Sorbonne Université

TER

Chris Cohadari

July 14, 2023

A heuristic for the maximum cut problem

Contents

1 Introduction

Throughout this text we consider the maximum cut problem, which is shown to be NP-hard by [GJS74]. To tackle this problem approximation algorithms and heuristic algorithms have been proposed. For a long time the best known approximation algorithm had an approximation factor of 0.5, until 1995, when Goemans and Williamson proposed an approximation algorithm with approximation factor 0.878 in [GW95], see [KV18, p. 420]. Their approach is based on relaxing the maximum cut problem to a semidefinite program (SDP), solving the SDP and rounding the obtained solution to a cut. Seven years later, Burer, Monteiro and Zhang published a very effective heuristic based on the rounding technique proposed by Goemans and Williamson but without having to solve a SDP. We will develop the theory for the heuristic presented by Burer, Monteiro and Zhang in [BMZ02], which we will call Burer heuristic. Furthermore we will present the computational results of running the Burer heuristic on the instances of the library [Mal23]. In subsection 2.1, we give an overview of an approximation algorithm proposed by Goemans and Williamson. This will be the basis for the Burer heuristic. That includes introducing a semidefinite programming problem relaxation for the maximum cut problem and showing how to derive a cut from a set of points on the unit sphere. We will discuss the performance guarantee in detail. However, the proof of polynomial runtime will not be treated here and we kindly refer to [KV18]. In subsection 2.2, we lay the theoretical foundation for the Burer heuristic. In particular we study the rank-two relaxation, which is a special case of the SDP relaxation introduced in subsection 2.1 for dimension two in polar coordinates. This leads to an unconstrained nonconvex optimisation problem, which has advantages and disadvantages. On the upside by using this relaxation the number of variables is not increased, i.e. the number of variables remains the number of vertices of the graph. This implies good scalability to large instances. However as the function is nonconvex, we cannot expect to solve this relaxation optimally. Therefore we face a trade-off between computational runtime and a theoretical guarantee, see [BMZ02, p. 506]. We will conduct experiments on the library [Mal23] asserting the effectiveness of the Burer heuristic in subsection 3. In subsection 2.3 we give a detailed exposition of the Burer heuristic. The heuristic combines the rank-two relaxation and Goemans-Williamson-type cuts. Given a set of points on the unit circle, we will provide a complete description of how to generate a best possible Goemans-Williamson-type cut in a deterministic manner. Furthermore we will describe an algorithm to improve a cut value by means of trial and error, where the termination criterion is given by the number of consecutive unsuccessful attempts to improve the cut value. Lastly we will end the subsection with the Burer heuristic. This is the algorithm on which the experimental data is based upon and it depends on two parameters M and N. The parameter M can be thought of as the number of seeds, i.e. the number of points to optimise from. The parameter N controls the number of consecutive attempts to improve the cut value from a given starting point. In subsection 2.4 we show that instances of the binary quadratic programming problem can be transformed to instances of the maximum cut problem. Thus this subsection shows how to provide heuristic solutions to instances of the BQP. The importance of this subsection stems from library [Mal23] containing BQP instances, which we will solve by running the Burer heuristic on the corresponding MCP graph.

Finally in section 3 we will present the results of running the Burer heuristic on the library [Mal23] for the 21 possible choices of $M \in \{1, 5, 10, 15, 20, 30, 40, 50\}$ and $N \in \{5, 10, 15\}$.

2 Theoretical aspects

2.1 A semidefinite relaxation for the Max cut problem

Throughout this text we consider undirected graphs G=(V,E) with $V = [1:n] := \{1, \dots, n\}$ and $E \subseteq \{(i,j) \in V \times V \mid i < j\}$. Note that we adopt this notation to stay coherent with [BMZ02] on which this text is based. By the condition $i < j$ there can only be one edge between the vertices i and j, thus we could have just as well considered $E \subseteq \{\{i,j\} \mid i,j \in V : i \neq j\}$. Let the edge weights $w_{ij} = w_{ji}$ be given such that $w_{ij} = 0$ if $(i,j) \notin E$. We then have $w_{ii} = 0$ for all $i \in V$. We are interested in studying the weights of cuts in the graph G. Given a bipartition of $\left(S, \overline{S}\right)$ of V, a cut is the set $\left\{e \in E \mid |S \cap e| = 1 \text{ and } \left|\overline{S} \cap e\right| = 1\right\}$. Clearly, a cut is not uniquely defined by $\left(S, \overline{S}\right)$, as $\left(\overline{S}, S\right)$ generates the same cut. Furthermore a cut is uniquely defined by a set X, as the bipartition is uniquely given by $(X, V \setminus X)$. The corresponding weight is obtained by summing the weights of the edges in the cut. The task of finding a cut of maximum weight is called the maximum cut problem, abbreviated by MCP, see [KV18]:

Maximum Weight Cut Problem

Instance:	An undirected weighted graph G.
Task:	Find a cut in G with maximum total weight.

It is well known that the MCP is NP-hard, see [GJS74]. Therefore there has been a lot of work on approximation algorithms and heuristics tackling the MCP. A groundbreaking contribution is the approximation algorithm described by Goemans and Williamson in [GW95]. In fact the Burer heuristic is based on this approximation algorithm. Therefore we will spend the rest of this subsection on developing the theory to understand the algorithm by Goemans and Williamson. We will focus on the main ideas and not go into, for example proving the polynomial runtime.
Following the description given by [Vaz03, p. 268 ff], we can motivate the formulation of the maximum cut problem as a binary quadratic program:
We give a for our purposes sufficient definition of binary quadratic program.

Definition 2.1 (Binary quadratic program) Let B be either $\{0,1\}$ or $\{-1,1\}$, and $Q \in \mathbb{R}^{n \times n}$. A (unrestricted) binary quadratic program is of the form:

$$\begin{aligned} &\text{minimize} && x^T Q x \\ &\text{subject to} && x_i \in B \quad \forall i \in [1:n] \end{aligned} \tag{1}$$

To every vertex i we assign a indicator variable x_i which is constrained to be in $\{-1,1\}$. This allows us to define the cut-defining partition $\left(S, \overline{S}\right)$ by $S := \{i \mid x_i = 1\}$ and $\overline{S} = \{i \mid x_i = -1\}$. For any edge $\{i,j\}$ in the cut we have $i \in S$ and $j \in \overline{S}$ or vice versa. Thus we have $x_i x_j = -1$ for every edge $\{i,j\}$ in the cut. On the other hand, for every edge $\{i,j\}$ that is not in the cut we have $i,j \in S$ or $i,j \in \overline{S}$, implying $x_i x_j = 1$.

$$\frac{1}{2}(1 - x_i x_j) = \begin{cases} 1 & \{i,j\} \text{ in the cut defined by } S \\ 0 & \text{otherwise} \end{cases}$$

This explains, why we can write the Maximum Cut Problem, abbreviated with MCP, as the following quadratic program.

$$\begin{aligned} &\text{maximize} \quad \frac{1}{2}\sum_{1\le i<j\le n} w_{ij}(1-x_ix_j) \\ &\text{subject to} \quad x_i \in \{-1,1\} \quad \forall i \in [1:n] \end{aligned} \tag{2}$$

Alternatively, we can solve the following binary quadratic program as Lemma 2.2. This will prove useful as we progress in the section.

$$\begin{aligned} &\text{minimize} \quad \sum_{1\le i<j\le n} w_{ij}x_ix_j \\ &\text{subject to} \quad |x_i| = 1 \quad \forall i \in [1:n] \end{aligned} \tag{3}$$

For the sake of convenience we will introduce some notation to denote sums, see [Aig07, p. 41]. Say we want to sum the even numbers between 0 and 100, we can write $\sum_{k=0}^{100} k\,[k \text{ even}]$. The expression in brackets, which is to be multiplied, means that

$$[k \text{ has property E}] = \begin{cases} 1 & \text{if } k \text{ satisfies property E} \\ 0 & \text{otherwise} \end{cases}$$

This expresses the same sum as

$$\sum_{\substack{k=0 \\ k \text{ even}}}^{100} k \quad \text{or} \quad \sum_{k=1}^{50} 2k$$

Lemma 2.2 The solutions of (2) and (3) coincide.

Proof. We first observe that the constraints of (2) and (3) coincide so it suffices to show that the target function of the BQP is minimised if and only if the target function of the MCP is maximised. The following computation uses that $x_ix_j \in \{-1,1\}$ for all $i,j \in [1:n]$:

$$\begin{aligned} &\sum_{1\le i<j\le n} w_{ij}x_ix_j = \sum_{1\le i<j\le n} w_{ij}x_ix_j\,[x_ix_j = 1] + \sum_{1\le i<j\le n} w_{ij}x_ix_j\,[x_ix_j = -1] \\ &= \sum_{1\le i<j\le n} w_{ij}\,[x_ix_j = 1] - \sum_{1\le i<j\le n} w_{ij}\,[x_ix_j = -1] = \sum_{1\le i<j\le n} w_{ij} - 2\sum_{1\le i<j\le n} w_{ij}\,[x_ix_j = -1] \end{aligned}$$

Using $\sum_{1\le i<j\le n} w_{ij}$ is constant and $\sum_{1\le i<j\le n} w_{ij}(1-x_ix_j) = 2\sum_{1\le i<j\le n} w_{ij}\,[x_ix_j = -1]$. We can thus conclude, that maximising $\frac{1}{2}\sum_{1\le i<j\le n} w_{ij}(1-x_ix_j)$ (and thus $\sum_{1\le i<j\le n} w_{ij}(1-x_ix_j)$) minimises $\sum_{1\le i<j\le n} w_{ij}x_ix_j$ and vice versa. □

Let us introduce some notation allowing for more concise formulation. Consider $W = (w_{ij})_{1\le i,j\le n}, X = (x_{ij})_{1\le i,j\le n} \in \mathbb{R}^{n\times n}$. We denote the sum of their entrywise product by $W \bullet X = \sum_{i=1}^n \sum_{j=1}^n w_{ij}x_{ij}$ and vector of diagonal elements by $\operatorname{diag}(X) = (x_{11},\ldots,x_{nn})^T \in \mathbb{R}^n$. Further we denote the vector of all ones by e and denote a symmetric positive semidefinite matrix X by $X \succeq 0$.

The following Lemma shows a useful reformulation of (3), which will form the basis of our construction of the Goemans-Williamson algorithm.

Lemma 2.3 The BQP (3) can be rewritten into the following matrix optimisation program

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} W \bullet X \\ \text{subject to} \quad & \operatorname{diag}(X) = e, \\ & \operatorname{rank}(X) = 1, \\ & X \succeq 0 \end{aligned} \tag{4}$$

Proof. First of all, we are going to show that a matrix X satisfies the conditions of (4) if and only if there exists a $x \in \{-1,1\}^n$ such that $X = xx^T$:
As $\operatorname{rank}(X) = 1$ implies that the columns of the matrix are scalar multiples of each other, there exist $x, y \in \mathbb{R}^n \setminus 0$ such that $X = xy^T$.
Claim: The vectors x and y are necessarily linearly dependent.
Assume otherwise. Then the Cauchy-Schwarz inequality (CS) is strict, i.e. $|\langle x, y\rangle| < \|x\| \, \|y\|$. Furthermore the point $\frac{x+y}{2}$ is linearly independent from x and y. By symmetry, we only show this for y: Given x, y linearly independent. Assume there exists $\lambda \in \mathbb{R}$ such that $\frac{x+y}{2} = \lambda y \Leftrightarrow x = (2\lambda - 1)y$ contradicting x, y linearly independent. Thus $\left|\left\langle x, \frac{x+y}{2}\right\rangle\right| < \|x\| \left\|\frac{x+y}{2}\right\|$ and $\left|\left\langle y, \frac{x+y}{2}\right\rangle\right| < \|y\| \left\|\frac{x+y}{2}\right\|$. Now we can show the claim using the following preparatory calculations. We consider the vector $v := y - \frac{\left\langle y, \frac{x+y}{2}\right\rangle}{\left\|\frac{x+y}{2}\right\|^2} \frac{x+y}{2}$ and calculate:

$$\langle y, v\rangle = \|y\|^2 - \frac{\left\langle y, \frac{x+y}{2}\right\rangle}{\left\|\frac{x+y}{2}\right\|^2} \left\langle y, \frac{x+y}{2}\right\rangle \overset{\text{CS}}{>} \|y\|^2 - \frac{1}{\left\|\frac{x+y}{2}\right\|^2} \|y\|^2 \left\|\frac{x+y}{2}\right\|^2 = 0$$

Furthermore we have:

$$\begin{aligned} \frac{1}{2}\left(\langle x, v\rangle + \langle y, v\rangle\right) &= \left\langle v, \frac{x+y}{2}\right\rangle = \left\langle y, \frac{x+y}{2}\right\rangle - \frac{\left\langle y, \frac{x+y}{2}\right\rangle}{\left\|\frac{x+y}{2}\right\|^2} \left\langle \frac{x+y}{2}, \frac{x+y}{2}\right\rangle \\ &= \left\langle y, \frac{x+y}{2}\right\rangle - \left\langle y, \frac{x+y}{2}\right\rangle = 0 \end{aligned}$$

Thus we have $\langle x, v\rangle + \langle y, v\rangle = 0$ and using the first calculation we get $\langle x, v\rangle < 0$ and $\langle y, v\rangle > 0$. The claim follows by observing that $v^T X v = v^T x y^T v < 0$ contradicting X positive semidefinite.

As x and y are linearly dependent, there exist an $\lambda \in \mathbb{R}$ such that $X = \lambda x x^T$ and $X_{ii} = \lambda x_i^2$ for all $i \in [1:n]$. Using the assumption $\operatorname{diag}(X) = e$ yields $1 = \lambda x_i^2$ for all $i \in [1:n]$. This implies $\lambda \neq 0$ and we can rewrite $x_i^2 = \frac{1}{\lambda}$ for all $i \in [1:n]$. In other words we have $x_i \in \left\{-\frac{1}{\sqrt{\lambda}}, \frac{1}{\sqrt{\lambda}}\right\}$ for all $i \in [1:n]$ and setting $\widetilde{x} := \sqrt{\lambda} x$ yields $|\widetilde{x}_i| = 1$ for all $i \in [1:n]$ and $X = \widetilde{x}\widetilde{x}^T$.

For the other direction: Let $x \in \{-1,1\}^n$ and set $X = xx^T$. We clearly have $\operatorname{rank}(X) = 1$, also $x_i \in \{-1,1\}$ for all $i \in [1:n]$ implies $x_i x_i = 1$ for all $i \in [1:n]$, which shows $\operatorname{diag}(X) = e$. That X is positive semidefinite follows from:

$$y^T X y = y^T x x^T y = |(x, y)|^2 \geq 0 \quad \forall y \in \mathbb{R}^n$$

Finally, we conclude the proof by showing that the objective functions coincide. Let $X = xx^T$, with $x \in \{-1,1\}$:

$$\frac{1}{2} W \bullet X = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} x_i x_j \overset{\substack{w_{ii}=0 \\ w_{ij}=w_{ji}}}{=} \frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij} x_i x_j$$

□

Remark In the previous proof, we showed that for linearly independent $x, y \in \mathbb{R}^n$ the matrix xy^T is not positive semidefinite, by an explicit construction. We will explain the intuition behind the construction: As x and y are different, by the geometric Hahn-Banach Theorem, there exists a hyperplane separating the points. However the points being different is not to sufficient to show that there exists a separating hyperplane containing the origin. This is where the linear independence is necessary. As the construction shows, there exists a hyperplane, generated by v, that contains the origin. Containing the origin is critical as this ensures that $x^T v, y^T v \neq 0$ and that $x^T v$ and $y^T v$ have different signs. Only then we can deduce $v^T x y^T v < 0$.

In order to describe the algorithm by Goemans and Williamson [GW95] we need to define what a semidefinite program is.

Definition 2.4 (Semidefinite program [Vaz03, p.258]) Let $C, D_1, \ldots, D_k \in \mathbb{R}^n$ symmetric and $d_1, \ldots, d_k \in \mathbb{R}$. The general semidefinite programming problem, abbreviated to SDP, is given by

$$\begin{aligned} \text{maximize} \quad & C \bullet X \\ \text{subject to} \quad & D_i \bullet X = d_i \quad \forall i \in [1:k], \\ & X \succeq 0 \end{aligned} \tag{5}$$

The following Theorem guarantees that SDP can be approximately solved in polynomial time. Showing this is a key point in showing that the algorithm presented by Goemans and Williamson has polynomial runtime. As we will not use SDP for the heuristic refer to [Vaz03, p. 258 ff] or [KV18, Theorem 16.10] for a proof.

Theorem 2.5 ([Vaz03, p. 259]) For any $\varepsilon > 0$ semidefinite programs can be solved up within an additive error of ε, in time polynomial in n and $\log(1/\varepsilon)$

By setting $C = -\frac{1}{2}W$ and $D_i = e_i e_i^T, d_i = 1$ for all $i \in [1:n]$ we see that we can relax the matrix optimisation program (4) into a SDP by simply dropping the rank constraint:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} W \bullet X \\ \text{subject to} \quad & \operatorname{diag}(X) = e, \\ & X \succeq 0 \end{aligned} \tag{6}$$

The first main idea is to relax BQP (3) in the following manner. We replace the unit scalars x_i by unit vectors $v_i \in \mathbb{R}^d$ (where the dimension $d > 1$ can be chosen freely). Furthermore the product $x_i x_j$ can be interpreted as the one dimensional scalar product, we therefore replace it by $v_i^T v_j$, which is the standard scalar product in $\mathbb{R}^d$. Noting $v_i \in \mathbb{R}^d$ the relaxation writes as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{1 \leq i < j \leq n} w_{ij} v_i^T v_j \\ \text{subject to} \quad & \|v_i\|_d = 1 \quad \forall i \in [1:n] \end{aligned} \tag{7}$$

Remark 1. We are going to show that (7) is a relaxation of (3):
Let $x \in \{-1, 1\}^n$ be a solution to (3). Let $e_1 \in \mathbb{R}^d$ be the first standard basis vector and set $v_i = x_i e_1 \in \mathbb{R}^d$ for all $i \in [1:n]$. Then the v_i's form a feasible solution as for every $i \in [1:n]$ we have $\|v_i\| = |x_i| \, \|e_1\|_d = 1$. Furthermore values of the objective functions coincide by $v_i^T v_j = x_i x_j e_1^T e_1 = x_i^T x_j$ for all $i, j \in [1:n]$.

2. Observe that for a feasible (and optimal) solution $v_1, \ldots, v_n$ each point v_i is located on the unit sphere $\mathbb{S}^{d-1}$ representing the vertex i in G.

Aiming to apply Lemma 2.3, we can change the variables $x_i x_j$ to $v_i^T v_j$. Therefore X_{ij} is given by $v_i^T v_j$ instead of $x_i^T x_j$.

Lemma 2.6 ([Vaz03, p. 259f]) We can rewrite (7) into the following SDP

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} W \bullet X \\ \text{subject to} \quad & \operatorname{diag}(X) = e, \\ & X \succeq 0 \end{aligned} \tag{8}$$

Proof. Throughout this proof we denote the i-th column of a matrix L by L_i.

Let X be a feasible solution of (7). As X is positive semidefinite there exists a $L \in \mathbb{R}^{n \times n}$ such that $X = L^T L$. Thus we have $X_{ij} = L_i^T L_j$ for all $i, j \in [1:n]$. For $i \in [1:n]$ we set $v_i := L_i$. As X is a feasible solution to (8) we have $1 = X_{ii} = L_i^T L_i = \|v_i\|_n^2$, which shows $\|v_i\|_n = 1$ for all $i \in [1:n]$. Thus the v_i's constitute a feasible solution to (7) of the same objective function value:

$$\frac{1}{2} \sum_{i,j=1}^{n} w_{ij} v_i^T v_j = \frac{1}{2} W \bullet L^T L = \frac{1}{2} W \bullet X.$$

Given a feasible solution $v_1, \ldots, v_n$ to (7) we define the matrix $L \in \mathbb{R}^{n \times n}$ by $L_i := v_i$. Now set $X = L^T L$, we then have $X_{ij} = v_i^T v_j$. As the v_i's are a feasible solution we have $X_{ii} = \|v_i\|_n^2 = 1$ for $i \in [1:n]$. Furthermore X is positive semidefinite since $y^T X y = \|Ly\|_n^2 \geq 0$ for all $y \in \mathbb{R}^n$, which shows that X is a feasible solution of (8). By construction we have that the two feasible solutions have the same value:

$$\frac{1}{2} W \bullet X = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} X_{ij} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} v_i^T v_j$$

□

Remark As this is essential, we reiterate how to retrieve a solution of (7) from a solution X of (8): As X is a positive semidefinite matrix, using the Cholesky decomposition we compute a matrix $L \in \mathbb{R}^{n \times n}$ such that $X = L^T L$. The columns of L are a solution to (7).

The algorithm presented by Goemans and Williamson can be stated as follows [KV18, p.424]: We emphasise that nonnegative edge weights are required. This requirement is essential in the proof of Lemma 2.10.

As we will not solve semidefinite programs in the heuristic presented later in the text, we will for simplicity's sake, assume that we can find an optimal solution to (8). That means that we have an optimal solution to (7). The inaccuracy of assuming an optimal solution to

Algorithm 1 Goemans-Williamson algorithm

Input: A graph with nonnegative edge weights
Output: A cut given by the set S

1: Find an approximately optimal solution X to (7)
2: Find vectors $y_1, ..., y_n \in S^d$ such that $y_i^T y_j = x_{ij}$ for all $i, j \in [1:n]$ (using Cholesky decomposition)
3: Choose a random point $r \in \mathbb{S}^d$
4: Set $S := \left\{ i \in [1:n] \mid r^T y_i \geq 0 \right\}$
5: **return** $(S, V \setminus S)$

(7) can be absorbed into the approximation factor; as stated in [Vaz03, p. 260]. A detailed account of how to handle nearly optimal solutions is given in [KV18]. Until the end of the subsection we will closely follow the description provided in [Vaz03, p. 260 ff]. In order to approximate the MCP we need to relate this optimum solution to (7) to a cut. As the matrix whose columns are an optimum solution to (7) is in general not of rank 1 we are faced with a rounding problem. The question boils down to the following: How can we derive a good cut from an optimum solution to (7)?

The second big idea by Goemans and Williamson, answers the question. As the heuristic presented later on in the text is based on this idea we will describe it in detail.

Let $v_1, \ldots, v_n$ be an optimal solution and denote the optimal value by OPT. We have $v_i \in \mathbb{S}^d$ for all $i \in [1:n]$ and for all $i, j \in [1:n]$ we have $\cos(\theta_{ij}) = v_i^T v_j$. Here θ_{ij} denotes the angle between v_i and v_j. The contribution of v_i and v_j to the value of OPT is, as we simply read off from the definition of the objective function:

$$\frac{1}{2} w_{ij} \left(1 - \cos(\theta_{ij})\right)$$

Therefore the contribution of v_i and v_j is the greater, the closer θ_{ij} is to π (, while considering the same w_{ij}). Geometrically this means, that diametrically opposed points have the highest contribution. This means that roughly speaking we want to separate diametrically opposed points. The cuts introduced in the next Definition achieve just that.

Definition 2.7 (Goemans-Williamson cut) A Goemans-Williamson cut for a direction $r \in \mathbb{S}^n$ is the cut given by $\left(S, \overline{S}\right)$, where we set $S := \left\{ i \in [1:n] \mid v_i^T r \geq 0 \right\}$ and $\overline{S} := \left\{ i \in [1:n] \mid v_i^T r < 0 \right\}$.

As the algorithm is based on rounding the values according to a uniformly chosen Goemans-Williamson cut we are naturally interested in knowing what the odds of separating two points are.

Lemma 2.8 Let $n \geq 2$ and $v_i, v_j \in \mathbb{S}^n$ different. We divide the unit sphere into two hemispheres by a hyperplane through the origin. The probability that v_i and v_j are separated is given by $P\left[v_i \text{ and } v_j \text{ are separated}\right] = \frac{\theta_{ij}}{\pi}$

Proof. Consider the plane that contains $0, v_i$ and v_j, given by $\text{span}\left(v_i, v_j\right)$. A hyperplane (through the origin) separates the two points if and only if its projection onto the plane intersects the arc connecting v_i, v_j on the sphere. As the hyperplane is chosen uniformly and the angle between v_i and v_j is given by θ_{ij} the probability, that the hyperplane separates the two points is $\frac{\theta_{ij}}{\pi}$. □

Lemma 2.9 The expression $\alpha := \min_{0<\theta\leq\pi} \frac{2}{\pi}\frac{\theta}{1-\cos(\theta)}$ is well defined and $\alpha > 0.87856$ holds.

Proof. The enumerator has a simple zero and the denominator a zero of order two. As the terms are nonnegative we have $\lim_{\theta\to 0} = \frac{2}{\pi}\frac{\theta}{1-\cos(\theta)} = \infty$. Thus we have a continuous function on an interval and the As the denominator and the numerator have simple zeroes in zero, the function can be continuously extended in 0. Thus we have a continuous function on an interval and the minimum will be attained. We take the derivative and get

$$\frac{\partial}{\partial\theta}\frac{2}{\pi}\frac{\theta}{1-\cos(\theta)} = \frac{2}{\pi}\frac{1-\cos(\theta)-\theta\sin(\theta)}{(1-\cos(\theta))^2}$$

Thus we have a critical point if and only if $1 = \cos(\theta) + \theta\sin(\theta)$ In $(0, \pi]$ there is only one solutions, namely 2.33112237041442. By the first part this has to be a local minimum with value 0.878567205784851604. By $\frac{2}{\pi}\frac{\pi}{1-\cos(\pi)} = 2$ the local minimum is a global minimum and $\alpha > 0.87856$ □

Lemma 2.10 The expected value of the weight of a Goemans-Williamson cut is greater or equal to $\alpha \cdot \mathrm{OPT}$, where OPT denotes the value of an optimal solution to (7).

Proof. Using Lemma 2.9 we have for all $\theta \in (0, \pi]$:

$$\frac{\theta}{\pi} \geq \alpha\frac{1-\cos(\theta)}{2}$$

We denote the weight corresponding to the edge (i, j) by w_{ij} and set $w_{ij} = 0$ otherwise. The expected value of a Goemans-Williamson cut is given by
$\sum_{1\leq i<j\leq n} w_{ij}P[v_i \text{ and } v_j \text{ are separated}]$, together with Lemma 2.8 we can compute:

$$\begin{aligned}\sum_{1\leq i<j\leq n} w_{ij}P[v_i \text{ and } v_j \text{ are separated}] &\geq \sum_{1\leq i<j\leq n} w_{ij}\frac{\theta_{ij}}{\pi}\\ &\geq \frac{\alpha}{2}\sum_{1\leq i<j\leq n} w_{ij}(1-\cos(\theta_{ij}))\\ &= \alpha\cdot\mathrm{OPT}\end{aligned}$$

□

Remark In particular Theorem 2.10 implies that there exists a Goemans-Williamson cut with value $0.8785 \cdot \mathrm{OPT}$.

Therefore we have discussed all the, for our purposes, important aspects and can state, as in [KV18, Theorem 16.12]:

Theorem 2.11 The Goemans-Williamson Max-Cut-Algorithm returns a set S for which the expected value of the associated cut is at least 0.878 times the maximum possible value.

2.2 A rank-two relaxation

In this subsection we will investigate relaxation (7) in dimension two. Instead of using Cartesian coordinates we will use polar coordinates. We will investigate the implications of these changes, following [BMZ02, Sec. 3]. We will introduce the notion of angular representation

of cuts, which are certain vectors that correspond to cuts. The subsection will culminate in giving a classification of cuts as stationary points of the objective function of the relaxation.

Using polar coordinates we can represent any point on the unit sphere by their angle. In other words for every $v \in S^1$, there exists a $\theta \in \mathbb{R}$ such that

$$v = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}.$$

Thus we can represent n points $v_1, \dots, v_n$ on the unit sphere, by a vector $\theta = (\theta_1, \dots, \theta_n) \in \mathbb{R}^n$, such that the i-th vector corresponds to the i-th coordinate:

$$v_i = \begin{pmatrix} \cos(\theta_i) \\ \sin(\theta_i) \end{pmatrix} \quad \text{for all } i \in [1:n] \tag{9}$$

By using the addition Theorem the inner product simplifies to:

$$v_i^T v_j = \cos(\theta_i)\cos(\theta_j) + \sin(\theta_i)\sin(\theta_j) = \cos(\theta_i - \theta_j) \tag{10}$$

Next we define the following map:

$$T : \mathbb{R}^n \to \mathbb{R}^{n \times n},\ \theta \mapsto (\theta_i - \theta_j)_{\substack{i \in [1:n] \\ j \in [1:n]}}$$

For every $\theta \in \mathbb{R}^n$ the resulting matrix $T(\theta)$ is skew-symmetric, since for all $i, j \in [1:n]$

$$T(\theta)_{ij} = \theta_i - \theta_j = -(\theta_j - \theta_i) = -T(\theta)_{ji}$$

Throughout the text, the application of a scalar function onto a matrix, is to be understood as entrywise application of the scalar function. We define the following function, which will turn out to be of central importance:

$$f : \mathbb{R}^n \to \mathbb{R},\ \theta \mapsto \frac{1}{2} W \bullet \cos(T(\theta)) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \cos(\theta_i - \theta_j)$$

Lemma 2.12 The function f is:

1. invariant with respect to simultaneous, uniform rotation on every component. That means for every $\theta \in \mathbb{R}^n$ and every $\tau \in \mathbb{R}$ we have $f(\theta) = f(\theta + \tau e)$.
2. 2π-periodic with respect to each variable, i.e., for all $\theta \in \mathbb{R}^n$ we have $f(\theta) = f(\theta + 2\pi e_i)$, where e_i denotes the i-th standard basis vector.

Proof. We begin by proving the first property. Let $\theta \in \mathbb{R}^n$ and $\tau \in \mathbb{R}$, it is sufficient to show $T(\theta + \tau e) = T(\theta)$, as this is the only term in f that depends on θ. This holds, as for any $i, j \in [1:n]$ we have $T(\theta + \tau e)_{ij} = \theta_i + \tau - (\theta_j + \tau) = \theta_i - \theta_j = T(\theta)_{ij}$. To show the second property let $\theta \in \mathbb{R}^n$ and $i \in [1:n]$ and compute using 2π-periodicity of the cosine:

$$\begin{aligned} &\cos(T(\theta + 2\pi e_i))_{ij} = \cos(\theta_i + 2\pi - \theta_j) = \cos(\theta_i - \theta_j) = \cos(T(\theta))_{ij} \text{ and} \\ &\cos(T(\theta + 2\pi e_i))_{ji} = \cos(\theta_j - \theta_i - 2\pi) = \cos(\theta_j - \theta_i) = \cos(T(\theta))_{ji} \end{aligned}$$

As all other entries of $\cos(T(\theta))$ remain unchanged, $f(\theta) = f(\theta + 2\pi e_i)$ is established. □

Recalling (10) and (9) we see that minimising f is nothing but solving (7) in polar coordinates. This observation is of central importance and we denote the relaxation of the MCP:

$$\min_{\theta \in \mathbb{R}^n} f(\theta) \tag{11}$$

This point of view has advantages and disadvantages. Since f is nonconvex we have no general tool to find a global minimum or even to decide whether a local minimum is a global minimum. In general, there can be multiple local but nonglobal minima. However, formulation (11) is an unconstrained minimisation problem, with a fairly easy analytical description of f. This allows for efficient computation of local minima. Making use of its simplicity we compute its partial derivatives by hand.

Lemma 2.13 The partial derivatives of f for any $j \in [1:n]$ is given by

$$\frac{\partial f}{\partial \theta_j}(\theta) = \sum_{k=1}^{n} w_{kj} \sin(\theta_k - \theta_j)$$

and the gradient can be written as:

$$\nabla f(\theta) = e^T \left(W \circ \sin(T(\theta)) \right) \tag{12}$$

Here the notation $\circ$ stands for the Hadamard product, i.e. the entrywise product of W and $\sin(T(\theta))$

Proof. We fix $j \in [1:n]$ as the index of the variable for which we compute the partial derivative. To avoid ambiguity we rename the indices of f yielding:

$$f(\theta) = \frac{1}{2} \sum_{k=1}^{n} \sum_{i=1}^{n} w_{ki} \cos(\theta_k - \theta_i)$$

From this expression we see that the term $k = i = j$ is equal to zero and we only need to distinguish between the following two cases:
In case of $k = j$ and $i \neq j$ we have:

$$\frac{\partial}{\partial j} w_{ji} \cos(\theta_j - \theta_i) = -w_{ji} \sin(\theta_j - \theta_i) \overset{\substack{w_{ij}=w_{ji} \\ \sin \text{ odd}}}{=} w_{ij} \sin(\theta_i - \theta_j)$$

In case of $i = j$ and $k \neq j$ we have:

$$\frac{\partial}{\partial j} w_{kj} \cos(\theta_k - \theta_j) = w_{kj} \sin(\theta_k - \theta_j)$$

In the case of $k \neq j \neq i$ the derivative is zero as the term is constant with respect to j. By linearity we therefore conclude:

$$\frac{\partial}{\partial j} f(\theta) = \frac{1}{2} \left(\sum_{k=1}^{n} w_{kj} \sin(\theta_k - \theta_j) + \sum_{i=1}^{n} w_{ij} \sin(\theta_k - \theta_j) \right) = \sum_{k=1}^{n} w_{kj} \sin(\theta_k - \theta_j)$$

To justify the gradient, we only need to observe that $\frac{\partial}{\partial j} f(\theta)$ is the sum of the entries of the j-th column of the matrix $W \circ \sin(T(\theta))$. This coincides with the j-th entry of $(W \circ \sin(T(\theta)))^T e = \left(e^T \left(W \circ \sin(T(\theta)) \right) \right)^T$. □

To be able to classify stationary points we now compute the Hessian matrix.

Lemma 2.14 The Hessian matrix H of f is given by

$$H(\theta) = W \circ \cos(T(\theta)) - \operatorname{diag}\left((W \circ \cos(T(\theta)))\, e\right) \tag{13}$$

and the partial derivatives of second order are explicitly give by:

$$\frac{\partial^2}{\partial\theta_i \partial\theta_j} f(\theta) = \begin{cases} w_{ij}\cos(\theta_i - \theta_j) & \text{if } i \neq j \\ -\sum_{k\neq j} w_{kj}\cos(\theta_k - \theta_j) & \text{if } i = j \end{cases} \quad \forall i, j \in [1:n]$$

Proof. We begin by computing the partial derivatives of second order:
Case $i \neq j$:
By $i \neq j$ we have that $w_{kj}\sin(\theta_k - \theta_j)$ is not constant with respect to θ_i if and only if $k = i$. Noting that $\frac{\partial}{\partial\theta_i} w_{ij}\sin(\theta_i - \theta_j) = w_{ij}\cos(\theta_i - \theta_j)$ we get

$$\frac{\partial^2}{\partial\theta_i \partial\theta_j} f(\theta) = w_{ij}\cos(\theta_i - \theta_j)$$

Case $i = j$:
In this case we have $w_{jj} = 0$ and $\frac{\partial}{\partial\theta_j}\sin(\theta_k - \theta_j) = -\cos(\theta_k - \theta_j)$. By linearity we thus get

$$\frac{\partial^2}{\partial\theta_j \partial\theta_j} f(\theta) = \sum_{k\neq j} -w_{kj}\cos(\theta_k - \theta_j)$$

The result follows, since the partial derivatives of second order coincide with the entries of

$$W \circ \cos(T(\theta)) - \operatorname{diag}\left((W \circ \cos(T(\theta))e\right).$$

□

Next we want to investigate the relationship between the function f and the cuts of a graph.

Definition 2.15 (Angular representation of a cut) A vector $\theta \in \mathbb{R}^n$ is called an angular representation of a cut, or simply a cut, if there exist integers k_{ij} such that $\theta_i - \theta_j = k_{ij}\pi$ for all $i, j \in [1:n]$.

Let $\overline{\theta}$ be an angular representation of a cut. Using that $\cos(\overline{\theta}_i - \overline{\theta}_j)$ evaluates to 1 if k_{ij} is even and to -1 if k_{ij} is odd, there exists a binary vector $\overline{x} \in \{-1, 1\}^n$ such that

$$\cos(\overline{\theta}_i - \overline{\theta}_j) = \overline{x}_i\overline{x}_j = \pm 1 \quad \text{for all } i, j \in [1:n]\,. \tag{14}$$

This can be seen by setting $x_1 = 1$ and

$$\overline{x}_i = \begin{cases} 1 & \text{if } \overline{\theta}_i - \overline{\theta}_1 \in 2\pi\mathbb{Z} \\ -1 & \text{otherwise} \end{cases} \quad \text{for } i \in [2:n]$$

The notation $2\pi\mathbb{Z}$ indicates the integer multiples of 2π. This construction is correct, as for any $i, j \in [1:n]$ we have k_{ij} is even if and only if k_{i1} and k_{j1} are both even or both odd if and only if $\overline{x}_i\overline{x}_j = 1$. The first equivalence holds by

$$k_{ij}\pi = \overline{\theta}_i - \overline{\theta}_j = \overline{\theta}_i - \overline{\theta}_1 - (\overline{\theta}_j - \overline{\theta}_1) = (k_{i1} - k_{j1})\pi$$

Then $\overline{x}$ can be viewed as the cut corresponding to $\overline{\theta}$. Moreover, the cut value corresponding to $\overline{\theta}$ is

$$\psi(\overline{\theta}) = \frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij} \left(1 - \cos(\overline{\theta}_i - \overline{\theta}_j)\right) \tag{15}$$

Furthermore this equation makes sense for all $\theta \in \mathbb{R}^n$ and we can interpret its value as a relaxed cut value. In the next Proposition we give a detailed account of this correspondence.

Proposition 2.16 Modulo the uniform rotation and the periodicity for each variable, there is an one-to-one correspondence between binary cuts and angular representations of a cut, given by

$$\overline{\theta}_i = \begin{cases} 0 & \text{if } \overline{x}_i = 1 \\ \pi & \text{if } \overline{x}_i = -1 \end{cases} \tag{16}$$

In the same manner we can set

$$\overline{x}_i = \begin{cases} 1 & \text{if } \overline{\theta}_i = 0 \\ -1 & \text{if } \overline{\theta}_i = \pi \end{cases}$$

Proof. It is clear that assignment (16) defines a one-to-one correspondence between binary cuts and angular representations in $\{0, \pi\}^n$. On the other hand every angular representation of a cut has a representative, where every variable is either 0 or π. Let $\overline{\theta}$ be an angular representation of a cut. By the second property of f, we can assume $\overline{\theta}_i \in [0, 2\pi)$ for all $i \in [1:n]$. We only need to show that there exists a $\tau \in \mathbb{R}$ such that $\overline{\theta}_i - \tau \in \{0, \pi\}$ for all $i \in [1:n]$. If $\overline{\theta}_i \in \{0, \pi\}$ for all $i \in [1:n]$ we simply pick $\tau = 0$ and are done. Otherwise there exists an $i \in [1:n]$, such that $\overline{\theta}_i \notin \{0, \pi\}$. If $\overline{\theta}_i \in (0, \pi)$ we choose $\tau := \overline{\theta}_i$, else we have $\overline{\theta}_i \in (\pi, 2\pi)$ and we choose $\tau := \overline{\theta}_i - \pi$. Now let $j \in [1:n]$. With this choice we just have to show that

$$\overline{\theta}_j = \begin{cases} \tau & \text{if } \overline{\theta}_j \in (0, \pi) \\ \pi + \tau & \text{if } \overline{\theta}_j \in (\pi, 2\pi) \end{cases} \tag{17}$$

holds. As $\overline{\theta}$ is an angular representation of a cut there exists a $k \in \mathbb{Z}$ such that $\overline{\theta}_i - \overline{\theta}_j = k\pi$. By rearranging the terms there exists a $\widetilde{k} \in \mathbb{Z}$ such that $\overline{\theta}_j = \overline{\theta}_i - k\pi = \widetilde{k}\pi + \tau$. This shows equation (17). This shows $\left(\overline{\theta} - \tau e\right)_i \in \{0, \pi\}$ for all $i \in [1:n]$. □

Using this correspondence we can represent a cut by both $\overline{\theta}$ and $\overline{x}$. Given a binary representation of a cut $\overline{x}$ (or an angular representation $\overline{\theta}$), we denote the angular representation by $\theta(\overline{x})$ (or the binary representation by $x(\overline{\theta})$ of that same cut. The next Proposition states an import property of angular representations of a cut.

Proposition 2.17 Every angular representation of a cut $\overline{\theta} \in \mathbb{R}^n$ is a stationary point of the function f.

Proof. As $\overline{\theta}$ is a cut we have $\overline{\theta}_i - \overline{\theta}_j \in \mathbb{Z}\pi$ for all $i, j \in [1:n]$. This directly implies that every entry of $\sin(T(\overline{\theta}))$ is zero. Plugging this into (12) yields $\nabla f(\overline{\theta}) = 0$. This shows that $\overline{\theta}$ is a stationary point. □

Definition 2.18 (Nonnegatively summable matrix) A matrix $M \in \mathbb{R}^{n\times n}$ is called nonnegatively summable if the sum of the entries in every principal submatrix of M is nonnegative, or equivalently, if $u^T M u \geq 0$ for every binary vector $u \in \{0,1\}^n$.

Put in words a principal submatrix is a matrix that we get by crossing out rows and corresponding columns, i.e. crossing out the i-th row implies crossing out the i-th column as well and vice versa. By definition, every semidefinite matrix is nonnegatively summable. However the other implication does not hold as the following example shows.

Example Let $n > 1$ and denote the identity matrix as $I \in \mathbb{R}^{n\times n}$. The matrix $ee^T - I$ is nonnegatively summable but not positive semidefinite. As all the entries of $ee^T - I$ are nonnegative the matrix is clearly nonnegatively summable. To show that $ee^T - I$ is not positive semidefinite we consider the vector $u := e_1 - e_n$, where e_1 is a standard basis vector. By $n > 1$ we have $1 \neq n$ and we get $u^T \left(ee^T - I\right) u = u^T \left(-1, 0, \cdots, 0, 1\right) = -2 < 0$.

Before we can provide a characterisation for maximum (and minimum) cuts in Lemma 2.20 we prove a little calculation rule:

Lemma 2.19 Let $W \in \mathbb{R}^{n\times n}$ be a symmetric matrix, $x \in \mathbb{R}^n$ and $\delta, \delta' \in \{0,1\}^n$. Then we have

$$(\delta \circ x)^T W (\delta' \circ x) = \delta^T \left(W \circ xx^T\right) \delta' = \delta'^T \left(W \circ xx^T\right) \delta$$

Proof. First we observe that for any $\delta \in \{0,1\}^n$ we have

$$(\delta \circ x)_i = \begin{cases} 0 & \text{if } \delta_i = 0 \\ x_i & \text{if } \delta_i = 1 \end{cases} = \delta_i x_i$$

Using this we can compute:

$$\begin{aligned}(\delta \circ x)^T W (\delta' \circ x) &= \sum_{i=1}^n (\delta \circ x)_i \sum_{j=1}^n w_{ij} (\delta' \circ x)_j = \sum_{i=1}^n \delta_i x_i \sum_{j=1}^n w_{ij} \delta'_j x_j \\ &= \sum_{i=1}^n \delta_i \sum_{j=1}^n w_{ij} x_i x_j \delta'_j = \delta^T A \delta'\end{aligned}$$

where $A_{ij} = w_{ij} x_i x_j$ for all $i, j \in [1:n]$. Thus we have $A = W \circ xx^T$ and the first equality holds. Having established the first equality the second follows by noting that W is symmetric:

$$(\delta \circ x)^T W (\delta' \circ x) = (\delta' \circ x)^T W^T (\delta \circ x) = \delta'^T \left(W \circ xx^T\right) \delta$$

□

Lemma 2.20 Let $\overline{x} \in \{-1,1\}^n$ be given and consider the matrix $M(\overline{x}) \in \mathbb{R}^{n\times n}$ defined as

$$M(\overline{x}) = W \circ \left(\overline{x}\overline{x}^T\right) - \operatorname{diag}\left(\left(W \circ \left(\overline{x}\overline{x}^T\right)\right) e\right) \tag{18}$$

Then, $\overline{x}$ is a maximum (respectively, minimum) cut if and only if $M(\overline{x})$ (respectively, $-M(\overline{x})$) is nonnegatively summable.

Proof. Consider the quadratic function $q : \mathbb{R}^n \to \mathbb{R},\ x \mapsto x^T W x/2$. By Lemma 2.2 we know that $\overline{x} \in \{-1,1\}^n$ is a maximum cut if and only if $\overline{x}$ satisfies $q(\overline{x}) \leq q(x)$ for all $x \in \{-1,1\}^n$. Consider a maximum cut $\overline{x} \in \{-1,1\}^n$ and an arbitrary $x \in \{-1,1\}^n$. As W is symmetric $\overline{x}^T W x = x^T W \overline{x}$ holds. Using this and the previous equivalence we have:

$$\begin{aligned} 0 \leq q(x) - q(\overline{x}) &= \frac{1}{2}\left(x^T W x - \overline{x}^T W \overline{x}\right) = \frac{1}{2}\left(x^T W x + \overline{x}^T W x - x^T W \overline{x} - \overline{x}^T W \overline{x}\right) \\ &= \frac{1}{2}(x+\overline{x})^T W (x-\overline{x}) = \left(\overline{x} + \frac{1}{2}(x-\overline{x}\right)^T W (x-\overline{x}) \\ &= \overline{x}^T W (x-\overline{x}) + \frac{1}{2}(x-\overline{x})^T W (x-\overline{x}) \end{aligned} \tag{19}$$

As we have $\overline{x}, x \in \{-1,1\}^n$ we know that either $x_i = \overline{x}_i$ or $x_i = -\overline{x}_i$ for all $i \in [1:n]$. With this observation we see that

$$(x-\overline{x})_i = \begin{cases} 0 & \text{if } x_i = \overline{x}_i \\ -2\overline{x}_i & \text{if } x_i \neq \overline{x}_i \end{cases} \text{ for all } i \in [1:n]$$

and by defining $\delta \in \mathbb{R}^n$ as

$$\delta_i = \begin{cases} 0 & \text{if } x_i = \overline{x_i} \\ 1 & \text{if } x_i \neq \overline{x_i} \end{cases}$$

we get the identity

$$x - \overline{x} = -2\delta \circ \overline{x}$$

With this we continue computation (19):

$$\begin{aligned} 0 &\leq \overline{x}^T W (x-\overline{x}) + \frac{1}{2}(x-\overline{x})^T W (x-\overline{x}) \\ &= -2\overline{x}^T W (\delta \circ \overline{x}) + 2(\delta \circ \overline{x})^T W (\delta \circ \overline{x}) \\ &= -2(e \circ \overline{x})^T W (\delta \circ \overline{x}) + 2(\delta \circ \overline{x})^T W (\delta \circ \overline{x}) \\ &\overset{2.19}{=} -2\delta^T \left(W \circ \overline{x}\overline{x}^T\right) e + 2\delta^T \left(W \circ \overline{x}\overline{x}^T\right) \delta \\ &= -2\delta^T \mathrm{diag}\left((W \circ \overline{x}\overline{x}^T)e\right)\delta + 2\delta^T \left(W \circ \overline{x}\overline{x}^T\right)\delta \overset{\text{def } M(\overline{x})}{=} 2\delta M(\overline{x})\delta \end{aligned}$$

In the second to last equality we used $\delta^T v = \delta^T \mathrm{diag}(v)\,\delta$, which follows immediately from

$$(\mathrm{diag}(v)\delta)_i = \begin{cases} 0 & \text{if } \delta_i = 0 \\ v_i & \text{if } \delta_i = 1 \end{cases} \quad \text{for all } i \in [1:n].$$

So far we have shown $0 \leq \delta^T M(\overline{x})\delta$, however for nonnegative summability we need to argue $0 \leq y^T M(\overline{x}) y$ for all $y \in \{0,1\}^n$.

Claim: For any $y \in \{0,1\}^n$ there exists a $x \in \{-1,1\}^n$ such that $\delta = y$.

First recall that $\delta \in \{0,1\}^n$ depends on $\overline{x}$, which is fixed, and x, which is a variable. To clarify this dependence we denote δ as $\delta_{\overline{x}}(x)$. As $x \in \{-1,1\}^n$ can be chosen arbitrarily, we choose

$$x_i = \begin{cases} \overline{x_i} & \text{if } y_i = 0 \\ -\overline{x_i} & \text{if } y_i = 1 \end{cases}$$

This choice directly implies $\delta_{\overline{x}}(x) = y$. The claim shows that $M(\overline{x})$ is nonnegatively summable. Now we want to show that $\overline{x}$ is minimum cut if and only if $-M(\overline{x})$ is nonnegatively summable. We do this in the same way as above only outlining the slight changes. Let q be the quadratic function we defined above, $\overline{x} \in \{-1,1\}^n$ a minimum cut and $x \in \{-1,1\}^n$ arbitrary. Thus we have $0 \geq q(x) - q(\overline{x})$ and the same computations as above yield $0 \geq \delta^T M(\overline{x})\delta \Leftrightarrow 0 \leq \delta^T \left(-M(\overline{x})\right)\delta$. The nonnegative summability of $-M(\overline{x})$ follows in the same manner as above. □

Remark 1. Let $\overline{x} \in \{-1,1\}^n$. Then the map

$$\delta_{\overline{x}} : \{-1,1\}^n \to \{0,1\}^n,\ x \mapsto \delta_{\overline{x}}(x),\ \text{where } \delta_{\overline{x}}(x)_i = \begin{cases} 0 & \text{if } x_i = \overline{x_i} \\ 1 & \text{if } x_i \neq \overline{x_i} \end{cases} \text{ for all } i \in [1:n]$$

is a bijection.
In the previous proof we have shown that $\delta_{\overline{x}}$ is surjective and as $\{-1,1\}^n, \{0,1\}^n$ are finite sets of the same cardinality we have bijectivity.

2. The matrix $M(\overline{x})$ is symmetric for every $\overline{x} \in \{0,1\}^n$:
The matrix $\operatorname{diag}\left((W \circ \left(\overline{x}\overline{x}^T\right) e\right)$ is symmetric as it is a diagonal matrix. As W is symmetric, we have for every $i, j \in [1:n]$:

$$\left(W \circ \overline{x}\overline{x}^T\right)_{ij} = w_{ij}\overline{x}_i\overline{x}_j = w_{ji}\overline{x}_j\overline{x}_i = \left(W \circ \overline{x}\overline{x}^T\right)_{ji}$$

This shows that $\left(W \circ \overline{x}\overline{x}^T\right)$ is symmetric and, as a difference of symmetric matrices, $M(\overline{x})$ is a symmetric matrix.

Comparing the Hessian of f as given in (13) to (18), we see that the two look similar. The difference is that $\cos(T(\theta))$ in (13) is replaced by xx^T in (18). Comparing the entries we see that $\cos(\theta_i - \theta_j) = x_i x_j$ for all $i, j \in [1:n]$ is a sufficient condition for $H(\theta) = M(x)$. This observation leads to the next Theorem providing a classification of angular representations of cuts as stationary points of the function f.

Theorem 2.21 Let $\overline{\theta}$ be an angular representation of a cut, or simply a cut, and let $\overline{x} \equiv x(\overline{\theta})$ be the associated binary cut. If $\overline{\theta}$ is a local minimum (respectively, local maximum) of $f(\theta)$, then $\overline{x}$ is a maximum (respectively, minimum) cut. Consequently, if $\overline{x}$ is neither a maximum cut nor a minimum cut, then $\overline{\theta}$ must be a saddle point of f.

Proof. First recall that every angular representation of a cut is a critical point of f and that $\cos(\overline{\theta_i} - \overline{\theta_j}) = \overline{x}_i\overline{x}_j$ for all $i, j \in [1:n]$. Therefore $H(\overline{\theta}) = M(\overline{x})$ holds. Let $\overline{\theta}$ be a local minimum of f. Since f is smooth, $H(\overline{\theta})$ is positive semidefinite and in particular nonnegatively summable. By $H(\overline{x}) = M(\overline{x})$ we have that $M(\overline{x})$ is nonnegatively summable. Now let $\overline{\theta}$ be a local maximum of f. Then $H(\overline{\theta})$ is negative semidefinite, that means that $-H(\overline{\theta})$ is nonnegatively summable. Therefore $-H(\overline{\theta}) = -M(\overline{x})$ implies that $-M(\overline{x})$ is nonnegatively summable. Lastly let $\overline{x}$ be neither a maximum cut nor a minimum cut. By Lemma 2.20 we know that neither $M(\overline{x})$ nor $-M(\overline{x})$ is nonnegatively summable. By $M(\overline{x}) = H(\overline{\theta})$ we have that $H(\overline{\theta})$ is neither positive semidefinite nor negative semidefinite. By the necessary condition for a local extremum we conclude that $\overline{\theta}$ is neither a local minimum nor a local maximum. As $\overline{\theta}$ is a critical point of f, it must be a saddle point. □

Although all cuts are stationary points of f (by Proposition 2.17) Theorem 2.21 shows that only the maximum cuts can be local minima of f, see [BMZ02, p. 508].

Example The converse of the two implications in the above Theorem do not hold. We will give an example showing that the angular representation of a maximum cut is not necessarily a local minimum of f.
Consider the complete graph with three vertices, where every edge has weight 1. Then the weight of a maximum cut is 2 and is achieved by $\overline{x} = \begin{pmatrix} 1 & -1 & -1 \end{pmatrix}$. We then have

$$W \circ \overline{x}\overline{x}^T = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & 1 \\ -1 & 1 & 1 \end{pmatrix} \text{ and } \left(W \circ \overline{x}\overline{x}^T\right) e = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}$$

Using (18) we get

$$M(\overline{x}) = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & 1 \\ -1 & 1 & 1 \end{pmatrix} - \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix}$$

Straight forward computation shows nonnegative summability, namely $u^T M(\overline{x}) u \geq 0$ for all $u \in \{0,1\}^n$. However, $M(\overline{x})$ is not positive semidefinite, since

$$\begin{pmatrix} 1 & 0 & 2 \end{pmatrix} M(\overline{x}) \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 2 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix} = -2 < 0$$

Using $H(\theta(\overline{x})) = M(\overline{x})$, we know that $H(\theta(\overline{x}))$ is not positive semidefinite. Thus $\theta(\overline{x})$ is not a local minimum of f.

However there are special classes of instances, where for every maximum cut $\overline{x}$, the angular representation $\theta(\overline{x})$ is a local minimum. The following Proposition gives a class for which this holds:

Proposition 2.22 For a bipartite graph with nonnegative edge weights, the global minimum value of f is attained by a maximum cut.

Proof. We denote the bipartite graph $G = (A, B, E)$. As all the edge weights are nonnegative, the cut (A, B), that cuts through all edges is a maximum cut. For that cut we have $\cos(\theta_i - \theta_j) = -1$ for all $\{i, j\} \in E$. For that cut f evaluates to $-\frac{1}{2} e^T W e$. This value must be a global minimum of f as all the entries of W are nonnegative. □

For instances, where a maximum cut $\overline{x}$ corresponds to a local minimum $\theta(\overline{x})$ of f, the optimality of x can be checked in polynomial time. That is because $\theta(\overline{x})$ local minimum implies $H(\theta(\overline{x}) = M(\overline{x})$ positive semidefinite. This in turn implies $M(\overline{x})$ nonnegatively summable, which shows $\overline{x}$ maximum cut by 2.20. A detailed proof showing that we can decide whether a symmetric matrix (- $M(\overline{x})$ is symmetric-) is positive semidefinite in polynomial time is provided in [KV18, Theorem 16.8].

Theorem 2.21 directly implies the following Corollary, which plays an important role for the heuristic we will develop in the next section.

Corollary 2.23 Let $x \in \{0,1\}^n$ be a nonmaximum cut. Then $\theta(x)$ cannot be a local minimum.

Proof. The statement is the contraposition of the first statement in Theorem 2.21 □

Therefore, as written in [BMZ02, p.509], a good minimisation algorithm would not be attracted to stationary points which are not local minima. We construct such an algorithm in subsection 2.3.

2.3 A heuristic algorithm for MCP

In this subsection we will describe the heuristic described by [BMZ02]. Again we assume nonnegative edge weights, in order to mirror the analysis done in 2.1.

Consider a local minimum θ of f. As θ is not necessarily an angular representation of a cut, we need to develop a method that associates a cut $x \in \{-1,1\}^n$ to any $\theta \in \mathbb{R}^n$. Using polar coordinates we can associate the entries of θ to points on the unit circle. Furthermore we can assume $\theta_i \in [0, 2\pi)$ for all $i \in [1:n]$. Inspired by the Goemans-Williamson cuts in subsection 2.1 we can generate a cut by cutting the unit circle into two halves. For any angle $\alpha \in [0, \pi)$ we can set

$$x_i = \begin{cases} 1 & \text{if } \theta_i \in [\alpha, \alpha + \pi) \\ -1 & \text{otherwise} \end{cases} \tag{20}$$

generating a cut. The weight of the obtained cut is given by

$$\gamma(x) = \frac{1}{2} \sum_{1 \le i < j \le n} w_{ij} \left(1 - x_i x_j\right)$$

Remark 1. Let $\theta \in [0, 2\pi)^n$ and $\alpha \in [0, \pi)$. Let x^α denote the cut generated by α and $x^{\alpha+\pi}$ the cut generated by $\alpha + \pi$. We then have $x^{\alpha+\pi} = -x^\alpha$. For $\alpha + \pi$ assignment (20) is to be understood as:

$$x_i = \begin{cases} 1 & \text{if } \theta_i \in [0, \alpha) \cup [\alpha + \pi, 2\pi) \\ -1 & \text{otherwise} \end{cases}$$

By observing $[0, 2\pi) = [\alpha, \alpha + \pi) \dot\cup ([0, \alpha) \cup [\alpha + \pi, 2\pi))$ we deduce $-x^\alpha = x^{\alpha+\pi}$. As x^α and $x^{\alpha+\pi}$ are of the same weight, it suffices to consider $\alpha \in [0, \pi)$ instead of $\alpha \in [0, 2\pi)$.

2. The cut generated by assignment (20) can also be generated by a Goemans-Williamson cut. This is achieved by slightly turning the diving line in the mathematically negative sense. Slightly means more than zero but strictly less than the smallest angle between two distinct points. As we are given a finite amount of points we know that such a turn exists.

3. Choosing the halfopen interval $[\alpha, \alpha + \pi)$ ensures, that diametrically opposed points are separated, i.e. the corresponding vertices are on different sides of the cut.

By increasing α we have a simple way to examine all cuts obtained by assignment (20), called Goemans-Williamson-type cuts. As we will describe in algorithm 2 we have a deterministic and computationally inexpensive way of finding a best possible Goemans-Williamson-type cut. The algorithm works by rotating the separating line in the mathematically positive sense, starting with angle 0. In the update step the angle is set to the angle of the line that first intersects a new point on the circle, while rotating. If the rotated line has not encountered a new point yet it generates the same cut as the line obtained by the update step. Therefore the algorithm finds all possible Goemans-Williamson-type cuts.

We will make two assumptions without loss of generality. First of all we note that by the 2π-periodicity in each variable of f, see Lemma 2.12 , we can assume $\theta_i \in [0, 2\pi)$ for all $i \in [1:n]$. Furthermore we assume $\theta_1 \le \theta_2 \le \cdots \le \theta_n$.

Algorithm 2 Procedure-Cut

Input: $\theta \in [0, 2\pi)^n$ satisfying $\theta_1 \leq \theta_2 \leq \cdots \leq \theta_n$, undirected weighted graph G
Output: A cut x^*

1: Let x^* trivial cut, i.e. all vertices on one side
2: Let $\alpha = 0, \Gamma = -\infty, i = 1$. Let j be the smallest index such that $\theta_j > \pi$ if there is one; otherwise set $j = n + 1$. Set $\theta_{n+1} = 2\pi$.
3: **while** $\alpha < \pi$ **do**
4: Generate cut x by (20) and compute $\gamma(x)$
5: If $\gamma(x) > \Gamma$, then let $\Gamma = \gamma(x)$ and $x^* = x$.
6: **if** $\theta_i \leq \theta_j - \pi$ **then**
7: Let $\alpha = \theta_i$ and increment i by 1.
8: **else**
9: Let $\alpha = \theta_j - \pi$ and increment j by 1.
10: **end if**
11: **end while**
12: **return** x^*

Remark The assumption $\theta_1 \leq \theta_2 \leq \cdots \leq \theta_n$ is not a real constraint, we just have to keep track of the original indices, as they determine the cut. Let σ be a permutation such that $\theta_{\sigma(1)} \leq \theta_{\sigma(2)} \leq \cdots \leq \theta_{\sigma(n)}$. We can now apply algorithm 2, with the only slight change that the cut in line 2.4 needs to be generated in the following manner:

$$x_i = \begin{cases} 1 & \text{if } \theta_{\sigma(i)} \in [\alpha, \alpha + \pi) \\ -1 & \text{otherwise} \end{cases}$$

Recall that our rank-two relaxation has the form of the Goemans and Williamson relaxation in dimension 2 and the Goemans-Williamson-type cuts can be generated by Goemans-Williamson cuts. Hence we can analyse the performance of our algorithm in a similar manner as in subsection 2.1.

Lemma 2.24 Let $\theta \in [0, 2\pi)^n$ such that $\theta_1 \leq \theta_2 \leq \cdots \leq \theta_n$. Then the cut value generated by algorithm 2 is at least 0.878 times the relaxed cut value $\psi(\theta)$ as defined in (15). Written in a formula:

$$\gamma(x^*) \geq 0.878\psi(\theta) \tag{21}$$

Proof. The entry θ_i represents the point $v_i = \begin{pmatrix} \cos(\theta_i) & \sin(\theta_i) \end{pmatrix}^T$ for $i \in [1 : n]$. We compute the probabilities that two points v_i and v_j are separated by a Goemans-Williamson-type cut, for any $i, j \in [1 : n]$. In dimension 2, we say that a Goemans-Williamson cut is of angle β, if it is generated by $r = \begin{pmatrix} \cos(\beta + \frac{\pi}{2}) & \sin(\beta + \frac{\pi}{2}) \end{pmatrix}$. A Goemans-Williamson cut of angle β induces the cut

$$x_i = \begin{cases} 1 & \theta_i \in [\beta, \beta + \pi] \\ -1 & \text{otherwise} \end{cases}$$

Now fix any angle β and distinguish the following cases.
Case $\theta_i = \theta_j$: Then the points cannot be separated neither for Goemans-Williamson-type cuts nor for Goemans-Williamson cuts.

Case $\theta_i \neq \theta_j$ and neither $\theta_i = \beta + \frac{\pi}{2}$ nor $\theta_i = \beta + \frac{\pi}{2}$: Then both the Goemans-Williamson-type cut and the Goemans-Williamson cut assign the points are assigned to the same value.
Case $\theta_i \neq \theta_j$ and $\theta_i = \beta + \frac{\pi}{2}$: Then v_i and v_j are separated by the Goemans-Williamson-type cut if and only if v_i and v_j are not separated.
Case $\theta_i \neq \theta_j$ and $\theta_j = \beta + \frac{\pi}{2}$: Then v_i and v_j are separated by the Goemans-Williamson-type cut if and only if v_i and v_j are not separated.
Since the set of separating angles only changes on a nullset the probabilities for Goemans-Williamson-type cuts remain the same as for Goemans-Williamson cuts. Therefore we can redo the proof of Lemma 2.10 by substituting Goemans-Williamson cut for Goemans-Williamson-type cut and OPT for $\psi(\theta)$. This yields that the expected value of a Goemans-Williamson-type cut for a given θ is greater or equal to $\alpha\psi(\theta)$. Since algorithm 2 finds all Goemans-Williamson-type cuts and returns a best one it finds a Goemans-Williamson-type cut of value greater or equal to $\alpha\psi(\theta)$.

□

This is not a performance guarantee, as we cannot guarantee that $\psi(\theta)$ is an upper bound on the maximum cut value. Recall, that in the Goemans-Williamson algorithm, the guarantee comes from (approximately) solving the SDP (8). The next Lemma can be interpreted as a weak performance guarantee

Lemma 2.25 Let x_a^* and x_b^* be two cuts generated by algorithm 2 from θ_a and θ_b respectively. If $\gamma(x_a^*) \leq \psi(\theta)$ and $\psi(\theta_b) > \psi(\theta_a)/0.878$, then x_b^* has a higher cut value than x_a^*.

Proof. Using the assumptions we confirm

$$\gamma(x_b^*) \overset{(21)}{\geq} 0.878 \cdot \psi(\theta_b) > \psi(\theta_a) \geq \gamma(x_a^*)$$

□

The next paragraph, describes the idea of algorithm 3. We minimise the function f and obtain a minimiser called θ^1. We can now start algorithm 2 and obtain a best possible cut x^1 associated with θ^1. At this point we could return the cut x^1. If θ^1 happens to be an angular representation of a cut we know, by Theorem 2.21, that x^1 is a maximum cut. Otherwise, we can attempt to improve the cut value by spending more computational resources. By Theorem 2.21 we know that the angular representation of x^1, denoted as $\theta(x^1)$, is a stationary point of f, most likely a saddle point. Therefore we can hope to find a deeper local minimum close to $\theta(x^1)$. As we use a gradient descend, restarting the minimisation from a stationary point is of no use. Thus we restart the minimisation of f from a slight perturbation of $\theta(x^1)$, in hopes of finding another local minimum $\theta^2 \neq \theta^1$ from which we hope to obtain a cut x^2 with a higher cut value than x^1, i.e. $\gamma(x^2) > \gamma(x^1)$. In case $\gamma(x^2) > \gamma(x^1)$ is achieved we deem our attempt successful, and unsuccessful otherwise. We can set the termination criterion to be N consecutive unsuccessful attempts of improving the value of the cut.

Assuming that we do not find a maximum cut, algorithm 3 can also be interpreted in the following way. From a seed we start minimising f. By means of Goemans-Williamson-type cuts, we look for nearby saddle points. Here it is important to note, that nearby is to be understood by the cuts that we can generate from the minimum. The so found saddle points is not necessarily the closest saddle point in the euclidean metric. The saddle point is chosen by the associated cut with the highest weight. From a slight perturbation of that

Algorithm 3 ImprovedCut

Input: Integer $N \in \mathbb{N}$, a seed $\theta^0 \in \mathbb{R}^n$, undirected weighted graph G
Output: A cut x^*

procedure ImprovedCut(input N, θ^0)
    Given $\theta^0 \in \mathbb{R}^n$ and integer $N \geq 0$, let $k = 0$ and $\Gamma = -\infty$
    while $k \leq N$ **do**
        Starting from θ^0, minimise f to get θ
        Compute a best cut x associated with θ by Algorithm 2
        if $\gamma(x) > \Gamma$ **then**
            Let $\Gamma = \gamma(x), x^* = x$ and $k = 0$
        else
            $k = k + 1$
        end if
        Set θ^0 to a random perturbation of the angular representation of x.
    end while
    return x^*
end procedure

point we search for a local minimum, which has a saddle point with lower f-value in its neighbourhood.

The following consideration explains the idea behind algorithm 4: In algorithm 3 we have θ^0 as input, which is the initial seed for the minimisation of f. We can try and improve our chances of finding a high quality heuristic solution by running algorithm 3 from different starting points, in other words increase the number of seeds. The number of seeds, will be given by the input M.

Algorithm 4 Burer heuristic

Input: Undirected weighted graph G, number of seeds M, number of consecutive attempts N
Output: A cut x^*

Let x^* trivial cut, i.e. all vertices on one side
for $i \leftarrow 0, n$ **do**
    Generate random $\theta^0 \in \mathbb{R}^n$
    $x \leftarrow$ ImprovedCut(N, θ^0)
    if $\gamma(x) > \gamma(x^*)$ **then**
        $x^* \leftarrow x$
    end if
end for
return x^*

Generally speaking the bigger M and N are the longer the algorithm will run and the better the returned cut will be. However as the termination criterion in the procedure ImprovedCut relies on the number of unsuccessful attempts we cannot predict the runtime. Since there are elements of randomness, namely in the choice of the seed and the random perturbation there can always be exceptions to the rule.

2.4 Solving BQP using MCP

Closely following [JM21, Section 2] we are going to give a cursory explanation, that binary quadratic problems of the form: Let $Q \in \mathbb{R}^{n\times n}$

$$\begin{aligned} &\text{minimize} && x^T Q x \\ &\text{subject to} && x_i \in \{0,1\} \forall i \in [1:n] \end{aligned} \tag{22}$$

can be solved by solving the MCP on a suitable graph with appropriately chosen edge weights, see Corollary 2.27.

To this end we formulate the MCP as an integer linear programming problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{e\in E} w_e z_e \\ \text{subject to} \quad & \sum_{e\in F} z_e - \sum_{e\in C\setminus F} z_e \leq |F| - 1 && F \subseteq C \subseteq E \text{ subject to } C \text{ cycle and } |F| \text{ odd}, \\ & 0 \leq z_e \leq 1 && \forall e \in E, \\ & z_e \in \mathbb{Z} && \forall e \in E \end{aligned} \tag{23}$$

An optimal solution to (23) gives us the following maximum cut $\{\{i,j\} \mid z_{ij} = 1\}$. Any cut of G is a feasible solution to (23) and vice versa. The cut polytope of a graph G can thus be denoted as

$$P^G_{\text{CUT}} := \left\{ z \in [0,1]^E \mid z \text{ is a feasible solution to (23)} \right\}.$$

For the BQP (22) consider the following undirected graph $G = (V, E)$. The vertices $V = \{v_1, \ldots, v_n\}$, where v_i corresponds to the variable x_i for every $i \in [1:n]$ and $E = \{\{v_i, v_j\} \mid i, j \in [1:n] : q_{ij} + q_{ji} \neq 0\}$ where each edge $\{v_i, v_j\}$ corresponds to the product $y_{ij} := x_i x_j$. We admit that the Boolean quadric polytope is given by

$$P^G_{\text{BQP}} = \left\{ (x,y) \in \{0,1\}^{|V|} \times \{0,1\}^{|E|} \mid (x,y) \text{ satisfies } y_{ij} = x_i x_j \right\}.$$

Without proof we state the following Theorem proven in [De 90].

Theorem 2.26 Let $Q \in \mathbb{R}^{n\times n}$ and consider $G = (V, E)$ with $V := [1:n]$ and $E := \{\{i,j\} \mid i, j \in [1:n] : q_{ij} + q_{ji} \neq 0\}$. Define $H = (W, F)$ where $W := \{v_0, \ldots, v_n\}$ and $F := \{\{v_i, v_j\} \mid \{i,j\} \in E\} \cup \{\{v_0, v_i\} \mid i \in [1:n]\}$. Let $f : \mathbb{R}^F \to \mathbb{R}^{V\cup E}$ be the bijective linear map defined by

$$\begin{aligned} & x_v = z_{0v} \text{ for all } v \in V \text{ , and} \\ & y_{vw} = x_v x_w = \frac{1}{2}\left(z_{0v} + z_{0w} - z_{vw}\right) \text{ for all } \{v,w\} \in E. \end{aligned}$$

Then $P^{\text{G}}_{\text{BQP}} = f\left(P^{\text{H}}_{\text{CUT}}\right)$.

Using this Theorem we can show a fundamental result allowing us to solve a BQP as a MCP.

Corollary 2.27 ([Mal22, p. 2]) Consider a matrix $Q \in \mathbb{R}^{n\times n}$ and set $E = \{\{i,j\} \mid i,j \in [1:n] \text{ such that } q_{ij} + q_{ji} \neq 0\}$. Then the corresponding BQP with the objective function x^TQx to be minimised can be solved as a MCP on $H = (W,F)$ where $W := \{v_0, \dots, v_n\}$ and $F := \{\{v_i, v_j\} \mid \{i,j\} \in E\} \cup \{\{v_0, v_i\} \mid i \in [1:n]\}$, with the objective function

$$\max \sum_{\substack{\{v_i,v_j\}\in F \\ v_i \neq v_0 \neq v_j}} \frac{1}{2}\left(q_{ij}+q_{ji}\right) z_{ij} - \sum_{i \in W\setminus\{v_0\}} \widetilde{c}_i z_{0i},$$

where $\widetilde{c}_i := q_{ii} + \sum_{j=1}^{n} \frac{1}{2}\left(q_{ij}+q_{ji}\right)[\{v_i,v_j\} \in F, v_i \neq v_0 \neq v_j]$ for all $i \in W \setminus \{v_0\}$.

Proof. We have the following chain of equalities:

$$\begin{aligned}
& xQx^T \\
&= \sum_{i,j=1}^{n} q_{ij}x_ix_j + \sum_{i=1}^{n} q_{ii}x_ix_i \\
&\overset{\substack{x_ix_i=x_i \\ \text{def } E}}{=} \sum_{\{i,j\}\in E} \left(q_{ij}+q_{ji}\right) x_ix_j + \sum_{i=1}^{n} q_{ii}x_i \\
&\overset{2.26}{=} \sum_{\{i,j\}\in E} \left(q_{ij}+q_{ji}\right)\left(\frac{1}{2}\left(z_{0i}+z_{0j}-z_{ij}\right)\right) + \sum_{i\in W\setminus\{v_0\}} q_{ii}z_{0i} \\
&= -\sum_{\{i,j\}\in E} \frac{1}{2}\left(q_{ij}+q_{ji}\right) z_{ij} + \frac{1}{2}\sum_{\{i,j\}\in E} \left(q_{ij}+q_{ji}\right)\left(z_{0i}+z_{0j}\right) + \sum_{i\in W\setminus\{v_0\}} q_{ii}z_{0i} \\
&\overset{*}{=} -\sum_{\{i,j\}\in E} \frac{1}{2}\left(q_{ij}+q_{ji}\right) z_{ij} + \sum_{i\in W\setminus\{v_0\}} \underbrace{\left(q_{ii} + \frac{1}{2}\sum_{\substack{j\neq 0 \\ \{v_i,v_j\}\in F}} \left(q_{ij}+q_{ji}\right)\right)}_{=:\tilde{c}_i} z_{0i} \\
&= -\sum_{\{i,j\}\in E} \frac{1}{2}\left(q_{ij}+q_{ji}\right) z_{ij} + \sum_{i\in W\setminus\{v_0\}} \tilde{c}_i z_{0i}
\end{aligned}$$

In $*$ we use that for for all $\{i,j\} \in E$ we have $i,j \neq 0$ and $i \neq j$ and move the middle term to the right term. The previous computation shows that the two objective functions are the negation of each other under the bijection f from Theorem 2.26. This shows the result. □

As we can see the vertex v_0 has a special role as it is connected to all other vertices and its weight of the incident edges is calculated by $\widetilde{c}_i$. The vertex v_0 is also called the sun vertex.

3 Experimentation

3.1 Experimental Setup

The goal of our experimental study is to compare the performance of the Burer heuristic for a range of different parameters. We recall that the Burer heuristic depends on two parameter: The parameter N, which determines the number of consecutive attempts to improve a cut and the parameter M, which controls the number of starting points. We are going to compare 21 solvers, which are the combinations of $M \in \{1, 5, 10, 15, 20, 30, 40, 50\}$ and $N \in \{5, 10, 15\}$. As these parameters are characteristic we have the following naming convention: The solver with the name Mxx.Nyy is the Burer heuristic with M=xx and N=yy. The solvers are run on the instances from [Mal23]. We will give a broad overview of the library for a detailed description we refer to [Mal23]. The library comprises 431 instances, split into 8 classes. The number of vertices ranges from 21 to 3000 with 25% of the instances having 100 or fewer vertices, 50% having 121 or fewer vertices and 75% having 251 or fewer vertices. The (rounded) densities range from 0.001 to 1 with 25% of the instances having density 0.1 or less, 50% having density 0.3 or less and 75% having density 0.8 or less. Thus we have a range of different sized graphs from low density to high density. The runtime of the solvers is measured in the number of generated cuts, since this metric is system independent.

We will distinguish between instances for which the optimal value is known and those for which the optimal value is not known. Since we are interested in the approximation ratio, the returned cut values are divided by the optimal value. For the instances without known optimal value, we normalise by the largest returned cut, see subsection 3.4. In the analysis we will consider three indicators for a given instance. The smallest and largest returned ratio over the 21 solvers. Lastly we will investigate the standard deviation, which indicates how sensitive the heuristic is to different choices of parameters.

The goal is to investigate the overall performance and find out for which parameters the Burer heuristic works best. Measuring the number of produced cuts will help us in evaluating the trade-off between improved solutions and increased runtime.

3.2 Instances with known optimal value

We are going to investigate the instances with known optimal value identifying instances, for which the Burer heuristic performs the worst and take a detailed look at those in subsection 3.3.

We are going to take a closer look at the performance of the 21 solvers on the 332 instances with known optimal value. There are 146 instances which solved optimally by each of the 21 solvers.

Furthermore there are 154 instances for which at least one of the solvers but not all finds the optimal value. Out of these instances for all but 6, every solver achieves an approximation ratio, or simply ratio, of at least 0.975207. The excluded 6 instances are of the form gka*b for $^* \in [3 : 8]$. Their worst case performance ranges from 0.248276 to 0.847458 and the standard deviations range from 0.059270 to 0.166898. Again excluding those instances the standard deviation of each instance decreases considerably to less or equal to 0.006419.

Lastly there are 32 instances for which none of the 21 solvers finds the optimal value. For these instances we first take a look at the highest achieved ratio over all the solvers. Doing this, we see that there are only, two instances namely gka10b with 0.863636 and gka9b with 0.875912 for which the highest ratio is less than 0.878. For every other instance the highest

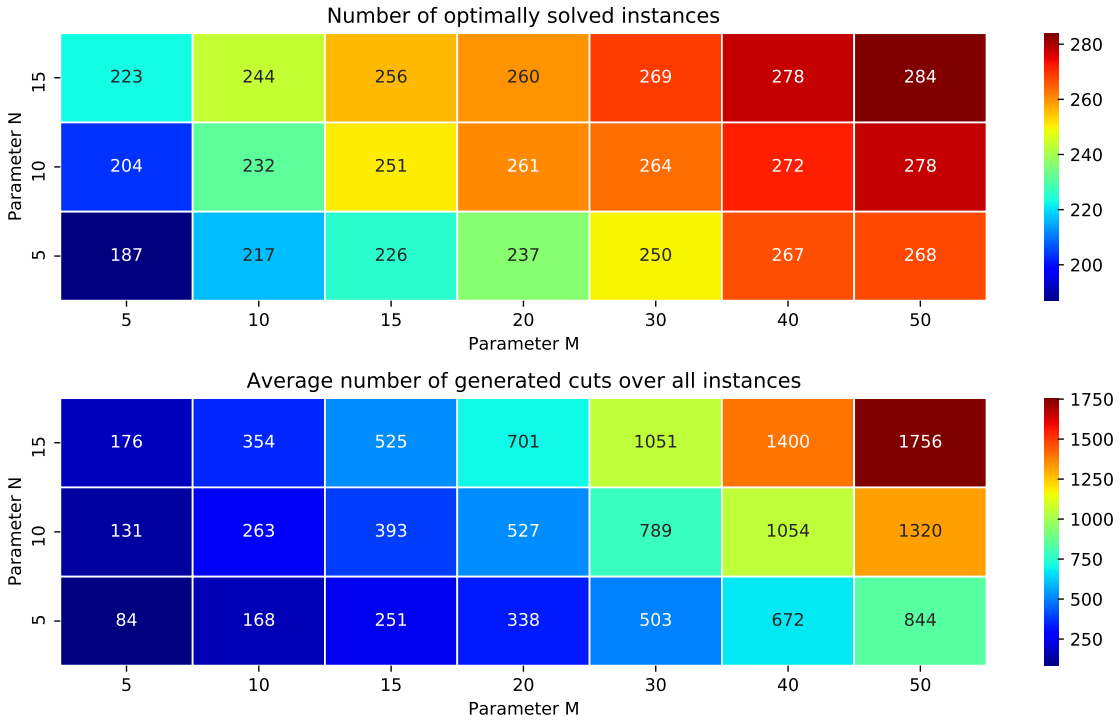


Figure 1: An overview of the performance of each solver ran on every instance with known optimal solution.

achieved ratio is greater than 0.99. Regarding the minimum the same phenomenon occurs. For the two instances gka10b and gka9b the lowest ratios are 0.000000 and 0.277372 respectively. For the other instance the lowest ratio is greater than 0.97 exceeding 0.878. This means that for 30 out of those 32 instances every solver returns a solution satisfying the performance guarantee of the Goemans-Williamson algorithm, although there is no theoretical guarantee to do so for the Burer heuristic, see subsection 2.3. Essentially the same applies to the standard deviations. The standard deviation for gka9b and gka10b, with 0.331967 and 0.154788 respectively, is a lot larger than for the other instances which have standard deviation less or equal to 0.004201.

This summary of the data clearly suggests, that we should take a careful look at the instances of the form gka*b for $* \in [1 : 10]$, see subsection 3.3.

Our previous analysis shows that for the majority of instances with known optimal value the optimal value is attained. Therefore it is sensible to count the number of optimally solved instances for each solver and the average number of generated cuts. This is done in the top plot in figure 1, where the entry M=xx and N=yy indicates the number of optimally solved instances by the Burer heuristic Mxx.Nyy. In the same manner, the bottom plot tracks the (floored) average number of generated cuts for each solver over all instances with known optimal value. The figure clearly shows that increasing the parameters individually or at the same time increases both the number of optimally solved instances and the average number of generated cuts on average. We can also see that the returns are diminishing, i.e. further increasing large parameters yields a few more optimally solved instances at the cost of having to generate a lot more cuts.

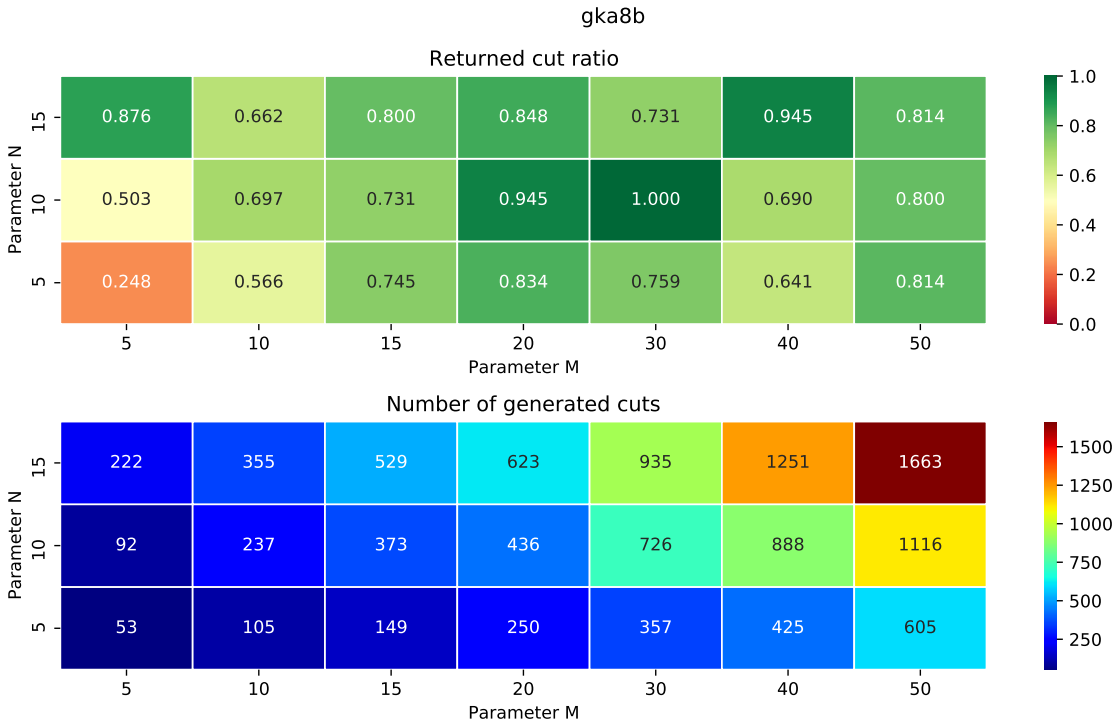


Figure 2: Overview of the returned ratios and number of generated cuts for each solver on the instance gka8b.

3.3 In-depth look at gka*b instances

As we have observed in subsection 3.2 the instances of the form gka*b with $^* \in [3:10]$ perform by far the worst. As stated in [Mal23] the instances gka*b are originally BQP instances. After the transformation to MCP the number of vertices ranges from 21 to 126. Except for instance gka10b, the number of vertices increases by 10 compared to the previous instance, i.e. gka3b has 31 vertices and gka4b has 41 vertices. All instances have a (rounded) density of 1. Looking at the distribution of weights we see that there are edges of different signs. A closer look reveals that the edges with negative weight are exclusively the edges incident to the sun vertex, i.e. the additional vertex connected to all other vertices. The other edges have weights that are within the order of magnitude of the maximum cut value. Thus it could be possible that different cut values have larger gaps compared to other instances. This provides a possible explanation for the much higher standard deviations. However, the instances gka*b with $^* \in [1:8]$ the same characteristics hold true but their performance is much better. Therefore it is worthwile to take a closer look at the performance of the instances and search for similarities.

The figures for each instance are provided in the appendix. In each figure the entry M=xx, N=yy of the top plot indicate the returned ratio of the solver Mxx.Nyy. The colormap is chosen such that dark green indicates a particularly good cut ratio and red a bad one. For the bottom plot the entries indicate the number of generated cuts. Unsurprisingly the data indicates that the Burer heuristic delivers worse approximation ratios as the size of the instance increases.

Throughout these instances we notice that increasing one parameter while keeping the

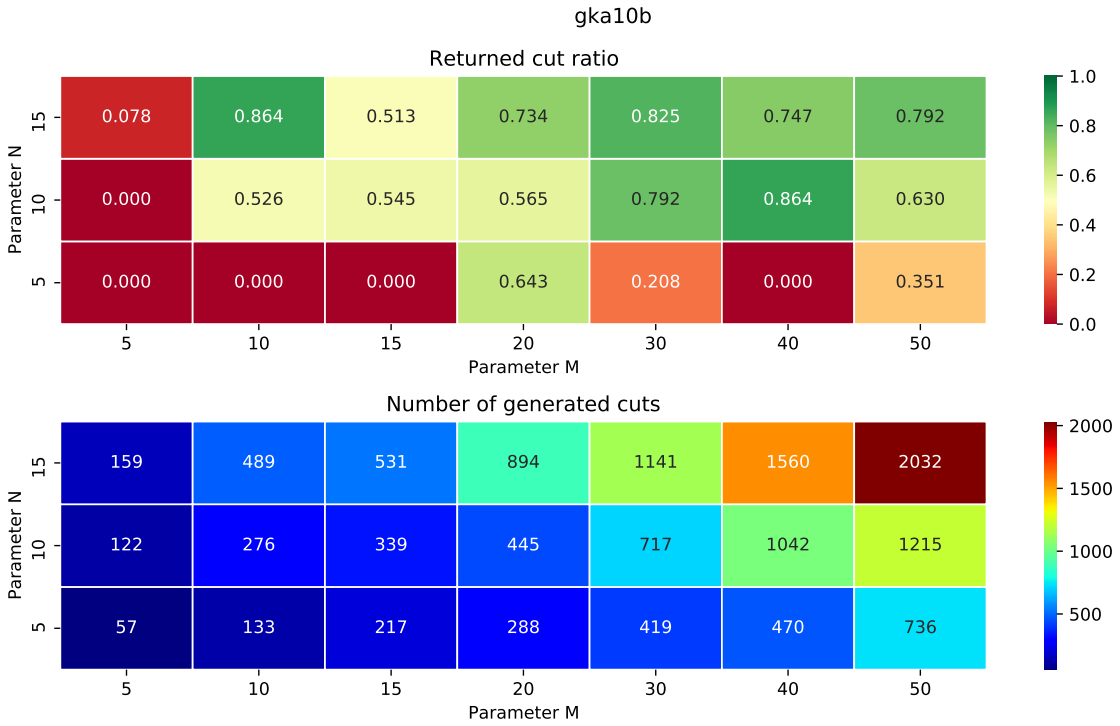


Figure 3: Overview of the returned ratios and number of generated cuts for each solver on the instance gka10b.

other the same sometimes yields a worse ratio. We will call this phenomenon a worsening, which can be observed in figure 3 fixing N=5 or M=20. Worsenings also occur in the instances, where one of the solvers finds the optimal solution,in figure 2 fixing N=10 or M=30. A worsening can be explained by recalling that the algorithm depends on a randomly chosen starting point and the improvement of the cut utilises random perturbations. We observe that the number of worsenings increases as the size of the instances increase. This indicates that the function f from subsection 2.2 "gets more complicated" for increasing sizes. By more complicated, we mean that the probability of a starting point leading to a good cut or a perturbation leading to a better cut decreases.

Although worsenings occur, it is the case that by averaging out the performance on smaller parameters, i.e. averaging the ratios of $M \in \{5, 10, 15, 20\}$ and $N \in \{5, 10\}$, opposed to larger parameters, yields a smaller ratio than the average of $M \in \{20, 30, 40, 50\}$ and $N \in \{10, 15\}$. This indicates that on average the intuition that larger parameters yield better results still holds.

Since the instance gka10b has by far the worst performance we will close this chapter by describing figure 3. In figure 3 we notice that the ratio strongly varies and that worsenings occur even for big parameters. For example M40.N10 to M50.N10 and M40.N10 to M40.N15 are both worsenings. We can even observe that the ratio 0.864 of M40.N10 is smaller than the ratio 0.792 of M50.N15. It is particularly interesting that gka10b is the only instance where the weight of the trivial cut is also the best found weight. There are 5 solvers for which the returned cut has the value of the trivial cut. Notably 4 out of those 5 solvers have $N = 5$.

3.4 Instances without known optimal value

In this subsection we are going to consider the instances, where there is no known optimal value, i.e. the library [Mal23] does not provide one. Although there is no optimal value we are still interested in the quality of a solution as a ratio. For each instance we therefore normalise the returned values by dividing by the maximal returned value. By construction the values will be between 0 and 1 with at least one value being 1. Looking at the smallest cut ratio of each instance we see that every instance has a ratio greater than 0.973214. Considering the largest standard deviation between the ratios returned for each solver we see that the standard deviation is less or equal to 0.006403. For these observations we deduce that for the instances without known optimal value the Burer heuristic is very robust. By robust we mean that the quality of the returned cut is not very sensitive to the choice of parameters.

3.5 Results

The experiments gather further evidence that the Burer heuristic is highly effective at finding high quality approximations. Out of the 332 test instances with known optimal value, there are only 6 instances, for which the heuristic returns a cut with ratio less than 0.878. Recall that 0.878 is the constant of the performance guarantee in the Goemans-Williamson algorithm. They are gka*b for $* \in [5 : 10]$.

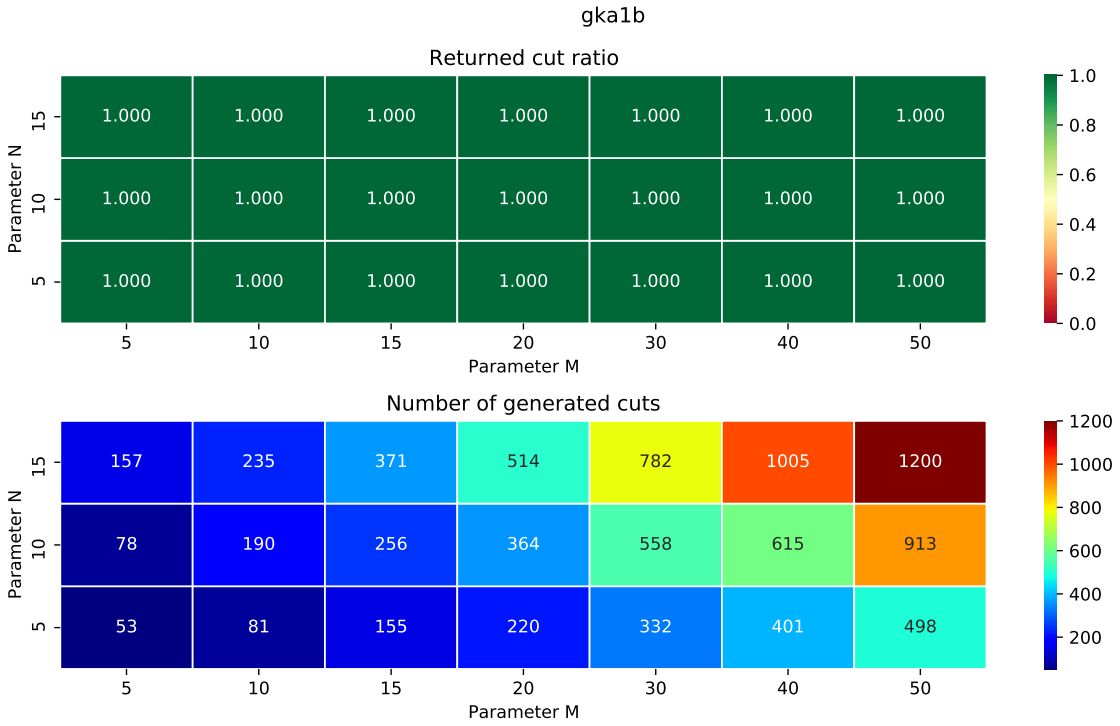
In particular the heuristic returns high quality cuts for instances with mixed sign weights. This is very impressive, as the performance guarantee of the Goemans-Williamson algorithm only holds for graphs with nonnegative weights. Therefore there is no theoretical reason for the approximation algorithm to perform well, let alone the Burer heuristic.

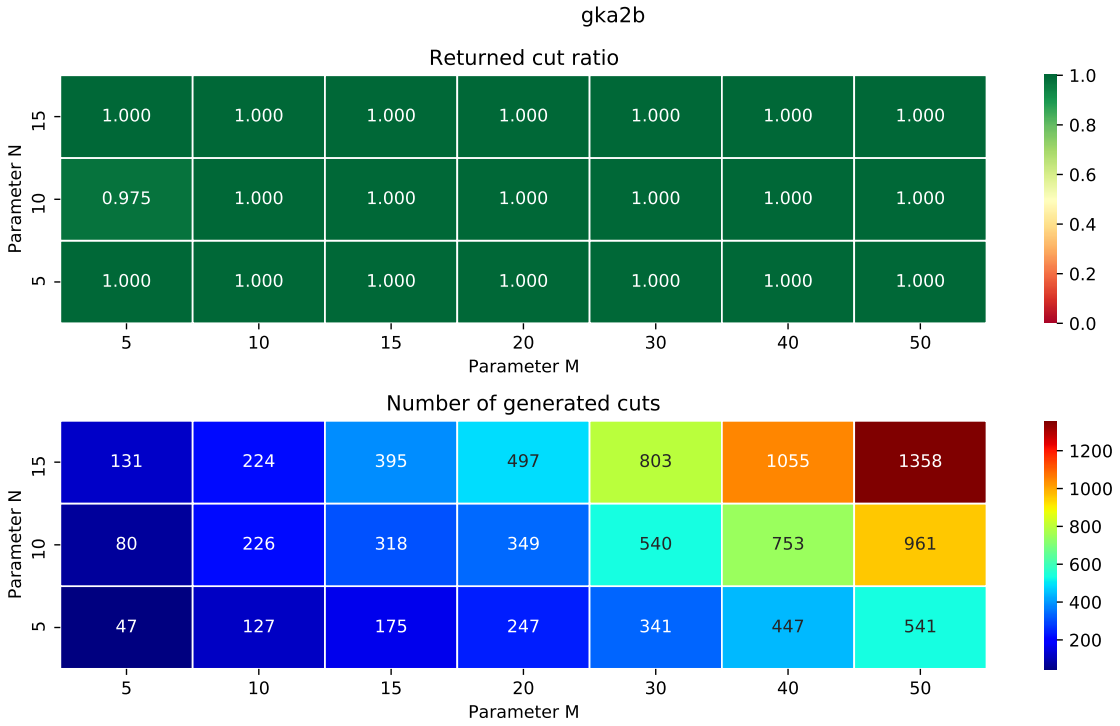
We point out that the standard deviation of the ratios of the returned cut values for different parameters is very low, except for the instances gka*b for $* \in [3 : 10]$. Therefore the data suggests, that the Burer heuristic is rather robust and that, generally speaking, if a run with large parameters finds a high quality solution, then a high quality solution can already be found with small parameters.

Finally we have identified, that the instances in the library [Mal23] that perform by far the worst are gka*b for $* \in [3 : 10]$. We have observed that those instances have the following common properties. They all have rounded density 1. They all have edges with negative weights and edges with positive weights. Again note that the edges with negative weights are the edges incident to the star vertex. Compared to the edge weights their optimal cut values are the smallest in the library [Mal23]. These instances have demonstrated the role that randomness plays in the Burer heuristic, as we have observed several occurences of worsenings, see subsection 3.3. However, we still observed that on average the returned cut value increases for larger parameters.

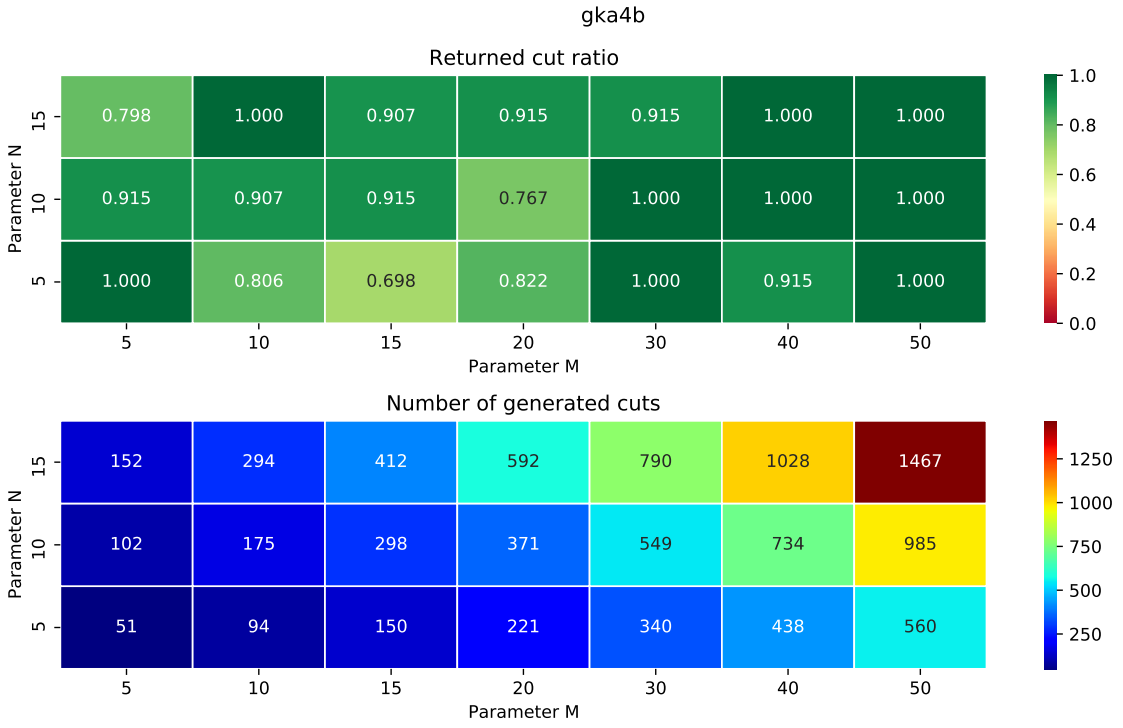
A Appendix

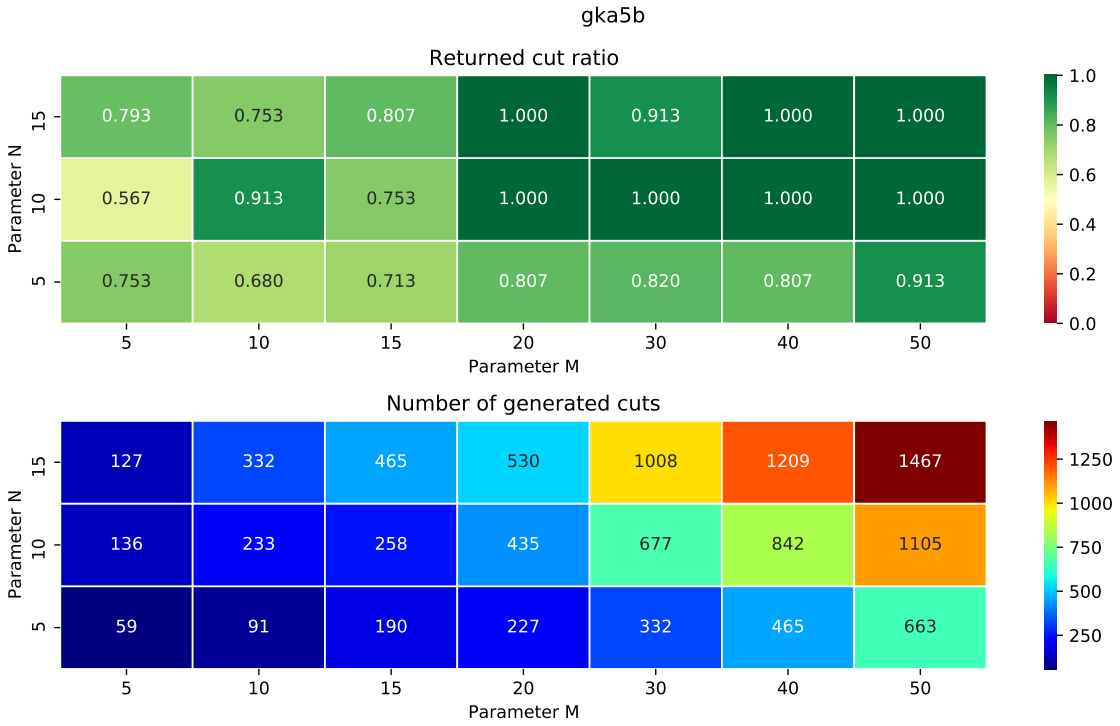
Plots of the instances gka*b for $* \in [5 : 10]$ where the entry with M=xx and N=yy stands for the solver Mxx.Nyy and the entry of the top plot shows the returned cut ratio (returned cut weight/optimal cut value) and the bottom plot shows the number of generated cuts.

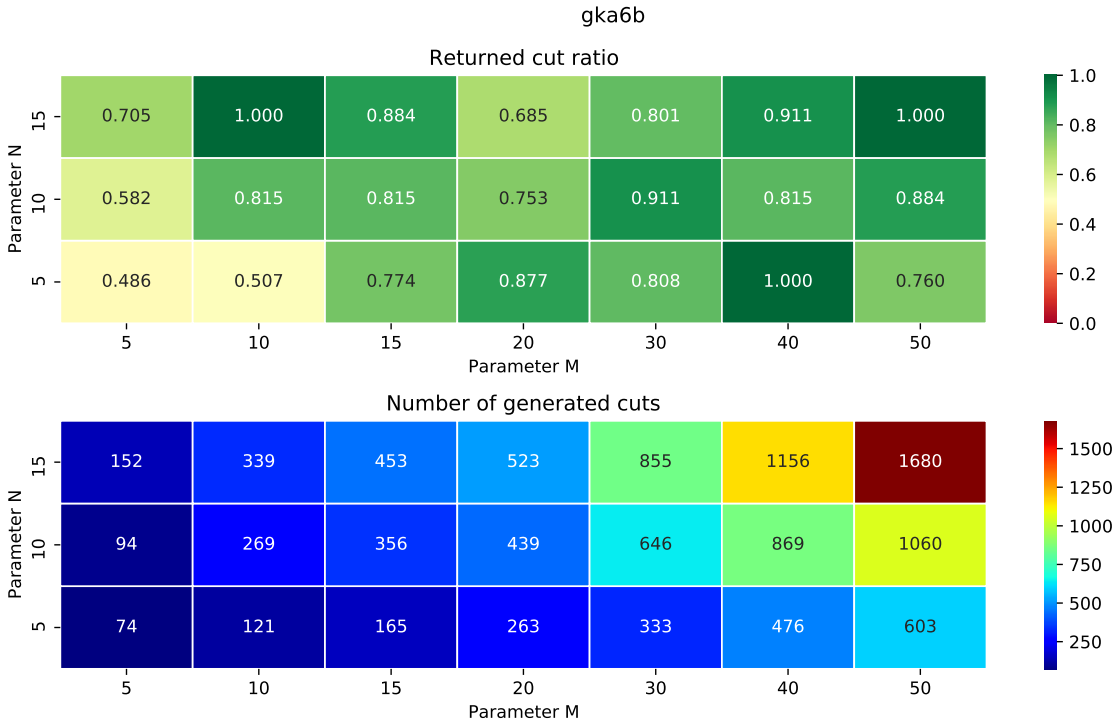

gka1b
Returned cut ratio
Parameter N
15
10
5
1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000
5 10 15 20 30 40 50
Parameter M
1.0 0.8 0.6 0.4 0.2 0.0
Number of generated cuts
157 235 371 514 782 1005 1200
78 190 256 364 558 615 913
53 81 155 220 332 401 498
1200 1000 800 600 400 200

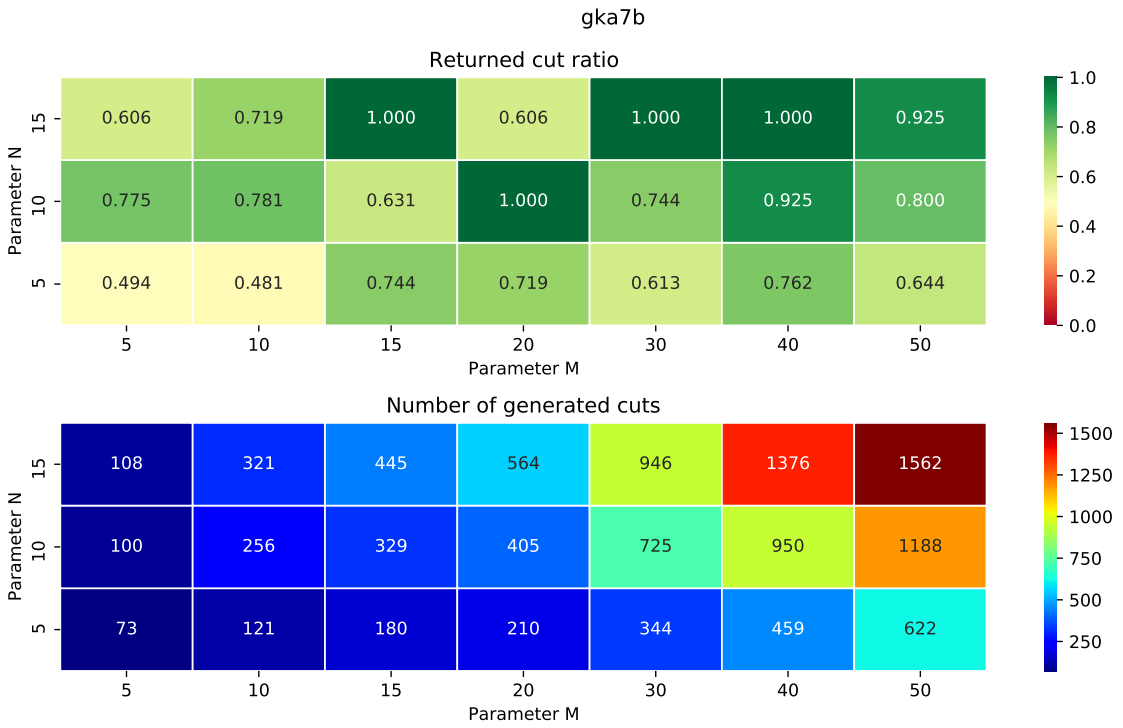

gka2b
Returned cut ratio
Parameter N
15
10
5
1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.975 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000
5 10 15 20 30 40 50
Parameter M
1.0 0.8 0.6 0.4 0.2 0.0
Number of generated cuts
131 224 395 497 803 1055 1358
80 226 318 349 540 753 961
47 127 175 247 341 447 541
1200 1000 800 600 400 200

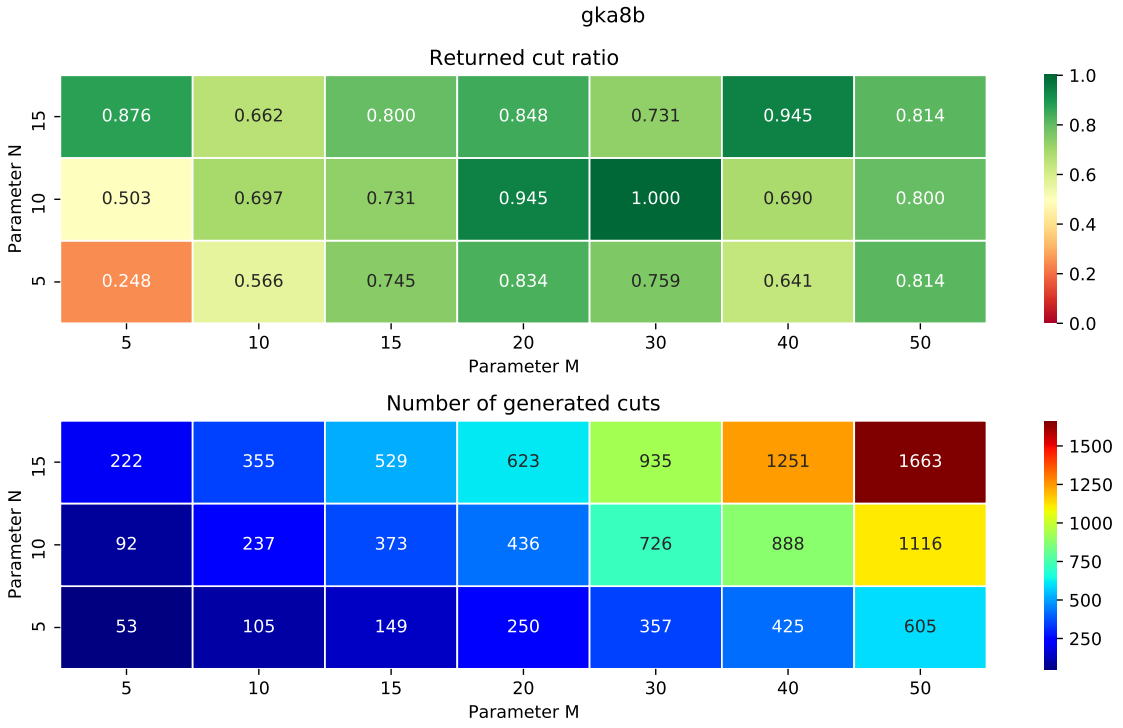

gka3b
Returned cut ratio
Parameter N
15
10
5
0.898 0.898 1.000 1.000 1.000 1.000 1.000
1.000 0.898 1.000 1.000 1.000 1.000 1.000
0.856 0.847 0.864 1.000 1.000 1.000 0.898
5 10 15 20 30 40 50
Parameter M
1.0 0.8 0.6 0.4 0.2 0.0
Number of generated cuts
141 244 444 523 778 1024 1314
113 200 254 350 593 801 967
58 85 176 247 342 514 502
1200 1000 800 600 400 200

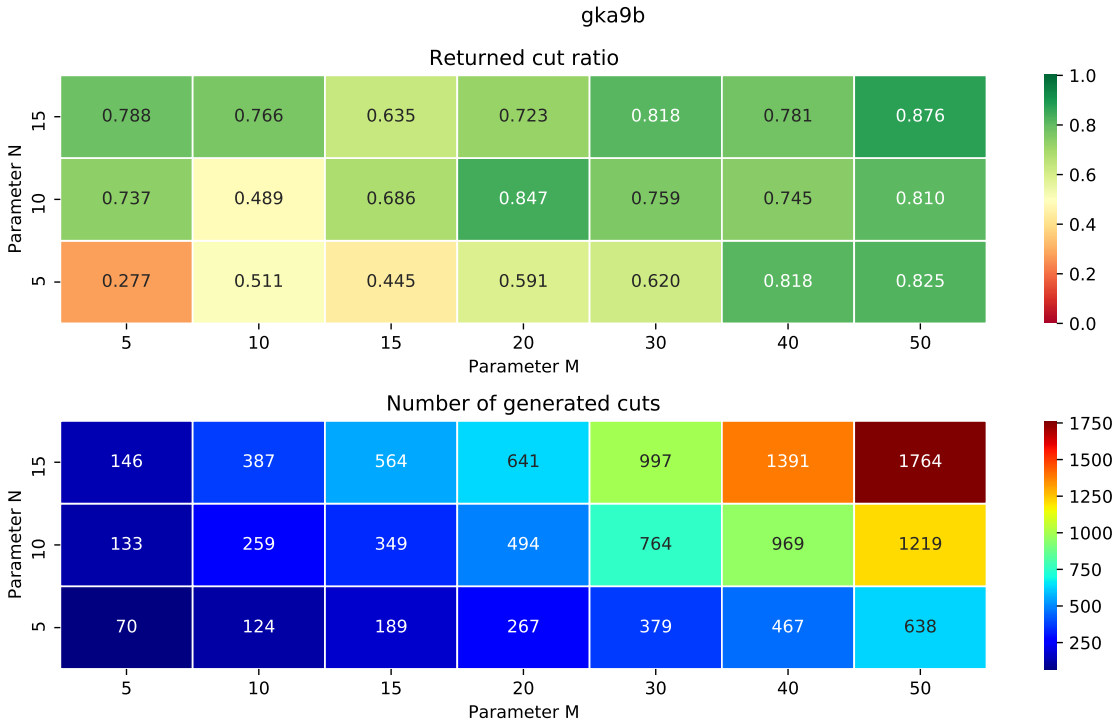

gka4b
Returned cut ratio
Parameter N
15
10
5
0.798 1.000 0.907 0.915 0.915 1.000 1.000
0.915 0.907 0.915 0.767 1.000 1.000 1.000
1.000 0.806 0.698 0.822 1.000 0.915 1.000
5 10 15 20 30 40 50
Parameter M
1.0 0.8 0.6 0.4 0.2 0.0
Number of generated cuts
152 294 412 592 790 1028 1467
102 175 298 371 549 734 985
51 94 150 221 340 438 560
1250 1000 750 500 250

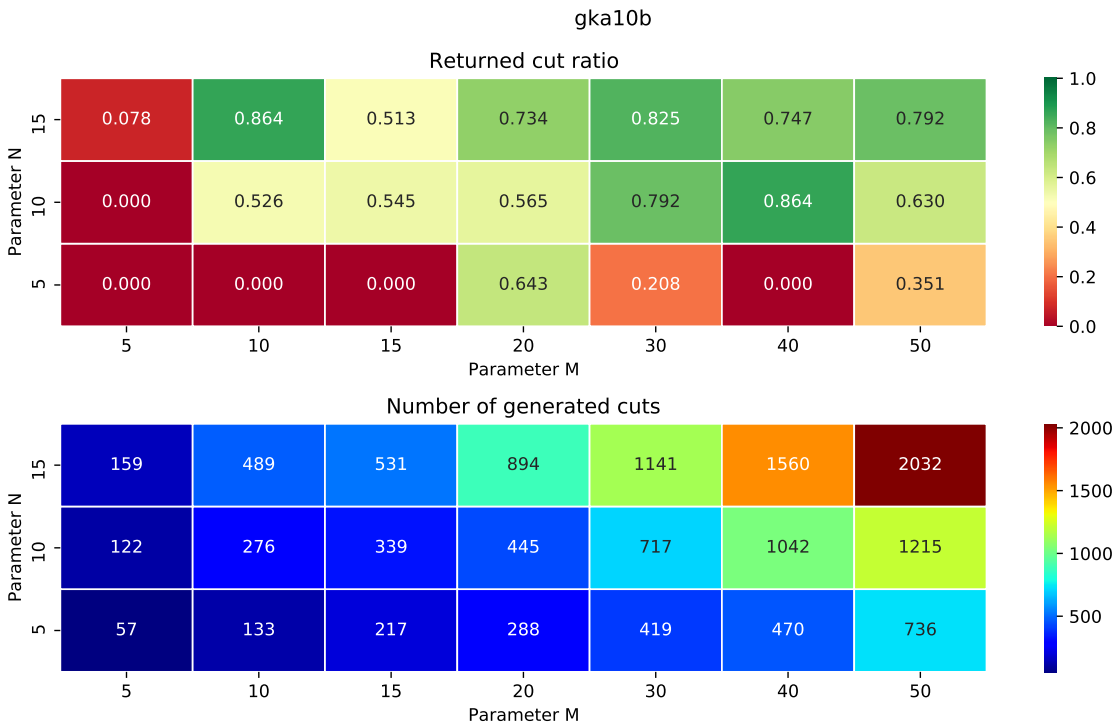

gka5b
Returned cut ratio
Parameter N
15
10
5
0.793 0.753 0.807 1.000 0.913 1.000 1.000
0.567 0.913 0.753 1.000 1.000 1.000 1.000
0.753 0.680 0.713 0.807 0.820 0.807 0.913
5 10 15 20 30 40 50
Parameter M
1.0 0.8 0.6 0.4 0.2 0.0
Number of generated cuts
127 332 465 530 1008 1209 1467
136 233 258 435 677 842 1105
59 91 190 227 332 465 663
5 10 15 20 30 40 50
Parameter M
1250 1000 750 500 250


gka6b
Returned cut ratio
Parameter N
15
10
5
0.705 1.000 0.884 0.685 0.801 0.911 1.000
0.582 0.815 0.815 0.753 0.911 0.815 0.884
0.486 0.507 0.774 0.877 0.808 1.000 0.760
5 10 15 20 30 40 50
Parameter M
1.0 0.8 0.6 0.4 0.2 0.0
Number of generated cuts
152 339 453 523 855 1156 1680
94 269 356 439 646 869 1060
74 121 165 263 333 476 603
5 10 15 20 30 40 50
Parameter M
1500 1250 1000 750 500 250


gka7b
Returned cut ratio
Parameter N
15
10
5
0.606 0.719 1.000 0.606 1.000 1.000 0.925
0.775 0.781 0.631 1.000 0.744 0.925 0.800
0.494 0.481 0.744 0.719 0.613 0.762 0.644
5 10 15 20 30 40 50
Parameter M
1.0 0.8 0.6 0.4 0.2 0.0
Number of generated cuts
108 321 445 564 946 1376 1562
100 256 329 405 725 950 1188
73 121 180 210 344 459 622
1500 1250 1000 750 500 250


gka8b
Returned cut ratio
Parameter N
15
10
5
0.876 0.662 0.800 0.848 0.731 0.945 0.814
0.503 0.697 0.731 0.945 1.000 0.690 0.800
0.248 0.566 0.745 0.834 0.759 0.641 0.814
5 10 15 20 30 40 50
Parameter M
1.0 0.8 0.6 0.4 0.2 0.0
Number of generated cuts
222 355 529 623 935 1251 1663
92 237 373 436 726 888 1116
53 105 149 250 357 425 605
1500 1250 1000 750 500 250


gka9b
Returned cut ratio
Parameter N
15
10
5
0.788 0.766 0.635 0.723 0.818 0.781 0.876
0.737 0.489 0.686 0.847 0.759 0.745 0.810
0.277 0.511 0.445 0.591 0.620 0.818 0.825
5 10 15 20 30 40 50
Parameter M
1.0 0.8 0.6 0.4 0.2 0.0
Number of generated cuts
146 387 564 641 997 1391 1764
133 259 349 494 764 969 1219
70 124 189 267 379 467 638
1750 1500 1250 1000 750 500 250


gka10b
Returned cut ratio
Parameter N
15
10
5
0.078 0.864 0.513 0.734 0.825 0.747 0.792
0.000 0.526 0.545 0.565 0.792 0.864 0.630
0.000 0.000 0.000 0.643 0.208 0.000 0.351
5 10 15 20 30 40 50
Parameter M
1.0 0.8 0.6 0.4 0.2 0.0
Number of generated cuts
159 489 531 894 1141 1560 2032
122 276 339 445 717 1042 1215
57 133 217 288 419 470 736
2000 1500 1000 500

References

[Aig07] Martin Aigner. *Discrete mathematics*. American Mathematical Society, 2007.

[BMZ02] Samuel Burer, Renato D. C. Monteiro, and Yin Zhang. Rank-two relaxation heuristics for max-cut and other binary quadratic programs. *SIAM Journal on Optimization*, 12(2):503–521, 2002.

[De 90] Caterina De Simone. The cut polytope and the boolean quadric polytope. *Discrete Mathematics*, 79(1):71–75, 1990.

[GJS74] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete problems. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, page 47–63, New York, NY, USA, 1974. Association for Computing Machinery.

[GW95] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, nov 1995.

[JM21] Michael Jünger and Sven Mallach. Exact facetial odd-cycle separation for maximum cut and binary quadratic optimization. *INFORMS Journal on Computing*, 33(4):1419–1430, 2021.

[KV18] Bernhard Korte and Jens Vygen. *Approximation Algorithms*, pages 423–469. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018.

[Mal22] Sven Mallach. Binary linear and quadratic optimization. Lecture Notes, 2022.

[Mal23] Sven Mallach. Instance library for the maximum cut and the unconstrained binary quadratic optimization problem, 2023.

[Vaz03] Vijay V. Vazirani. *Semidefinite Programming*, pages 255–269. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.