, we are able to give a modified algorithm, namely Algorithm BB-UI, which needs at most as many evaluations of h as the multiplicity of the largest eigenvalue of $(A + \mathrm{diag}(d^{\mathrm{opt}}))_Q$. The idea is to remove edges from the graph. If a cut edge is removed, the multiplicity of the largest eigenvalue decreases by one. If an inner edge is removed, we are able to reconstruct the bisection immediately. Let us note that this algorithm is more of theoretical interest. In practice, our experiments indicate that in case this algorithm is able to find and certify the bisection, then Step 2 already reveals a largest eigenvalue with multiplicity 1. Then, the algorithm is essentially the same as Algorithm BB.

Algorithm 5.1: Boppana-based algorithm for unique optimum bisections with improved performance BB-UI

```
   Input: Graph $G$ with $n$ vertices.
1  Initialize $G' = G$, $h^{\text{prev}} = \infty$, $z^{\text{prev}} = \infty$ and $d^{\text{prev}} = \delta = \mathbf{0}$.
2  Compute $h(G')$: numerically find a vector $d^{\text{opt}}$ which maximizes
     $g(G', d)$. Let $A'$ denote the adjacency matrix of $G'$ and let $Z$ be a set
     of basis vectors of the eigenspace corresponding to the largest
     eigenvalue of $(A' + \text{diag}(d^{\text{opt}}))_Q$.
3  If $h(G') < 0$: output "fail" and terminate.
4  If $|Z| = 1$: let $x' \in Z$ and set $x = Qx'$. Go to Step 9.
5  If $h(G') > h^{\text{prev}} - 1$: set $x = d^{\text{prev}} + \delta - d^{\text{opt}}$. Go to Step 9.
6  If $h(G') = h^{\text{prev}} - 1$ and $|Z| \geq z^{\text{prev}}$: set $x = d^{\text{prev}} - \delta - d^{\text{opt}}$. Go to
     Step 9.
7  Store $h^{\text{prev}} = h(G')$, $d^{\text{prev}} = d^{\text{opt}}$ and $z^{\text{prev}} = |Z|$. Find an edge
     $\{u, v\} \in E(G')$ where vectors $x, w \in Z$ exist with $(Qx)_u \neq (Qx)_v$ and
     $(Qw)_u \neq -(Qw)_v$. Remove edge $\{u, v\}$, i. e. set $G' \leftarrow G' \setminus \{\{u, v\}\}$.
     Set $\delta \in \mathbb{R}^n$ to $\delta_u = \delta_v = 1$ and $\delta_i = 0$ for $i = \{1, \ldots, n\} \setminus \{u, v\}$. Then
     go to Step 2.
8  If no such edge is found, take any $x' \in Z$ and set $x = Qx'$. Proceed
     with Step 9.
9  Construct a bisection vector $\hat{x}$ by splitting at the median $\bar{x}$ of $x$, i. e. let
     $\hat{x}_i = +1$ if $x_i \geq \bar{x}$ and $\hat{x}_i = -1$ if $x_i < \bar{x}$. If $\sum_i \hat{x}_i > 0$, move
     (arbitrarily) $\frac{1}{2} \sum_i \hat{x}_i$ vertices $i$ with $x_i = \bar{x}$ to part $-1$ letting $\hat{x}_i = -1$;
10 Output $\hat{x}$; if $\text{cw}_G(\hat{x}) = h(G)$, output "optimum bisection", else output
     "fail".
```

Lemma 5.6. *For every graph G, Algorithm BB-UI terminates in polynomial time.*

Proof. Each single step can be computed in polynomial time. The algorithm has a loop, when Step 7 performs a goto to Step 2. In Step 5, the algorithm leaves the loop when the value of h has decreased less than 1. Thus, as long as the loop is active, h decreases at least by one at each iteration. Finally, Step 3 lets the algorithm terminate when h is below 0. Recall that $h(G)$ is a lower bound on the bisection width of G and trivially below n^2. Thus, after at most n^2 loops, the algorithm terminates. □

Lemma 5.7. *Let G be a graph chosen at random from $\mathcal{G}_n(p,q)$ with probabilities fulfilling Eq. (2.13). Let A denote the adjacency matrix of G and d^{opt} be a vector that maximizes $g(G,d)$. Let k be the multiplicity of the largest eigenvalue of $(A + \text{diag}(d^{\text{opt}}))_Q$. Then, w.h.p. Algorithm BB-UI finds and certifies the optimum bisection with at most k evaluations of the function h.*

Proof. Let us start to consider the base cases. In Step 4, we derive a bisection in case $|Z| = 1$. From Lemma 4.3 we conclude that in this case, for $x' \in Z$, the vector Qx' is w.h.p. the bisection vector. Next let us consider Step 8. We show that this step is unlikely to occur. If no suitable edge has been found in the preceding step, for each edge $\{u,v\} \in E(G')$ holds either $\forall x \in Z : (Qx)_u = (Qx)_v$, or it holds $\forall x \in Z : (Qx)_u \neq -(Qx)_v$. If the graph is connected, this is a contradiction to $|Z| > 1$, since there cannot be two such linear independent vectors. By assuming Eq. (2.13), the graph is w.h.p. connected, and also the subgraphs induced by the parts of the optimum bisection are connected w.h.p. Consequently, when removing a series of cut edges, the graph remains connected until the bisection width gets zero and Step 8 does not take an action w.h.p. Nevertheless, taking any vector in case the step occurs seems to be a reasonable fall back action (again due to Lemma 4.3).

We now discuss the iterative removal of edges from the graph. Assuming Eq. (2.13), we have w.h.p. that $h(G) = \text{bw}(G)$ (see Theorem 3.2) and for every edge $e \in E(G)$ that $h(G \setminus e) = \text{bw}(G \setminus e)$ (see Lemma 4.6). Furthermore, the bisection is w.h.p. unique [24]. We show that the algorithm then finds and certifies an optimum bisection with at most k evaluations of h. When we obtain a graph G' from G by removing a series of cut edges and a single inner edge, we obtain the same graph as when we first remove the inner edge and then the cut edges. Since $h(G \setminus e) = \text{bw}(G \setminus e)$, after removing the inner edge, the bound is still tight. Then, removing the cut edges is an adversarial change and the bound remains tight (see Theorem 5.1).

Step 7 either removes an inner edge or it removes a cut edge. First consider it removes an inner edge $\{u,v\}$ in Step 7. Denote by G^{prev} the graph before removing the edge and G' afterwards. Since we assumed that the bound remains tight when removing a single inner edge, Step 2 gives $h(G') = h^{\text{prev}}$. For the optimal vectors d^{prev} and d^{opt} computed in Step 2 then holds

$$g(G^{\text{prev}}, d^{\text{prev}}) = g(G', d^{\text{opt}}) = \text{bw}(G'). \tag{5.1}$$

By assumption of the algorithm there is a unique bisection vector y, so that by Lemma 4.4 we can decompose the vectors $d^{\text{prev}} = d^{\text{prev},y} + \alpha^{\text{prev}} y + \beta^{\text{prev}} \mathbf{1}$ and $d^{\text{opt}} = d^{(y)} + \alpha y + \beta \mathbf{1}$. We define $\delta \in \mathbb{R}^n$, $\delta_u = \delta_v = 1$ and $\delta_i = 0$ for $i = \{1, \ldots, n\} \setminus \{u, v\}$. Since we removed an inner edge, it holds $d^{\text{prev},y} + \delta = d^{(y)}$.

For contradiction, assume that $\alpha^{\text{prev}} = \alpha$. Then $d^{\text{opt}} = d^{\text{prev}} + \delta + (\beta - \beta^{\text{prev}})\mathbf{1}$. We insert this in Eq. (5.1) and obtain

$$g(G^{\text{prev}}, d^{\text{prev}}) = g(G', d^{\text{prev}} + \delta + (\beta - \beta^{\text{prev}})\mathbf{1})$$

and since g is invariant under shifting (Lemma 2.7), we get

$$\Leftrightarrow \quad g(G^{\text{prev}}, d^{\text{prev}}) = g(G', d^{\text{prev}} + \delta).$$

Applying the definition of g, we conclude with $B_Q^{\text{prev}} = (A + \operatorname{diag}(d^{\text{prev}}))_Q$:

$$\Leftrightarrow \max_{x \in \mathbb{R}^{n-1} \setminus \{\mathbf{0}\}} \frac{x^T B_Q^{\text{prev}} x}{\|x\|^2} = \max_{x \in \mathbb{R}^{n-1} \setminus \{\mathbf{0}\}} \frac{x^T B_Q^{\text{prev}} x + ((Qx)_u - (Qx)_v)^2}{\|x\|^2}.$$

Vectors maximizing the left term are the eigenvectors corresponding to the largest eigenvalues of B_Q^{prev}. From Step 7 we know that there is some eigenvector of B_Q^{prev} with $(Qx)_u \neq (Qx)_v$, such that $((Qx)_u - (Qx)_v)^2 > 0$. Hence, the right maximum is larger than the left one. This contradicts that $\alpha^{\text{prev}} = \alpha$. Now, since $\alpha^{\text{prev}} \neq \alpha$, we can apply Corollary 4.8 and obtain a (stretched and shifted) bisection vector via $d^{\text{prev}} + \delta - d^{(y)}$. Hence, after the algorithm removed an inner edge, Step 5 is able to detect this case and continues with Step 9, revealing the optimum bisection.

We remain to discuss the case of removing a cut edge. We show that either we are able to reveal the optimum bisection similar as above, or $|Z|$ decreases by at least one. For removing a cut edge we assumed $h(G^{\text{prev}}) = h(G') + 1$, such that

$$g(G^{\text{prev}}, d^{\text{prev}}) = g(G', d^{\text{opt}}) + 1. \tag{5.2}$$

Again, we can decompose the vectors $d^{\text{prev}} = d^{\text{prev},y} + \alpha^{\text{prev}} y + \beta^{\text{prev}} \mathbf{1}$ and $d^{\text{opt}} = d^{(y)} + \alpha y + \beta \mathbf{1}$. Using the same δ as above, and since we removed a cut edge, it holds $d^{\text{prev},y} - \delta = d^{(y)}$. We show that, if $\alpha^{\text{prev}} = \alpha$, then $|Z| < z^{\text{prev}}$: assuming $\alpha^{\text{prev}} = \alpha$, we express d^{opt} in terms of d^{prev} as $d^{\text{opt}} = d^{\text{prev}} - \delta + (\beta - \beta^{\text{prev}})\mathbf{1}$. We insert this in Eq. (5.2) and obtain

$$\begin{aligned}
&\Leftrightarrow g(G^{\text{prev}}, d^{\text{prev}}) = g(G', d^{\text{prev}} - \delta + (\beta - \beta^{\text{prev}})\mathbf{1}) + 1 \\
&\Leftrightarrow g(G^{\text{prev}}, d^{\text{prev}}) = g(G', d^{\text{prev}} - \delta) + 1
\end{aligned}$$

$$\Leftrightarrow \max_{x\in\mathbb{R}^{n-1}\setminus\{\mathbf{0}\}} \frac{x^T B_Q^{\text{prev}} x}{\|x\|^2} = \max_{x\in\mathbb{R}^{n-1}\setminus\{\mathbf{0}\}} \frac{x^T B_Q^{\text{prev}} x - ((Qx)_u + (Qx)_v)^2}{\|x\|^2}.$$

From Step 7 we know that there is some eigenvector of B_Q^{prev} with $(Qx)_u \neq -(Qx)_v$, such that $((Qx)_u + (Qx)_v)^2 > 0$. This vector obtains a smaller value for the right term than for the left one. Consequently, the vector is no longer eigenvector in the span of Z for graph G'. The size of Z has decreased by at least one. If still $|Z| > 1$, the algorithm continues with removing another edge.

If $|Z|$ did not decrease, i. e. $|Z| \geq z^{\text{prev}}$, we conclude from above that $\alpha \neq \alpha^{\text{prev}}$. Step 6 then provides a reconstruction: apply Corollary 4.8 and the (stretched and shifted) bisection vector is obtained as $d^{\text{prev}} - \delta - d^{(y)}$.

Hence, removing an edge either provides a solution, or reduces $|Z|$ by at least one. This completes the proof. $\square$

5.3 The Limitations of the Spectral Approach

We have seen that Boppana's approach works well on several classes of random graphs. However, we did not see so far which graph properties force the algorithm to fail. For example, for the considered planted bisection model, we require a small bisection width. On the other hand, as we have seen in Sections 4.4 and 5.1, Boppana's approach works for hypercubes and their semirandom modifications – graphs that have large minimum bisection sizes.

In the following, we present newly discovered structural properties from inside the algorithm, which provide a framework for a better analysis of the algorithm itself. Since the proofs for this section are rather technical, we moved them to the separate Section 5.4.

Lemma 5.8. *Let G be a graph with $h(G) = \text{bw}(G)$ and assume there is more than one optimum bisection in G. Then (up to constant translation vectors $c\mathbf{1}$) there exists a unique vector d^{opt} with $g(G, d^{\text{opt}}) = \text{bw}(G)$. Additionally, for every bisection vector y of an arbitrary optimum bisection of G, there exists a unique $\alpha^{(y)}$ and the corresponding $d^{(y)}$ with $g(G, d^{(y)} + \alpha^{(y)} y) = \text{bw}(G)$.*

Thus, if there are two optimum bisections represented by y and y' with $d^{(y)} \neq d^{(y')}$, then the difference of the d-vectors in component i is only dependent on y_i and y'_i, since we have $d^{(y)} - d^{(y')} = \alpha' y' - \alpha y$ for some constants α and α'.

This structural property allows us to show the following limitation for the sparse planted bisection model $\mathcal{G}_n(p, q)$.

Theorem 5.9. *Boppana's approach fails w.h.p. in the subcritical phase of $\mathcal{G}_n(p,q)$, defined in [24] as $n(p-q) = o(\sqrt{np \cdot \log n})$.*

In the planted bisection model $\mathcal{G}_n(p,q)$, if the graphs are dense, e. g. $p = 1/n^c$ for a constant c with $0 < c < 1$, the constraints for the density difference $p - q$ assumed in Boppana's [12] and Coja-Oghlan's [24] algorithms are essentially the same. However, for sparse graphs, e. g. such that $q = \mathcal{O}(1)/n$, the situation changes drastically. Now, e. g. $p = \sqrt{\log n}/n$ satisfy Coja-Oghlan's constraint $p - q \geq \Omega(\sqrt{p \log(pn)}/\sqrt{n})$, but the condition on the difference $p - q$ assumed by Boppana is not true any more. Theorem 5.9 shows that Boppana's algorithm indeed fails under this setting.

The proof of this theorem relies on the following observation, which can be derived from our newly discovered structural properties in Lemma 5.8.

Lemma 5.10. *Let G be a graph with $h(G) = \mathrm{bw}(G)$ and let (Y_1, Y_{-1}) be an arbitrary optimum bisection. Let $e(v, Y)$ denote the number of edges incident to vertex v and to a vertex from the set Y. Then, for each pair of vertices $v_1 \in Y_1$ and $v_{-1} \in Y_{-1}$, not connected by an edge ($\{v_1, v_{-1}\} \notin E$), we have: if $e(v_i, Y_1) = e(v_i, Y_{-1})$ for $i \in \{1, -1\}$ (the vertices have balanced degree), then $N(v_1) = N(v_{-1})$, i. e. both vertices have the same neighbors.*

In other words, if we have two balanced vertices in different parts of an optimum bisection, not connected by an edge, then the two vertices must have the same neighborhood as a necessary criterion for Boppana's approach to work. In the subcritical phase in Theorem 5.9, there exist most likely many of such pairs of vertices, but they are unlikely to have all even the same degree.

We can also provide forbidden substructures which make Boppana's approach fail. This is e. g. the case, when the graph contains a path segment located across an optimum bisection:

Corollary 5.11. *Let G be a graph, as illustrated in Figure 5.1 (left), with $n \geq 10$ vertices containing a path segment $\{u', u\}, \{u, w\}, \{w, w'\}$, where u and w have no further edges. If there is an optimum bisection y, s. t. $y_u = y_{u'} = +1$ and $y_w = y_{w'} = -1$ (i. e. $\{u, w\}$ is a cut edge), then $h(G) < \mathrm{bw}(G)$.*

To prove this corollary, we use the following more general but also more technical Lemma 5.12 from which Corollary 5.11 follows, setting parameters $\tilde{C}_{+1} = \{u\}$ and $\tilde{C}_{-1} = \{w\}$.

Lemma 5.12 (Necessity for many edges)**.** *Let $G = (V, E)$ be a graph and y be an optimum bisection vector of G. For $i \in \{+1, -1\}$ let $C_i = \{u \mid y_u =$*

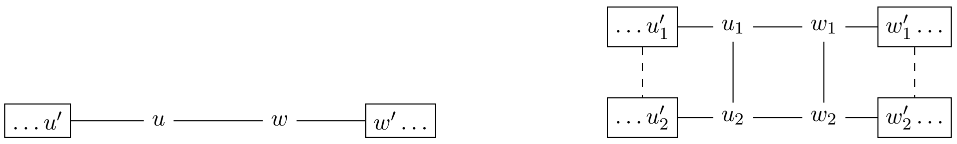


Figure 5.1: Forbidden graph structures as in Corollary 5.11 (left) and in Corollary 5.13 (right).

$i \wedge \exists v : y_v = -i \wedge \{u, v\} \in E\}$ be the set of vertices in part i incident to the cut edges. If there exist non-empty $\tilde{C}_i \subseteq C_i$ with $k = \min\{|\tilde{C}_{+1}|, |\tilde{C}_{-1}|\}$, $k + \delta = \max\{|\tilde{C}_{+1}|, |\tilde{C}_{-1}|\}$, $l = |V| - (k + \delta)$, s. t.

- $(3k < l \wedge \delta = 0)$ *or* $(4k < l \wedge \delta < \min\{\frac{4k^2}{l-4k}, \frac{7}{128}l\})$
- $2|E(\tilde{C}_{+1}, \tilde{C}_{-1})| \geq |E(\tilde{C}_{+1} \cup \tilde{C}_{-1}, V \setminus (\tilde{C}_{+1} \cup \tilde{C}_{-1}))|$,

then $h(G) < \mathrm{bw}(G)$.

An illustration of the condition stated in Lemma 5.12 is given in Figure 5.2. The parameter δ allows for some unbalanced size of the subsets.

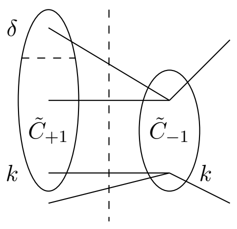


Figure 5.2: Forbidden graph structures as in Lemma 5.12.

Lemma 5.12 can also be applied for $2 \times c$ lattices:

Corollary 5.13. *Let G be a graph with $n \geq 10c$ vertices containing a $2 \times c$ lattice with vertices u_i and w_i, as illustrated in Figure 5.1 (right). (The construction is similar to Corollary 5.11, but now we have a lattice instead of a single cut edge.) If there is an optimum bisection y, s. t. $y_{u_i} = y_{u'_i} = +1$ and $y_{w_i} = y_{w'_i} = -1$, for $i = 1, 2$, then $h(G) < \mathrm{bw}(G)$.*

Furthermore, Boppana's approach fails if there are isolated vertices in both parts of an optimum bisection:

Theorem 5.14. *Let G be a graph with $h(G) = \mathrm{bw}(G)$. Let G' be the graph G with two additional isolated vertices, then $h(G') \leq h(G) - \frac{4\,\mathrm{bw}(G)}{n^2}$.*

5.4 Technical Proofs

Now we provide the proofs left out in the previous section. Corollary 5.11 and Corollary 5.13 follow easily from Lemma 5.12, thus we skip the proofs.

Proof of Lemma 5.8. Consider two optimum bisections with bisection vectors y and y'. (Note that we consider y and $-y$ as the same bisection.) For contradiction, assume there are two different $d_1^{\mathrm{opt}} \neq d_2^{\mathrm{opt}}$, while for normalization we require both vectors to have mean zero (see Lemma 2.7). Due to Lemma 4.4 we have that, for every y representing an optimum bisection, the vectors d_1^{opt} and d_2^{opt} can be expressed as $d_1^{\mathrm{opt}} = d^{(y)} + \alpha_1 y$ and $d_2^{\mathrm{opt}} = d^{(y)} + \alpha_2 y$. The difference is then

$$d_1^{\mathrm{opt}} - d_2^{\mathrm{opt}} = (\alpha_1 - \alpha_2)y.$$

For y' representing an optimum bisection, we have analogously $d_1^{\mathrm{opt}} = d^{(y')} + \beta_1 y'$ and $d_2^{\mathrm{opt}} = d^{(y')} + \beta_2 y'$ with difference

$$d_1^{\mathrm{opt}} - d_2^{\mathrm{opt}} = (\beta_1 - \beta_2)y'.$$

We conclude

$$(\alpha_1 - \alpha_2)y = (\beta_1 - \beta_2)y'.$$

Since y and y' are linearly independent, we conclude $\alpha_1 = \alpha_2$ and $\beta_1 = \beta_2$. This means, if there are two optimum bisections, then there is only one d^{opt} and α is unique! $\square$

Proof of Theorem 5.9. Let G be a graph sampled from the subcritical phase and (V_1, V_{-1}) be the planted bisection. Coja-Oghlan [24] defines two sets of vertices (see [23, page 112]):

$$\begin{aligned} N_i &= \{v \in V_i : e(v, V_i) = e(v, V_{-i})\} \quad \text{and} \\ N_i^* &= \{v \in N_i : N(v) \setminus \mathrm{core}(G) = \emptyset\}, \end{aligned}$$

where core(G) is a subgraph whose connected components are, w.h.p., not cut by any optimum bisection.

Let further (Y_1, Y_{-1}) be an optimum bisection. Coja-Oghlan claims that, w.h.p., $\#(Y_i \cap N_i^*) \geq \mu/8$ (eventually swap the parts), where $\mu = \mathbb{E}(\#N_1 + \#N_{-1})$ and $\mu \geq n^{1-\Theta(\gamma)}$ with $n(p-q) = \sqrt{np \cdot \gamma \log n}$, $\gamma = \mathcal{O}(1)$ [23, page 122]. Then there are $\exp(\Omega(\mu))$ many optimum bisections. A growing set size $\#(Y_i \cap N_i^*)$ contradicts then the following fact, which we are going to show: assuming that Boppana's approach works on G, the probability that $\#(Y_i \cap N_i^*) \geq 2$ will tend to 0.

Consider any pair of vertices $v_1 \in Y_1 \cap N_1^*$ and $v_{-1} \in Y_{-1} \cap N_{-1}^*$. v_1 and v_{-1} are not connected by an edge, since they have only neighbors in the core of G. Furthermore, they both have balanced degree. Thus, we can apply Lemma 5.10 and conclude that v_1 and v_{-1} have the same neighbors. In direct consequence, all vertices in $Y_i \cap N_i^*$, $i \in \{1, -1\}$ have the same neighbors and the same number of edges to each part as well. We denote this number by $k = e(v_1, V_1)$.

In the following, we consider sets of 4 vertices, while two are chosen from $Y_1 \cap N_1^*$ and two from $Y_{-1} \cap N_{-1}^*$. By our assumption of $\#(Y_i \cap N_i^*) \geq 2$, we can choose at least one such set w.h.p.

Let us first rule out two edge cases. In the first case, the vertices have degree $k = 0$. Then Boppana's approach does not work due to Theorem 5.14. In the second case, the vertices have maximal many edges, i. e. $k = n/2 - 2$ many edges to each part. W.h.p., a graph does not even have two vertices in each part with k edges:

$$\frac{(n/2)^2(n/2-1)^2}{4}(p^{n/2-2}q^{n/2-2})^4(1-q)^4(1-p)^2 \to 0.$$

Thus, we have to consider $1 \leq k \leq n/2 - 2$. Let $C_i^{(k)} = \{v \in V_i : e(v, V_i) = e(v, V_{-i}) = k\}$ be the set of vertices with a balanced number of exactly k edges to each part. With the k from above, we have $Y_i \cap N_i^* \subseteq C_i^{(k)}$.

We want to estimate the expected number of 4-element sets

$$\{v_1, u_1, v_{-1}, u_{-1}\} \subseteq C_1^{(k)} \cup C_{-1}^{(k)} \text{ with } v_1, u_1 \in C_1^{(k)} \text{ and } v_{-1}, u_{-1} \in C_{-1}^{(k)},$$

where all vertices have the same neighbors. Let us take v_1 as a reference vertex and thus the k edges from v_1 to V_1 as well as k edges to V_{-1} are given. Now we estimate the probability that v_{-1}, u_1, u_{-1} have exactly the same neighbors. For each vertex and each part, the k neighbors are chosen independently, since the four vertices are not connected to each other. In both parts, there are $n/2 - 2$ possible neighbors. This makes $\binom{n/2-2}{k} \geq \binom{n/2-2}{1} = n/2 - 2$ possibilities for the k edges in one part and only one of them coincides with the edges of v_1. For 3 vertices to have the same neighbors as v_1 in two parts each, the probability is at most $\frac{1}{(n/2-2)^6}$. The expected number of 4 vertices as described with the same neighbors is therefore

$$\mathbb{E}(\#4 - \text{elem} - \text{set}) \leq \binom{n/2}{2}^2 \cdot \frac{1}{(n/2-2)^6} \leq \frac{(n/2)^4}{(n/2-2)^6} \to 0.$$

This means, w.h.p. we do not find any 4-element set. In consequence, $\#(Y_i \cap N_i^*) \geq 2$ may not be true w.h.p. □

Proof of Lemma 5.10. Let y be the bisection vector corresponding to the optimum bisection in the lemma. Let $v_i \in Y_i$, $i \in \{1, -1\}$ be vertices as in the lemma, which fulfill $e(v_i, Y_i) = e(v_i, Y_{-i})$. We obtain the bisection vector y' as vector corresponding to $(Y_1 \setminus \{v_1\} \cup \{v_{-1}\}, Y_{-1} \setminus \{v_{-1}\} \cup \{v_1\})$. Due to the balanced degree, this bisection is optimal as well.

Hence, we have two optimum bisections and from Lemma 5.8 we know that the d^{opt} is unique and there are unique $\alpha^{(y)}$ and $\alpha^{(y')}$ corresponding to y and y', resp. It holds

$$\begin{aligned} & d^{(y)} + \alpha^{(y)} y = d^{(y')} + \alpha^{(y')} y' \\ \Leftrightarrow \quad & d^{(y)} - d^{(y')} = \alpha^{(y')} y' - \alpha^{(y)} y \end{aligned}$$

Since v_1 has balanced degree and is only connected to vertices, which are in the same part in y and y', we have $d^{(y)}_{v_1} = d^{(y')}_{v_1} = 0$. Furthermore, $y_{v_1} = 1$, $y'_{v_1} = -1$. Thus we conclude by the equation above that $-\alpha^{(y')} - \alpha^{(y)} = 0$.

Since y and y' are optimum bisections and $e(v_i, Y_i) = e(v_i, Y_{-i})$, we have

$$\sum_{i \in Y_1 \setminus \{v_1\}} d^{(y)}_i - \sum_{i \in Y_1 \setminus \{v_1\}} d^{(y')}_i = 0$$

because

$$\begin{aligned} & \sum_{i \in Y_1 \setminus \{v_1\}} d^{(y)}_i \\ &= \text{bw}(G) - e(v_1, Y_{-1}) - 2 \cdot |(Y_1 \setminus \{v_1\}) \times (Y_1 \setminus \{v_1\}) \cap E(G)| - e(v_1, Y_1) \\ &= \sum_{i \in Y_1 \setminus \{v_1\}} d^{(y')}_i. \end{aligned}$$

But we have also

$$\sum_{i \in Y_1 \setminus \{v_1\}} d^{(y)}_i - d^{(y')}_i = (n/2 - 1)(\alpha^{(y')} - \alpha^{(y)}) = -2\alpha^{(y)}(n/2 - 1).$$

Thus, $\alpha^{(y)} = \alpha^{(y')} = 0$. It follows $d^{(y)}_i - d^{(y')}_i = 0$ for any i, so that each vertex must have no edge to v_1 and v_{-1} or must have an edge to both of them. Hence, the v_1 and v_{-1} have exactly the same neighbors. □

Proof for Lemma 5.12. For contradiction, we assume $h(G) = \text{bw}(G)$. For the bisection vector y, we then have $d^{\text{opt}} = d^{(y)} + \alpha y$ for some $\alpha \in \mathbb{R}$, such that $g(G, d^{\text{opt}}) = \text{bw}(G)$. Let A denote the adjacency matrix of G and let

$B = A + \operatorname{diag}(d^{\mathrm{opt}})$. Referring to the definition of g in Eq. (2.10), and since $\operatorname{sum}(B) = 4\operatorname{bw}(G)$, we conclude from $g(G, d^{\mathrm{opt}}) = \operatorname{bw}(G)$ that $\lambda(B_Q) = 0$. We contradict this by choosing a vector x and then show that the Rayleigh quotient for x and B_P is larger than 0 (for any α). Note, since we assume $\lambda(B_Q) = 0$, we can consider B_P instead of B_Q (see Fact 2.1). This makes our analysis using explicit vectors more readable. W. l. o. g. we assume $|\tilde{C}_{+1}| \geq |\tilde{C}_{-1}|$. We choose

$$x_i = \begin{cases} -1 & \text{if } y_i = +1 \wedge i \notin \tilde{C}_{+1}, \\ z & \text{if } i \in \tilde{C}_{+1} \cup \tilde{C}_{-1}, \\ -\beta z & \text{if } y_i = -1 \wedge i \notin \tilde{C}_{-1}, \end{cases}$$

with $\beta = \sqrt{\frac{\delta + l/z^2}{\delta + l}}$ and $z = \frac{2kl + \delta l + 2\sqrt{kl(k+\delta)(l+\delta)}}{4k^2 + 4\delta k - \delta l}$. Note that for $\delta = 0$, we have $z = l/k > 3$, $\beta = 1/z < 1/3$ and $-\beta z = -1$.

First, we derive the z above by enforcing $\sum_i x_i = 0$ and choosing β as above:

$$\begin{aligned} \sum_i x_i &= l(-1) + (k+\delta)z + kz + (\delta + l)(-\beta z) \\ &= -l + (k+\delta)z + kz - (\delta + l)\sqrt{\frac{\delta z^2 + l}{\delta + l}} \\ &= -l + (2k+\delta)z - \sqrt{(\delta + l)(\delta z^2 + l)} \overset{!}{=} 0 \end{aligned}$$

$$\begin{aligned} \Leftrightarrow \quad && \sqrt{(\delta + l)(\delta z^2 + l)} &= (2k+\delta)z - l \\ \Rightarrow \quad && (\delta + l)(\delta z^2 + l) &= ((2k+\delta)z - l)^2 \\ \Leftrightarrow \quad && \delta^2 z^2 + \delta l + \delta l z^2 + l^2 &= (2k+\delta)^2 z^2 + l^2 - 2(2k+\delta)lz \\ \Leftrightarrow \quad && \delta^2 z^2 + \delta l + \delta l z^2 &= 4k^2 z^2 + \delta^2 z^2 + 4k\delta z^2 - 4klz - 2\delta lz \\ \Leftrightarrow \quad && 0 &= (4k^2 + 4k\delta - \delta l)z^2 + (-4kl - 2\delta l)z - \delta l \\ \Leftrightarrow \quad && z &= \frac{2kl + \delta l \pm \sqrt{(2kl + \delta l)^2 + \delta l(4k^2 + 4k\delta - \delta l)}}{4k^2 + 4k\delta - \delta l} \\ \Leftrightarrow \quad && z &= \frac{2kl + \delta l \pm \sqrt{4(kl)^2 + 4kl\delta l + \delta l(4k^2 + 4k\delta)}}{4k^2 + 4k\delta - \delta l} \\ \Leftrightarrow \quad && z &= \frac{2kl + \delta l \pm 2\sqrt{kl(kl + \delta l + \delta(k+\delta))}}{4k^2 + 4k\delta - \delta l} \\ \Leftrightarrow \quad && z &= \frac{2kl + \delta l \pm 2\sqrt{kl(k+\delta)(l+\delta)}}{4k^2 + 4k\delta - \delta l}. \end{aligned}$$

We take the larger z-solution with the $+$.

We show that by our choice of β, the sum of squares for both parts is the same:

$$\begin{aligned}\sum_{i:y_i=+1} x_i^2 - \sum_{i:y_i=-1} x_i^2 &= (l(-1)^2 + (k+\delta)z^2) - (kz^2 + (\delta + l)(-\beta z)^2) \\ &= l + (k+\delta)z^2 - kz^2 - (\delta + l)\frac{\delta z^2 + l}{\delta + l} \\ &= l + (k+\delta)z^2 - kz^2 - (\delta z^2 + l) = 0.\end{aligned}$$

Thus, α will have no effect:

$$\begin{aligned}\frac{x^T B_P x}{\|x\|^2} &= \frac{x^T B x}{\|x\|^2} \\ &= \frac{x^T (A + \operatorname{diag}(d^{(y)} + \alpha y) x}{\|x\|^2} \\ &= \frac{x^T (A + \operatorname{diag}(d^{(y)}))x + x^T(\alpha y)x}{\|x\|^2} && \Big| \; x^T(\alpha y)x = \alpha \sum_i y_i x_i^2 = 0 \\ &= \frac{x^T (A + \operatorname{diag}(d^{(y)}))x}{\|x\|^2}. && (5.3)\end{aligned}$$

From now we consider the case $4k < l$ and $\delta < \min\{\frac{4k^2}{l-4k}, \frac{7}{128}l\}$. Next, we show $z > 4$. From $\delta < \frac{4k^2}{l-4k}$, we get for the denominator of z that $4k^2 + 4k\delta - \delta l > 0$. For the enumerator, we have:

$$\begin{aligned}&2kl + \delta l + 2\sqrt{kl(k+\delta)(l+\delta)} \\ &= 2kl + 5\delta l + 2\sqrt{kl(k+\delta)(l+\delta)} - 4\delta l \\ &> 8k^2 + 20\delta k + 4\sqrt{k^2(k+\delta)(4k+\delta)} - 4\delta l && \Big| \; \text{Assumption } 4k < l \\ &= 8k(k+\delta) + 12\delta k + 4\sqrt{k^2 4k^2} - 4\delta l \\ &> 16k(k+\delta) - 4\delta l \\ &= 4(4k(k+\delta) - \delta l). && \Big| \; 4 \text{ times denominator of } z\end{aligned}$$

Since the enumerator is more than 4 times larger then the denominator and both are positive, we conclude $z > 4$. From $\delta < \frac{7}{128}l$ follows further that $\beta < 1/3$:

$$\beta^2 = \frac{\delta + l/z^2}{\delta + l} \le \frac{1}{9} = \left(\frac{1}{3}\right)^2$$

$$\begin{aligned}
\Leftrightarrow \quad & 9(\delta + l/z^2) \leq \delta + l \\
\Leftrightarrow \quad & 8\delta \leq l - \frac{9l}{z^2} \\
\Leftarrow \quad & 8\delta \leq l - \frac{9l}{16} && \Big| \, z > 4 \\
\Leftrightarrow \quad & \delta \leq \frac{7}{16 \cdot 8} l.
\end{aligned}$$

Now we want to show that Eq. (5.3) is larger than zero. For this we decompose $B = A + \mathrm{diag}(d^{(y)})$ into $B = \sum_{e \in E} B^e$ and analyze $x^T B^e x$ for each edge e separately. Note that $d_i^{(y)}$ is for vertex i the number of neighbors in the other part minus the number of neighbors in the same part. For the decomposition, we set $B^e_{ii} = B^e_{jj} = 1$, if $e = \{i, j\}$ is a cut edge and $B^e_{ii} = B^e_{jj} = -1$, if e is a inner edge. Further, $B_{ij} = B_{ji} = 1$.

If $e = \{i, j\}$ is a cut edge, we have $x^T B^e x = 2x_i x_j + x_i^2 + x_j^2 = (x_i + x_j)^2$. Thus, cut edges always contribute positive. We only consider the edges $E(\tilde{C}_{+1}, \tilde{C}_{-1})$. Since $x_i = x_j = z$, they contribute $4z^2$ each.

If $e = \{i, j\}$ is a inner edge, we have $x^T B^e x = 2x_i x_j - x_i^2 - x_j^2$. For inner edges in $V \setminus (\tilde{C}_{+1} \cup \tilde{C}_{-1})$, $x_i = x_j$ and the contribution is 0. The same holds for inner edges in $\tilde{C}_{+1}$ and $\tilde{C}_{-1}$. Thus, we only have to consider the edges $E(\tilde{C}_{+1} \cup \tilde{C}_{-1}, V \setminus (\tilde{C}_{+1} \cup \tilde{C}_{-1}))$. One vertex is z, the other -1 or $-\beta z < -1$. Thus, the contribution is $-2z - 1 - z^2$ or $-2\beta z^2 - \beta^2 z^2 - z^2 = -(3\beta + 1)z^2$. Since $0 < \beta < 1/3$, both are larger than $-2z^2$.

We conclude:

$$x^T B x > |E(\tilde{C}_{+1}, \tilde{C}_{-1})| \cdot 4z^2 + |E(\tilde{C}_{+1} \cup \tilde{C}_{-1}, V \setminus (\tilde{C}_{+1} \cup \tilde{C}_{-1}))| \cdot (-2z^2).$$

By the assumption in the lemma, this is greater or equal to zero. □

Proof of Theorem 5.14. Let A be the adjacency matrix of G and

$$A' = \left(\begin{array}{c|cc} \mathrm{A} & 0 & 0 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right)$$

be the adjacency matrix of G', where we added two isolated vertices to G. Since $h(G) = \mathrm{bw}(G)$, there exists a d^{opt}, such that $g(G, d^{\mathrm{opt}}) = \mathrm{bw}(G)$.

$$\begin{aligned}
& h(G) - h(G') \\
= & g(G, d^{\mathrm{opt}}) - \max_{d'} g(G', d')
\end{aligned}$$

$$
\begin{aligned}
&= \frac{\mathrm{sum}(A) + \mathrm{sum}(d^{\mathrm{opt}})}{4} - \max_{d'} \frac{\mathrm{sum}(A') + \mathrm{sum}(d') - (n+2)\lambda(B'_Q)}{4} && \Big| \; B'_Q = (A' + \mathrm{diag}(d'))_Q \\
&= \frac{\mathrm{sum}(d^{\mathrm{opt}})}{4} - \max_{d'} \frac{\mathrm{sum}(d') - (n+2)\lambda(B'_Q)}{4} && \Big| \; \mathrm{sum}(A) = \mathrm{sum}(A') \\
&= \frac{\mathrm{sum}(d^{\mathrm{opt}})}{4} - \max_{z} \frac{\mathrm{sum}(z + (d_{\mathrm{opt}}^T, 0, 0)^T) - (n+2)\lambda(B'_Q)}{4} && \Big| \; d' = z + (d_{\mathrm{opt}}^T, 0, 0)^T \\
&= -\max_{z} \frac{\mathrm{sum}(z) - (n+2)\lambda(B'_Q)}{4} \\
&= \min_{z} \left(\frac{n+2}{4} \lambda(B'_Q) - \frac{\mathrm{sum}(z)}{4} \right) \\
&= \min_{z} \left(\frac{n+2}{4} \max_{x \in S \setminus \{\mathbf{0}\}} \frac{x^T (A' + \mathrm{diag}(d'))x}{\|x\|^2} - \frac{\mathrm{sum}(z)}{4} \right) \\
&= \min_{z} \left(\frac{n+2}{4} \max_{x \in S \setminus \{\mathbf{0}\}} \left(\frac{x^T (A' + \mathrm{diag}(z + (d_{\mathrm{opt}}^T, 0, 0)^T))x}{\|x\|^2} \right) - \frac{\mathrm{sum}(z)}{4} \right).
\end{aligned}
$$

We restrict ourselves to two kinds of vectors: $x_a = (x_1, \ldots, x_n, 0, 0)^T$ with $\sum_{i=1}^n x_i = 0$, and $x_b = (1, \ldots, 1, -\frac{n}{2}, -\frac{n}{2})$, and proceed to

$$
\geq \min_{z} \left(\frac{n+2}{4} \max_{x \in \{x_a, x_b\}} \left(\frac{x^T (A' + \mathrm{diag}(z + (d_{\mathrm{opt}}^T, 0, 0)^T))x}{\|x\|^2} \right) - \frac{\mathrm{sum}(z)}{4} \right). \tag{5.4}
$$

We want to show that this term is at least $\frac{4\,\mathrm{bw}(G)}{n^2}$. Therefore, we analyze the max-term separately and then show, for which d' we have to choose which of the x_a and x_b.

Firstly, consider vector x_a. Let $z^{(n)}$ denote the first n components of vector z. Then

$$
\begin{aligned}
&\max_{x_a = (x_1, \ldots, x_n, 0, 0)^T, \sum_{i=1}^n x_i = 0} \left(\frac{x_a^T (A' + \mathrm{diag}(z + (d_{\mathrm{opt}}^T, 0, 0)^T))x_a}{\|x_a\|^2} \right) \\
&= \max_{\sum_{i=1}^n x_i = 0} \left(\frac{x^T (A + \mathrm{diag}(d^{\mathrm{opt}}))x}{\|x\|^2} + \frac{x^T \mathrm{diag}(z^{(n)})x}{\|x\|^2} \right) \\
&= \max_{\sum_{i=1}^n x_i = 0} \left(\frac{x^T B x}{\|x\|^2} + \frac{x^T \mathrm{diag}(z^{(n)})x}{\|x\|^2} \right).
\end{aligned}
$$

We choose an optimum bisection vector y of G:

$$\geq \frac{y^T B y}{\|y\|^2} + \frac{y^T \operatorname{diag}(z^{(n)}) y}{\|y\|^2} = \frac{\sum_{i=1}^n z_i}{n}. \qquad \Big| \text{Lemma 4.3} \tag{5.5}$$

Secondly, we consider $x_b = (1, \ldots, 1, -\frac{n}{2}, -\frac{n}{2})^T$:

$$\begin{aligned}
&\frac{x_b^T (A' + \operatorname{diag}(z + (d_{\text{opt}}^T, 0, 0)^T)) x_b}{\|x_b\|^2} \\
=&\frac{\operatorname{sum}(A) + \sum_i d_i^{\text{opt}} + x_b^T \operatorname{diag}(z) x_b}{\|x_b\|^2} \\
=&\frac{4 \operatorname{bw}(G) + x_b^T \operatorname{diag}(z) x_b}{\|x_b\|^2} && \Big| \sum_i d_i^{\text{opt}} = 4 \operatorname{bw}(G) - 2|E| \\
=&\frac{4 \operatorname{bw}(G)}{(n+2)\left(\frac{n}{2}\right)} + \frac{\sum_{i=1}^n z_i + (z_{n+1} + z_{n+2}) \left(\frac{n}{2}\right)^2}{(n+2)\left(\frac{n}{2}\right)}.
\end{aligned} \tag{5.6}$$

We insert the result Eq. (5.5) for x_a and Eq. (5.6) for x_b into Eq. (5.4):

$$\begin{aligned}
(5.4) \geq \min_z \Bigg(\frac{n+2}{4} \max \Bigg(\frac{\sum_{i=1}^n z_i}{n}, \frac{4 \operatorname{bw}(G)}{(n+2)\left(\frac{n}{2}\right)} \\
+ \frac{\sum_{i=1}^n z_i + (z_{n+1} + z_{n+2}) \left(\frac{n}{2}\right)^2}{(n+2)\left(\frac{n}{2}\right)} \Bigg) - \frac{\operatorname{sum}(z)}{4} \Bigg).
\end{aligned}$$

We again simplify the terms separately for Eq. (5.5)

$$\begin{aligned}
&\frac{n+2}{4} \frac{\sum_{i=1}^n z_i}{n} - \frac{\sum_{i=1}^{n+2} z_i}{4} \\
=&\frac{(n+2) \sum_{i=1}^n z_i - n \sum_{i=1}^n z_i - n(z_{n+1} + z_{n+2})}{4n} \\
=&\frac{2 \sum_{i=1}^n z_i - n(z_{n+1} + z_{n+2})}{4n} \\
=&\frac{\sum_{i=1}^n z_i}{2n} - \frac{z_{n+1} + z_{n+2}}{4} = \frac{1}{2} \delta && \Big| \delta = \frac{\sum_{i=1}^n z_i}{n} - \frac{z_{n+1} + z_{n+2}}{2}
\end{aligned}$$

and Eq. (5.6)

$$\frac{n+2}{4} (5.6) - \frac{\sum_{i=1}^{n+2} z_i}{4}$$

$$
\begin{aligned}
&=\frac{4\,\mathrm{bw}(G)}{2n}+\frac{\sum_{i=1}^{n} z_i+(z_{n+1}+z_{n+2})\left(\frac{n}{2}\right)^2}{2n}-\frac{\sum_{i=1}^{n} z_i}{4}-\frac{z_{n+1}+z_{n+2}}{4}\\
&=\frac{4\,\mathrm{bw}(G)}{2n}+\left(\frac{1}{2n}-\frac{1}{4}\right)\sum_{i=1}^{n} z_i+\left(\frac{n}{8}-\frac{1}{4}\right)(z_{n+1}+z_{n+2})\\
&=\frac{4\,\mathrm{bw}(G)}{2n}+\frac{2-n}{4n}\sum_{i=1}^{n} z_i+\frac{n-2}{8}(z_{n+1}+z_{n+2})\\
&=\frac{4\,\mathrm{bw}(G)}{2n}+\frac{2-n}{4}\left(\frac{\sum_{i=1}^{n} z_i}{n}-\frac{z_{n+1}+z_{n+2}}{2}\right)\\
&=\frac{2\,\mathrm{bw}(G)}{n}+\frac{2-n}{4}\delta.
\end{aligned}
$$

In both cases, the minimization over z could be reduced to a minimization over δ and we conclude

$$
h(G)-h(G')\geq(5.4)\geq\min_{\delta}\max\left(\frac{1}{2}\delta,\frac{2b}{n}+\frac{2-n}{4}\delta\right).
$$

The first term in the maximum is monotone increasing and the second one monotone decreasing (for $n\geq 3$). Hence, the minimum is at the intersection point of these two lines:

$$
\begin{aligned}
\frac{1}{2}\delta_{\min}&=\frac{2\,\mathrm{bw}(G)}{n}+\frac{2-n}{4}\delta_{\min}\\
\frac{2-2+n}{4}\delta_{\min}&=\frac{2\,\mathrm{bw}(G)}{n}\\
\frac{n}{4}\delta_{\min}&=\frac{2\,\mathrm{bw}(G)}{n}\\
\delta_{\min}&=\frac{8\,\mathrm{bw}(G)}{n^2}.
\end{aligned}
$$

It follows

$$
h(G)-h(G')\geq\frac{1}{2}\delta_{\min}=\frac{4\,\mathrm{bw}(G)}{n^2}.
$$

□

6 Boppana's Approach: SDP Characterizations

In this chapter we compare Boppana's approach with the Algorithm 2.2 (FK) from Feige and Kilian. Feige and Kilian express the Minimum Bisection Problem for an instance graph G as a semidefinite programming problem (SDP) with solution $h_p(G)$ and prove that the function $h_d(G)$, which is the solution to the dual SDP, reaches $\mathrm{bw}(G)$ w.h.p. Since $\mathrm{bw}(G) \geq h_p(G) \geq h_d(G)$, they conclude that $h_p(G)$ as well reaches $\mathrm{bw}(G)$ w.h.p. The proposed Algorithm FK computes $h_p(G)$ and reconstructs the minimum bisection of G from the optimum solution of the primal SDP. The authors conjecture in [31, Sec. 4.1.] the following: "Possibly, for every graph G, the function $h_p(G)$ and the lower bound $h(G)$ computed in Boppana's algorithm give the same value, making the lemma that $h_p(G) = \mathrm{bw}(G)$ w.h.p. a restatement of the main theorem of [12]." In this chapter we answer this question affirmatively.

To compare the algorithms, we provide a primal SDP formulation for Boppana's approach and prove that it is equivalent to the dual SDP of Feige and Kilian. Next, we give a dual program to the primal formulation of Boppana's approach and prove that the optima of the primal and dual programs are equal to each other. Note that, unlike in linear programming, for semidefinite programs there may be a duality gap. Thus, we show that the bisection Algorithm 2.2 (FK) of Feige and Kilian provides exactly the same results as Boppana's approach with Algorithm 4.1 (BB-U).

We also conduct a series of experiments, which show that the SDP formulation of Feige and Kilian can be solved more efficiently by state of the art SDP solvers than the problem formulation of Boppana, although the SDP used about n^2 variables instead of n variables in the eigenvalue based approach. However, in Chapter 7 we provide a new eigenvalue based heuristic that will allow us to solve instances with up to 10^6 vertices, while the SDP approach is limited to around 2000 vertices.

A problem related to Minimum Graph Bisection is the Cluster Recovery Problem, which asks to find the planted bisection from which the graph was generated. Since we show that the method by Feige and Kilian is equivalent to Boppana's, we get, as a consequence, that Algorithm BB-U achieves the sharp threshold for exact cluster recovery in the stochastic block model. This threshold has been obtained recently by Abbe et al. [2] and independently by Mossel et al. [63].

In [2, 63] it is proved that in the (binary) stochastic block model, with $p = \alpha \log(n)/n$ and $q = \beta \log(n)/n$ for fixed constants $\alpha \neq \beta$, if $(\sqrt{\alpha}-\sqrt{\beta})^2 > 2$, the planted clusters can be exactly recovered (up to a permutation of cluster indices) with probability converging to one; if $(\sqrt{\alpha}-\sqrt{\beta})^2 < 2$, no algorithm can exactly recover the clusters with probability converging to one. Note that the choice of p and q is well justified: Mossel et al. show that, if $q < p = \log(n)/n$, then the exact recovery is impossible for these parameters. In [42] Hajek et al. proved that the SDP of Feige and Kilian achieves the optimal threshold, i. e. if $(\sqrt{\alpha} - \sqrt{\beta})^2 > 2$, the SDP reconstructs communities w.h.p. From our result we get that Algorithm BB-U achieves the threshold, too.

6.1 An SDP Formulation for Boppana's Approach

The semidefinite programming approach for optimization problems was studied by Alizadeh [3], who as first provided an equivalent SDP formulation of Boppana's algorithm.

To prove that for any graph G Boppana's function $h(G)$ gives the same value as $h_p(G)$ we formulate the function h as a (primal) SDP. We provide also its dual program and prove that the optimum solutions of primal and dual are equal in this case. Then, we show that the dual formulation of the Boppana's optimization is equivalent to the primal SDP defined by Feige and Kilian [31].

Below, $G = (V, E)$ denotes a graph, A the adjacency matrix of G and for a given vector d, as usually, let $D = \text{diag}(d)$, for short. We provide the SDP for the function h (Eq. (2.12)) that slightly differs from that one given in [3].

Proposition 6.1. *For any graph $G = (V, E)$, the objective function*

$$h(G) \ = \ \max_{d \in \mathbb{R}^n} \frac{\text{sum}(A + D) - n\lambda((A + D)_Q)}{4}$$

maximized by Boppana's approach can be characterized as an SDP as follows:

$$\begin{cases} p(G) = \min\limits_{z \in \mathbb{R}, d \in \mathbb{R}^n} (nz - \mathbf{1}^T d) \quad \textit{subject to} \\ zI - A + \frac{JA+AJ}{n} - \frac{\text{sum}(A)J}{n^2} - D + \frac{\mathbf{1}d^T + d\mathbf{1}^T}{n} - \frac{\text{sum}(D)J}{n^2} \ \succeq \ 0, \end{cases} \tag{6.1}$$

with the relationship $h(G) = \frac{|E|}{2} - \frac{1}{4}p(G)$. The dual program to the program in Eq. (6.1) *can be expressed as follows:*

$$\begin{cases} d(G) = \max_{Y \in \mathbb{R}^{n\times n}} \Big(A \bullet Y - \frac{1}{n}\sum_j \big(\deg(j) \sum_i y_{ij}\big) \\ \qquad\qquad\qquad\qquad -\frac{1}{n}\sum_i \big(\deg(i)\sum_j y_{ij}\big) + \frac{1}{n^2}\sum_{i,j} y_{ij}\Big) \\ \textit{subject to} \\ \qquad\qquad\qquad\qquad\qquad \sum_i y_{ii} \;=\; n, \\ \forall i \quad y_{ii} - \frac{1}{n}\sum_j y_{ji} - \frac{1}{n}\sum_j y_{ij} + \frac{1}{n^2}\sum_{k,j} y_{kj} \;=\; 1, \\ \qquad\qquad\qquad\qquad\qquad Y \;\succeq\; 0. \end{cases} \tag{6.2}$$

Proof. To obtain an SDP formulation we start with Boppana's function $h(G)$ and transform it successively as follows:

$$\begin{aligned} h(G) &= \max_{d\in\mathbb{R}^n} \frac{\mathrm{sum}(A + \mathrm{diag}(d)) - n\lambda((A+\mathrm{diag}(d))_Q)}{4} \\ &= \max_{d\in\mathbb{R}^n} \frac{J \bullet A + \mathbf{1}^T d - n\lambda(Q^T(A+\mathrm{diag}(d))Q)}{4} \\ &= \frac{J\bullet A}{4} + \frac{1}{4}\max_{d\in\mathbb{R}^n}\big(\mathbf{1}^T d - n\lambda(Q^T(A+\mathrm{diag}(d))Q)\big) \\ &= \frac{J\bullet A}{4} - \frac{1}{4}\min_{d\in\mathbb{R}^n}\big(n\lambda(Q^T(A+\mathrm{diag}(d))Q) - \mathbf{1}^T d\big). \end{aligned} \tag{6.3}$$

Obviously it holds (6.3) $\leq$ (6.4). We want to show that we have "=" by showing that every d which is optimal in Eq. (6.4) fulfills $\lambda(\cdot) \geq 0$, such that the maximization has no effect. For contradiction assume that the eigenvalue $z = \lambda(\cdot) < 0$ is negative for some d optimizing the term in Eq. (6.4). Then choose $d' = d - \mathbf{1}\frac{z}{n}$. For d' we also have $\max(\lambda(\cdot), 0) = 0$, and the term within the minimization is $0 - \mathbf{1}^T d' = 0 - (\mathbf{1}^T d - z) = 0 - \mathbf{1}^T d + z < 0 - \mathbf{1}^T d$. Thus, for the minimization, d has not been optimal – a contradiction. Hence, we continue with "=":

$$= \frac{J\bullet A}{4} - \frac{1}{4}\min_{d\in\mathbb{R}^n}\big(n\max(\lambda(Q^T(A+\mathrm{diag}(d))Q), 0) - \mathbf{1}^T d\big) \tag{6.4}$$

$$= \frac{J\bullet A}{4} - \frac{1}{4}\min_{d\in\mathbb{R}^n}\big(n\lambda(P^T(A+\mathrm{diag}(d))P) - \mathbf{1}^T d\big) \qquad \Big|\ \text{Fact 2.1}$$

$$= \frac{J\bullet A}{4} - \frac{n}{4}\min_{d\in\mathbb{R}^n}\lambda\left(P^T(A+\mathrm{diag}(d))P - \frac{\mathbf{1}^T d}{n} I\right)$$

$$=\frac{J \bullet A}{4} - \frac{n}{4} \min_{d \in \mathbb{R}^n} \lambda(M(d)),$$

where $M(d) = P^T(A + \mathrm{diag}(d))P - \frac{\mathbf{1}^T d}{n} I$. Hence, we want to solve the following problem: minimize the largest eigenvalue of the matrix $M(d)$ for $d \in \mathbb{R}^n$. For this problem, [74] gives the SDP formulation:

$$\min z \quad \text{s.t.} \quad zI - M(d) \succeq 0,$$

with $z \in \mathbb{R}, d \in \mathbb{R}^n$. Inserting $M(d)$ and then substituting z with $z - \frac{\mathbf{1}^T d}{n}$, we get

$$\min_{z \in \mathbb{R}, d \in \mathbb{R}^n} \left(z - \frac{\mathbf{1}^T d}{n} \right) \quad \text{s.t.} \quad zI - P^T(A + \mathrm{diag}(d))P \succeq 0. \tag{6.5}$$

It is easy to see that the constraint matrix above is equal to the constraint matrix of Eq. (6.1), since $P = I - \frac{J}{n}$. This completes the proof that $h(G)$ maximized by Algorithm BB-U gives the same value as the optimum solution of Eq. (6.1), because under the constraints we have

$$h(G) = \frac{J \bullet A}{4} - \frac{1}{4} \min_{z \in \mathbb{R}, d \in \mathbb{R}^n} (nz - \mathbf{1}^T d). \tag{6.6}$$

To obtain the formulation for a dual program, consider the primal SDP in the form:

$$\min_{z \in \mathbb{R}, d \in \mathbb{R}^n} (nz - \mathbf{1}^T d) \quad \text{s.t.} \quad -P^T A P + zI - \sum_i d_i P^T I_i P \ \succeq\ 0,$$

where I_i denotes the matrix which has a single 1 in the ith row and the ith column, and zero everywhere else. The dual can be derived by using the rules in Eq. (2.16). We obtain:

$$\max_{Y \in \mathbb{R}^{n \times n}} (P^T A P) \bullet Y \quad s.t. \quad I \bullet Y = n, \quad \forall i: \ -P^T I_i P \bullet Y = -1, \quad Y \succeq 0.$$

Thus, since $P = I - \frac{J}{n}$, we get the following formulation for the dual SDP:

$$\max_{Y \in \mathbb{R}^{n \times n}} \left(A - \frac{JA + AJ}{n} + \frac{\mathrm{sum}(A) J}{n^2} \right) \bullet Y$$

under the constraints:

$$\sum_i y_{ii} = n,$$

$$\forall i \quad y_{ii} - \frac{1}{n}\sum_j y_{ji} - \frac{1}{n}\sum_j y_{ij} + \frac{1}{n^2}\sum_{k,j} y_{kj} = 1,$$

$$Y \succeq 0.$$

Here we can note that the second constraint is equal to $(P^T Y P)_{ii} = 1$, for all i. Note further that $(AJ) \bullet Y = \sum_i \big(\deg(i) \sum_j y_{ij}\big)$ and an analogous holds for $(JA) \bullet Y$. Hence, we can reformulate the objective function as follows:

$$\max_{Y \in \mathbb{R}^{n\times n}} \Big(A \bullet Y - \frac{1}{n}\sum_j \big(\deg(j) \sum_i y_{ij}\big) - \frac{1}{n}\sum_i \big(\deg(i)\sum_j y_{ij}\big) + \frac{1}{n^2}\sum_{i,j} y_{ij}\Big).$$

This completes the proof. □

Using these formulations, we prove that the primal and dual SDPs attain the same optima.

Theorem 6.2. *For the semidefinite programs of Proposition 6.1 the optimal value p^* of the primal SDP in Eq.* (6.1) *is equal to the optimal value d^* of the dual SDP in Eq.* (6.2)*. Moreover, there exists a feasible solution (z, d) achieving the optimal value p^*.*

Proof. Consider the primal SDP in Eq. (6.1) of Boppana in the form of Eq. (6.5)

$$\min_{z\in\mathbb{R}, d\in\mathbb{R}^n} z - \frac{\mathbf{1}^T d}{n} I \quad \text{s.t.} \quad zI - M(d) \succeq 0,$$

with $M(d) = P^T(A + \operatorname{diag}(d))P$ and, recall, $P = I - \frac{J}{n}$. Note that this formulation is equivalent to Eq. (6.1), as we have shown as Eq. (6.6). We show that this primal SDP problem is strictly feasible, i. e. that there exist a z' and a d' with $z'I - M(d') \succ 0$. To this aim we choose an arbitrary d' and then some $z' > \lambda(M(d'))$. From [74, Theorem 3.1] follows that the optima of the primal and dual problem obtain the same value.

To prove the second part of the theorem, i. e. there exists a feasible solution achieving the optimal value p^*, consider the following. The function $h(G)$ maximizes $g(G, d)$ over vectors $d \in \mathbb{R}^n$. The function g is concave and goes to $-\infty$ for vectors d with some component going to ∞. Thus, g reaches its maximum at some finite d^{opt}. Now we choose $d = d^{\text{opt}} - \text{mean}(d^{\text{opt}})\mathbf{1} + n\mathbf{1}$ and $z = \lambda(M(d))$, while the shifting of d does not change the value of $g(G, d)$ due to Lemma 2.7. Clearly, this solution is feasible. We show that it also obtains the optimal value. We insert z and d into Eq. (6.6) and obtain

$$\frac{J \bullet A}{4} - \frac{n\lambda(M(d)) - \mathbf{1}^T d}{4} = \frac{\mathrm{sum}(A) + \mathbf{1}^T d - n\lambda(M(d))}{4}. \tag{6.7}$$

Let us compare this with the definition of $g(G, d)$ as found in Eq. (2.10). The shifting of d ensures $\lambda(M(d)) \geq 0$ and by Fact 2.1 we can use B_P instead of B_Q in function g. We see that Eq. (6.7) is the same as $g(G, d)$, which means Eq. (6.6) gives us the optimal value $g(G, d^{\mathrm{opt}})$. □

6.2 Feige and Kilian's SDP equals SDP for Boppana's Approach

For a graph $G = (V, E)$, Feige and Kilian express the minimum bisection problem as an SDP over an $n \times n$ matrix Y as follows:

$$h_p(G) = \min_{Y \in \mathbb{R}^{n\times n}} h_Y(G) \quad \text{s.t.} \quad \forall i \; y_{ii} = 1, \; \sum_{i,j} y_{ij} = 0, \text{ and } Y \succeq 0, \tag{6.8}$$

where $h_Y(G) = \sum_{\substack{\{i,j\} \in E \\ i<j}} \frac{1-y_{ij}}{2}$. For proving that the SDP takes as optimum the bisection width w.h.p. on $\mathcal{G}_n(p, q)$, the authors consider the dual of their SDP:

$$h_d(G) = \max_{x \in \mathbb{R}^n} \left(\frac{|E|}{2} + \frac{1}{4} \sum_i x_i \right) \quad \text{s.t.} \quad M = -A - x_0 J - \mathrm{diag}(x) \succeq 0, \tag{6.9}$$

where A is the adjacency matrix of G. They show that the dual takes the value of the bisection width w.h.p. and bounds the optimum of the primal SDP. Although we know that their SDP and Boppana's algorithm both work well on $\mathcal{G}_n(p, q)$, it was open so far how they are related to each other. Below we answer this question showing that the formulations are equivalent. We start with the following theorem:

Theorem 6.3. *The primal SDP in Eq.* (6.8) *is equivalent to the dual SDP in Eq.* (6.2)*, with the relationship* $h_p(G) = \frac{|E|}{2} - \frac{1}{4} d(G)$.

For proving this theorem, we need the following:

Claim 6.4. *Let X be a positive semidefinite matrix. Then the conditions* (*a*) $\forall i : \sum_j x_{ij} = 0$ *and* (*b*) $\sum_{i,j} x_{ij} = 0$ *are equivalent.*

Proof. We show two directions. The implication from (a) to (b) is obvious. We proceed with proving of the second direction and assume that (b) holds.

Each positive semidefinite matrix X can be represented as a Gram matrix, i. e. as matrix of scalar products $x_{ij} = \langle u_i, u_j \rangle$ of vectors u_i. Thus, we have

$$\sum_{i,j} x_{ij} = \sum_{i,j} \langle u_i, u_j \rangle = \sum_i \langle u_i, \sum_j u_j \rangle = \langle \sum_i u_i, \sum_j u_j \rangle = 0,$$

where we used condition (b). The scalar product of the vector $\sum_i u_i$ with itself is zero and we conclude that it is the zero vector: $\sum_i u_i = \mathbf{0}$. Now we compute

$$\sum_j x_{ij} = \sum_j \langle u_i, u_j \rangle = \langle u_i, \sum_j u_j \rangle = \langle u_i, \mathbf{0} \rangle = 0$$

which gives condition (a). □

Now we are ready to prove Theorem 6.3:

Proof of Theorem 6.3. For convenience we restate the primal SDP in Eq. (6.8) as follows:

$$h_p(G) = \min_Y \left(\frac{|E|}{2} - \frac{1}{4}(A \bullet Y) \right) \quad \text{s.t.} \quad \forall i \; y_{ii} = 1, \; \sum_{i,j} y_{ij} = 0, \text{ and } Y \succeq 0.$$

We show that for the following program

$$h'_p(G) = \max_Y A \bullet Y \tag{6.10}$$

under the constraints:

$$\begin{aligned} \forall i : y_{ii} &= 1, \\ \sum_{i,j} y_{ij} &= 0, \\ Y &\succeq 0, \end{aligned}$$

we have $h'_p(G) = d(G)$, where recall, $d(G)$ is the objective function of Eq. (6.2). Then we conclude $h_p(G) = \frac{|E|}{2} - \frac{1}{4} h'_p(G) = \frac{|E|}{2} - \frac{1}{4} d(G)$.

Consider an optimum solution matrix Y for the SDP. We show that Y is a solution to the dual program in Eq. (6.2) as well, with the value for the objective function equal to Eq. (6.10).

Since $y_{ii} = 1$, the first constraint of Eq. (6.2) is fulfilled. Due to Claim 6.4 and since $\sum_{i,j} y_{ij} = 0$, we have $\sum_j y_{ij} = 0$ for all i. Hence, the second constraint of (6.2):

$$\forall i \quad y_{ii} - \frac{1}{n}\sum_j y_{ji} - \frac{1}{n}\sum_j y_{ij} + \frac{1}{n^2}\sum_{k,j} y_{kj} = 1$$

is fulfilled as well. In the objective function of Eq. (6.2), the second and third term are zero, since $(AJ) \bullet Y = \sum_i \left(\deg(i) \sum_j y_{ij}\right) = 0$. Obviously, the fourth term is zero due to the constraints as well. Hence, we obtain the same value as $h'_p(G)$.

For the other direction, consider an optimum solution matrix Y of SDP in Eq. (6.2). First, we show that the first and second constraint of Eq. (6.2) imply $\sum_{i,j} y_{ij} = 0$:

$$\begin{aligned}
&\forall i \quad y_{ii} - \frac{1}{n}\sum_j y_{ji} - \frac{1}{n}\sum_j y_{ij} + \frac{1}{n^2}\sum_{k,j} y_{kj} = 1 && \Big|\ \text{second constraint of (6.2)}\\
\Rightarrow\quad &\sum_i y_{ii} - \frac{1}{n}\sum_{i,j} y_{ji} - \frac{1}{n}\sum_{i,j} y_{ij} + \frac{n}{n^2}\sum_{i,j} y_{ij} = n && \Big|\ \text{sum all } n \text{ constraints}\\
\Rightarrow\quad &n - \frac{1}{n}\sum_{i,j} y_{ji} - \frac{1}{n}\sum_{i,j} y_{ij} + \frac{n}{n^2}\sum_{i,j} y_{ij} = n && \Big|\ \text{use first constraint of (6.2)}\\
\Rightarrow\quad &\sum_{i,j} y_{ij} = 0.
\end{aligned}$$

Next, due to Claim 6.4 we know that $\sum_j y_{ij} = 0$ for all i. Again from the second constraint of Eq. (6.2), we conclude that $y_{ii} = 1$. Hence, the constraints of the SDP in Eq. (6.8) are fulfilled. Obviously, the second, third, and fourth term in the objective function of Eq. (6.2) are zero again and the objective values of both SDPs are the same as well. □

From Theorems 6.2 and 6.3 we get

Corollary 6.5. *Let G be an arbitrary graph. Then, for the lower bound $h(G)$ of Boppana's algorithm and for the objective functions $h_p(G)$ of the primal SDP in Eq. (6.8), resp. $h_d(G)$ of the dual SDP in Eq. (6.8) of Feige and Kilian [31], it holds*

$$h(G) = h_p(G) = h_d(G).$$

Thus, for any graph G, both approaches provide the same objective value. Regarding the algorithms, we obtain the following corollary:

Theorem 6.6. *For a given graph G, the algorithms FK and BB-U either both find and certify an optimum bisection, or both fail.*

Proof. The algorithms FK and BB-U are based on the bounding functions $h_p(G)$ and $h(G)$, respectively. Due to Corollary 6.5, they both are the same. The reconstruction process itself is identical for both algorithms. Hence, they output the same results. $\square$

6.3 Comparison of Running Times

Theorem 6.6 raises the interesting question, how the approaches of Boppana and of Feige and Kilian compare in their running time. We implemented both methods: for Boppana's approach we used the original Algorithm BB. Although for a guarantee on a good reconstruction performance, we would need Algorithm BB-U or better BB-UI, we noticed in our experiments that the largest eigenvalue computed by the algorithm had almost always multiplicity 1, in which case BB-UI does the same as BB. For implementing BB, we used Matlab's built-in optimization function `fmincon` and the `eig` function for computing the largest eigenvalue. For the implementation of the SDP approach of Feige and Kilian, we used Algorithm FK*. We then used the SDP solver from [72, 69] with Matlab integration. Table 6.1 shows the running times for both of our implementations on an AMD Opteron 6272 CPU using a single core. For each n we performed 20 runs, where in each run a graph with n vertices is generated and then both algorithms are executed on this graph. On almost all graphs, both algorithms were able to find and certify the optimum bisection. The observed standard deviations have been below 0.2 times the mean value.

Table 6.1: Average running times (in seconds) of 20 runs of the algorithms BB and FK* on graphs from $\mathcal{G}_n(p = \frac{9\log(n)}{n}, q = \frac{2\log(n)}{n})$. The "-" indicates a timeout after 10 minutes. Standard deviations are below 0.2 times mean.

n	50	100	150	500	1000	1500	2000
BB	3.80	29.68	112.16	-	-	-	-
FK*	0.19	0.48	1.00	9.97	50.15	140.43	312.09

As we can see, the SDP formulation is much faster, although the primal problem has n^2 variables for a graph with n vertices. In contrast, Boppana's algorithm uses n variables in the convex optimization problem. However, Boppana's algorithm needs to solve an eigenvalue problem in each evaluation step within the convex optimization. In Chapter 7 we address the problem of the slow convex optimization of Boppana's approach and develop a heuristic which solves input instances with up to 10^6 vertices in a reasonable time. We present more experimental data for the comparison of FK* with our new heuristic and show that the new heuristic clearly outperforms FK*.

6.4 Optimality of Boppana's Approach on the Cluster Recovery Problem

So far we have extensively studied the Minimum Bisection Problem, which asks to partition the vertices into two equally sized sets, such that the number of edges crossing the sets is minimized. A related and also well studied problem is the Cluster Recovery Problem. It is usually considered along with the stochastic block model, which is a generalization of the planted bisection model and is the same in the case of two communities V_1 and V_2 and symmetric probabilities p for an edge within a cluster and q for an edge between two clusters. The Cluster Recovery Problem now asks to find the partition which has been used in the graph generation process, only using the graph itself.

Note an important difference between the Minimum Bisection Problem and the Cluster Recovery Problem: while a minimum bisection always exists and the difficulty is only in finding it, a cluster recovery might be impossible, e .g. if $p = q$. Furthermore, while we usually restrict ourselves to $q < p$ for analyzing minimum bisections, the cluster recovery problem allows for $q > p$. For $q > p$ we can consider the complement graph, which is the same as taking a cut edge probability of $1 - q$ and an inner edge probability of $1 - p$. Then $1 - q < 1 - p$.

The formal definition of Exact Cluster Recovery, e. g. found as strong consistency in [63], considers the asymptotic regime $n \to \infty$ and allows p and q to depend on n:

Definition 6.7 (Strong Consistency [63]). *Let y_n denote the planted bisection of a graph G_n. Given two sequences $p_n, q_n \in [0, 1]$, and given a map $\mathcal{A}$ from graphs to vertex labellings, we say that $\mathcal{A}$ is consistent if*

$$\Pr(\mathcal{A}(G_n) = y_n \text{ or } \mathcal{A}(G_n) = -y_n) \to 1,$$

where the probability is taken with respect to the distribution of $(G_n, y_n) \sim \mathcal{G}_n(p_n, q_n)$.

An intensive study has been carried out on providing lower bounds on $|p-q|$ to ensure recoverability of the planted bisection. Mossel et al. [63] provide a characterization, for which sequences p_n, q_n an estimator for strong consistency exists:

Theorem 6.8 (Characterization of Consistency [63])**.** *Given n, p and q, let $X \sim \mathrm{Binom}(n, \max(p,q))$ and $Y \sim \mathrm{Binom}(n, \min(p,q))$ be independent. We define*

$$P(n,p,q) = \Pr(Y \geq X).$$

Consider sequences $p_n, q_n \in [0,1]$. There is a strongly consistent estimator for $\mathcal{G}_n(p_n, q_n)$ if and only if $P(n, p_n, q_n) = o(n^{-1})$.

In [63], a series of more specific regimes in terms of p_n and q_n is provided. We focus on the model $\mathcal{G}_n(p,q)$ with $p = \alpha \log(n)/n$ and $q = \beta \log(n)/n$ for fixed constants α, β, $\alpha > \beta$, since we can later compare the results to results on the Minimum Bisection Problem. For this regime, Mossel et al. and independently Abbe et al. [2] have shown that there is a sharp threshold phenomenon: exact recovery of the planted bisection is possible if and only if $(\sqrt{\alpha} - \sqrt{\beta})^2 > 2$. Hajek et al. [42] show than an SDP equivalent to the one of Feige and Kilian achieves this bound. Since, due to Corollary 6.5, we know that the SDP is equivalent to Boppana's algorithm, we conclude that also Boppana's algorithm achieves the optimal threshold for finding and certifying the optimum bisection in the considered model. We get:

Theorem 6.9. *Let α and β, $\alpha > \beta$, be constants. Consider the graph model $\mathcal{G}_n(p,q)$ with $p = \alpha \log(n)/n$ and $q = \beta \log(n)/n$. Then, as $n \to \infty$, if $(\sqrt{\alpha} - \sqrt{\beta})^2 > 2$, Algorithm BB-U recovers the planted bisection w.h.p. If $(\sqrt{\alpha} - \sqrt{\beta})^2 < 2$, no algorithm is able to recover the planted bisection w.h.p.*

Proof. The second part of the theorem is exactly the statement from [2]. The first part, i. e. that Boppana's algorithm is able to recover the bisection, follows from [42, Theorem 2]. Hajek et al. show that for $(\sqrt{\alpha} - \sqrt{\beta})^2 > 2$ the SDP of Feige and Kilian obtains the optimal solution. Due to Theorem 6.6, the same holds for Algorithm BB-U. $\square$

7 A New Fast Heuristic for Certified Minimum Graph Bisections

Boppana's algorithm, though good in theory[1], is too slow in practice. This is due to the huge computational complexity of the required high-dimensional convex optimization, which needs a large amount of eigenvalue computations. This severely limits the practical applicability of the algorithm. Numerical experiments done so far, e. g. by Tu and Cheng [70] and Tu et al. [71], were performed on small graphs only. Also, the experiments did not analyze the extent to which Boppana's algorithm certifies the optimality.

In this chapter we present a method to eliminate the time-consuming convex optimization used in Boppana's Algorithm BB. We propose a simple heuristic, named FB (for fast spectral-based bisection), which, performing a single computation of an eigenvector, is able to construct a bisection having in most cases the same width as a candidate bisection determined by Boppana's algorithm. Next, based on new properties of the convex optimization of Boppana's method, FB is able to compute in a single step a lower bound proving the optimality of the given bisection. This bound is close to the value found in the convex optimization, thus allowing to achieve similar certificates as in the original algorithm. However, there is a huge difference in time efficiency between Boppana's algorithm and our method. For example, on graphs with 100 vertices, the convex search of Boppana's algorithm needs around 10^4 eigenvalue evaluations, while our heuristic does a single evaluation to compute both the bisection and the lower bound. This results in a speedup of factor 10^4 compared to plain Boppana in this case. As a consequence, the lack of need of a high-dimensional optimization enables us to perform a large series of experiments on graphs with up to millions of vertices.

As main result we provide a strong experimental evidence that our heuristic works reasonably well in practice on graphs from the planted bisection model and on random regular graphs. To compare the performance of the new heuristic

[1]The theoretical analysis forced us to provide Algorithm BB-UI to cope with a multiplicity of the largest eigenvalue larger than one, but in practice BB does not run into this issue. Thus, we use the original BB in this chapter.

FB and plain Boppana on $\mathcal{G}_n(p,q)$, we estimate, for fixed values n, the spaces of parameters $(p,q) \in [0,1] \times [0,1]$ for which the algorithms find and certify minimum size bisections with high empirical probability. Our experiments demonstrate that for graphs with up to 100 vertices the parameter spaces obtained by both methods almost coincide. Unfortunately, due to huge time complexity of Boppana's original Algorithm BB, we were not able to perform this comparative study for larger values n. In the same way, we compare the performance of our heuristic with a state-of-the-art algorithm by Chen, Sanghavi, and Xu [17, 18] and demonstrate for graphs with up to $n = 300$ that the corresponding parameter spaces are almost the same, but FB is much faster than the algorithm by Chen et al. Furthermore, our experiments show that the parameter spaces (r,b) for the random regular model $\mathcal{R}_n(r,b)$, for which our heuristic works well, nicely capture the cases which satisfy Eq. (2.14), i. e. for which plain Boppana theoretically guarantees certified optimum solutions.

We provide also experiments which demonstrate that our heuristic achieves the sharp threshold for exact cluster recovery in the stochastic block model. From our new theoretic results, we know that the original algorithm of Boppana achieves the optimal threshold (Theorem 6.9), i. e. if $(\sqrt{\alpha} - \sqrt{\beta})^2 > 2$ then the algorithm reconstructs the communities w.h.p. Our experimental results in this chapter demonstrate that in the stochastic model with $p = \alpha \log(n)/n$ and $q = \beta \log(n)/n$ our heuristic FB exactly recovers communities for parameters α and β which achieve the provably optimum parameter space satisfying $(\sqrt{\alpha} - \sqrt{\beta})^2 > 2$.

Related Work.
Delling et al. [29] provide a branch-and-bound heuristic for finding a provably minimum bisection. Their experimental results show that the heuristic dominates other previously known exact algorithms as [68, 14, 5, 41, 47]. Since in [29] the authors do not examine the performance of their algorithm on the planted bisection model and random regular graphs, it would be interesting to see a comparison to our polynomial time heuristic. Unfortunately, neither the source code nor the binary of [29] is available. Thus, it was impossible to compare the performances of both algorithms on the specific families of random graphs. Since the experimental data used by Delling et al. is on weighted graphs only, we don't see a way to relate the performance of the algorithms to each other on instances used in [29].

Organization of this Chapter.
In Section 7.1 we discuss theoretical properties on the lower bound of Boppana and provide some experimental findings indicating how to avoid the

extensive convex search. Our new heuristic FB is described in Section 7.2. In Sections 7.3–7.5 we analyze the performance of the heuristic with focus on the success probability of finding and certifying an optimum bisection. In Section 7.6 we propose a modification FB* of our heuristic which improves the theoretical properties of FB. Finally, we compare the running times of our new heuristic and several other algorithms.

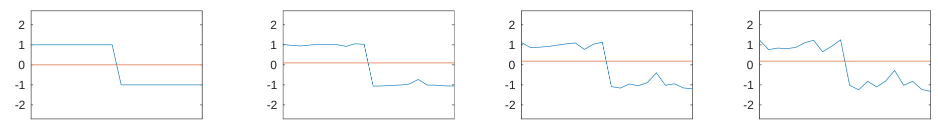
7.1 Avoiding High-Dimensional Optimization

Consider Boppana's original Algorithm BB. The high-dimensional optimization to compute d^{opt} in Step 1 seems to be indispensable for this algorithm. Firstly, the bisection vector $\hat{x}$ is obtained from an eigenvector corresponding to the eigenvalue $(A + \text{diag}(d^{\text{opt}}))_Q$. Secondly, to certify the optimality of the solution $\hat{x}$ the algorithm uses $h(G) = g(G, d^{\text{opt}})$ as a lower bound. However, the optimization step comes with high computational costs. In this section we discuss how to avoid the optimization step despite its inherent advantages. Particularly, we show how to avoid the optimum vector d^{opt} both in the computation of good candidates for bisection vectors and in the lower bound $h(G) = g(G, d^{\text{opt}})$. We also provide experimental and theoretical arguments which justify our approach.

Finding a Relaxed Bisection Vector without d^{opt}.
In our experiments, we have analyzed the impact of the optimal correction vector d^{opt} on the bisection vector $\hat{x}$ generated in Step 2 of BB. Therefore, we examine the relaxed bisection vectors Qx', where x' denotes the eigenvector corresponding to the largest eigenvalue of $\lambda((A + \beta \cdot \text{diag}(d^{\text{opt}}))_Q)$ and we vary β from 1 to 0. Note that for $\beta = 1$ we get just the computation from BB. We have discovered that for $\beta = 0$ the relaxed bisection vector corresponding to $\lambda(A_Q)$ carries almost the same information as the vector corresponding to $\lambda((A + \text{diag}(d^{\text{opt}}))_Q)$. In Figure 7.1 we show some results which illustrate and clarify our findings. We will use the property from Lemma 4.3.

We consider the example graph G_{ex} presented in Figure 4.1. The graph is 5-regular, has 20 vertices and a unique optimum bisection of width 6. The vertex labels are arranged in such way that the first ten and the last ten vertices form the optimum partition. The graph satisfies $h(G_{\text{ex}}) = g(G_{\text{ex}}, d^{\text{opt}}) = \text{bw}(G_{\text{ex}})$, thus, Boppana's algorithm works well on G_{ex}. This means that in Step 3 of BB we get vector $\hat{x}$ of the first ten components equal to $+1$ and the remaining ones equal to -1. For this $\hat{x}$ we have $\text{cw}(\hat{x}) = 6$ and $h(G_{\text{ex}}) = g(G_{\text{ex}}, d^{\text{opt}}) = \text{cw}(\hat{x})$, where d^{opt} is the optimum correction vector found in maximization Step 1 of BB.

Now we analyze for graph G_{ex} to what extent the relaxed bisection vectors still carry the bisection-relevant information. We compute the eigenvectors x'_β for $\lambda((A + \beta \cdot \operatorname{diag}(d^{\text{opt}}))_Q)$, with $\beta = 1$, $\beta = 0.9$, $\beta = 0.5$, and $\beta = 0$. The normalized (unique, up to a sign) vectors Qx'_β for these eigenvectors x'_β are presented in Figure 7.1. From Lemma 4.3 we know that the optimum bisection vector for G_{ex} is equal to $\pm Qx'_\beta$ in the case $\beta = 1$ (see Figure 7.1, first plot). As could be expected, for $\beta = 0.9$ the vector Qx'_β with eigenvector x'_β for $\lambda(A + \beta \cdot \operatorname{diag}(d^{\text{opt}}))_Q)$ remains almost unchanged (second plot). Also the distortion in Qx'_β, with $\beta = 0.5$, seems to be negligible (third plot). However, surprisingly, we can observe that also the vector Qx'_β with eigenvector x'_β for $\lambda(A_Q) = \lambda((A + \beta \cdot \operatorname{diag}(d^{\text{opt}}))_Q)$, with $\beta = 0$, saves the full bisection information, though with some noise (fourth plot). Based on these observations, we will derive a heuristic which tries to gain the optimum bisection from eigenvector x' for $\lambda(A_Q)$.



Boppana ($\beta = 1$) $\beta = 0.9$ $\beta = 0.5$ $\beta = 0$

Figure 7.1: Vectors Qx'_β obtained from eigenvectors x'_β for graph G_{ex} presented in Figure 4.1 with adjacency matrix A: The plots show components of the normalized transformed (by Q) eigenvectors for (unique) eigenvalues $\lambda((A+\operatorname{diag}(d^{\text{opt}}))_Q)$ (left, this corresponds to Boppana's algorithm), and $\lambda((A + \beta \cdot \operatorname{diag}(d^{\text{opt}}))_Q)$, for $\beta = 0.9$ and $\beta = 0.5$, resp. for $\lambda((A)_Q)$ (right). In the plots the components are displayed as line graphs instead of scatter plots.

A Lower Bound without d^{opt}.

Now we discuss how to handle the further issue to eliminate the need for the computation of d^{opt}, namely an estimation of a good lower bound to certify the optimality of the solution. Let y be a bisection vector of G (not necessarily the optimal one). We assign a specific correction vector $d^{(y)}$ to y that is defined as follows (see also Eq. (3.1) and below for the intuition):

$$d^{(y)} = -\operatorname{diag}(y)Ay.$$

From Lemma 5.8 we know that for graphs G for which $h(G) = \operatorname{bw}(G)$, every optimum bisection vector y satisfies the inequality

$$\mathrm{cw}_G(y) - g(G, d^{(y)} + \alpha^{(y)}y) < 1. \tag{7.1}$$

In this case, i. e. if $h(G) = \mathrm{bw}(G)$, we even have $\mathrm{cw}(y) - g(G, d^{(y)} + \alpha^{(y)}y) = 0$. For graphs G with $h(G) < \mathrm{bw}(G)$, inequality Eq. (7.1) may not be true. However, one can still try to verify the optimality of a bisection vector y by testing if Eq. (7.1) holds for y. Importantly, to do this no computation of d^{opt} is needed. Actually, in our new heuristic we will use Eq. (7.1) to certify the optimality of a bisection.

Note that, when Boppana's algorithm is considered to work, the lower bound $h(G)$ exactly reaches the bisection width and the (normalized) eigenvector for the largest eigenvalue is unique and it composes an optimum bisection vector. From the theoretical analysis (see Lemma 5.8) we then know that the optimum correction vector d^{opt} can be derived from an optimum bisection y and has always the form $d^{\mathrm{opt}} = -\operatorname{diag}(y)Ay + \alpha^{(y)}y$ for some scalar $\alpha^{(y)}$. Quite often, as in our example graph G_{ex} (Figure 4.1), we can set $\alpha^{(y)} = 0$. Hence, in our heuristic we will test the optimality of $\hat{x}$ verifying if $\mathrm{cw}(\hat{x}) - g(G, -\operatorname{diag}(\hat{x})A\hat{x}) < 1$.

7.2 A New Heuristic FB

Motivated from the observations of the previous section, we propose a fast bisection heuristic, named FB and based on the function g, that avoids the expensive convex search used by Boppana's algorithm. It can be found in Algorithm 7.1. The idea is to derive the optimum bisection simply using the adjacency matrix and afterwards using the function g to certify the optimality. In an experimental analysis we will show that this new heuristic works well on classes of random graphs.

While Boppana's algorithm finds d using the convex search and afterwards derives a bisection, we first derive a bisection from A and based on this bisection determine some d which hopefully suffices to certify the optimality. The lack of need of a convex search enables us to perform a large series of experiments for FB's performance analysis on graphs with up to several millions of vertices. In our numerical experiments we use Matlab with machine precision computations. In Step 3 we set the tolerance to 10^{-4}, i. e. in our implementation we test if $\mathrm{cw}(\hat{x}) - g(G, d^{(\hat{x})}) < 1 - 10^{-4}$. For the numerical stability of the eigenvalue problem and the reliability of the results, we refer to [65, Chapter 13.7]. A vector iteration algorithm for computing the largest eigenvalue and corresponding

Algorithm 7.1: Fast (Spectral-based) Bisection FB

```
Input: Graph $G$ with adjacency matrix $A$.
1 Compute the eigenvector $x'$ corresponding to the largest eigenvalue of
    the matrix $A_Q$. Let $x = Qx'$. Construct a bisection vector $\hat{x}$ by
    splitting at the median $\bar{x}$ of $x$, i. e. let $\hat{x}_i = +1$ if $x_i \geq \bar{x}$ and $\hat{x}_i = -1$ if
    $x_i < \bar{x}$. If $\sum_i \hat{x}_i > 0$, move (arbitrarily) $\frac{1}{2}\sum_i \hat{x}_i$ vertices $i$ with $x_i = \bar{x}$
    to part $-1$ letting $\hat{x}_i = -1$;
2 Let $d^{(\hat{x})} = -\operatorname{diag}(\hat{x})A\hat{x}$. Compute the function $g(G, d^{(\hat{x})})$ as a lower
    bound;
3 Output $\hat{x}$; if $\mathrm{cw}(\hat{x}) - g(G, d^{(\hat{x})}) < 1$, output “optimum bisection”, else
    “fail”.
```

eigenvector, as done by our heuristic, works well for real symmetric matrices. Rounding errors even prove to work in favor of the algorithm, since they help to overcome problems when the start vector is perpendicular to the largest eigenspace.

7.3 Performance in the Planted Bisection Model

We analyze the performance of FB on graphs from the planted bisection model $\mathcal{G}_n(p, q)$ and provide a strong evidence that our heuristic achieves similar results as plain Boppana. To compare the performance between both algorithms we estimate, for fixed values n, the spaces of parameters $(p, q) \in [0, 1] \times [0, 1]$ for which the heuristics find and certify optimum bisections with high empirical probability.

In our experiment, we proceed for FB resp. BB as follows: we fix $n = 100$ and we take the number of trials to be 5. Next at each trial and for fixed values $(p, q) \in [0, 1] \times [0, 1]$ we choose a random graph from $\mathcal{G}_n(p, q)$ and run the algorithm on this instance. Then we count how many times the algorithm succeeds and compute the empirical probability of success by dividing this value by the number of trials. We iterate this, starting with $(p, q) = (0, 0)$ and increasing the parameters independently by 0.02 until $(1, 1)$. Figure 7.2 summarizes our results. It demonstrates that for $n = 100$ the parameter spaces obtained by both methods almost coincide. Due to the huge complexity of Boppana's algorithm, we were not able to perform the experiments for larger n.

Next, we analyze the performance of FB on graphs from $\mathcal{G}_n(p, q)$ for growing values of n and with parameters $p(n)$ and $q(n)$ for which Boppana's algorithm

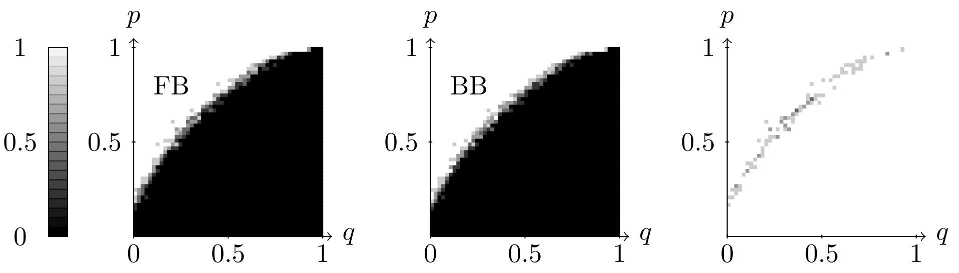


Figure 7.2: Empirical probabilities that FB (left), resp. BB (middle) succeed ($n = 100$). The right plot shows the differences between the probabilities for BB and FB: white pixel means 0 and shades of gray illustrate differences > 0.

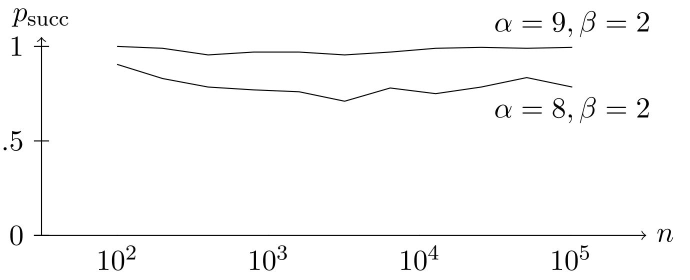


Figure 7.3: The planted bisection model $\mathcal{G}_n(p, q)$: empirical probabilities that the heuristic FB finds and certifies the optimum bisection, using $p = \alpha \log(n)/n$ and $q = \beta \log(n)/n$.

theoretically works, i. e. the assumption Eq. (2.13) is satisfied. We perform our experiments in the regime where $p = \alpha \log(n)/n$ and $q = \beta \log(n)/n$ (for justification see also [2, 63]). We estimate the probability p_{succ} that the optimum bisection of a graph can be found and certified. For this purpose, we generate and analyze 200 random graphs for each set of parameters with n up to 10^5.

Figure 7.3 presents the results for $\alpha = 9, \beta = 2$, as well as for $\alpha = 8, \beta = 2$. While in the first case, the success probability is near one, the latter case already results in a drop of the performance. In Theorem 3.2, we derived that Boppana's algorithm works w.h.p. assuming Eq. (2.13) with $c = 42$. When we insert the p and q from our experiment in Eq. (2.13), we obtain

$$(\alpha - \beta)/\sqrt{\alpha} \geq c.$$

This gives for $\alpha = 9, \beta = 2$ the bound $7/3 \approx 2.333 \geq c$ and for $\alpha = 8, \beta = 2$

the bound $6/\sqrt(8) \approx 2.12 \geq c$. This indicates that our theoretical constant still gives room for improvement.

We can further observe that the graphs for which our heuristic FB works well, nicely capture the threshold for exact cluster recovery in the stochastic block model: for $\alpha = 8, \beta = 2$ the success probability is smaller, but this is exactly the threshold case with $(\sqrt{\alpha} - \sqrt{\beta})^2 = 2$ (see also Section 6.4).

7.4 Performance in the Regular Graph Model

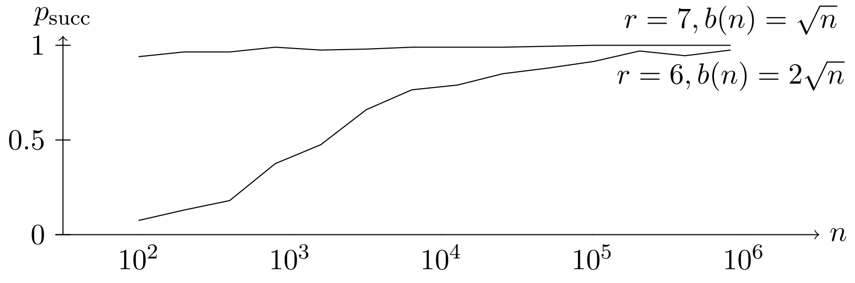


Figure 7.4: Graph model $\mathcal{R}_n(r, b)$: empirical probabilities that the heuristic FB finds and certifies the optimum bisection.

Now we consider the model $\mathcal{R}_n(r, b)$ – regular graphs with degree r and planted bisection width b. The random instances used in our experiments are generated by an algorithm proposed by Bui et al. [15]. We estimate the empirical success probabilities on $\mathcal{R}_n(r, b)$ with increasing number of vertices n and with parameters r and $b(n)$ satisfying Eq. (2.14), i. e. where Boppana claims that BB works well. Figure 7.4 shows the empirical success probability for 6- and 7-regular graphs with bisection width $2\sqrt{n}$, resp. $\sqrt{n}$. We observe that for regular graphs, the bisection width may grow in the order of $\sqrt{n}$ for a success probability going to 1 asymptotically. Note that this is the border case in assumption in Eq. (2.14) for $r = 6$ and $r = 7$. This justifies the relatively slow growth of the plot for $b(n) = 2\sqrt{n}$.

7.5 Achieving the Optimal Threshold for Cluster Recovery

We present that FB achieves the optimal threshold $(\sqrt{\alpha} - \sqrt{\beta})^2 > 2$ for exact cluster recovery in the stochastic block model (SBM) with $p = \alpha \log(n)/n$ and

$q = \beta \log(n)/n$ for fixed constants $\alpha \neq \beta$ (see also Section 6.4).

Due to Theorem 6.6, we already know that Boppana's approach, there using BB-U, achieves the threshold. For the experimental evaluation of FB, we compute for $\alpha \in [1, 40]$ and $\beta \in [0.5, 25]$ the probability that FB finds the planted bisection. For $n = 1000$ we sample 5 graphs for each combination of α and β and count the number of successful recoveries by FB. For results see Figure 7.5. Note that we do not test if FB found an (certifiable) optimum bisection. Instead we check if the output bisection is the same as the planted one, which generally is the task in the community detection literature.

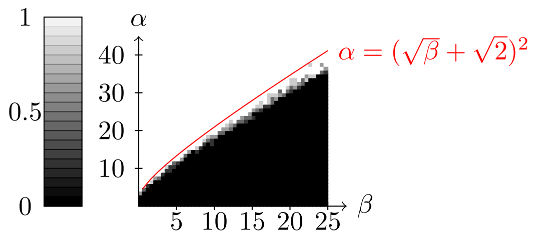


Figure 7.5: Empirical probabilities that FB exactly recovers the planted bisection ($n = 1000$). The red line represents the theoretical threshold.

We compare the performance of our heuristic with a state-of-the-art algorithm for exact cluster recovery in the SBM by Chen, Sanghavi, and Xu [17, 18]. We proceed analogously as in Section 7.3 and demonstrate in Figure 7.6 that, for graphs with $n = 300$, the corresponding parameter spaces are almost the same.

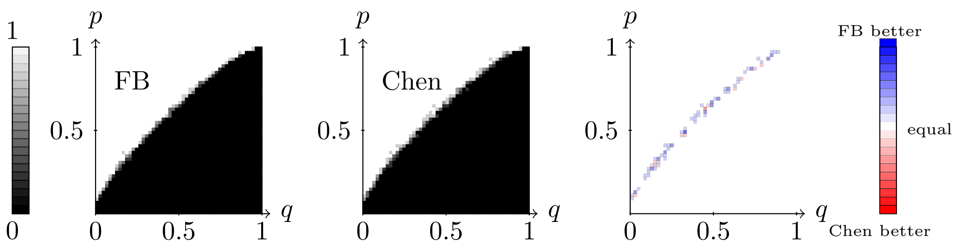


Figure 7.6: A comparison between FB and the algorithm of Chen et al. [17, 18] for exact cluster recovery in the stochastic block model (experiments performed for $n = 300$). The meaning of the plots is analogous to that presented in Figure 7.2.

7.6 An Improved Heuristic FB*

In the following we propose an extension of FB, named as FB*, which additionally performs a one-dimensional convex search. The experiments we conducted have shown that in the cases where $h(G) = \mathrm{bw}(G)$, the vector $\hat{x}$ obtained by our heuristic FB corresponds indeed to an optimum bisection vector. Although FB is also able to certify the optimality of $\hat{x}$ in most cases, there are cases where Boppana's Algorithm BB gives a certificate but FB does not. FB* fills this gap and provides a certificate similar as Boppana does, but only using a one-dimensional instead of an n-dimensional convex search.

For the eigenvector x' of A_Q corresponding to the largest eigenvalue $\lambda(A_Q)$, set $x = Qx'$ and let $\hat{x}$ be the bisection vector obtained by splitting at the median $\bar{x}$ of x. Our new heuristic FB* computes as a lower bound the function

$$h^*(G) \;=\; \max_{\alpha\in\mathbb{R}} g(G, -\operatorname{diag}(\hat{x})A\hat{x} + \alpha\hat{x}). \tag{7.2}$$

For every graph G, both FB and FB* output the same bisection vector. The only difference is that for some graphs FB*, using for optimality test the function h^*, can certify the solution as an optimal one, while FB labels the computed bisection with "fail". Algorithm 7.2 provides the details of FB*.

Algorithm 7.2: FB with one-dimensional convex search FB*

```
  Input: Graph $G$ with adjacency matrix $A$.
1 Do Step 1 of FB; Let $\hat{x}$ be the computed bisection vector and let
    $d^{(\hat{x})} = -\operatorname{diag}(\hat{x})A\hat{x}$;
2 Try to find a better lower bound than $g(G, d^{(\hat{x})})$: to this aim perform a
    convex search on $\alpha$ to compute $h^*(G) = \min_\alpha g(G, d^{(\hat{x})} + \alpha\hat{x})$;
3 Output $\hat{x}$; if $\mathrm{cw}(\hat{x}) - h^*(G) < 1$, output "optimum bisection", else "fail".
```

Lemma 7.1. *Let G be a graph with $h(G) = \mathrm{bw}(G)$. If the vector $\hat{x}$ obtained by FB* is an optimum bisection vector, then FB* certifies the optimality of $\hat{x}$.*

Proof. From $h(G) = \mathrm{bw}(G)$ and with Lemma 4.4, we know that there is a vector $d^{\mathrm{opt}} = d^{(y)} + \alpha^{(y)}y$, for any optimum bisection vector y, such that $g(G, d^{\mathrm{opt}}) = \mathrm{bw}(G)$. By assumption, $y = \hat{x}$ is an optimum bisection vector. Thus, maximizing $h^*(G)$ will find such a d^{opt} and provide the certificate. $\square$

However, FB* is more of theoretical interest, since the gain in the success probability is only marginal, while the additional computational effort of the one-dimensional convex search makes it significantly slower than FB.

7.7 Comparison of Running Times

Finally, we compare the running times of our heuristic FB, Boppana's Algorithm BB, the SDP approach from Feige and Kilian FK*, and the algorithm of Chen et al. Additionally to the graphs from the planted bisection model $\mathcal{G}_n(p, q)$, we tested the algorithms on regular random graphs $\mathcal{R}_n(r, b)$. The results are shown in Table 7.1. For each n we performed 20 runs on an AMD Opteron 6272 CPU using a single core, as we already did in the experiments in Section 6.3. Again, in each run a graph is generated and then the algorithms are executed on this graph. On almost all graphs, the algorithms were able to find and certify the optimum bisection, or, in case of the algorithm of Chen et al., the algorithm was able to recover the planted bisection. The observed standard deviations have been below 0.2 times the mean value.

Table 7.1: Average running times (in seconds) of 20 runs of the algorithms on graphs from the corresponding family if the time did not exceed the limit of 10 minutes. Otherwise we mark such cases by "-". Standard deviations are below 0.2 times mean.

	$\mathcal{G}_n(p = \frac{9\log(n)}{n}, q = \frac{2\log(n)}{n})$				$\mathcal{R}_n(r = 7, b = \sqrt{n})$			
n	FB	BB	FK*	Chen	FB	BB	FK*	Chen
10	0.001	0.13	0.12	0.04	0.001	0.13	0.15	0.04
10^2	0.010	29.68	0.48	0.89	0.009	20.62	0.49	0.87
10^3	0.073	-	50.15	256.44	0.050	-	44.61	252.81
10^4	0.949	-	-	-	0.230	-	-	-
10^5	16.818	-	-	-	2.931	-	-	-
10^6	221.886	-	-	-	34.671	-	-	-

To conclude, we recall that our new heuristic FB finds on graphs from $\mathcal{G}_n(p, q)$ and $\mathcal{R}_n(r, b)$ almost the same (optimal) bisections as BB. The results from Table 7.1 moreover demonstrate that our heuristic FB significantly outperforms each of the other algorithms, while it still is able to provide optimality certificates. Let us note here that the algorithm by Chen et al. also finds the optimum bisection, since w.h.p.the optimum bisection equals the planted bisection.

8 Hard Instances with Provable Large Bisection Width: Ramanujan Graphs

The bisection width is an important graph parameter with direct impact on the worst case running time analysis of many algorithms. For the graphs studied in this chapter, i. e. cubic graphs, the bisection width is also closely related to the pathwidth of a graph – another parameter playing a prominent role in parameterized complexity. Due to Fomin and Høie [34], we know that the pathwidth of a cubic graph with n vertices is bounded from below by $\mathrm{bw}_3(n)$ – the worst case bisection width over all cubic graphs with n vertices. Furthermore, using the best known upper bound $0.167n$ on $\mathrm{bw}_3(n)$ by Monien and Preis [60], Fomin and Høie improved the worst case time analysis for a number of exact exponential algorithms on graphs of maximum vertex degree three, including Independent Set, Dominating Set and Max-Cut. Accordingly, an exact estimation of $\mathrm{bw}_3(n)$ is an important theoretical challenge to address.

An important task for experimental analysis of parameterized algorithms is to construct effectively graphs with bisection width equal or close to $\mathrm{bw}_3(n)$, and thus of largest pathwidth. We investigate cubic Ramanujan graphs as good deterministic candidates for graphs with largest bisection width.

Known Results

The term of Ramanujan graphs was coined by Lubotzky, Phillips, and Sarnak [52]. We denote by $\lambda_2(G)$ the second largest eigenvalue of the adjacency matrix of G, and define:

Definition 8.1. *A connected r-regular graph G is Ramanujan if*

$$\lambda_2(G) \leq 2\sqrt{r-1}.$$

The lower bound by Alon-Boppana implies that for every (n, r)-graph G we have ([64, 36]) $\lambda_2(G) \geq 2\sqrt{r-1} \cdot (1 - \mathcal{O}(1/\log^2 n)))$. Here (n, r)-graph means r-regular with n vertices. Thus, Ramanujan graphs are optimal in the sense of a large spectral gap. A crucial property of Ramanujan graphs discovered by Lubotzky et al. [52] and independently by Margulis [56] is that arbitrarily large r-regular Ramanujan graphs exist when $r-1 \geq 5$ is prime congruent to $1 \mod 4$, and moreover they can be explicitly constructed. In [19] Chiu extends the construction of [52] and shows the existence of 3-regular Ramanujan graphs, which are the main interests of this chapter. For a prime $q \neq 2, 13$, with $\sqrt{-2} \in \mathbb{Z}_q$ and $\sqrt{13} \in \mathbb{Z}_q$, Chiu provides an explicit construction, named $X^{2,q}$, that is a 3-regular Ramanujan graph with vertex set of size $q(q^2-1)$ if $\left(\frac{2}{q}\right) = 1$ and $q(q^2-1)/2$ if $\left(\frac{2}{q}\right) = -1$, where $\left(\frac{a}{q}\right)$ is the Legendre symbol of a and q. Chiu does not write explicitly if $X^{2,q}$ can be used to obtain a richer family, e. g. allowing graphs of sizes q. Morgenstern [62] extended the construction of [52] to the case when $p \geq 2$ is a prime power.

It was long an open conjecture whether for every degree $r \geq 3$ there exist r-regular Ramanujan graphs of any size n. Very recently Marcus et al. have shown in a seminal series of papers [55, 54] that there exist Ramanujan bipartite graphs for all degrees and all number of vertices. In a follow-up to the work of Marcus et al., Cohen [20] has shown a polynomial time algorithm to find bipartite Ramanujan multi-graphs of all degrees via the method of interlacing families. However, it is not clear whether this algorithm can be reasonably implemented [21].

The importance of Ramanujan graphs follows from peculiar properties of these graphs. For example, they belong to the class of expander graphs (see e. g. [45]). Moreover, the explicit constructions by Lubotzky, Phillips, and Sarnak [52] and by Margulis [56] share many of the extremal properties of random graphs, e. g. they have girth, i. e. the length of the shortest circle, at least $6 \log_{r-1} n$. In this chapter we analyze the bisection width of explicitly constructed cubic Ramanujan graphs.

The best known upper bound on $\mathrm{bw}_3(n)$ is due to Monien and Preis [60]: For any $\epsilon > 0$ and sufficiently large 3-regular graphs, it holds

$$\mathrm{bw}_3(n) \leq (\frac{1}{6} + \epsilon)n \approx (0.167 + \epsilon)n.$$

The proof is constructive and based on a polynomial time algorithm doing local improvements on an initial bisection. On the other hand, there are lower bounds on the worst case bisection width which limit the improvement of the upper bound. The best known lower bound of this kind is due to Kostochka and Melnikov [48]. They show that almost all 3-regular graphs have a bisection width of at least $\frac{1}{9.9}n \approx 0.101n$:

$$0.101n \leq \mathrm{bw}(G) \quad \text{for almost all } (n,3)\text{-graphs } G. \tag{8.1}$$

As a consequence, we get $0.101 \leq \mathrm{bw}_3(n) \leq 0.167n$ for sufficiently large n. However, since the proof by Kostochka and Melnikov uses a probabilistic approach, it does not provide any graph construction satisfying the lower bound. Consequently, we know there exist a large number of $(n, 3)$-graphs G with $0.101n \leq \mathrm{bw}(G)$, but we do not know any explicit graph family satisfying this inequality or how to compute such worst case graphs effectively. Note that generating graphs randomly one gets with high probability G with $\mathrm{bw}(G)$ satisfying the lower bound in Eq. (8.1), but it is unknown how to verify this effectively.

A partial solution of this problem was given by Bezrukov et al. [7] who proved for the cubic Ramanujan graphs the lower bound $0.082n$ on the bisection width:

$$0.082n \leq \mathrm{bw}(G) \quad \text{for all Ramanujan } (n, 3)\text{-graphs } G. \tag{8.2}$$

Obviously, the upper bound $0.167n$ for the bisection width on the class of cubic Ramanujan graphs also applies. It can be shown (cf. [34, 35]) that the lower bounds Eq. (8.1) and Eq. (8.2) on the worst case bisection width also yield the lower bounds $0.101n$, resp. $0.082n$ for pathwidth of graphs with maximum degree three. It is an interesting challenge to reduce the gap between $0.101n$ and $0.167n$ in the general case, and between $0.082n$ and $0.167n$ for explicitly constructed graphs. A solution to this problem would give an impact on the question what is the largest possible pathwidth of an n-vertex cubic graph.

Our Achievements

In our main theoretical results, we show explicit constructions for cubic Ramanujan graphs whose number of vertices is growing linear. In the first construction, we start with Chiu's graphs $X^{2,q}$ and transform them in the linear fractional way, similarly as Lubotzky et al. [52] in case of $X^{p,q}$ graphs, with $p \geq 5$. This way we obtain cubic Ramanujan *multigraphs*, i. e. graphs which have multiple edges and self-loops.

Theorem 8.2. *Let $q \neq 2, 13$ be a prime with $\sqrt{-2} \in \mathbb{Z}_q$ and $\sqrt{13} \in \mathbb{Z}_q$. Then there exists an explicitly constructed multigraph $Y^{2,q}$ that is a non-bipartite cubic Ramanujan graph with $q + 1$ vertices. The graphs $Y^{2,q}$ can be constructed in linear time.*

Next, based on structural properties of $Y^{2,q}$, we show how to modify these graphs into simple graphs such that they remain Ramanujan. This is the main theoretical result of this chapter. Note that the algorithm proposed by Cohen [22] generates (bipartite) multigraphs as well.

Theorem 8.3. *Let $q \neq 2, 13$ be a prime with $\sqrt{-2} \in \mathbb{Z}_q$, $\sqrt{13} \in \mathbb{Z}_q$ and $\sqrt{-15} \notin \mathbb{Z}_q$. Then there exists an explicitly constructed simple graph $G^{2,q}$ that is a non-bipartite cubic Ramanujan graph with $q+1$ vertices. The graphs $G^{2,q}$ can be constructed in linear time.*

Tables 8.1 and 8.2 illustrate quantitative differences between the constructions by Morgenstern [62], the graphs $X^{2,q}$ by Chiu [19], and our families.

Table 8.1: Number of cubic Ramanujan graphs with n vertices for the specific graph classes. The first three families allow self-loops and multiple edges, while $G^{2,q}$ consists of simple graphs.

	Number of graphs with n vertices				
Graph class	$n \leq 10^3$	$n \leq 10^4$	$n \leq 10^5$	$n \leq 10^6$	$n \leq 10^7$
1. Morgenstern [62]	1	2	2	3	3
2. $\#X^{2,q}$ Chiu [19]	1	2	3	4	11
3. $\#Y^{2,q}$ Our work	40	298	2389	19616	165882
4. $\#G^{2,q}$ Our work	23	156	1191	9849	83049

Table 8.2: Sizes of the eight smallest cubic Ramanujan graphs of Morgenstern [62] (line 1), the graphs $X^{2,q}$ by Chiu [19] (line 2), and our families $Y^{2,q}$ and $G^{2,q}$ (line 3, resp. 4).

1.	60	4080	262080	16776960	$\approx 10^9$	$\approx 7 \cdot 10^{10}$	$\approx 4 \cdot 10^{12}$	$\approx 3 \cdot 10^{14}$
2.	24	2448	79464	721392	1224936	2247960	2685480	5735160
3.	4	18	44	108	114	132	140	180
4.	44	132	180	252	284	314	338	420

In Theorem 8.3, in addition to the constraints on $Y^{2,q}$ given in Theorem 8.2, we require that $\sqrt{-15} \notin \mathbb{Z}_q$. In this chapter we show how to modify the multigraphs $Y^{2,q}$ to obtain simple cubic graphs, which we call $\bar{G}^{2,q}$, such that the construction allows q with $\sqrt{-15} \in \mathbb{Z}_q$, and such that the bisection width of the resulting graph $\bar{G}^{2,q}$ is not smaller than $\mathrm{bw}(Y^{2,q})$.

These constructions allow us to conduct an extensive experimental analysis on the bisection width. We estimate the asymptotic bisection width of our graphs at approximately $0.12n$, leading to the conjecture that the cubic Ramanujan graphs fulfill a better lower bound than in Eq. (8.2). Furthermore, we compare the results with these of random graphs and observe a similar behavior as for the Ramanujan graphs.

The graphs constructed in this chapter might be interesting as benchmark graphs for the Parameterized Algorithms and Computational Experiments Challenge (PACE), in the track on optimal tree decomposition.
(http://pacechallenge.wordpress.com)
In the 2017's competition [28], the two best algorithms were able to solve all instances from the benchmark set which contains graphs of sizes between 48 and 3104. We ran both algorithms on our newly constructed graphs and observed that from the cubic graphs they could only solve instances with a number of vertices below 100 within the time limit used in the competition. Thus, for our graphs it seems to be hard to compute an optimal tree decomposition even for a very small number of vertices.

Organization of this Chapter

In the next section we recall the known constructions of Ramanujan graphs. Then, in Section 8.2 we show the transformation of the graphs $X^{p,q}$ to $Y^{p,q}$ and in Section 8.3 we analyze structural properties of the cubic Ramanujan graphs $Y^{2,q}$. Both sections provide proofs for Theorem 8.2 and 8.3: Theorem 8.2 follows from Theorem 8.7 and Lemma 8.9, while Theorem 8.3 is a consequence of Theorem 8.12, Lemma 8.9, and Theorem 8.10. In Section 8.3 we show also how to obtain the simple cubic graphs $\bar{G}^{2,q}$. Finally, in Section 8.4 we provide our experimental results.

8.1 Known Constructions of Explicit Ramanujan Graphs

Our constructions provided in this chapter are based on works by Lubotzky, Phillips, and Sarnak [52] and by Chiu [19]. In this section we recall the most relevant results of [52, 19] and we briefly present the extension by Morgenstern [62].

Lubotzky et al. [52] and Chiu [19] construct Ramanujan graphs as Cayley graphs of the projective linear group $PGL(2, \mathbb{Z}_q)$ and the projective special linear group $PSL(2, \mathbb{Z}_q)$. A Cayley graph is a directed graph which is induced by a group and a generating set. Each vertex corresponds to a group element. Edges are created by applying the elements of the generating set to the element corresponding to a vertex. $\mathbb{Z}_q$ denotes the group of integers modulo a prime q. We represent the group $PGL(2, \mathbb{Z}_q)$ as 2×2 matrices $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$, where all multiples kM of M, $k \in \mathbb{Z}_q$, represent the same group element in $PGL(2, \mathbb{Z}_q)$. For the $PSL(2, \mathbb{Z}_q)$, we require the unit determinant $ab - cd = 1$. The group operation is the usual matrix multiplication. The constructions by

Lubotzky et al. [52] and Chiu [19] differ in how the generating sets are chosen. Once the generating set is determined, they both construct the Cayley graphs over the mentioned groups. Lubotzky et al. [52] construct regular graphs with degree $p+1$ for $p = 5, 13, \ldots$, while Chiu explicitly solves the problem for $p = 2$.

We start with the definition of the generating set by Lubotzky, Philips, and Sarnak. Let $p \neq q$ be two primes with $p, q \equiv 1 \mod 4$. Let further i be such that $i^2 \equiv -1 \mod q$. There are $8(p+1)$ solutions to $a_0^2 + a_1^2 + a_2^2 + a_3^2 = p$ (see [52]), and thereof $p+1$ solutions with $a_0 > 0$ odd and a_1, a_2, a_3 even. Associate a generator matrix to each such solution $(a_{k,0}, a_{k,1}, a_{k,2}, a_{k,3})$, $k \in \{1, \ldots, p+1\}$:

$$S_k^{p,q} = \begin{pmatrix} a_{k,0} + ia_{k,1} & a_{k,2} + ia_{k,3} \\ -a_{k,2} + ia_{k,3} & a_{k,0} - ia_{k,1} \end{pmatrix}. \tag{8.3}$$

These $p+1$ matrices form the generating set $S^{p,q} = \{S_1^{p,q}, \ldots, S_{p+1}^{p,q}\}$.

This construction does not work for $p = 2$, such that we cannot generate cubic graphs that way. Chiu [19] proposes generators explicitly for this case using the following generating set $S^{2,q}$. Let $q \neq 2, 13$ be a prime such that $\sqrt{-2} \in \mathbb{Z}_q$ and $\sqrt{13} \in \mathbb{Z}_q$. The set $S^{2,q} = \{S_1^{2,q}, S_2^{2,q}, S_3^{2,q}\}$ is defined as follows:

$$\begin{aligned} S_1^{2,q} &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \ S_2^{2,q} = \begin{pmatrix} 2+\sqrt{-2} & \sqrt{-26} \\ \sqrt{-26} & 2-\sqrt{-2} \end{pmatrix}, \\ S_3^{2,q} &= \begin{pmatrix} 2-\sqrt{-2} & -\sqrt{-26} \\ -\sqrt{-26} & 2+\sqrt{-2} \end{pmatrix}. \end{aligned} \tag{8.4}$$

We combine the constructions from Lubotzky et al. and the one of Chiu by using primes $p \neq q$ with

$$(i)\ p = 2, q \neq 2, 13 \text{ with } \sqrt{-2}, \sqrt{13} \in \mathbb{Z}_q \text{ or } (ii)\ p, q \equiv 1 \mod 4 \tag{8.5}$$

and then construct Ramanujan graphs $X^{p,q}$ as Cayley graphs using the generator sets $S^{p,q}$ from above. The graph $X^{p,q}$ is defined as follows:

$$\text{Set of vertices } \ V(X^{p,q}) = \begin{cases} PSL(2, \mathbb{Z}/q\mathbb{Z}) \text{ if } \left(\frac{p}{q}\right) = 1, \\ PGL(2, \mathbb{Z}/q\mathbb{Z}) \text{ if } \left(\frac{p}{q}\right) = -1 \end{cases}$$

$$\text{Set of edges } \ E(X^{p,q}) = \{(M, M \cdot S_i^{p,q}) \mid M \in V(X^{p,q}),\ i = 1, \ldots, p+1\},$$

where $\left(\frac{p}{q}\right)$ is the Legendre symbol of p and q.

Theorem 8.4 ([52], [19])**.** *Let p, q be primes fulfilling condition Eq.* (8.5)*. Then $X^{p,q}$ is a $(p+1)$-regular Ramanujan graph of order $q(q^2-1)$ if $\left(\frac{p}{q}\right) = -1$ and of order $q(q^2-1)/2$ if $\left(\frac{p}{q}\right) = 1$. Moreover, $X^{p,q}$ is bipartite if and only if $\left(\frac{p}{q}\right) = -1$.*

The definitions above allow the construction of $(p+1)$-regular graphs for primes p satisfying Eq. (8.5). Morgenstern [62] extended these results and provides a construction for prime powers. Particularly, he provides a construction such that for any p which is a power of 2 and any even d, the resulting Cayley graph is a $(p+1)$-regular Ramanujan graph with $(p^{3d} - p^d)$ vertices. However, in the case of 3-regular graphs, as illustrated in Tables 8.1 and 8.2, the obtained class is much sparser than $X^{2,q}$. For Morgenstern's construction, it is also possible to define smaller graphs analogously to our constructions in the next section. We would then use the vertex set $\mathbb{F}_{q^d} \cup \{\infty\}$ and obtain $(q+1)$-regular multigraphs with $q^d + 1$ vertices. However, the number of vertices is still growing exponentially.

8.2 Construction of Almost Dense Families – General Case

A drawback of the graph family $X^{p,q}$ is the growth of the number of vertices, which is of order $\mathcal{O}(q^3)$. Lubotzky et al. already mentioned the possibility to construct an almost dense family $Y^{p,q}$ for $p \geq 5$ of graphs using linear fractional transformations on their construction [52, page 14]. This way we obtain multigraphs of sizes $\mathcal{O}(q)$. We will use their idea and show that this family can be generalized to also capture the case $p = 2$ covered by Chiu's construction, i. e. we add the case $Y^{2,q}$. For the sake of completeness, we present here the transformation for any prime $p \geq 2$. Furthermore, we show that the construction of these dense graphs can be computed in time linear in $p \cdot q$.

Based on the generator sets $S^{p,q}$ from above, we construct graphs $Y^{p,q}$ with $q+1$ vertices. A set of generators becomes a set of Möbius transformations, which we first define as follows: let $\mathbb{F}$ be a field and $a, b, c, d \in \mathbb{F}$ with $ad - bc \neq 0$. We define the Möbius transformation over the extension $\mathbb{F} \cup \{\infty\}$ as $\phi(z) = \frac{az+b}{cz+d}$. For $c \neq 0$, we define $\phi\left(\frac{-d}{c}\right) = \infty$ and $\phi(\infty) = \frac{a}{c}$. For $c = 0$, we define $\phi(\infty) = \infty$.

Now let p, q be primes fulfilling condition Eq. (8.5). The multigraph $Y^{p,q}$

is defined as follows. The vertex set is $V(Y^{p,q}) = \mathbb{Z}_q \cup \{\infty\}$. Then, for each generator matrix $S_i^{p,q} = \left(\begin{smallmatrix} a_i & b_i \\ c_i & d_i \end{smallmatrix}\right)$, we define a Möbius transformation

$$\phi_i^{p,q}(z) = \frac{d_i z + b_i}{c_i z + a_i}.$$

The edges are the following multiset: $E(Y^{p,q}) = \{(z, \phi_i(z)) \mid z \in V, i = 1, \dots, p+1\}$.

The families $Y^{p,q}$ with $p, q \equiv 1 \mod 4$ are quite dense: Let $\pi_{a,b}(x)$ denote the number of primes in the arithmetic progression $b + ja$ for $j = 0, 1, , \dots$ less than x. The Siegel-Walfisz Theorem gives a good estimate for the distribution of primes in the residue classes. We have in particular: $\pi_{4,1}(x) \sim \pi_{4,3}(x)$. Empirically the primes congruent to 3 are more numerous and are nearly always ahead in this sequence; the first reversal occurs at $x = 26861$. However, Littlewood showed in 1914, that there are infinitely many sign changes for the function $\pi_{4,1}(x) - \pi_{4,3}(x)$ (see paper by Granville and Greg [39]).

Surely, $X^{p,q}$ and $Y^{p,q}$ are related to each other. In the graphs $X^{p,q}$, we labeled the vertices with matrices of the type $\left(\begin{smallmatrix} 1 & x \\ y & z \end{smallmatrix}\right)$ or $\left(\begin{smallmatrix} 0 & 1 \\ y & z \end{smallmatrix}\right)$. In $Y^{p,q}$, instead of using matrices, we identify the vertices by some $z \in \mathbb{Z}_q \cup \{\infty\}$. The relation can be viewed as simply ignoring the second row of the matrix labels from $X^{p,q}$ to obtain the corresponding labels of $Y^{p,q}$, while preserving all edges.

In the following we formally give the mapping from $X^{p,q}$ to $Y^{p,q}$, and then we show that $Y^{p,q}$ is Ramanujan.

Definition 8.5. *We define the map* $h : PGL(2, \mathbb{Z}/q\mathbb{Z}) \to \mathbb{Z}_q \cup \{\infty\}$ *as*

$$h : \begin{pmatrix} a & b \\ c & d \end{pmatrix} \to \frac{b}{a}.$$

Since $PSL(2, \mathbb{Z}/q\mathbb{Z})$ is a subgroup of $PGL(2, \mathbb{Z}/q\mathbb{Z})$, the map h can be used for graphs defined on $PSL(2, \mathbb{Z}/q\mathbb{Z})$ as well.

Lemma 8.6. *The map h is a homomorphism: $h(MS_i) = \phi_i(h(M))$.*

Proof. A computation from both sides ends in the same term:

$$h(MS_i) = h\left(\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix}\right) = \frac{ab_i + bd_i}{aa_i + bc_i},$$

$$\phi_i(h(M)) = \phi_i(h\left(\begin{pmatrix} a & b \\ c & d \end{pmatrix}\right)) = \phi_i\left(\frac{b}{a}\right) = \frac{d_i\frac{b}{a} + b_i}{c_i\frac{b}{a} + a_i} = \frac{d_i b + b_i a}{c_i b + a_i a}.$$

□

Theorem 8.7. *If $X^{p,q}$ is Ramanujan, then $Y^{p,q}$ is Ramanujan.*

Proof. Both graphs are regular with same degree, since the number of generators is the same. Let A^X denote the adjacency matrices of $X^{p,q}$ and let $f : V(X^{p,q}) \to \mathbb{N}$ map the vertices of the graph to the corresponding row index in the adjacency matrix. Analogously, we define A^Y and $g : V(Y^{p,q}) \to \mathbb{N}$ for the graph $Y^{p,q}$. We show that every eigenvalue of A^Y is an eigenvalue of A^X. Thus, A^Y has a spectral gap at least as large as A^X.

Consider an arbitrary eigenvector y of A^Y. We define the vector x as follows: for every $v \in V(X^{p,q})$, set $x_{f(v)} := y_{g(h(v))}$. Now, for every $w \in V(X^{p,q})$:

$$\begin{aligned}
\sum_{v \in V(X^{p,q})} A^X_{f(w),f(v)} \cdot x_{f(v)} &= \sum_{i=1,\ldots,p+1} x_{f(w \cdot S_i)} && \Big| \text{Def. of } X^{p,q} \\
&= \sum_{i=1,\ldots,p+1} y_{g(h(w \cdot S_i))} && \Big| \text{Def. of } x \\
&= \sum_{i=1,\ldots,p+1} y_{g(\phi_i(h(w)))} && \Big| \text{Lemma 8.6} \\
&= \sum_{v \in V^Y} A^Y_{g(h(w)),g(v)} \cdot y_{g(v)}. && \Big| \text{Def. of } Y^{p,q}
\end{aligned}$$

Hence, x describes an eigenvector of A^X with the same eigenvalue as y for A^Y. □

Some edges coincide or form self-loops. However, the number of occurrences is bounded in p, the prime which determines the degree, and does not grow with the number of vertices:

Lemma 8.8. *The multigraph $Y^{p,q}$ has at most $2(p+1)$ self-loops and $(p+1)p$ multi-edges.*

Proof. Each self-loop corresponds to a solution of an equation $\phi_i^{p,q}(z) = z$, $i = 1, \ldots, p+1$. Since the equations are of degree ≤ 2, they have at most two solutions each. Therefore, we have at most $2(p+1)$ self-loops. Each multi-edge corresponds to a solution of an equation $\phi_i^{p,q}(z) = \phi_j^{p,q}(z)$ with $i, j = 1, \ldots, p+1$, $i \neq j$, which again are quadratic equations. Therefore, we have at most $(p+1)p$ multi-edges. □

Lemma 8.9. *The graphs $Y^{p,q}$ can be constructed in time $\mathcal{O}(pq)$.*

Proof. First we need to compute the generators $S^{p,q}$. For $p = 2$, the generators are already provided (see Eq. (8.4)). For other p, we need to solve the equation $a_0^2 + a_1^2 + a_2^2 + a_3^2 = p$ (see Eq. (8.3)). This can be done via bruteforce for $a_i = -\sqrt{p}, \ldots, \sqrt{p}$, with a_1 odd and a_2, a_3, a_4 even and has a running time of $\mathcal{O}(p^2)$. Since we generate $(p+1)$-regular graphs with $q+1$ vertices, we can assume $p < q$, and thus $\mathcal{O}(p^2) \subseteq \mathcal{O}(pq)$.

Using the generators and the corresponding Möbius transformations, we proceed with the graph generation. The vertices are the numbers $0, \ldots, q-1$ and the additional element ∞. We assume multiplication and addition in $\mathbb{Z}_q$ can be done in constant time. Roots can be computed via brute force in time $\mathcal{O}(q)$. For the division we precompute the inverse of each element. This is done as follows:

1. Find all distinct prime factors $p_1, \ldots, p_k$ of $q-1$. This can be done by successively testing if $2, 3, \ldots$ divide $z = q-1$. If a factor is found, it is added to a list and the factor is eliminated from z. The time complexity is $\mathcal{O}(q)$.
2. Find a primitive root modulo q: for $m = 2, 3, \ldots$, test if m is a primitive root. For this test we compute $m^{(q-1)/p_i} \mod q$ for every prime factor p_i. If the result is not congruent 1 for every p_i, then m is a primitive root. This is due to the fact that the order of an element $m \in \mathbb{Z}_q$ is a divisor of the order $q-1$ of the group $\mathbb{Z}_q$ itself. We have $\mathcal{O}(\log q)$ prime factors to test and the modular exponentiation can be done in $O(\log q)$. Thus, a single test is in $\mathcal{O}(\log^2 q)$. If q is large enough, it is known that the least primitive root is at most $\sqrt{q} - 2$ [22]. Thus, we obtain a total running time of $\mathcal{O}(\sqrt{q} \log^2 q)$ for finding a primitive root.
3. Compute the multiplicative modular inverse r^{-1} of the primitive root r as $r^{q-2} \mod q$. This is $\mathcal{O}(\log q)$. The result is stored in an array at position r. Then successively compute r^i and its inverse $(r^{-1})^i$ for $i = 2, \ldots, q-2$. This has order $\mathcal{O}(q)$ operations.

Thus, the table of inverse elements can be created using $\mathcal{O}(q)$ operations.

To obtain the edges, we need to evaluate the p Möbius transformations $\phi_i^{p,q}$, $i = 1, \ldots, p$ for each of the $q+1$ vertices. Using the precomputed values, a single Möbius transformation can be evaluated in constant time. Thus, we obtain $\mathcal{O}(pq)$ for creating the graph $Y^{p,q}$. □

8.3 Almost Dense Explicit Cubic Ramanujan Graphs

We analyze the family of dense cubic Ramanujan graphs $Y^{2,q}$ in more detail and propose a construction of simple graphs $G^{2,q}$ based on them. The multigraphs $Y^{2,q}$ are constructed using the following Möbius transformations, which generate graphs on the vertex set $\mathbb{Z}_q \cup \{\infty\}$:

$$\phi_b(z) = -z,\ \phi_r(z) = \frac{(2-\sqrt{-2})z + \sqrt{-26}}{\sqrt{-26}z + (2+\sqrt{-2})},\ \phi_g(z) = \frac{(2+\sqrt{-2})z - \sqrt{-26}}{-\sqrt{-26}z + (2-\sqrt{-2})}.$$

Recall that the graphs are constructed as directed graphs, having edges from z to $\phi_i(z)$, $i \in \{r, g, b\}$. In this section we use color coded edges to illustrate which Möbius transformation is used for a particular edge. Starting from a vertex z, we have a blue edge $z \rightarrow \phi_b(z)$, a red edge $z \rightarrow \phi_r(z)$ and a green edge $z \rightarrow \phi_g(z)$. Note that, if there is a blue edge $z \rightarrow y$, then there is a blue edge $z \leftarrow y$ as well, since $\phi_b(\phi_b(z)) = \phi_b(-z) = z$. On the other hand, a red edge $z \rightarrow y$ implies a green edge $z \leftarrow y$, since $\phi_g(\phi_r(z)) = z$. Thus, the graph can be reinterpreted as an undirected graph, while each undirected edge corresponds to two directed edges, one in each direction. Note, however, that either two blue edges or a red and a green edge form an undirected edge.

Theorem 8.10. *The graph $Y^{2,q}$ has*

1. *a self-loop at vertex* 0 *and at vertex* ∞*,*
2. *if $\sqrt{-7} \in \mathbb{Z}_q$, another two self-loops at each of the two vertices $z_{1/2} = -\frac{\sqrt{-2}\pm\sqrt{-28}}{\sqrt{-26}}$,*
3. *if $\sqrt{-15} \in \mathbb{Z}_q$, with $z_{3/4} = \frac{2\pm\sqrt{30}}{\sqrt{-26}}$, a double edge between the vertices $\{z_3, -z_3\}$ and $\{z_4, -z_4\}$.*

There are no further self-loops or multi-edges.

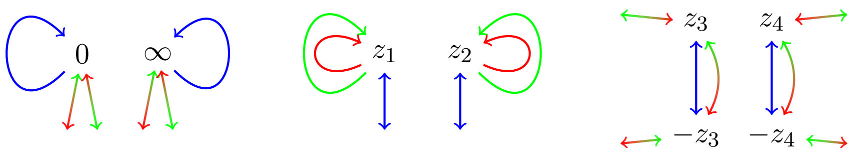


Figure 8.1: All possible self-loops and multi-edges in $Y^{2,q}$.

Proof. Self-loops occur exactly at the vertices which correspond to solutions of the equations $\phi_b(z) = z$, $\phi_r(z) = z$, and $\phi_g(z) = z$. For determining the multi-edges, we solve the equations $\phi_b(z) = \phi_r(z)$, $\phi_b(z) = \phi_g(z)$ and $\phi_r(z) = \phi_g(z)$. A straight forward computation directly leads to the results stated in the theorem:

- Case $\phi_b(z) = z$
  It holds $-z = z$ and thus, we have self-loops at $z = 0$ and $z = \infty$.

- Case $\phi_r(z) = z$

$$\begin{aligned}
\Leftrightarrow \quad & \frac{(2-\sqrt{-2})z + \sqrt{-26}}{\sqrt{-26}z + (2+\sqrt{-2})} = z \\
\Leftrightarrow \quad & (2-\sqrt{-2})z + \sqrt{-26} = z(\sqrt{-26}z + (2+\sqrt{-2})) \\
\Leftrightarrow \quad & (2-\sqrt{-2})z + \sqrt{-26} = \sqrt{-26}z^2 + (2+\sqrt{-2})z \\
\Leftrightarrow \quad & 0 = z^2 + \frac{2\sqrt{-2}}{\sqrt{-26}}z - 1 \\
\Rightarrow \quad & z_{1/2} = -\frac{\sqrt{-2} \pm \sqrt{-28}}{\sqrt{-26}}.
\end{aligned}$$

  We have self-loops at z_1 and z_2, iff $\sqrt{-28} = 2\sqrt{-7}$ exists. Note that $\sqrt{-2}$ and $\sqrt{13}$ are guaranteed to exist due to the choice of q.

- Case $\phi_g(z) = z$

$$\begin{aligned}
\Leftrightarrow \quad & \frac{(2+\sqrt{-2})z - \sqrt{-26}}{-\sqrt{-26}z + (2-\sqrt{-2})} = z \\
\Leftrightarrow \quad & (2+\sqrt{-2})z - \sqrt{-26} = z(-\sqrt{-26}z + (2-\sqrt{-2})) \\
\Leftrightarrow \quad & (2+\sqrt{-2})z - \sqrt{-26} = -\sqrt{-26}z^2 + (2-\sqrt{-2})z \\
\Leftrightarrow \quad & 0 = z^2 + \frac{2\sqrt{-2}}{\sqrt{-26}}z - 1.
\end{aligned}$$

  This is the same equation as in the case above, so z_1 and z_2 also have self-loops from ϕ_g.

- Case $\phi_b(z) = \phi_r(z)$

$$\Leftrightarrow \quad -z = \frac{(2-\sqrt{-2})z + \sqrt{-26}}{\sqrt{-26}z + (2+\sqrt{-2})}$$

$$\begin{aligned}
\Leftrightarrow\quad & -z(\sqrt{-26}z + (2+\sqrt{-2})) = (2-\sqrt{-2})z + \sqrt{-26} \\
\Leftrightarrow\quad & -\sqrt{-26}z^2 - (2+\sqrt{-2})z = (2-\sqrt{-2})z + \sqrt{-26} \\
\Leftrightarrow\quad & 0 = z^2 + \frac{4}{\sqrt{-26}}z + 1 \\
\Rightarrow\quad & z_{3/4} = -\frac{2 \pm \sqrt{30}}{\sqrt{-26}}.
\end{aligned}$$

We have a double edge between $\{z_3, \phi_b(z_3)\}$ and $\{z_4, \phi_b(z_4)\}$.

- Case $\phi_b(z) = \phi_g(z)$

$$\begin{aligned}
\Leftrightarrow\quad & -z = \frac{(2+\sqrt{-2})z - \sqrt{-26}}{-\sqrt{-26}z + (2-\sqrt{-2})} \\
\Leftrightarrow\quad & -z(-\sqrt{-26}z + (2-\sqrt{-2})) = (2+\sqrt{-2})z - \sqrt{-26} \\
\Leftrightarrow\quad & \sqrt{-26}z^2 - (2-\sqrt{-2})z = (2+\sqrt{-2})z - \sqrt{-26} \\
\Leftrightarrow\quad & 0 = z^2 - \frac{4}{\sqrt{-26}}z + 1.
\end{aligned}$$

This equation has solutions $-z_3$ and $-z_4$.

- Case $\phi_r(z) = \phi_g(z)$

$$\begin{aligned}
\Leftrightarrow\quad & \frac{(2-\sqrt{-2})z + \sqrt{-26}}{\sqrt{-26}z + (2+\sqrt{-2})} = \frac{(2+\sqrt{-2})z - \sqrt{-26}}{-\sqrt{-26}z + (2-\sqrt{-2})} \\
\Leftrightarrow\quad & -(2-\sqrt{-2})\sqrt{-26}z^2 + 26z + (2-\sqrt{-2})^2 z + \sqrt{-26}(2-\sqrt{-2}) = \\
& (2+\sqrt{-2})\sqrt{-26}z^2 + 26z + (2+\sqrt{-2})^2 z - \sqrt{-26}(2+\sqrt{-2}) \\
\Leftrightarrow\quad & -2\sqrt{-26}z^2 - 4\sqrt{-2}z + 2\sqrt{-26} = \\
& 2\sqrt{-26}z^2 + 4\sqrt{-2}z - 2\sqrt{-26} \\
\Leftrightarrow\quad & 0 = z^2 + \frac{2\sqrt{-2}}{\sqrt{-26}}z - 1 \\
\Rightarrow\quad & z_{1/2} = -\frac{\sqrt{-2} \pm \sqrt{-28}}{\sqrt{-26}}.
\end{aligned}$$

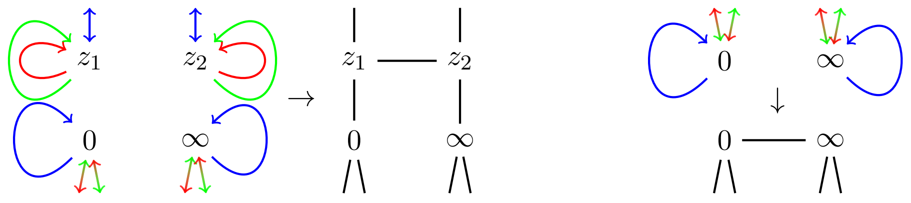
This case coincides with the self-loop cases $\phi_r(z) = z$ and $\phi_g(z) = z$.

□

In order to create a simple Ramanujan graph $G^{2,q}$ with $q+1$ vertices, we define the following graph construction:

Definition 8.11. *Let $q > 13$ be a prime with $\sqrt{-2} \in \mathbb{Z}_q$, $\sqrt{13} \in \mathbb{Z}_q$, and $\sqrt{-15} \notin \mathbb{Z}_q$. Graph $G^{2,q}$ is constructed as follows: start with the multigraph $Y^{2,q}$. Then remove all self-loops. To obtain a cubic graph, perform the following further steps:*

1. *If $\sqrt{-7} \in \mathbb{Z}_q$, then with $z_{1/2} = -\frac{\sqrt{-2}\pm\sqrt{-28}}{\sqrt{-26}}$, add the edges $\{z_1, z_2\}$, $\{z_1, 0\}$, and $\{z_2, \infty\}$.*
2. *If $\sqrt{-7} \notin \mathbb{Z}_q$, add the edge $\{0, \infty\}$.*



Note that we replaced the condition of Chiu, q to be a prime unequal 2 and 13, by the condition prime $q > 13$. Due to the other constraints, this only eliminates the choice of $q = 3$.

Theorem 8.12. *Let $q > 13$ be a prime with $\sqrt{-2}, \sqrt{13} \in \mathbb{Z}_q$ and $\sqrt{-15} \notin \mathbb{Z}_q$ as used in the construction of the graphs $G^{2,q}$. Then $G^{2,q}$ is simple, cubic, and Ramanujan.*

Proof. Since by assumption $\sqrt{-15} \notin \mathbb{Z}_q$, we know from Theorem 8.10 that there are no multi-edges, except for the self-loops. Now, if $\sqrt{-7} \notin \mathbb{Z}_q$, there are only two self-loops at 0 and ∞. We need to show that replacing them by an edge $\{0, \infty\}$ does not create a multi-edge. We examine where the two edges starting from vertex 0 lead to: this is $\phi_r(0) = \sqrt{-26}/(2+\sqrt{-2})$ and $\phi_g(0) = -\sqrt{-26}/(2-\sqrt{-2})$. For contradiction, assume $\phi_r(0) = \infty$ or $\phi_g(0) = \infty$. Then

$$2 \pm \sqrt{-2} = 0 \quad \Rightarrow \quad -2 = \pm\sqrt{-2} \quad \Rightarrow \quad 4 = -2 \quad \Rightarrow \quad 6 = 0.$$

Since $q > 13$, this is a contradiction.

Next, we consider $\sqrt{-7} \in \mathbb{Z}_q$, where we place three new edges. At vertex z_1, we remove two self-loops and add two new edges. We need to show that none of these two edges coincide with the already existing blue edge. The blue edge leads to $\phi_b(z_1) = -z_1$. Let us assume this would be vertex 0, such that $-z_1 = 0$. Then

$$\frac{\sqrt{-2}+\sqrt{-28}}{\sqrt{-26}} = 0 \quad \Rightarrow \quad \sqrt{-2}+\sqrt{-28} = 0 \;\Rightarrow\; -2 = -28 \;\Rightarrow\; 0 = -26.$$

Since $q > 13$, this is a contradiction. Let us assume the blue edge leads from z_1 to z_2. Then $\phi(z_1) = z_2$ and

$$\frac{\sqrt{-2}+\sqrt{-28}}{\sqrt{-26}} = -\frac{\sqrt{-2}-\sqrt{-28}}{\sqrt{-26}} \quad \Rightarrow \quad 2\sqrt{-2} = 0 \quad \Rightarrow \quad -2 = 0.$$

Again a contradiction. It remains to show that the blue edge at z_2 does not coincide with the new edge $\{z_2, \infty\}$. For contradiction, assume $\phi_b(z_2) = \infty$.

$$\frac{\sqrt{-2}-\sqrt{-28}}{\sqrt{-26}} = \infty \quad \Rightarrow \quad \sqrt{-26} = 0 \quad \Rightarrow \quad -26 = 0.$$

We conclude that none of the added edges have been in the graph before, such that the graph $G^{2,q}$ is cubic and simple.

Last, we show that $G^{2,q}$ is Ramanujan. We know that $Y^{2,q}$ is Ramanujan. Let A_G and A_Y denote the adjacency matrices of these two graphs. We compute the second largest eigenvalue of A_G for the case $\sqrt{-7} \notin \mathbb{Z}_q$, while x_0 and x_∞ denote the vector components corresponding to the vertices labeled by 0 and ∞, respectively.

$$\begin{aligned}
\lambda_2(A_G) &= \max_{x \perp 1} \frac{x^T A_G x}{\|x\|^2} \\
&= \max_{x \perp 1} \frac{x^T A_Y x - x_0^2 - x_\infty^2 + 2x_0x_\infty}{\|x\|^2} \\
&= \max_{x \perp 1} \frac{x^T A_Y x - (x_0 - x_\infty)^2}{\|x\|^2} \\
&\leq \max_{x \perp 1} \frac{x^T A_Y x}{\|x\|^2} = \lambda_2(A_Y).
\end{aligned}$$

Thus, $\lambda_2(A_G) \leq \lambda_2(A_Y)$ and since $Y^{2,q}$ is Ramanujan, $G^{2,q}$ is Ramanujan as well.

For the case $\sqrt{-7} \in \mathbb{Z}_q$, we have

$$\begin{aligned}
&\lambda_2(A_G)\\
&= \max_{x \perp 1} \frac{x^T A_G x}{\|x\|^2}\\
&= \max_{x \perp 1} \frac{x^T A_Y x - x_0^2 - x_\infty^2 - 2x_{z1}^2 - 2x_{z2}^2 + 2x_0 x_{z1} + 2x_{z1}x_{z2} + 2x_{z2}x_\infty}{\|x\|^2}\\
&= \max_{x \perp 1} \frac{x^T A_Y x - (x_0 - x_{z1})^2 - (x_{z1} - x_{z2})^2 - (x_{z2} - x_\infty)^2}{\|x\|^2}\\
&\leq \max_{x \perp 1} \frac{x^T A_Y x}{\|x\|^2} = \lambda_2(A_Y).
\end{aligned}$$

□

Theorem 8.13. *For all x, there are $\Theta(x/\log(x))$ many primes $q < x$ with $\sqrt{-2}, \sqrt{13} \in \mathbb{Z}_q$ and $\sqrt{-15} \notin \mathbb{Z}_q$. This means there are $\Theta(x/\log(x))$ graphs $G^{2,q}$ with $q+1 < x+1$ vertices.*

For the proof, we need the following

Fact 8.14 (see e. g. [4])**.** *For every odd prime p, it holds:*

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2} = \begin{cases} 1 & \text{if } p \equiv 1 \mod 4 \\ -1 & \text{if } p \equiv 3 \mod 4, \end{cases}$$

$$\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8} = \begin{cases} 1 & \text{if } p \equiv \pm 1 \mod 8 \\ -1 & \text{if } p \equiv \pm 3 \mod 8. \end{cases}$$

Now we are ready to prove Theorem 8.13:

Proof. The proof is similar to the argumentation by Chiu in [19]. We consider primes of the form $q = 104 \cdot 30m + 16 \cdot 13 + 1$ with $m \in \mathbb{N}$. Let $\pi(x)$ denote the number of primes $\leq x$ and let $\pi_a(x)$ denote the number of primes $\leq x$ of the form $km + a$, $m = 1, 2, \ldots$. Further, denote by $\varphi(k)$ Euler's totient function, i. e. the number of relative primes smaller than k. Using the prime number theorem for arithmetic progression, it holds

$$\pi_a(x) \sim \frac{\pi(x)}{\varphi(k)} \sim \frac{1}{\varphi(k)} \frac{x}{\log x}, \quad x \to \infty,$$

if the largest common divisor $(a, k) = 1$ [4]. In our case, for $k = 104 \cdot 30$ and $a = 16 \cdot 13 + 1$, we have $\varphi(104 \cdot 30) = 768$ and thus

$$\pi_{16\cdot 13+1}(x) \sim \frac{1}{768}\frac{x}{\log x}.$$

We now need to show that -2 and 13 are quadratic residues modulo q and -15 is a nonresidue. From Fact 8.14 follows directly that -2 is quadratic residue modulo q:

$$\left(\frac{-2}{q}\right) = \left(\frac{-1}{q}\right)\left(\frac{2}{q}\right) = 1 \cdot 1 = 1.$$

Using the quadratic reciprocity law $\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{(p-1)(q-1)/4}$, we obtain

$$\begin{aligned}\left(\frac{13}{q}\right) &= \left(\frac{q}{13}\right)(-1)^{(13-1)(q-1)/4} = \left(\frac{q}{13}\right)(-1)^{3(q-1)} = \left(\frac{q}{13}\right)(-1)^{(q-1)}\\ &= \left(\frac{q}{13}\right)(-1)^{104\cdot 30m+16\cdot 13} = \left(\frac{q}{13}\right)\\ &\equiv q^{(13-1)/2} \mod 13\\ &\equiv (104\cdot 30m + 16\cdot 13 + 1 \mod 13)^6 \mod 13\\ &\equiv 1 \mod 13.\end{aligned}$$

Hence, 13 is quadratic residue modulo q.

For -15, we first compute the following:

$$\begin{aligned}\left(\frac{5}{q}\right) &= \left(\frac{q}{5}\right)(-1)^{(5-1)(q-1)/4} = \left(\frac{q}{5}\right)(-1)^{104\cdot 30m+16\cdot 13} = \left(\frac{q}{5}\right)\\ &\equiv q^{(5-1)/2} \mod 5 \equiv q^2 \mod 5\\ &= (104\cdot 30m + 16\cdot 13 + 1 \mod 5)^2 \mod 5 \equiv 1 \mod 5,\end{aligned}$$

$$\begin{aligned}\left(\frac{3}{q}\right) &= \left(\frac{q}{3}\right)(-1)^{(5-1)(q-1)/4} = \left(\frac{q}{3}\right)(-1)^{104\cdot 30m+16\cdot 13} = \left(\frac{q}{5}\right)\\ &\equiv q^{(3-1)/2} \mod 3 \equiv q \mod 3\\ &= 104\cdot 30m + 16\cdot 13 + 1 \mod 3 \equiv -1 \mod 3.\end{aligned}$$

Now,

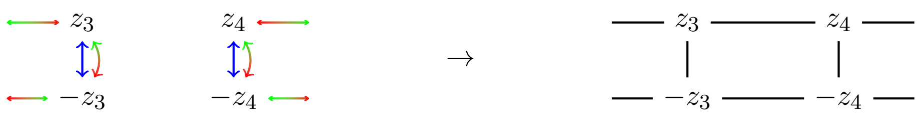
$$\left(\frac{-15}{q}\right) = \left(\frac{-1}{q}\right)\left(\frac{3}{q}\right)\left(\frac{5}{q}\right) = 1 \cdot (-1) \cdot 1 = -1.$$

Hence, -15 is a nonresidue. □

In the definition of the graph class $G^{2,q}$ (Definition 8.11), we require $\sqrt{-15} \notin \mathbb{Z}_q$ additionally to the constraints for $Y^{2,q}$. However, we can extend the definition of $G^{2,q}$ to allow q with $\sqrt{-15} \in \mathbb{Z}_q$ as follows:

Definition 8.15. *Let $q > 13$ be a prime with $\sqrt{-2} \in \mathbb{Z}_q$, $\sqrt{13} \in \mathbb{Z}_q$. For $\bar{G}^{2,q}$, use the construction from Definition 8.11, extended by*

3. If $\sqrt{-15} \in \mathbb{Z}_q$, then with $z_{3/4} = \frac{2 \pm \sqrt{30}}{\sqrt{-26}}$, add the edges $\{z_3, z_4\}$ and $\{-z_3, -z_4\}$.



We are now able to show that the resulting graphs are simple and cubic:

Theorem 8.16. *Let $q > 13$ be a prime with $\sqrt{-2}, \sqrt{13} \in \mathbb{Z}_q$. Then $\bar{G}^{2,q}$ is simple and cubic.*

Proof. Most of the proof is covered by Theorem 8.12. We are left to discuss the changes at the vertices $\pm z_3$ and $\pm z_4$, i. e. that the edges $\{z_3, z_4\}$ and $\{-z_3, -z_4\}$ do not exist in $Y^{2,q}$. Consider z_3: since $\phi_b(z_3) = \phi_r(z_3) = -z_3$, we only have to check if $\phi_g(z_3) \neq z_4$. For contradiction, assume $\phi_g(z_3) = z_4$. We apply ϕ_r and obtain $\phi_r(\phi_g(z_3)) = \phi_r(z_4)$. Now the left hand side is z_3 again, since red and green are opposite to each other, and the right hand side is $-z_4$ in the considered case. Thus, $z_3 = -z_4$ and

$$\frac{2 + \sqrt{30}}{\sqrt{-26}} = -\frac{2 - \sqrt{30}}{\sqrt{-26}} \quad \Rightarrow \quad 4 = 0,$$

a contradiction. Analogously, there is no edge $\{-z_3, -z_4\}$ in $Y^{2,q}$. $\square$

Unfortunately, the author was not able to show that the graphs $\bar{G}^{2,q}$ are all Ramanujan. However, we can show that they have a bisection width which is at least as large as the one of $Y^{2,q}$:

Theorem 8.17. *Let $q > 13$ be a prime with $\sqrt{-2}, \sqrt{13} \in \mathbb{Z}_q$. Then* $\mathrm{bw}(\bar{G}^{2,q}) \geq \mathrm{bw}(Y^{2,q})$.

Proof. Let y be an optimum bisection vector for graph $G^{2,q}$. We show that there exists a bisection of the same or smaller width in $Y^{2,q}$. Therefore, we analyze how $G^{2,q}$ is changed back to obtain $Y^{2,q}$. For modification 1, the three edges $\{0, z_1\}$, $\{z_1, z_2\}$, and $\{z_2, \infty\}$ are removed and replaced by several self-loops. These modifications clearly do not increase the width of any bisection. The same applies for modification 2, where the edge $\{0, \infty\}$ is replaced by two self-loops. We are left to discuss modification 3, where edges between the vertices $z_3, -z_3, z_4, -z_4$ are rearranged. We discuss all possible labellings for these four vertices, and since y and $-y$ describe the same bisection, we say z_3 has label 1. For $z_3, -z_3, z_4, -z_4$ being labeled as $(1, 1, 1, 1)$, the cutwidth does not change with the modifications. The same applies if one or three vertices are labeled with -1, since we cut two edges before and after within our four vertices. Thus, we are left with the labellings $(1, 1, -1, -1)$, $(1, -1, -1, 1)$ and $(1, -1, 1, -1)$. In the first case, the cutwidth of y for $Y^{2,q}$ is smaller by two edges, in the second case the width is the same. Finally, we discuss $(1, -1, 1, -1)$. Considering $G^{2,q}$, we cut two edges within the four vertices. When we change the labeling to $(1, 1, -1, -1)$ and transform our graph to $Y^{2,q}$, no edge is cut within the four vertices anymore. On the other hand, the new labeling might cut the two other edges adjacent to $-z_3$ and z_4. However, in total the number of newly cut edges is at most two, while we removed two cut edges. Thus, the cutwidth less or equal. □

8.4 Computing the Bisection Width of Ramanujan Graphs

In this section we experimentally estimate the bisection width of our constructive Ramanujan graphs $G^{2,q}$ and compare it with the best known lower and upper bounds for largest bisection width. This will provide an idea how large the actual worst case bisection width might be, since the exact bisection width is not even known for explicit Ramanujan graph constructions such as the one of Chiu. A way to determine an optimum bisection is to explicitly construct the graph and then formulate an Integer Linear Program (ILP) for the Minimum Bisection Problem on this specific instance.

A standard approach would be to introduce a variable $x_v \in \{0, 1\}$ for each vertex v, indicating the part the vertex belongs to, and a variable $e_{u,v} \in \{0, 1\}$ for each edge $\{u, v\}$, indicating if the edge is a cut edge. The objective function would then be to minimize the number of cut edges, i. e. $\min \sum_{\{u,v\}\in E} e_{u,v}$.

We obtain the following ILP:

$$\min \sum_{\{u,v\}\in E} e_{u,v} \quad \text{w.r.t.} \sum_{v\in V} x_v = n/2$$
$$\text{for each } \{u,v\} \in E: \quad e_{u,v} \geq x_u - x_v \quad \text{and} \quad e_{u,v} \geq x_v - x_u,$$
$$e_{u,v},\ x_v \in \{0,1\}.$$

In an optimal solution, the variable $e_{u,v}$ is 1 iff the edge $\{u,v\}$ is a cut edge. The value of the objective function then corresponds to the bisection width.

However, we use another, less convenient way to formulate the bisection problem as ILP, which turned out to perform better in the upcoming experiments. In regular graphs, the number of edges $\{u,v\}$ with $u,v \in V_1$ is the same as the number of edges $\{u,v\}$ with $u,v \in V_2$ for every bisection $V = V_1 \dot\cup V_2$: we have the same number of valencies in both sets and every cut edge binds one valency in each set. Consequently, in both sets the same number of valencies has to be satisfied by edges within the parts. Furthermore, each edge is either a cut edge or an inner edge. Using these observations, the problem of minimizing the number of cut edges is equivalent to maximizing the number of inner edges of lets say part V_1. In our ILP, we use as above the variables x_v and $e_{u,v}$, but this time $e_{u,v}$ is constructed to be 1 iff the edge $\{u,v\}$ is an inner edge of part V_1. We obtain the following ILP:

$$\min |E| - 2 \sum_{\{u,v\}\in E} e_{u,v} \quad \text{w.r.t.} \sum_{v\in V} x_v = n/2 \tag{8.6}$$
$$\text{for each } \{u,v\} \in E: e_{u,v} \leq x_u \quad \text{and} \quad e_{u,v} \leq x_v,$$
$$e_{u,v},\ x_v \in \{0,1\}.$$

Solvers for linear programs are highly developed and optimized. They can also prove the solution to be optimal, which is not the case for e. g. doing a simple random search. However, this simple approach comes to an end for even small graphs $G^{2,q}$ in the order of 100 vertices, since the ILP solver does not terminate in a reasonable time. For graphs with around 1000 vertices, we can even observe that the best solution found within the first hour does not even come close to the solutions found by the algorithms we present below. Thus, we need a more sophisticated approach.

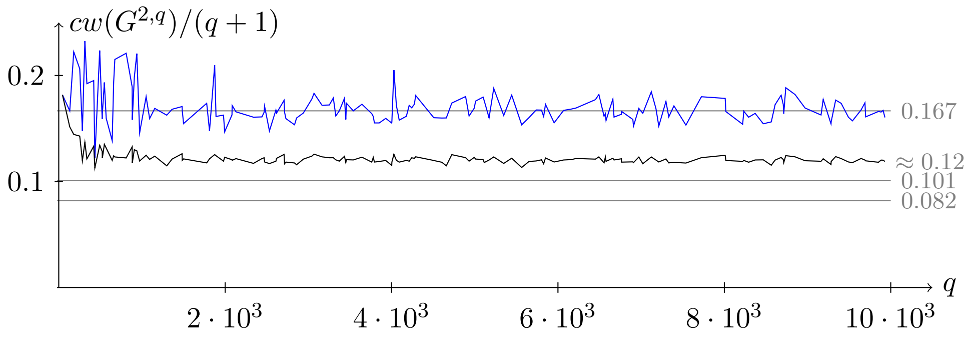
As a starting point, we use a spectral based heuristic which is commonly known to work well on many graph classes. For a given graph G, the bisection is derived from the eigenvector corresponding to the second largest eigenvalue of the adjacency matrix of G, as described in Algorithm 8.1.

```
Algorithm 8.1: Second-Eigenvalue-based Heuristic
  Input: Graph $G$ with adjacency matrix $A$.
1 Compute the eigenvector $x$ to the second largest eigenvalue of the
    adjacency matrix $A$.
2 Splitting at the median $\bar{x}$ of vector $x$, we put all vertices $i$ with $x_i \geq \bar{x}$
    to part $V_1$ and all other vertices to part $V_2$. If $|V_1| > |V_2|$, we move
    arbitrarily $(|V_1| - |V_2|)/2$ vertices from $V_1$ to $V_2$. Output the partition
    $(V_1, V_2)$.
```

Next, we combine the second-eigenvalue-based heuristic with the idea of using an ILP in Algorithm 8.2. We start with the spectral based heuristic and then perform a greedy improvement. We then formulate the ILP as in Eq. (8.6) but force some vertices to be in a certain part. For this we use the bisection obtained after the greedy improvement has been performed, and compute for each vertex its distance to the cut, starting with distance 1 for vertices incident with a cut edge. All vertices with a distance of 3 or more will be fixed in the ILP. Thus, the ILP only decides where to put vertices with distance 1 or 2. This choice was made empirically for our graphs and results in fixing about one-third of the vertices. Fixing less vertices results in worse solutions of the ILP within the given time.

```
Algorithm 8.2: Spectral-ILP Heuristic
  Input: Graph $G$ with adjacency matrix $A$. Timeout $t_1$.
1 Use Algorithm 8.1 to obtain an initial partition $(V_1, V_2)$
2 Perform a greedy improvement: find a vertex $u$ in $V_1$ which has more
    neighbors in $V_2$ than in $V_1$. Next, find a vertex $v$ in $V_2$ which has more
    neighbors in $V_1$ than in $V_2$. Move $u$ to $V_2$ and $v$ to $V_1$. Repeat, until
    no such vertices are found anymore.
3 Compute the distance of each vertex to the cut: vertices adjacent to a
    cut edge have distance 1, and all other vertices have a recursively
    defined distance of the minimum of its neighbors plus 1. Formulate the
    ILP given in Eq. (8.6), and then add constraints which set all vertices
    with distance of 3 or more to the part determined by the previously
    computed partition. Solve the ILP with timeout $t_1$. Output the best
    solution found.
```

In our experiments, we generated the cubic Ramanujan graph $G^{2,q}$ and cubic random graphs with up to 10000 vertices. We then used the Algorithms 8.1



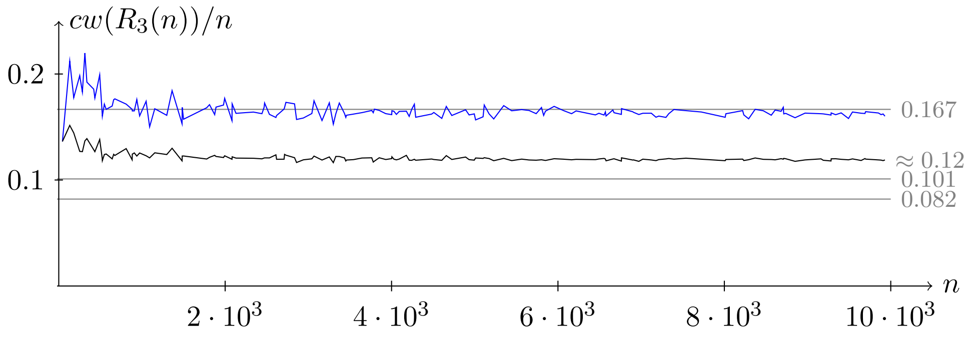


Figure 8.2: Bisection estimates for cubic Ramanujan graphs $G^{2,q}$ (upper) and cubic random graphs $R_3(n)$ (lower), obtained via the Second-Eigenvalue-based Heuristic from Algorithm 8.1 (blue) and the Spectral-ILP Heuristic from Algorithm 8.2 (black).

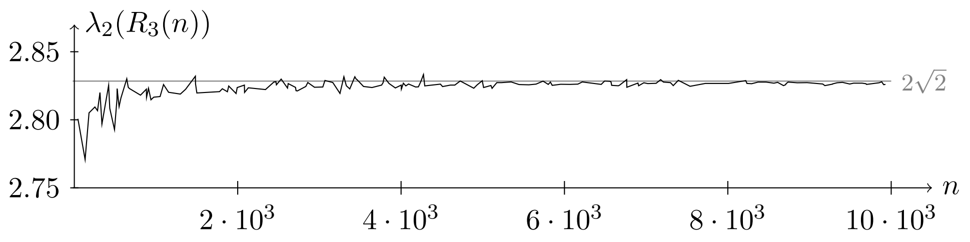


Figure 8.3: Second largest eigenvalues $\lambda_2(R_3(n))$ of the cubic random graphs $R_3(n)$ used in Figure 8.2. Graphs with $\lambda_2 \leq 2\sqrt{2}$ are Ramanujan.

and 8.2 to estimate the bisection width. The results can be found in Figure 8.2. There we plot the relative bisection width $\mathrm{bw}(G)/n$. For the ILP step we chose a timeout t_1 of $n/100$ seconds, i. e. linear in the number of vertices. For small graphs, we observed that a nearly-optimal solution was found very quickly, so we believe that the bisection widths are nearly-optimal for all graphs in our experimental analysis. However, the found bisections provide an upper bound on the actual bisection width. Interestingly, the determined relative bisection width does not increase with increasing n, which indicates that the quality of the solution found by Algorithm 8.2 remains good even for large n. We might add another step to Algorithm 8.2: first, we start the ILP solver on the restricted ILP using the timeout as described in the algorithm. Thereafter, as an additional step we remove the distance-based restrictions and resume the ILP solver. However, we could observe no substantial improvement even with a timeout of one hour for this additional step.

Except for the graphs with only very few vertices, the ratio of the bisection width and the number of vertices seems to be about the constant of 0.12. For comparison we did the same experiment for random cubic graphs. We can observe roughly the same behavior and obtain the same estimation of the constant of 0.12. Let us note that most of the random graphs are Ramanujan as well: Figure 8.3 shows the second largest eigenvalue of the adjacency matrices of the random cubic graphs from our experiment. Eigenvalues below $2\sqrt{2}$ indicate that the corresponding graph is Ramanujan.

9 Discussion and Open Problems

In this thesis we have seen that Boppana's Algorithm BB is a powerful method for graph bisection with certifying the optimality of a solution. We provided a full proof that the algorithm obtains a tight lower bound on the planted bisection model w.h.p. and proposed several modifications to guarantee that the algorithm also finds an optimum bisection. We were able to implement the algorithms and demonstrated that they work well even for small graphs. We conducted further experiments on the graph model $\mathcal{R}_n(r,b)$, which indicated that Boppana's algorithm also works for $r=5$ but not for $r=3$ and $r=4$. An interesting question arising is, which properties of 3- and 4-regular graphs from the planted bisection model let the algorithm fail.

Further, we proved that the SDP approach due to Feige and Kilian [31] is equivalent to Boppana's approach. More precisely the dual SDP from Feige and Kilian can be seen as formulation of the eigenvalue problem contained in Boppana's algorithm. Although their primal SDP uses about n^2 variables and the eigenvalue based approach only n variables, our SDP implementation of FK* has shown to be much faster and was able to solve larger instances than our implementation of BB. The explanation of this issue is that Boppana's Algorithm BB needs to solve a huge amount of eigenvalue problems within the convex optimization. The limitation of the graph sizes the SDP in FK* can handle is indeed dictated by the large number of variables. It is an interesting task if one can find a solution to the dual SDP with n variables without solving the primal SDP explicitly. Another question is if the eigenvalue optimization can be done better and obtain similar performance as FK* but without using the primal SDP.

There are still open questions about the interpretation of the variable matrix X of the primal SDP, when trying to relate X to the vector d in the dual SDP as well as to the eigenvalue problem, and there especially to the eigenvectors corresponding to the largest eigenvalue. We know that e. g. in the planted bisection model $\mathcal{G}_n(p,q)$ with appropriate parameters p and q, the algorithms BB and FK* find and certify with high probability the optimum bisection. In this case, an optimal vector d can be derived from the bisection vector y as $d = -\operatorname{diag}(y)Ay$. We also know that other optimal d might exist, which must be of the form $d = -\operatorname{diag}(y)Ay + \alpha y$. The eigenvector to the unique largest eigenvalue of the matrix B constructed in Boppana's algorithm is then a stretched vector y. For the primal SDP it is known that the matrix X is a matrix with 1 or -1 entries, indicating if two vertices belong to the same or to different parts. This gives us for both approaches a reconstruction possibility. However, if we have a graph where the bound $h(G)$ is not tight, i. e. $h(G) < \operatorname{bw}(G)$, can we then use an optimal solution to Boppana's approach to obtain an optimal solution for the primal SDP?

The SDP implementation of FK* was able to solve graph instances only up to 2000 vertices. To overcome this issue, we proposed a new heuristic FB, which tries to guess an optimal vector d^{opt} for Boppana's approach and then verify and certify the solution. This heuristic showed to be practically implementable as well and was able to solve instances with up to 10^6 vertices. Our new heuristic FB finds on graphs from $\mathcal{G}_n(p,q)$ and $\mathcal{R}_n(r,b)$ almost the same (optimal) bisections as BB. It would be interesting to find a better candidate for the correction vector than $d^{(\hat{x})}$, as used in FB, but which still does not require an optimization search for d^{opt}. Another interesting future work would be to analyze theoretical properties of FB. We conjecture that for random graphs G from $\mathcal{G}_n(p,q)$, with $p-q \geq c(\sqrt{p\log n}/\sqrt{n})$, our Algorithm FB certifies the optimality revealing w.h.p. the bisection vector $\hat{x}$ of $\operatorname{cw}(\hat{x}) - g(G, -\operatorname{diag}(\hat{x})A\hat{x}) < 1$.

The heuristics FB, FB*, BB and FK handle graphs with bisection width not too large, but for certain graph classes with bisection width $\Theta(n)$ they do not work at all. For the graphs considered in Chapter 8 for example, the bisection obtained by these algorithms has a much larger width than the best solution we found by Algorithm 8.2. However, when considering minimum bisections the question for the worst case of bisection width arises. In order to analyze graphs with largest possible bisection width, we focused on explicitly constructed Ramanujan graphs in comparison to random regular graphs. We tried to estimate the asymptotic largest bisection width of this graph classes. Based on our experimental analysis for the bisection width of cubic Ramanujan graphs, we conjecture the lower bound to be around $0.12n$ for the graph class $G^{2,q}$. We observed the same result for 3-regular random graphs. This motivates further research to improve the best known lower bound $0.082n$ on the worst case bisection width by Bezrukov et al. [7] (see Eq. (8.2)) and simultaneously shows that the lower bound by Kostochka and Melnikov [48] (see Eq. (8.1)) holds already for Ramanujan graphs. It is an interesting challenge to prove the conjecture that $0.12n \leq \operatorname{bw}_{3,R}(n)$. Since for any graph its bisection width is less than or equal to the pathwidth of the graph, this would also imply that the largest possible pathwidth of an n-vertex cubic graph is at least $0.12n$. Note that due to Fomin and Høie [34], we know that the upper bound on the pathwidth is $0.167n$. Another interesting question, we leave open, is to prove that our graphs $\bar{G}^{2,q}$ are Ramanujan.

For determining the bisection width of our Ramanujan graphs, we used an algorithm combining spectral techniques, greedy improvements and ILP optimization. This heuristic seems to perform well, but a thorough analysis on its quality is still an open task. Since we have an explicitly constructed graph class of cubic Ramanujan graphs, it is an interesting question if the bisection width of these graphs can be computed or estimated analytically.

In this work we restricted ourselves to graph with an even number of vertices and the task to obtain equally sized bisections. A more general formulation of the problem allows for odd vertex degree. The partition should then have parts which differ at most by 1 in size. A direct application of the algorithms discussed in this work is not possible, since we need to obtain bisection vectors $y \in \{1, -1\}^n$ which sum up to 0. An open task is to modify the algorithms and allow for odd numbers of vertices, e. g. by carefully adding another vertex.

An even more general setting could allow the parts to differ in sizes by some relative factor c, i. e. instead of a bisection we might search for a partition V_1, V_2 with $||V_1| - |V_2|| < cn$. For this problem, an algorithmic approach could combine the spectral techniques from this work with decomposition techniques, e. g. as used in [24].

Instead of minimizing the number of edges crossing a bisection, we can ask to maximize them, i. e. consider the Maximum Bisection Problem. This problem is NP-hard but allows e. g. for a polynomial-time approximation scheme (PTAS) on planar graphs [46].

List of Algorithms

List of Figures

List of Tables

Bibliography

[1] Emmanuel Abbe. Community detection and stochastic block models: recent developments. *CoRR*, abs/1703.10146, 2017.

[2] Emmanuel Abbe, Afonso S. Bandeira, and Georgina Hall. Exact recovery in the stochastic block model. *IEEE Transactions on Information Theory*, 62(1):471–487, 2016.

[3] Farid Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

[4] Tom M. Apostol. *Introduction to Analytic Number Theory*. Undergraduate Texts in Mathematics. Springer New York, 1998.

[5] Michael Armbruster, Marzena Fügenschuh, Christoph Helmberg, and Alexander Martin. A comparative study of linear and semidefinite branch-and-cut methods for solving the minimum graph bisection problem. In *Proceedings of the 13th International Conference on Integer Programming and Combinatorial Optimization, IPCO 2008*, pages 112–124, 2008.

[6] Sanjeev Arora, David R. Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of np-hard problems. *Journal of Computer and System Sciences*, 58(1):193–210, 1999.

[7] Sergei L. Bezrukov, Robert Elsässer, Burkhard Monien, Robert Preis, and Jean-Pierre Tillich. New spectral lower bounds on the bisection width of graphs. *Theoretical Computer Science*, 320(2-3):155–174, 2004.

[8] Sandeep N. Bhatt and Frank Thomson Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28(2):300–343, 1984.

[9] Avrim Blum and Joel Spencer. Coloring random and semi-random k-colorable graphs. *Journal of Algorithms*, 19(2):204–234, 1995.

[10] Robert D. Blumofe. Spectral methods for bisecting graphs. Unpublished Manuscript, 1993.

[11] Béla Bollobás and Alex D. Scott. Max cut for random graphs with a planted partition. *Combinatorics, Probability and Computing*, 13(4-5):451–474, 2004.

[12] Ravi B. Boppana. Eigenvalues and graph bisection: An average-case analysis (extended abstract). In *Proceedings of the 28th Symposium on Foundations of Computer Science, FOCS 1987*, pages 280–285, 1987.

[13] Andrei Z. Broder and Eli Shamir. On the second eigenvalue of random regular graphs (preliminary version). In *Proceedings of the 28th Symposium on Foundations of Computer Science, FOCS 1987*, pages 286–294, 1987.

[14] Lorenzo Brunetta, Michele Conforti, and Giovanni Rinaldi. A branch-and-cut algorithm for the equicut problem. *Mathematical Programming*, 77:243–263, 1997.

[15] Thang Nguyen Bui, Soma Chaudhuri, Frank Thomson Leighton, and Michael Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987.

[16] Ted Carson and Russell Impagliazzo. Hill-climbing finds random planted bisections. In *Proceedings of the 12th Symposium on Discrete Algorithms, SODA 2001*, pages 903–909, 2001.

[17] Yudong Chen, Sujay Sanghavi, and Huan Xu. Clustering sparse graphs. In *Advances in Neural Information Processing Systems 25: 26th Conference on Neural Information Processing Systems, NIPS 2012*, pages 2213–2221, 2012.

[18] Yudong Chen, Sujay Sanghavi, and Huan Xu. Improved graph clustering. *IEEE Transactions on Information Theory*, 60(10):6440–6455, 2014.

[19] Patrick Chiu. Cubic ramanujan graphs. *Combinatorica*, 12(3):275–285, 1992.

[20] Michael B. Cohen. Ramanujan graphs in polynomial time. In *Proceedings of the 57th Symposium on Foundations of Computer Science, FOCS 2016*, pages 276–281, 2016.

[21] Michael B. Cohen. Personal communication, 2017-06-28.

[22] Stephen D. Cohen, Tomás Oliveira e Silva, and Tim Trudgian. On grosswald's conjecture on primitive roots, 2015.

[23] Amin Coja-Oghlan. Spectral techniques, semidefinite programs, and random graphs. *Habilitationsschrift, Humboldt Universität zu Berlin, Institut für Informatik*, 2005.

[24] Amin Coja-Oghlan. A spectral heuristic for bisecting random graphs. *Random Structures and Algorithms*, 29(3):351–398, 2006.

[25] Amin Coja-Oghlan, Oliver Cooley, Mihyun Kang, and Kathrin Skubch. The minimum bisection in the planted bisection model. *Theory of Computing*, 13(1):1–22, 2017.

[26] Anne Condon and Richard M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116–140, 2001.

[27] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Minimum bisection is fixed parameter tractable. In *Proceedings of the 46th Symposium on Theory of Computing, STOC 2014*, pages 323–332, 2014.

[28] Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The PACE 2017 Parameterized Algorithms and Computational Experiments Challenge: The Second Iteration. In *Proceedings of the 12th International Symposium on Parameterized and Exact Computation, IPEC 2017*, pages 30:1–30:12, 2018.

[29] Daniel Delling, Daniel Fleischman, Andrew V. Goldberg, Ilya P. Razenshteyn, and Renato F. Werneck. An exact combinatorial algorithm for minimum graph bisection. *Mathematical Programming*, 153(2):417–458, 2015.

[30] Martin E. Dyer and Alan M. Frieze. The solution of some random np-hard problems in polynomial expected time. *Journal of Algorithms*, 10(4):451–489, 1989.

[31] Uriel Feige and Joe Kilian. Heuristics for semirandom graph problems. *Journal of Computer and System Sciences*, 63(4):639–671, 2001.

[32] Uriel Feige and Robert Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM Journal on Computing*, 31(4):1090–1118, 2002.

[33] Uriel Feige, Robert Krauthgamer, and Kobbi Nissim. Approximating the minimum bisection size (extended abstract). In *Proceedings of the 32nd Symposium on Theory of Computing, STOC 2000*, pages 530–536, 2000.

[34] Fedor V. Fomin and Kjartan Høie. Pathwidth of cubic graphs and exact algorithms. *Information Processing Letters*, 97(5):191–196, 2006.

[35] Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010.

[36] Joel Friedman et al. *Some geometric aspects of graphs and their eigenfunctions.* Princeton University, Department of Computer Science, 1991.

[37] Zoltán Füredi and János Komlós. The eigenvalues of random symmetric matrices. *Combinatorica*, 1(3):233–241, 1981.

[38] M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[39] Andrew Granville and Greg Martin. Prime number races. *The American Mathematical Monthly*, 113(1):1–33, 2006.

[40] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[41] William W. Hager, Dzung T. Phan, and Hongchao Zhang. An exact algorithm for graph partitioning. *Mathematical Programming*, 137(1-2):531–556, 2013.

[42] Bruce E. Hajek, Yihong Wu, and Jiaming Xu. Achieving exact cluster recovery threshold via semidefinite programming. *IEEE Transactions on Information Theory*, 62(5):2788–2797, 2016.

[43] Frank Harary, John P. Hayes, and Horng-Jyh Wu. A survey of the theory of hypercube graphs. *Computers and Mathematics with Applications*, 15(4):277 – 289, 1988.

[44] Paul W. Holland, Kathryn B. Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[45] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

[46] Klaus Jansen, Marek Karpinski, Andrzej Lingas, and Eike Seidel. Polynomial time approximation schemes for MAX-BISECTION on planar and geometric graphs. *SIAM Journal on Computing*, 35(1):110–119, 2005.

[47] Stefan E. Karisch, Franz Rendl, and Jens Clausen. Solving graph bisection problems with semidefinite programming. *INFORMS Journal on Computing*, 12(3):177–191, 2000.

[48] Alexandr V. Kostochka and Leonid S. Melnikov. On a lower bound for the isoperimetric number of cubic graphs. *Probabilistic Methods in Discrete Mathematics, Proceedings of the 3rd International Petrozavodsk Conference, Progress in Pure and Applied Discrete Mathematics*, 1:251–265, 1993.

[49] Vivek Kwatra, Arno Schödl, Irfan A. Essa, Greg Turk, and Aaron F. Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics*, 22(3):277–286, 2003.

[50] Thomas Lengauer. *Combinatorial algorithms for integrated circuit layout.* Springer Science and Business Media, 2012.

[51] Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980.

[52] Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[53] Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Approximation algorithms for semi-random partitioning problems. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012*, pages 367–384, 2012.

[54] Adam Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families I: bipartite ramanujan graphs of all degrees. In *Proceedings of the 54th Symposium on Foundations of Computer Science, FOCS 2013*, pages 529–537, 2013.

[55] Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families IV: bipartite ramanujan graphs of all sizes. In *Proceedings of the*

56th Symposium on Foundations of Computer Science, FOCS 2015, pages 1358–1377, 2015.

[56] Grigorii Aleksandrovich Margulis. Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Problemy peredachi informatsii*, 24(1):51–60, 1988.

[57] Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006.

[58] Frank McSherry. Spectral partitioning of random graphs. In *Proceedings of the 42nd Symposium on Foundations of Computer Science, FOCS 2001*, pages 529–537, 2001.

[59] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, New York, NY, USA, 2005.

[60] Burkhard Monien and Robert Preis. Upper bounds on the bisection width of 3- and 4-regular graphs. *Journal of Discrete Algorithms*, 4(3):475–498, 2006.

[61] Cristopher Moore. The computer science and physics of community detection: Landscapes, phase transitions, and hardness. *CoRR*, abs/1702.00467, 2017.

[62] Moshe Morgenstern. Existence and explicit constructions of $q + 1$ regular ramanujan graphs for every prime power q. *Journal of Combinatorial Theory, Series B*, 62(1):44–62, 1994.

[63] Elchanan Mossel, Joe Neeman, and Allan Sly. Consistency thresholds for the planted bisection model. In *Proceedings of the 47th Symposium on Theory of Computing, STOC 2015*, pages 69–75, 2015.

[64] Alon Nilli. On the second eigenvalue of a graph. *Discrete Mathematics*, 91(2):207–210, 1991.

[65] Robert Plato. *Numerische Mathematik kompakt - Grundlagenwissen für Studium und Praxis.* Vieweg, 2000.

[66] Huzur Saran and Vijay V. Vazirani. Finding k cuts within twice the optimal. *SIAM Journal on Computing*, 24(1):101–108, 1995.

[67] Kirk Schloegel, George Karypis, and Vipin Kumar. *Graph partitioning for high performance scientific simulations.* Army High Performance Computing Research Center, 2000.

[68] Norbert Sensen. Lower bounds and exact algorithms for the graph partitioning problem using multicommodity flows. In *Proceedings of the 9th European Symposium on Algorithms, ESA 2001*, pages 391–403, 2001.

[69] Kim-Chuan Toh, Michael J. Todd, and Reha H. Tütüncü. Sdpt3 — a matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

[70] Chih-Chien Tu and Hsuanjen Cheng. Spectral methods for graph bisection problems. *Computers and OR*, 25(7-8):519–530, 1998.

[71] Chih-Chien Tu, Ce-Kuen Shieh, and Hsuanjen Cheng. Algorithms for graph partitioning problems by means of eigenspace relaxations. *European Journal of Operational Research*, 123(1):86–104, 2000.

[72] Reha H. Tütüncü, Kim-Chuan Toh, and Michael J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Math. Program.*, 95(2):189–217, 2003.

[73] René van Bevern, Andreas Emil Feldmann, Manuel Sorge, and Ondrej Suchý. On the parameterized complexity of computing graph bisections. In *Graph-Theoretic Concepts in Computer Science - 39th International Workshop, WG 2013, Revised Papers*, pages 76–87, 2013.

[74] Lieven Vandenberghe and Stephen P. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[75] Zhenyu Wu and Richard M. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993.