On the Study of Fitness Landscapes and the Max-Cut Problem

Der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

DISSERTATION

zur Erlangung des akademischen Grades

Doctor rerum naturalium
(Dr. rer. nat.)

im Fachgebiet

Informatik

vorgelegt von

Master of Science Informatik Angel Eduardo Rodriguez Fernandez
geboren am 18.07.1991 in Mexiko

Die Annahme der Dissertation wurde empfohlen von:

1. Prof. Dr. Peter F. Stadler, Universität Leipzig
2. Prof. Dr. Christian Reidys, Biocomplexity Institute and Department of Mathematics of the University of Virginia, USA

Die Verleihung des akademischen Grades erfolgt mit Bestehen
der Verteidigung am 24.11.2021 mit dem Gesamtprädikat cum laude

Bibliographic Description

Title:	On the Study of Fitness Landscapes and the Max-Cut Problem
Type:	Dissertation
Author:	Angel Eduardo Rodriguez Fernandez
Year:	2021
Professional discipline:	Computer Science
Language:	English
Pages in the main part:	97
Chapter in the main part:	8
Number of Figures:	30
Number of Tables:	8
Number of Appendices:	3
Number of Citations:	84
Key Words:	Max-Cut, Max-k-Cut, Fitness Landscape, Approximation algorithms, Clustering, Randomized Rounding

This thesis is in part based on the following publications.

A. E. Rodriguez-Fernandez, B. Gonzalez-Torres, R. Menchaca-Mendez, and P. F. Stadler (2020). "Clustering Improves the Goemans–Williamson Approximation for the Max-Cut Problem". In: *Computation* 8.3, p. 75. *DOI*: `10.3390/computation8030075`. *URL*: `https://doi.org/10.3390/computation8030075`.

Abstract

The goal of this thesis is to study the complexity of NP-Hard problems, using the Max-Cut and the Max-k-Cut problems, and the study of fitness landscapes. The Max-Cut and Max-k-Cut problems are well studied NP-hard problems specially since the approximation algorithm of Goemans and Williamson (1995) which introduced the use of Semidefinite Programming (SDP) to solve relaxed problems. In order to prove the existence of a performance guarantee, the rounding step from the SDP solution to a Max-Cut solution is simple and randomized. For the Max-k-Cut problem, there exist several approximation algorithms but many of them have been proved to be equivalent. Similarly as in Max-Cut, these approximation algorithms use a simple randomized rounding to be able to get a performance guarantee.

Ignoring for now the performance guarantee, one could ask if there is a rounding process that takes into account the structure of the relaxed solution since it is the result of an optimization problem. In this thesis we answered this question positively by using clustering as a rounding method.

In order to compare the performance of both algorithms, a series of experiments were performed using the so-called G-set benchmark for the Max-Cut problem and using the Random Graph Benchmark of Goemans and Williamson (1995) for the Max-k-Cut problem. With this new rounding, larger cut values are found both for the Max-Cut and the Max-k-Cut problems, and always above the value of the performance guarantee of the approximation algorithm. This suggests that taking into account the structure of the problem to design algorithms can lead to better results, possibly at the cost of a worse performance guarantee. An example for the vertex k-center problem can be seen in Garcia-Diaz et al. (2017), where a 3-approximation algorithm performs better than a 2-approximation algorithm despite having a worse performance guarantee.

Landscapes over discrete configurations spaces are an important model in evolutionary and structural biology, as well as many other areas of science, from the physics of disordered systems to operations research. A *landscape* is a function defined on a very large discrete set V that carries an additional metric or at least topological structure into the real numbers $\mathbb{R}$. We will consider landscapes defined on the vertex set of undirected graphs. Thus let $G = G(V, E)$ be an undirected graph and $f : V \to \mathbb{R}$ an otherwise arbitrary function. We will refer to the triple (V, E, f) as a *landscape* over G.

We say two configurations $x, y \in V$ are *neutral* if $f(x) = f(y)$. We colloquially refer to a landscape as “neutral” if a substantial fraction of adjacent pairs of configurations are neutral. A *flat* landscape is one where f is constant. The opposite of flatness is *ruggedness* and it is defined as the number of local optima or by means of pair correlation functions.

These two key features of a landscape, ruggedness and neutrality, appear to be two sides of the same coin. Ruggedness can be measured either by correlation properties, which are sensitive to monotonic transformation of the landscape, and by combinatorial properties such as the lengths of downhill paths and the number of local optima, which are invariant under monotonic transformations. The connection between the two views has remained largely unexplored and poorly understood. For this thesis, a survey on fitness landscapes is presented, together with the first steps in the direction to find this connection together with a relation between the covariance matrix of a random landscape model and its ruggedness.

Acknowledgment

First and foremost I would like to thank my advisor Peter Stadler for his support, trust and always finding the time to discuss. I am grateful to him for his understanding, interest and guidance in the project and for the fast and great communications, all of which made me feel supported during the duration of my PhD.

Additionally I want to express my gratitude to Ricardo and Bernardo, for all the useful discussions (both personal and academic) and the motivation to keep going no matter what.

I acknowledge the financial support of the International Max Planck Research School "Mathematics in the Sciences" and the Consejo Nacional de Ciencia y Tecnología (CONACyT). I am thankful to Heike Rackwitz and Valeria Hünniger for helping me with the administrative processes during my PhD.

I am glad to have met many amazing people during my PhD. Thank you Micha and Michelle for all the support, laughs, great memories and the tasty food, I can say that you are like a second family to me. Many thanks to Cristian, Anahy, Su, Mario, Bruno, Karsten, Roger, Tobi, Kike and all the friends that I found during my studies, the time spent with you has been definitely a highlight of my stay.

Many thanks to my family and my friends back in Mexico, for the warm welcomes and constant support. I also would like to thank David, Micha and Rubén for being the best friends one could ask for, and last but not least, thanks to Andrea and Karina for the friendship during all these years and the help with "el Piña".

Contents

Part I

Introduction and Theoretical Background

CHAPTER 1

Introduction

Several combinatorial problems are of interest not only theoretically but also for its wide variety of applications, however many of these problems are hard to solve. The NP-Hard and NP-Complete problems fall into these category, and are of interest because of the applications in efficient routing of transport vehicles, network design, cryptography and in general in computer science. For these problems, it is not known if an optimal solution can be found in polynomial time and this question has been unanswered for more than 45 years.

However, for many problems there exists a structure that can be used to find solutions efficiently, which are close to the global optima, or at least guarantee local optimality. For application purposes this is enough, since the *worst case scenario* happens rarely in real-life applications. For some particular types of instances it may even happen that the problems can be solved in polynomial time, like planar graphs for the MAX-CUT problem (Hadlock, 1975).

objective function
configurations

The fitness landscape of a discrete optimization problem can be seen as the behavior of the objective function among all possible solutions. Let f be the **objective function**, i.e., the function we want to maximize (or minimize) and let X be a finite set containing all the possible valid **configurations**, i.e., an element $x \in X$ is a valid solution of the problem with value $f(x)$. For example, in evolution, an element of X is the particular chain of DNA of an individual and f is a function that measures its *fitness*. Another example in physics are spin glasses; assume there are n particles with only 2 spin values represented as ± 1, then the set of configurations X is an n dimensional vector where each entry can only be 1 or -1, and the function to optimize (minimize in this case) is the Hamiltonian energy of the system (V. M. d. Oliveira and Fontanari, 1997; Thouless, Anderson, and Palmer, 1977).

The analysis of fitness landscapes provides a hindsight of what makes a problem difficult since informally it can be interpreted as the plot of the objective function f over the space of configurations X (Figure 1). For example Krząkała and Zdeborová (2008), Krząkała, Montanari, et al. (2007), and Zdeborová and Krząkała (2007) show that increasing the number of constraints in random CSPs (constraint satisfaction problems) decreases the number of viable solutions in the same way a phase transition occur in physical phenomena. Additionally, properties of the fitness landscape of a problem can help in the design of better algorithms or heuristics, specially methods that use local information to find the solution (Neidhart, Szendro, and Krug, 2013). Properties such as **ruggedness** (large number of local optima as seen in Figure 1a) or **neutrality** (large number of points with similar fitness values as seen in Figure 1b) are clear examples of properties to consider, since a rugged landscape has a large number of local optima making local search algorithms stop before reaching a global optimum; meanwhile for flat landscapes most algorithms will find the global optimum without a problem. One extreme case happens in convex optimization (Figure 1c), where the landscape of the problem has only one local optimum which coincides with the global optimum.

For some problems, describing the landscape from the underlying structure may not be possible, therefore probabilistic models of fitness landscapes are used (Stadler and Happel, 1999). There are several models that can be defined, and for some of them properties such as the number of local optima has been found. The

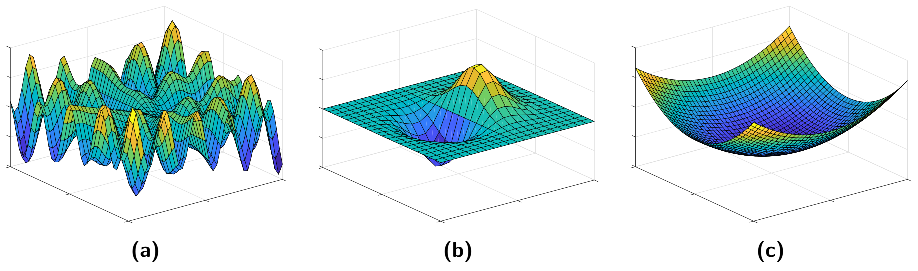


Figure 1: Approximate representations of rugged (a), neutral (b) and convex (c) fitness landscapes. The landscape of a discrete optimization problem is not a smooth function, but rather a set of discrete points such as the intersections on the grid drawn over the plot in these examples.

correlation matrix of a fitness landscape should contain most of the structure and information of the problem, therefore one may ask if properties such as the number of local optima can be seen as a function of the entries of this matrix (Stadler, 1996).

Unless $P = NP$, there cannot be polynomial time algorithms that solve NP-Complete and NP-hard problems for any instance, however we can design an algorithm which *guarantees* that the solution obtained is in an interval around the global optimum. These algorithms are called *approximation algorithms* and utilize several techniques such as probability theory, semidefinite programming and convex optimization among others (Williamson and Shmoys, 2011).

The Max-Cut problem is an NP-hard problem and its objective is to find a 2-partition of the node set of a graph such that it maximizes the number of edges that have end points in different partitions. Formally, let $G = (V, E)$ be an undirected graph where V and E are the set of nodes and edges respectively, let $i \in V$ be a node and let $(i, j) \in E$ represent an edge with endpoints i and j. Then the goal of the Max-Cut problem is to find a bipartition $A, B \subseteq V$ such that the number of edges (i, j) with $i \in A$ and $j \in B$ or vice versa is maximized (Karp, 1972). An example for a bipartite graph can be seen in Figure 2

One of the most known approximation algorithms, was done by Goemans and Williamson (1995) and instead of directly solving Max-Cut, it solves a *relaxed* problem which can be solved in polynomial time and then it *rounds* this solution to turn it into a valid solution to the Max-Cut problem. This approximation algorithm returns a solution that is guaranteed to be ≈ 0.87 times the global optimum. The rounding process of this approximation algorithm is done so that a proof of the performance guarantee can be done. This leads to ask if different rounding processes can give better solutions than the approximation algorithm, and in this thesis it was shown that it can be done using clustering. The proof of whether this approach still is an approximation algorithm has yet to be found, but the numerical data suggest a positive answer.

This approach can be generalized to the Max-k-Cut problem. Instead of partitioning the node set of a graph into two sets, it is now partitioned into k sets. It has the same objective, find a k partition that maximizes the number of edges

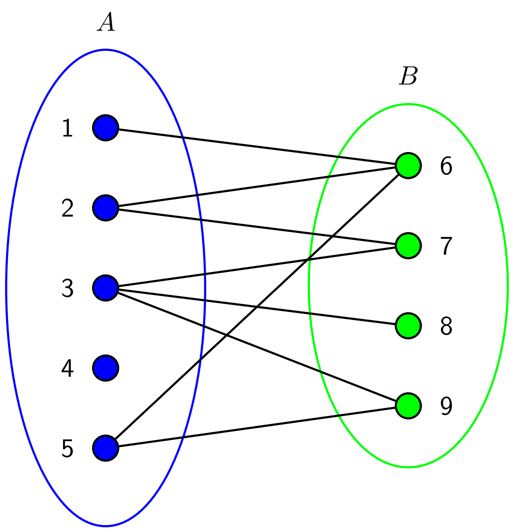


Figure 2: A bipartite graph together with the maximum cut partition A and B.

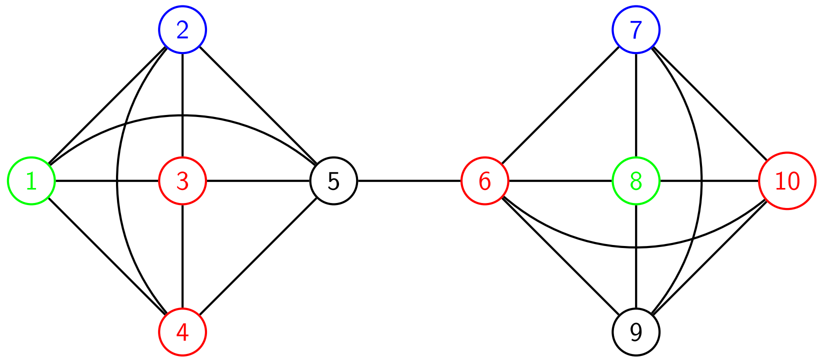


Figure 3: A graph consisting of two cliques of size 5 joined with one edge. Each color represent a partition and we have $k = 4$ partitions. A clique of size m will need $k = m$ in order to have all edges in the cut, therefore this is the best result possible for $k = 4$.

with end nodes in different partitions (Figure 3). There are several approximation algorithms for this problem, but most of them are equivalent versions of each others (Klerk, Pasechnik, and Warners, 2004; Newman, 2018). However, the rounding procedure remains the same as in the MAX-CUT ($k = 2$) case, and can still be replaced with clustering in order to find better solutions (Chapter 7). Similarly as for MAX-CUT, the data suggest that an approximation guarantee exists, but it has yet to be found.

Document Structure

The thesis is divided in three main parts. The first part, "Introduction and Theoretical Background", contains definitions and concepts that are needed in order to explain the methods and results found. The topic of Chapter 2 is NP-Complete and NP-Hard problems, how to find approximate solutions for them and some examples and iconic problems. In Chapter 3 positive semidefinite matrices are introduced together with some properties, followed by the definition of Semidefinite Programming problems. Chapter 4 expands on fitness landscapes, examples of some models and the problems encountered with trying to relate the covariance matrix with the number of local optima for the Sherrington-Kirkpatrick model. The last chapter of the first part of the thesis, Chapter 5 explains all the clustering methods used in this work.

The second part,"Methods and Findings" consists of the main results of this work. In Chapter 6 the MAX-CUT problem is introduced together with the best known approximation algorithm for it. Then, a variation of the approximation algorithm using clustering is presented together with numerical results. The Subsection 6.3.2 is based on Rodriguez-Fernandez et al. (2020) and apart from those results, spectral clustering was used and is presented in Subsection 6.3.3. Additionally, a Laplacian and Gram matrix spectrum analysis was performed to give more intuition about the structure and difficulty of the problem (Section 6.4). Finally an algorithm based on constantly minimizing in the distortion space of clustering and maximization in cut space was designed and presented in the last section. For Chapter 7, the generalization MAX-k-CUT is presented together with the best approximation algorithms. The new approach of using clustering as a rounding method is presented together with numerical results in Subsection 7.3.4 and Section 7.4.

Finally, the last part of the thesis contains the conclusions and future work for some of the problems discussed above.

CHAPTER 2

NP-Complete and NP-Hard Problems

Contents

One of the most studied problems in computational complexity are the NP-complete and NP-hard problems. There is a wide variety of problems that fall into this category, for example the traveling salesman problem, edge coloring, maximum satisfiability and maximum cut among others, with a wide variety of applications such as network design, efficient routing of vehicles, and many others.

2.1 Definitions

A problem belongs to the complexity class **P** if it can be solved in polynomial time for any instance. The class **NP** consists of all decision problems that can check if a given solution is correct in polynomial time. A problem in the **NP-Complete** class belongs to the NP class and any other problem in NP can be reduced to it, i.e., it can be seen as a particular case of it. A problem in the **NP-Hard** class does not need to be in NP, but all problems in NP can be reduced to it. Clearly P is contained in NP (since problems itself can be solved in polynomial time), but until now, a proof of whether the NP class is contained or not in the P class has not been found. This is the "P versus NP" problem and it is one of the most important question in the field, one of the Millennium Prize Problems, and has been unsolved for more than 45 years. The implications of the answer will have an impact on many fields including cryptography, mathematics, algorithm design and many others, specially if $P = NP$.

NP-Complete

NP-Hard

2.2 Methods to Approximate Solutions

Although the answer to $P = NP$ is still unknown, there are many approaches to obtain "good" solutions in polynomial time. NP-hard problems can not be, simultaneously:

- Solved exactly,
- in polynomial time and
- for any instance.

One approach is to ignore the last constraint, and focus on a particular type of instances. For example, the Max-Cut problem can be solved exactly and in polynomial time for planar graphs (Hadlock, 1975). Another approach is to solve a problem in polynomial time, but instead of finding the optimal solution, give one that is guaranteed to be inside an interval close to the optimum. This is called an **Approximation Algorithm**, and the solution obtained is guaranteed to be between the optimum and α times the optimum, where $\alpha \in \mathbb{R}$ is the **performance guarantee** and takes the values $\alpha < 1$ for maximization problems and $\alpha > 1$ for minimization problems. Formally, if OPT is the optimal solution of the problem, and S is the solution obtained by an approximation algorithm, then:

Approximation Algorithm

$$\begin{aligned} \alpha OPT \leq S \leq OPT &\quad \text{for maximization with } \alpha < 1 \\ OPT \leq S \leq \alpha OPT &\quad \text{for minimization with } \alpha > 1 \end{aligned}$$

There are several methods and tools used in the design of approximation algorithms, and it depends on the problem to be solved. For example: greedy algorithms for the set cover and the k-center problem, linear programming for the Steiner tree and the facility location, local search for finding a minimum degree spanning tree, and semidefinite programming for the MAX-CUT, MAX-k-CUT and coloring problem.

The MAX-CUT problem can be formulated as a quadratic integer optimization problem, and in order to find an approximation algorithm, one consider a *relaxation* of the problem, in which the variables are no longer integers but vectors in a real space. This new *relaxed* problem is a particular instance of a semidefinite programming problem, which can be solved in polynomial time. However, the solution obtained may not be a valid solution for the original problem, therefore one must *round* it. In Chapter 6 this approximation algorithm is discussed in detail and in Chapter 7 a similar algorithm for the MAX-k-CUT problem is presented.

The approximation algorithm for the MAX-CUT problem was found by Goemans and Williamson (1995), and several approximation algorithm follow the same procedure: first a problem is relaxed into another one that contains it but can be solved efficiently, then solve the new problem and finally find a way to round the solution of the relaxation to the original problem.

Problem Relaxation

Formally, a **relaxation** of a problem PR is another problem PS for which all instances of problem PR are valid, but not necessarily the other way around. This implies that the optimum found by the relaxation PS gives an upper bound of the optimum of PR for maximization and a lower bound for minimization. Using relaxations to find approximation algorithms is frequently used, and often the relaxed problems are instances of linear programming or semidefinite programming (Chapter 3) which can be solved in polynomial time with interior point methods (Karmarkar, 1984).

Let us take the set cover problem as an example of both relaxing and rounding a problem. Let $E = \{e_1, \ldots, e_n\}$ be a set containing n elements and let $S_1, \ldots, S_m \subseteq E$ be subsets of E with associated weights $w_1, \ldots, w_m$. The goal of the set cover problem is to select a collection $I \subseteq \{1, \ldots, m\}$ of subsets such that all elements of E are contained in at least one subset ($e_i \in S_j$ for some $j \in I$) and the sum of weights of all the selected subsets is minimized. We can formulate this problem as an *integer program*, i.e., an optimization problem where the variables can only take integer values. Let $x_i \in \{0, 1\}$ be an integer variable where the value 1 represents that the subset S_i is used in the set cover ($i \in I$); and 0 that S_i is not used in the set cover ($i \notin I$). The integer programming version of the set cover problem reads as follows:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{m} w_i x_i & & \\ \text{subject to:} \quad & \sum_{j|e_i \in S_j} x_j \geq 1 & & \forall i = 1, \ldots, n \\ & x_i \in \{0, 1\} & & \forall i = 1, \ldots, m \end{aligned} \tag{2.1}$$

where the first constraint is to assure that every element belong to at least one S_j. In general, integer optimization problems cannot be solved in polynomial time, therefore we need to *relax* this problem into another that can be solved efficiently and contains the original one. The main difficulty of an integer optimization problem is the last constraint, i.e., the integer part, and if we "relax" this constraint allowing x_i to be a real number y_i we obtain the following *linear programming* problem:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{m} w_i y_i & & \\ \text{subject to:} \quad & \sum_{j|e_i \in S_j} y_j \geq 1 & & \forall i = 1, \ldots, n \\ & y_i \geq 0 & & \forall i = 1, \ldots, m \end{aligned} \tag{2.2}$$

which can be solved in polynomial time. In order for a problem to be a *relaxation*, two things must be satisfied: first, all the feasible solutions of the original problem have to be feasible in the relaxed problem; and second, the value of the objective function for solutions of the original problem must be the same as the value for the relaxed problem. We can give an intuition of this relaxation process in the context of fitness landscapes. Imagine that Figure 4 is the plot of the landscape of some function f, i.e., it is the plot of the objective function f over the search space. In Figure 4a we can see the integer problem together with its relaxation on Figure 4b. Since many algorithms utilize local information, we can see why it may be easier to work with the relaxed problem; we are making the search space larger but more *well-behaved*.

Returning to the set cover example, we can easily see that an instance of Equation 2.1 is feasible on the relaxed problem (Equation 2.2), and since the objective functions are the same, Equation 2.2 is a valid relaxation for the set cover problem. However, since the relaxed problem has a larger search space, in general the optimum of the relaxed problem is not feasible for the original problem. We need to **round** this solution to one of its nearest feasible solutions of the original problem. Looking at Figure 4b, the minimum point of the surface may not be one of the green points which are the feasible values for the integer programming. Therefore, we must devise a method to convert this optimum into one of the nearest feasible (green) points. Let Z_{LP} and Z_{IP} be the global minimum for the relaxed and the original problem respectively, then $Z_{LP} \leq Z_{IP}$ since it contains all the solutions of the integer program and more. Let Z_R be the value obtained after

Rounding

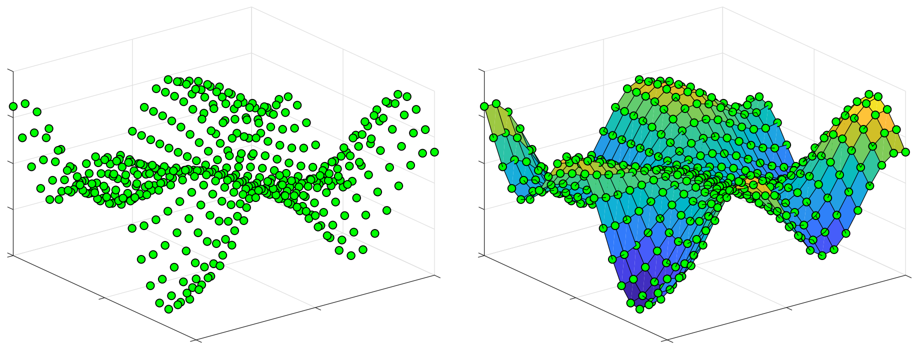

(a) Integer optimization problem **(b)** Relaxed problem

Figure 4: Fitness landscape of an integer optimization problem and its relaxation. Any point on the surface represents a feasible solution for the relaxed problem and the green circles represent the solutions where the variables are restricted to be integers.

rounding the optimal solution of the relaxed problem Z_{LP}, if there exists a number $\alpha \geq 1$ (for minimization problems), such that $Z_R \in [Z_{IP}, \alpha Z_{IP}]$, then the whole procedure is called an α-approximation algorithm. There exist several methods for rounding solutions, which depend on the properties and structure of each particular problem. However there are two classes of rounding methods, deterministic and randomized. The rounding method for the set cover problem is simple, letting t be the maximum number of times an element appears among all the sets S_j, we convert the optimum of Equation 2.2 y_i^* into a feasible solution for the set cover problem by setting $x_i^* = 1$ if $y_i \geq 1/t$ and 0 otherwise. Then, one needs to show that the set of x_i^* are feasible for Equation 2.1, that the solution obtained is guaranteed to be in the interval $[Z_{IP}, \alpha Z_{IP}]$ with $\alpha = t$, and that it is found in polynomial time. For more detail on these proofs, see Williamson and Shmoys (2011). This procedure has to be done in general for any rounding, i.e., show that the rounded solution is feasible in the original problem and that there exists an approximation guarantee α.

Another way of modifying the search space to guide local search algorithms towards the global optimum is discussed in Klemm, Mehta, and Stadler (2012). Assume we have an integer minimization problem and let us rename the objective function as the *energy*, allowing us to view the optimization problem in the context of an energy-minimization problem. The idea is to change the original encoding of a problem X into a larger space Y together with a mapping $\alpha : Y \to X \bigcup \emptyset$, where $\emptyset$ represents configurations that are non-feasible in the original problem. If the encoding Y have a larger density of low-energy states than X, even a random selection of configurations in Y will give lower energy values than in X. This idea was performed for the traveling salesman, the number partition and the MAX-CUT problems in Klemm, Mehta, and Stadler (2012).

Additionally, since the relaxation of a problem can be solved in polynomial time, it can be used in the design of **branch-and-bound** algorithms (Land and Doig, 1960). A branch-and-bound algorithm starts with the entire solution space of a problem and then starts to partition it, for example by adding constraints on variables. Assume we have integer binary variables $x_i = \pm 1$, then we can partition the solution space into two subspaces by adding the restrictions $x_i = 1$ and $x_i = -1$. If we see this procedure as a graph, the starting node and root is the complete problem, and it will **branch** into two nodes based on the restrictions defined below. Each subsequent node can branch into smaller subproblems the same way as before and if we continue this procedure we will end with the decision tree of the problem, where the leaves are all possible solutions of the problem. At some point during the search space division, we can obtain reasonable size subproblems for which we can use the optimum of the relaxation to find a **bound** of the optimal value at that particular node, which can be cut from the decision tree if it does not contain the optimum of the original problem, thus eliminating it together with all of its subproblems and reducing the size of the solution space. A more detailed explanation of this process can be found in Bader, Hart, and Phillips (2005) and Benaïchouche et al. (1996).

2.3 Examples and Iconic Problems

In this section we will present a list of NP-Complete and NP-Hard problems together with one of the best known performance guarantees and the type of method this approximation algorithm uses (Williamson and Shmoys, 2011).

- **Set Cover**. This NP-Hard problem consists of a set of elements $E = \{e_1, e_2, \dots, e_n\}$ together with m subsets $S_1, S_2, \dots, S_m \subseteq E$ with associated positive weights $w_1, \dots, w_m \geq 0$. The goal of Set Cover, as its name implies, is to find a group of sets S_i such that they "cover" all elements in E with a minimal sum of weights. Formally, we want to find $I \subseteq \{1, \dots, m\}$ such that $\bigcup_{i \in I} S_i = E$ and the sum $\sum_{i \in I} w_i$ is minimized. For this problem, there are both a **Linear Programming (LP)** and a **greedy** algorithms. The LP method relaxes the problem to an instance of LP which can be solved in polynomial time, then this solution can be rounded to be valid for the Set Cover problem and it will give an f-approximation algorithm, where f is the maximum number of times any element e_i repeats among different sets S_j. The greedy algorithm (valid only for unitary weight sets) is simple, just add a subset S_j if at the current point of the algorithm it is the one that contains the maximum number of uncovered elements e_j. This will give a performance guarantee of $H(s)$, where H is the n-th harmonic number and s is the maximum cardinality among all sets in S. Refer to Slavík (1996) for a more detailed analysis of the greedy algorithm.

- **Scheduling Jobs with Deadlines on a Single Machine**. For this problem, n jobs $J_1, \dots, J_n$ must be processed on a single machine. These jobs have a release time r_i, which means that job J_i cannot be processed before time r_i, and a deadline or due date d_i, which marks the latest point in time for which it has to be finished. Additionally, each job has a processing time p_i, i.e., the amount of time job i takes to be processed. Assume that the starting time is 0, and all $r_i \geq 0$. Assume we have a scheduling of the jobs and job J_i finishes at time C_i in this assignment. Then the *lateness* L_i for this job is equal to $C_i - d_i$. The goal of this problem is to find a scheduling such that the maximum lateness among all jobs is minimized. A simple greedy algorithm consists of processing the job with earliest due date as soon as the machine is free. This **greedy** algorithm is a 2-approximation algorithm, i.e., it has a performance guarantee of $\alpha = 2$.

- **k-center Problem**. Let $G = (V, E)$ be a complete undirected graph with positive distances d_{ij} between any pair of nodes i and j. We will require that the distance measure satisfies the triangle inequality, that $d_{ii} = 0$ and $d_{ij} = d_{ji}$ for all $i, j \in V$. The objective is to form k groups such that nodes belonging in one group are *similar* or close to each other. Let $S \subseteq V$ such that $|S| = k$ be the set that contains the k centers for all the groups. Then a node i will be assigned to its closest center in S. In order to quantify how good this assignment is, we define the distance from a node i to the set S as $d(i, S) = \min_{j \in S} d_{ij}$, and the radius of S as $\max_{i \in V} d(i, S)$. The goal of the k-center problem is to find a set of centers S with minimum radius. There is a **greedy** 2-approximation algorithm, which starts by randomly selecting a node as first center of S, then it includes as next center the node with largest distance to the current set S and repeat until $|S| = k$.

- **Traveling Salesman Problem (TSP)**. This problem is one of the most studied NP-complete problems, and it consists of n cities $\{1, 2, \dots, n\}$ with a matrix of costs C, where c_{ij} represent the cost or distance to go from city i to j. This matrix is assumed to be symmetric and with diagonal 0. This can also be seen as a complete undirected graph where the nodes are the cities and the edge weights are the costs c_{ij}. The goal of the Traveling Salesman Problem (TSP) is to find a tour with minimum cost that visits all cities only once and starts and ends on the same city. In the context of graphs, this is the equivalent of finding a Hamiltonian cycle with minimum cost. Christofides (1976) designed a $3/2$-approximation algorithm using minimum spanning trees, perfect matchings and Eulerian circuits. This was the best known approximation algorithm until 2020, where Karlin, Klein, and Gharan (2020) designed a variation with a performance guarantee of $3/2 - 10^{-36}$.

- **Hamiltonian Path**. As discussed above, finding a Hamiltonian cycle is NP-Hard, but even finding a Hamiltonian path is NP-hard. A Hamiltonian path must visit all nodes once, but it is not necessary to start and end on the same node.

- **Minimum Degree Spanning Tree**. A **spanning tree** of a graph $G = (V, E)$ is a minimal subset of edges $F \subseteq E$ for which there is a path between any pair of nodes using the edges of F. For this problem, the goal is to find a spanning tree T such that the maximum degree of nodes in T is minimized. This problem is NP-hard, and there exists an algorithm that finds an optimal tree of maximum degree at most $\text{OPT} + 1$, which is the best possible result unless P=NP. The algorithm starts with an arbitrary tree and then does **local changes** in order to reduce the degree of nodes.

- **Edge Coloring**. Let $G = (V, E)$ be an undirected graph. We will say G is k-edge-colorable if there exist an assignment of k colors to the edges such that edges with the same endpoint have different colors. The goal of edge coloring is to find the smallest k for which a k-edge-coloring exists. Let Δ be the maximum degree among all the vertices of G. Note that Δ is a lower bound on k. For graphs with $\Delta = 3$, the edge coloring decision problem for $k = 3$ is NP-complete. There is a **greedy/local search** approximation algorithm with performance guarantee of $\Delta + 1$ for this case. The greedy part is to find uncolored edges and color them until a coloring with $\Delta + 1$ colors is not possible anymore. Then, doing local changes of some edge colors, the coloring can be corrected.

- **Vertex Coloring**. It is the analogous of the edge coloring but for vertices. In this problem, the idea is to find the smallest number of colors to assign to the vertices such that no pair of vertices joined by an edge share the same color. The smallest number of colors required to color a graph G is called the chromatic number, denoted by $\chi(G)$. This problem is NP-complete, however it has been shown that a coloring of $\Delta + 1$ exists, where Δ is the maximum degree among the nodes of G. There is a **greedy** algorithm that greatly depends on the initial ordering of the nodes. The idea is to start assigning colors $0, 1, 2, \ldots$ to nodes using the smallest color value that has no conflict, i.e., assign to node i the smallest color value such that its neighbors have all different colors.

- **Bin-Packing Problem**. For this NP-hard problem, we have n objects with sizes $1 > s_1 \geq s_2 \geq \cdots \geq s_n > 0$ and the idea is to pack this objects into the minimum number possible of size 1 bins, i.e., a bin can contain several objects as long as their sum of sizes is less than 1. This is related to the **partition problem**, an NP-complete decision problem that consists of n positive integer numbers $b_1, \ldots, b_n$ whose total sum B is even, and the goal is to find a bipartition S and T of the numbers such that the sum of numbers in S is equal to the sum of numbers in T. The partition problem can be reduced to the bin-packing problem by setting the sizes $a_i = 2b_i/B$ and checking if the objects can be packed using only two bins. For the *First-Fit-Decreasing* algorithm, the pieces have to be ordered in a non-increasing size manner, then the algorithm packs each piece in order in the first bin until a piece does not fit anymore, then it opens a new bin and continues packing the pieces in order and in the first bin in which it fits, until a new piece cannot be packed in any of the existing bins and so on. Assuming OPT is the optimal number of bins needed, the First-Fit-Decreasing algorithm is a $(11/9)\text{OPT} + 6/9$ (Dósa, 2007) approximation algorithm.

- **Weighted Completion Times on a Single Machine**. This NP-hard problem is similar to the scheduling jobs with deadlines on a single machine problem.

Assume we have n jobs with processing time $p_i \geq 0$, release date $r_i \geq 0$ and weight $w_i \geq 0$. The goal is to find the minimum weighted sum of completion times C_j, i.e., minimize $\sum_{j=1}^{n} w_j C_j$, where C_j is the time when job j finishes processing. There is a 3-approximation algorithm using a relaxed LP problem, and a 2-approximation algorithm using a randomized rounding of an Integer Programming (IP) relaxation.

- **Minimum-Cost Steiner Tree**. Let $G = (V, E)$ be an undirected graph with edge cost c_{ij} for all edges $(i, j) \in E$. Let $R \subseteq V$ a subset of nodes called *terminals*. The goal of this NP-hard problem is to find a tree that contains all terminals with minimum cost. Note that this problem includes both the shortest path (if R has only two terminals) and the minimum spanning tree (if $R = V$) problems. Byrka et al. (2010) designed a $\ln(4) + \epsilon \leq 1.39$ approximation algorithm by doing a iterative randomized rounding technique on a LP relaxation of the problem.

- **Maximum Satisfiability**. Let $x_1, \ldots, x_n$ be n Boolean variables, i.e., each variable x_i can be `true` or `false`. Let $C_1, \ldots, C_m$ be m clauses, where each clause C_i has an associated weight w_i and is a disjunction of variables x_j and their negation $\bar{x_j}$. For the general Maximum Satisfiability (MAX SAT) problem, the number of variables in a clause is not fixed, but if clauses must have at most k variables, then we refer to this problem as MAX kSAT. The goal of MAX SAT is to find values of the variables x_i such that the weight sum of the satisfied clauses is maximized. A simple **random assignment** of the variables gives an expected $1/2$ approximation algorithm. Rounding a **LP relaxation** of the problem gives a $1 - 1/e$ approximation algorithm.

- **Max-Cut**. Let $G = (V, E)$ be an undirected weighted graph. The goal of $\textsc{Max-Cut}$ is to partition the node set into two disjoint sets A and B, such that the weight of edges having one end node in A and the other one in B is maximized. For the $\textsc{Max-}k\textsc{-Cut}$ problem, the goal is to partition the node set into k disjoint sets such that the weights of edges with end nodes in different sets is maximized. A broader definition and some approximation algorithms for the $\textsc{Max-Cut}$ and the $\textsc{Max-}k\textsc{-Cut}$ problem are presented in Chapter 6 and Chapter 7 respectively.

CHAPTER 3

Semidefinite Programming

Contents

Semidefinite Programming (SDP) is a convex optimization problem that appears often as the relaxation of several combinatorial optimization problems, such as graph coloring, Max-Cut and Max-k-Cut. The objective of an SDP problem is to optimize a linear function of a positive semidefinite matrix subject to linear constraints. An instance of SDP can be solved efficiently (in polynomial time) using interior point methods, making it a very useful tool in the design of approximation algorithms. Before talking about SDP, we need to introduce the concept of a Positive Semidefinite (PSD) matrix.

3.1 Positive Semidefinite Matrices

Positive Semidefinite Matrix

Let $X \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Then X is a **positive semidefinite** matrix if

$$v^T X v \geq 0 \quad \text{for any } v \in \mathbb{R}^n$$

Let S^n denote the set of symmetric $n \times n$ matrices and let S^n_+ be the set of symmetric $n \times n$ positive semidefinite matrices. Some properties of PSD matrices are (Helmberg, 2000):

- If $X \in S^n$, then $X = QDQ^T$ for some orthonormal matrix Q and some diagonal matrix D. A matrix Q is **orthonormal** if $Q^{-1} = Q^T$.
- If $X = QDQ^T$ as above, then the columns of Q form a set of n orthogonal eigenvectors.
- $X \succeq 0$ if and only if $X = QDQ^T$ where the eigenvalues (i.e. the diagonal entries of D) are all nonnegative.
- If $X \succeq 0$ and if $X_{ii} = 0$, then $X_{ji} = X_{ij} = 0 \quad \forall j = 1, \ldots, n$.
- Consider the matrix $M = \begin{pmatrix} P & \nu \\ \nu^T & d \end{pmatrix}$, where $P \succeq 0$, ν is a vector and d is a scalar. Then $M \succeq 0$ if and only if $d - \nu^T P^{-1} \nu \geq 0$.
- $X = vv^T \succeq 0 \quad \forall v \in R^n$.
- If $X \succeq 0$, X can be written as $X = N^T N$.
- If X is symmetric, then $\sum_{j=1}^n X_{jj} = \sum_{j=1}^n \lambda_j$.
- If X is symmetric, then $\det(X) = \prod_{i=1}^n \lambda_i$.

Linear Function of a Matrix

We also need to define a linear function of a matrix. Let $X \in S^n_+$ and $C \in \mathbb{R}^{n\times n}$, then a **linear function** $C(X) = C \cdot X$ is defined as:

$$C \cdot X := \mathrm{Tr}(C^T X) = \sum_{i=1}^n \sum_{i=1}^n C_{ij} X_{ij} \tag{3.1}$$

where $\text{Tr}(X)$ is the trace of matrix X. Since X is symmetric, without loss of generality we can assume that C is also symmetric.

Lastly, let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$. Then, the Kronecker product is the block matrix $A \otimes B \in \mathbb{R}^{pm\times qn}$ defined as:

$$A \otimes B = \begin{pmatrix} A_{11}B & \cdots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \cdots & A_{mn}B \end{pmatrix}$$

3.2 Definition

Semidefinite Program

Let $X \in S^n_+$, $C \in S^n$, $A_1, A_2, \ldots, A_m \in S^n$ and $b \in \mathbb{R}^n$. Then a **semidefinite program** is the following optimization problem:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & C \cdot X \\ \text{subject to} \quad & A_i \cdot X = b_i, \quad i = 1, \ldots, m \\ & X \succeq 0 \end{aligned} \tag{3.2}$$

For an SDP problem, the matrix X is the variable, the objective function is $C \cdot X$, X must satisfy m linear equations and it must lie in the cone of PSD matrices. The $m+1$ matrices $C, A_1, \ldots, A_m$ and the vector b are the data of the SDP problem.

SDP is a particular problem of convex optimization and it can be solved efficiently with interior point methods, therefore it is widely used as relaxed problems of NP-complete or NP-hard problems.

The dual problem of Equation 3.2 (SDD) is:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} y_i b_i \\ \text{subject to} \quad & \sum_{i=1}^{m} y_i A_i + S = C, \\ & S \succeq 0 \end{aligned} \tag{3.3}$$

Given multipliers $y_1, \ldots, y_m$, the objective is to maximize the linear function $\sum_{i=1}^{m} y_i b_i$. The constraints restrict the matrix $S = C - \sum_{i=1}^{m} y_i A_i$ to be positive semidefinite ($S = C - \sum_{i=1}^{m} y_i A_i \succeq 0$).

3.3 Complex Semidefinite Programming

In the previous sections we worked with matrices over the reals, however SDP can be generalized over the complex space. Instead of working with symmetric matrices $M \in \mathbb{R}^{n\times n}$, we will consider Hermitian matrices $Z \in H_n$, where H_n is the set of Hermitian matrices of dimension $n \times n$. An instance of a Complex Semidefinite Programming (CSDP) problem is defined as:

$$\begin{aligned} \underset{Z}{\text{maximize}} \quad & C \cdot Z \\ \text{subject to} \quad & A_i \cdot Z = b_i, \quad i = 1, \ldots, m \\ & Z \succeq 0 \\ & Z \in H_n \end{aligned} \tag{3.4}$$

where $A \cdot B = \sum_{i,j} \bar{B}_{ij} A_{ij}$ is the generalization of Equation 3.1 for complex numbers. As shown in Goemans and Williamson (2004), a $Z \in H_n$ CSDP problem is reducible to an SDP problem with a symmetric matrix $X \in S^{2n}$. Defining the mapping $\mathbb{T}(Z) : H_n \to S^{2n}$ as

$$\mathbb{T}(Z) = \begin{pmatrix} \text{Re}Z & -\text{Im}Z \\ \text{Im}Z & \text{Re}Z \end{pmatrix}$$

the equivalent SDP version of Equation 3.4 is

$$\begin{aligned} \underset{Y}{\text{maximize}} \quad & \mathbb{T}(C) \cdot Y \\ \text{subject to} \quad & \mathbb{T}(A_i) \cdot Y = 2b_i, \quad i = 1, \ldots, m \\ & \begin{pmatrix} E_{ij} & 0 \\ 0 & -E_{ij} \end{pmatrix} \cdot Y = 0, \quad i, j = 1, \ldots, n, i < j \\ & \begin{pmatrix} 0 & E_{ij} \\ E_{ij} & 0 \end{pmatrix} \cdot Y = 0, \quad i, j = 1, \ldots, n, i < j \\ & Y \succeq 0 \\ & Y \in S^{2n} \end{aligned} \tag{3.5}$$

where the matrix E_{ij} has 1 in positions (i, j) and (j, i) and 0 elsewhere. Since CSDP can be reduced to an instance of SDP, it implies that it can also be solved in polynomial time up to a given degree of accuracy.

CHAPTER 4

Fitness Landscapes

Contents

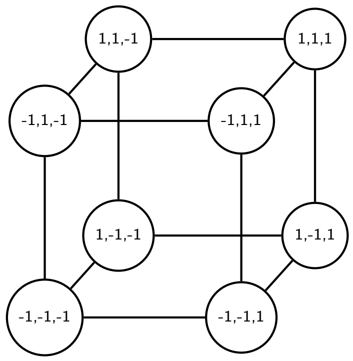


Figure 5: Hamming graph $\mathbb{H}_2^3$ where each node represent a particular spin configuration of 3 binary spin particles.

The fitness landscape of a problem is closely related to its complexity or difficulty, and in order to understand what makes a problem hard to solve, it is convenient to analyze its fitness landscape. We can analyze the fitness landscape of several *discrete optimization* problems, such as genetic algorithms, evolution of DNA, physics of disordered systems and many computational problems such as MAX-CUT or the Traveling Salesman Problem (TSP).

4.1 Definition

Fitness Landscape

A **fitness landscape** of a problem Π consists of two ingredients (Stadler, 1995):

- a finite set V of configurations with a notion of **neighborhood** and
- a cost or **fitness function** $f : V \to \mathbb{R}$.

We can see V as a graph Γ, where each vertex is a solution of problem Π, and the edges are the neighborhood: if two solutions (or nodes of Γ) are joined by an edge, it means that we can go from one solution to the other with a *simple step*. This simple step depends on the problem Π.

For example, if problem Π models the energy of n, 2-spin particles, a configuration or a node of V is the vector Σ containing all the spin values of the n particles, i.e, $\Sigma = (\sigma_1, \ldots, \sigma_n)$ where $\sigma_i = \pm 1$ is the spin value of particle i. Different values of σ_i will give a different Σ vector and therefore it will represent a different node of V. One particular neighborhood for this problem is to put an edge between spin configurations if you can go from one to the other by flipping (changing the sign) only one spin, i.e., Σ_i and Σ_j will differ in only one entry. With this notion of neighborhood, the graph Γ will be a Hamming graph $\mathbb{H}_2^n$ and an example for $n = 3$ can be seen in Figure 5. Finally, the fitness function f for the 2-spin n particles problem is the energy Hamiltonian and the problem consists of finding the configuration Σ^* with lowest energy.

Locus

We will denote a configuration $x \in V$ as $x = (x_1, x_2, \ldots, x_n)$. We will call the entry x_i a **locus**, and it can only take the values $+1$ or -1. Therefore $x \in \{\pm 1\}^n$.

4.2 Properties

The main goal of analyzing the landscape of a problem is to find the "best" possible value of the cost function among all the configurations in V, which can be the lowest state of energy for the spin particles problem, or the highest fitness function of an individual. A configuration $x \in V$ is a **local minimum** if $f(x) \leq f(y) \quad \forall y$ neighbor of x; and $x \in V$ is a **global minimum** if $f(x) \leq f(y) \quad \forall y \in V$.

Local Minimum
Global Minimum

Since many approaches utilize local information to optimize, it is easy for an algorithm to get stuck in a *local optimum*, therefore the global optimum will be hard to compute in a landscape with a large number of local optima. One extreme case is a "convex" landscape, which only has one local optimum which coincides with the global optimum and can be found efficiently. One of the definitions of **Ruggedness** of a landscape is the number of local optima, and it is a crucial property of landscapes as it "measures" the difficulty of a problem, but it is not trivial to calculate (Stadler, 2002).

Ruggedness

Basins of local minima are separated by saddle points and fitness barriers. Let x and y be two local minima and let p be a path in V from x to y. A **fitness barrier** is defined as:

Fitness Barrier

$$f[x,y] = \min\{\max[f(z) \mid z \in p] \mid p \text{ is a path from } x \text{ to } y\} \tag{4.1}$$

and points $z \in V$ that satisfy Equation 4.1 are called **saddle points**. The **barrier** enclosing a local minimum is the height of the lowest saddle point that give access to a more favorable minimum:

Saddle Points

$$B(x) = \min\{f[x,y] - f(x) \mid y \text{ is a local minimum and } f(y) < f(x)\} \tag{4.2}$$

4.3 Random Field Models

For some problems, we cannot hope to describe landscapes from their underlying biological, chemical or geometrical structure, therefore one approach taken is to consider probabilistic models of fitness landscapes.

The set $\{f : V \to \mathbb{R}\}$ together with a measure $\mu\{f\}$ form the probability space Ξ, which we call a **random field** on the graph Γ (Stadler and Happel, 1999). This measure can be seen as the form $P(c_1, c_2, \ldots, c_{|V|})$ which is the probability that for all configurations x_i, it holds simultaneously that $f(x_i) \leq c_i$, where $c_i \in \mathbb{R}$. Since we are working with random models, we need to define the expected value of a random variable X defined on the random field, which is given by

Random Field

$$\mathbb{E}[X] := \int_{\mathbb{R}^{|V|}} X \mathrm{d}P(c_1, \ldots, c_n)$$

Given the Adjacency matrix A, and the diagonal degree matrix D, the Laplacian matrix $-\Delta$ of Γ is

$$-\Delta := D - A$$

which is useful since we can do Fourier expansions of random fields using an orthonormal set of eigenvectors of the graph Laplacian.

Covariance Matrix The **covariance matrix** C of a random field is defined component wise as

$$C_{x,y} = \mathsf{Cov}[f(x), f(y)] = \mathbb{E}[f(x)f(y)] - \mathbb{E}[f(x)]\mathbb{E}[f(y)] \tag{4.3}$$

and is symmetric and non-negative definite. This matrix contains all the structure of the problem, therefore it can be that a relation between the covariance matrix and the number of local optima exists.

4.4 Landscape Models

In this section we define some of the most common landscape models:

- The **House of Cards (HoC)** (Kingman, 1978). One of the simplest models. The fitness value $f(x)$ for each configuration x is assigned independently at random from some probability distribution.

- **Sherrington Kirkpatrick (or 2-spin model)** (Sherrington and Kirkpatrick, 1975). This models a spin glass of n particles with 2 spin states, represented by $\{+1, -1\}$:

$$H_{SK}(x) := \sum_{i<j} J_{ij} x_i x_j \tag{4.4}$$

  where the coupling constants J_{ij} are i.i.d. Gaussian random variables with mean 0 and variance 1.

- **P-spin model** (Amitrano, Peliti, and Saber, 1989). A generalization of the Sherrington Kirkpatrick model in which each spin interacts with another $p-1$ spins, with random energy function (Derrida, 1981; Gross and Mezard, 1984):

$$H_p(x) := -\sum_{1 \le i_1 < \cdots < i_p \le n} J_{i_1 i_2 \ldots i_p} x_{i_1} \ldots x_{i_p} \tag{4.5}$$

  where the coupling constants $J_{i_1 i_2 \ldots i_p}$ are i.i.d. Gaussian random variables with distribution

$$\mathsf{P}(J_{i_1 i_2 \ldots i_p}) = \sqrt{\frac{n^{p-1}}{\pi p!}} \exp\left[-\frac{J^2_{i_1 i_2 \ldots i_p} n^{p-1}}{p!} \right]$$

- **NK Model** (Kauffman and Levin, 1987). In this model, the fitness $f(x)$ of a configuration x is given by

$$f(x) = \sum_{i=1}^{n} f_i(x_{b_{i,1}}, x_{b_{i,2}}, \ldots, x_{b_{i,K}}) \tag{4.6}$$

  where f_i is an independent HoC landscape for the locus i, depending on K loci and the indices $b_{i,j}$ represent which K loci are being considered.

- **Rough Mount Fuji model** (Neidhart, Szendro, and Krug, 2014). It is a simplified version of the model introduced by (Aita and Husimi, 2000) which is designed to have *tunable ruggedness*. It consists of two parts, an additive fitness landscape and an uncorrelated random HoC landscape. Let x and x' be two configurations, then we will denote as $D(x, x') = \sum_i^n (x_i - x'_i)^2$ the distance between the two configurations, which is the number of different loci values between x and x'. Let x^* be a configuration with maximal fitness of the additive part of the landscape. Then, the fitness for a configuration x is defined as $F(x) = -cD(x, x^*) + \eta(x)$, where $c > 0$ and $\eta(x)$ is a HoC model, i.e., the value of $\eta(x)$ is assigned independent and identically at random for all configurations x. If $c = 0$ the model becomes an uncorrelated HoC landscape, and with large values of c the additive part of the landscape dominates. The parameter that controls the ruggedness of the landscape is

$$\theta = \frac{c}{\sqrt{\mathrm{var}(\eta)}}$$

and increasing the value of θ will decrease the ruggedness.

4.4.1 NK Model

Note that the values of all the $\mathbf{b}_i$ were left undefined in Equation 4.6, and exactly these values model the interaction scheme between loci. Some models of interaction shown in Figure 6 (Krug and Schmiegelt, 2013; Nowak and Krug, 2015) are:

- **Adjacent neighborhood**. Each sub-landscape f_i depends on the i-th locus and its $K-1$ following neighbors. That is,

$$\mathbf{b}_i = (x_i, x_{i+1}, \ldots, x_{i+K})$$

  each element modulo n.

- **Random neighborhood**. The neighborhood set b_i contains i and $K-1$ other numbers, which are chosen at random from $\{1, 2, \ldots, n\}$.

- **Block neighborhood**. With n an integer multiple of K, n is divided into n/K disjoint K-subsets and each block effectively behaves as an independent HoC landscape.

4.5 Computing the Number of Local Optima

Since the ruggedness of a landscape give intuition of the difficulty of a problem, it is convenient to try to compute the number of local optima.

In order to find the number of local optima, the probability π_{max} that a randomly chosen configuration x is a maximum needs to be computed. Then, the number of maxima $\#_{max}$ is obtained by multiplying this probability by the total number of configurations:

$$\#_{max} = 2^n \pi_{max}$$

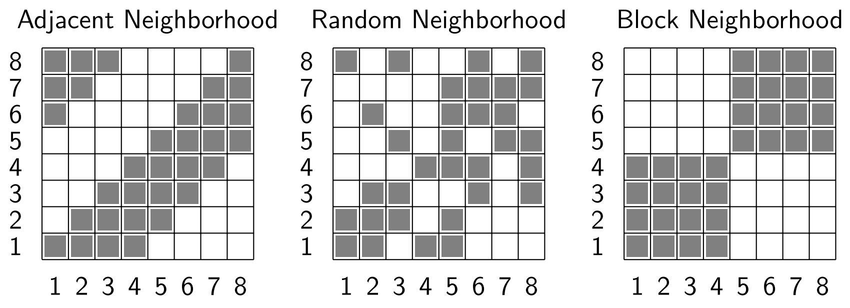


Figure 6: Examples of interactions between loci for the NK model with $n = 8$ and $k = 4$.

4.5.1 House of Cards Model

The easiest example is the House of Cards model. Since $f(x)$ is random for every configuration, π_{max} is just the probability that x is the maximum among $n+1$ random values, which is

$$\pi_{max} = \frac{1}{n+1}$$

However, there is a general formalism for computing π_{max} in the HoC and other models. This subsection follows the procedure of Hwang et al. (2018). Let us define the operator $\Delta_l : \mathbb{H}_2^n \to \mathbb{H}_2^n$ for all $l \in \{1, 2, \ldots, n\}$ as

$$(\Delta_l x)_m = (1 - 2\delta_{lm})x_m$$

which changes the l-th entry of the configuration x. Let h_0 and h_l be the fitness values for x and $\Delta_l x$ respectively, i.e., $h_0 = f(x)$ and $h_l = f(\Delta_l x)$. Then, x is a local maximum if $h_0 > h_l$ or $u_l \equiv h_0 - h_l > 0$ for all $1 \leq l \leq n$. With vector notation, $\mathbf{u} \equiv (u_1, u_2, \ldots, u_n)$, the **joint probability density** of the u_l is given by

$$P(\mathbf{u}) = \int \prod_{l=0}^{n} dh_l p_f(h_l) \prod_{l=1}^{n} \delta(u_l - (h_0 - h_l)),$$

with characteristic function

$$\Phi(\mathbf{q}) = \int dy \; p_f(y) \left(\prod_{l=1}^{n} \phi_f(-q_l) \right) \exp\left(iy \sum_{l=1}^{n} q_l \right)$$

where $\phi_f(-q_l)$ is the individual characteristic function of $p_f(h)$. By performing the inverse Fourier transform of $\Phi(q)$ and then integrating over only positive values of u_l (condition for x to be maximum and represented by $\theta(\mathbf{u} > 0)$), we obtain:

$$\pi_{max} = \prod_{l=1}^{n} \int_0^{\infty} du_l \; P(\mathbf{u}) = \int \frac{DuDq}{(2\pi)^n} e^{-i\mathbf{q}\cdot\mathbf{u}} \theta\left(\mathbf{u} > 0\right) \Phi(\mathbf{q})$$

With these expresions one can recover the number of local optima for the HoC model and more importantly, one can get the number of maximum for some neighborhood models of the NK-model.

4.5.2 NK Model

In the NK model, the total fitness $F(x)$ of a configuration x is a sum of individual HoC fitness values defined on each particular block or neighborhood. Again, following the procedure of Hwang et al. (2018), the approach will be based on the characteristic function of the NK blocks, which has the form:

$$\Phi(\mathbf{q}) = \prod_{r=1}^{N} \Phi_r(\mathbf{q}) \tag{4.7}$$

where $\Phi_r(\mathbf{q})$ denotes the characteristic function of $\mathbf{u}$ of a NK block B_r. With the help of an incidence matrix notation $b_{l,r}$ that indicates the presence (absence) of a locus l in a neighborhood set r, i.e. $b_{l,r} = 1(0)$ if $l \in B_r (l \notin B_r)$; the characteristic function ϕ_r can be rewritten as:

$$\Phi_r(\mathbf{q}) = \int dy_r \; p_f(y_r) \prod_{l=1}^{n} \left[\phi_f(-q_l) e^{iy_r q_l}\right]^{b_{l,r}}$$

Then, after obtaining the characteristic function of Equation 4.7, the probability π_{max} can be obtained by using the inverse Fourier transform the same way as in the HoC model, and for the Adjacent Neighborhood reads as:

$$\pi_{max}^{AN} = \int D\mathbf{y} P(\mathbf{y}) \frac{D\mathbf{u} D\mathbf{q}}{(2\pi)^n} e^{-i\mathbf{q}\cdot\mathbf{u}} \theta(\mathbf{u} > 0) \prod_{l=1}^{n} \phi_f(-q_l)^k e^{iq_l \sum_{r=0}^{k-1} y_{(l+r)\mathtt{mod}(n)}}$$

With this formulation, one could search for a relation between the covariance matrix defined in Equation 4.3 and the number of local optima for simple cases and neighborhoods, for instance, $K = 2$ and the block neighborhood. If this relation exists, the following step would be to find it for a general K. The covariance matrix contains the structure and properties of the landscape, strongly suggesting that this relation should exist. Additionally, see Evans and Steinsaltz (2002) for an asymptotic analysis of the global optima and a representation of the probability that a random point is a local maximum for general K.

4.5.3 Sherrington-Kirkpatrick Model

The Sherrington-Kirkpatrick model is also known as the 2-spin model for spin glasses. For this problem, $\sigma = \{-1, 1\}^n$ will represent a spin configuration, the cost function will be the energy Hamiltonian

$$H(\sigma) := \sum_{1 \le i < j \le n} \sigma_i \sigma_j W_{ij}$$

where W_{ij} are independent standard normal random variables. For the neighborhood E, we will put an edge between two configurations σ and σ' if and only if they differ in only one spin.

Let $\sigma^{(i)}$ denote the operation of flipping the i-th spin of σ, i.e.,

$$\sigma_j^{(i)} := \begin{cases} -\sigma_i, & j = i \\ \sigma_j, & j \neq i \end{cases}$$

Additionally, let Z be a vector in which the i-th entry Z_i has the change in energy obtained by flipping the i-th spin:

$$Z_i(\sigma) := \frac{H(\sigma^{(i)}) - H(\sigma)}{2} = -\sum_{j \neq i} \sigma_i \sigma_j W_{i,j}$$

Z is a multivariate normal vector, with 0 mean and has the property that σ is a local minimum if and only if $Z_i(\sigma) \geq 0 \ \forall i = 1, \dots, n$.

We are interested in calculating the ruggedness of the landscape for this problem, and to do so, we need to compute first the probability that a configuration σ is a local minimum following the procedure of Addario-Berry et al. (2019). This is equivalent to asking the probability that $Z_i \geq 0$ for all $i = 1, \dots, n$, and since Z is a multivariate normal vector, we need to compute the covariance matrix C before. The covariance matrix of Z is:

$$\begin{aligned} & C = \mathbb{E}[ZZ^T] - \mathbb{E}[Z]\mathbb{E}[Z^T] = \mathbb{E}[ZZ^T] \\ \Rightarrow\ & C_{ij} = \mathbb{E}[Z_i Z_j] \end{aligned}$$

For the diagonal entries,

$$C_{ii} = \mathbb{E}[Z_i^2] = \mathbb{E}\left[\left(\sum_{j \neq i}^{n} \sigma_i \sigma_j W_{ij}\right)^2\right],$$

and since the W_{ij} are independent normal variables, the only terms that are not 0 are W_{ij}^2, since $\mathbb{E}[W_{ij}W_{kl}] = \mathbb{E}[W_{ij}]\mathbb{E}[W_{kl}] = 0$; therefore the diagonal entries are $C_{ii} = n - 1$. For the off diagonal entries $i \neq j$,

$$C_{ij} = \mathbb{E}[Z_i Z_j] = \mathbb{E}\left[(\sigma_i\sigma_1 W_{i1} + \dots + \sigma_i \sigma_n W_{in})(\sigma_j \sigma_1 W_{j1} + \dots + \sigma_j \sigma_n W_{jn})\right],$$

and all the terms $\mathbb{E}[W_{ik}W_{jl}] = 0$ unless $k = l$, therefore the only term left is $\mathbb{E}\left[(\sigma_i\sigma_j W_{ij})^2\right] = 1$. Then, the covariance matrix is

$$C_{i,j} = \begin{cases} n - 1, & i = j \\ 1, & i \neq j \end{cases}$$

or in compact form:

$$C = (n-2)\mathbb{I}_n + \vec{1}_n \vec{1}_n^T$$

where $\mathbb{I}_n$ is the $n \times n$ identity matrix and $\vec{1}_n$ is a vector with 1 in each entry.

The eigenvalues of C are easy to compute. Let $v \in \mathbb{R}^n$ an eigenvector of C. Then, v must satisfy:

$$(C - \lambda \mathbb{I}_n)v = 0 \ \Rightarrow \ \left((n-2-\lambda)\mathbb{I}_n + \vec{1}_n \vec{1}_n^T\right) v = 0$$

which can be rewritten as

$$(n-2-\lambda)\mathbb{I}_n v = -\left(\sum_{i=1}^{n} v_i\right) \vec{1}_n$$

which gives as solution the eigenvalues of $\lambda = 2n-2$ with multiplicity 1 and $\lambda = n-2$ with multiplicity $n-1$. The determinant of C is $\det(C) = (2n-2)(n-2)^{n-1}$ and with the Sherman Morrison formula, the inverse of C is

$$C^{-1} = \frac{1}{n-2}\left(\mathbb{I}_n - \frac{1}{2n-2}\vec{1}_n \vec{1}_n^T\right)$$

With the inverse of the covariance matrix, we can finally compute the probability that a configuration σ is a local optimum:

$$\begin{aligned} \mathbb{P}\{\sigma \text{ is locally optimal}\} &= \mathbb{P}\{\cap_{i=1}^n \{Z_i \geq 0\}\} \\ &= \frac{1}{(2\pi)^{n/2}\det(C)^{1/2}} \int_{[0,\infty)^n} \exp\left(\frac{-x^T C^{-1} x}{2}\right) dx \\ &= 2^{-n}\sqrt{\frac{n-2}{2n-2}}\mathbb{E}\left[\exp\left(\frac{\|N\|_1^2}{4(n-1)}\right)\right] \end{aligned}$$

with N a vector of n independent standard normal random variables (see Addario-Berry et al. (2019) for a more detailed derivation). It can be shown that in the limit,

$$\lim_{n\to\infty} \frac{1}{n}\mathbb{P}\{\sigma \text{ is locally optimal}\} = \alpha^* - \log 2$$

where $\alpha \approx 0.199$.

Short Range Interaction

All of the above was assuming **Infinite Range (IR)** interactions. For **Short Range (SR)** interactions the probability distribution of the matrix $W^{(SR)}$ changes:

$$\mathbb{P}\left(W_{i,j}\right)^{(SR)} = \mathbb{P}\left(W_{i,j}\right)^{(IR)} \Theta_{i,j}$$

where

$$\Theta_{i,j} = \begin{cases} 1, & \text{particle } i \text{ interacts with particle } j \\ 0, & \text{otherwise} \end{cases}$$

is the SR interaction pattern. Assuming the SR interaction is that a spin particle interacts with exactly k neighbors, the diagonal entries of the covariance matrix are:

$$C_{ii} = \mathbb{E}[Z_i^2] = \mathbb{E}\left[\left(\sum_{j\neq i}^{n} \sigma_i \sigma_j W_{ij}\right)^2\right] = \sum_{k\neq i} \delta_{ik} = k$$

where δ_{ik} is the Kronecker delta. The off diagonal entries have the form:

$$C_{ij} = \mathbb{E}[Z_i Z_j] = \mathbb{E}\left[(\sigma_i\sigma_1 W_{i1} + \cdots + \sigma_i\sigma_n W_{in})(\sigma_j\sigma_1 W_{j1} + \cdots + \sigma_j\sigma_n W_{jn})\right]$$

and again the only terms $W_{ik}W_{jl}$ that are not 0 are the ones with $k = l$ but in the short range model, $\mathbb{E}\left[(\sigma_i\sigma_j W_{ij})^2\right] = \delta_{ij}$. Therefore, for the SR interaction, the covariance matrix becomes:

$$C_{i,j}^{SR} = \begin{cases} k, & j = i \\ \Theta_{i,j}, & j \in [n]\backslash\{i\} \end{cases}$$

$$C^{SR} = k\mathbb{I}_n + \Theta$$

instead of

$$C_{i,j}^{IR} = \begin{cases} n-1, & j = i \\ 1, & j \in [n]\backslash\{i\} \end{cases}$$

$$C^{IR} = (n-2)\mathbb{I}_n - \vec{1}_n\vec{1}_n^T.$$

The structure of C^{IR} makes calculating the eigenvalues trivial, since:

$$(C - \lambda\mathbb{I}_n)\, v = 0 \quad \implies \quad \left((n-2-\lambda)\mathbb{I}_n + \vec{1}_n\vec{1}_n^T\right) v = 0$$

$$\implies (n-2-\lambda)v = -\vec{1}_n\vec{1}_n^T v = \left(\sum_{i=1}^{n} v_i\right)\vec{1}_n$$

which defines an easy solvable set of n equations with n unknowns that gives the eigenvalue $\lambda = 2n-2$ with multiplicity 1. And for $\lambda = n-2$, we have 1 equation with n unknowns, giving us the second eigenvalue with multiplicity $n-1$.

Trying this for the short range interactions, the eigenvalue problem translates to:

$$(k-\lambda)v = -\Theta v$$

Assume that, for instance, particle i interacts with its following k particles. With this particular interaction, the system becomes:

$$(k-\lambda)\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = -\begin{pmatrix} 0 & 1 & 1 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & \dots & 0 & 0 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$$

and adding up all the equations:

$$(k-\lambda)\sum_{i=1}^{n} v_i = -k\sum_{i=1}^{n} v_i$$

implies that $\lambda = 2k$ with multiplicity 1. This was the same procedure than the IR model for this eigenvalue, however one cannot do the same for the other eigenvalues. If $\lambda = k$, making the left size of the equation 0 in the same way as for the IR model, then the system of equations becomes

$$\begin{cases} 0 = -x_2 - \cdots - x_{k+1} \\ 0 = -x_3 - \cdots - x_{k+2} \\ \quad\vdots \\ 0 = -x_1 - \cdots - x_k \end{cases}$$

which does not help for finding the eigenvalues as clearly as the Infinite Range case, because in the IR all the equations turn into $0 = \sum_{i=1}^{n} v_i$.

With the eigenvalues of the covariance matrix, the same procedure as the IR model can be done in order to find a value of the probability that a configuration σ is a local optima.

4.5.4 p-Spin Model

The p-spin model was introduced by Amitrano, Peliti, and Saber (1989) as an alternative for the NK model with more application to physics. Calculating the number of local optima of the model in Equation 4.5 is equivalent of solving the TAP equations of Thouless, Anderson, and Palmer, 1977, which have been solved for $p = 2$ (Bray and Moore, 1980) and general p (Rieger, 1992). In V. M. d. Oliveira, Fontanari, and Stadler (1999) they calculate directly the expected value of number of local optima for sohrt range interactions. In V. M. d. Oliveira and Fontanari (1997), they find the number of local optima of the p-spin model with an external magnetic field, i.e., Equation 4.5 plus an additional term:

$$H_p(x) := -\sum_{1\le i_1<\cdots<i_p\le n} J_{i_1 i_2 \ldots i_p} x_{i_1} \ldots x_{i_p} - h\sum_{i}^{n} x_i \tag{4.8}$$

corresponding to the external magnetic field h.

CHAPTER 5

Clustering

Contents

Clustering is a technique with the task to organize a collection of objects (usually points in a vector space) into groups using either a *simmilarity* or a *difference* measure between them. The idea is that objects inside groups or **clusters** are similar between them but different from objects in other groups. An example of clustering can be seen in Figure 7, in which points with the same color represent a group.

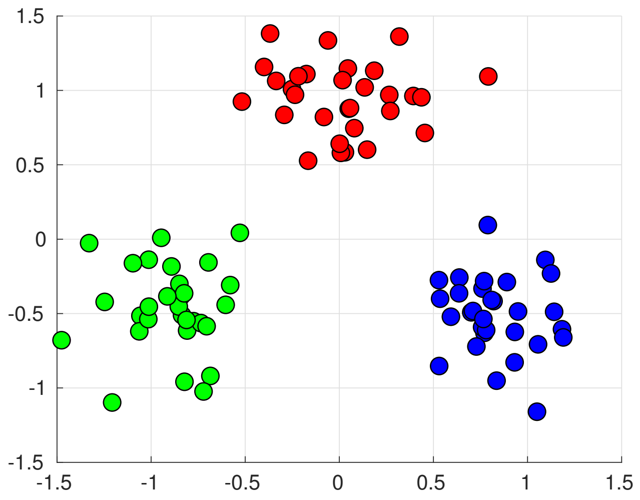


Figure 7: An example of clustering. The set of points in the figure are grouped in 3 clusters, the points in red, in green and in blue.

Different clustering methods can be obtained by using different similarity or distance measures, by changing the criteria to group objects or even by using transformations of the data points. However, we can define two main types of clustering:

- *Hard clustering*. Hard clustering is the natural idea of grouping objects into sets, and it assigns each data point to a single cluster. It can also be thought as assigning a point to a cluster with probability of 1.

- *Soft or Fuzzy clustering*. Contrary to hard clustering, in this type of clustering each point has a *membership degree* value between 0 and 1 for each cluster. The values 0 and 1 correspond to the hard version of clustering, where a value of 1 means that a point belongs entirely to that cluster and a value of 0 that it does not. In the case of soft clustering, higher membership degree means a higher belonging to that cluster and vice versa.

Three hard clustering methods (k-means, k-medoids and Minimum Spanning Tree Clustering) and one soft clustering method (Fuzzy C-means) are discussed in this chapter.

Algorithm 1: k-means Algorithm

```
Require: Set of objects $\mathbf{X}$, number of clusters $k$.
Ensure: Set of centroids $\mathbf{c}_j$
1: Select $k$ initial centers $\mathbf{c}_j \in \mathbb{R}^n \quad \forall j \in 1, 2, \ldots, k$
2: Create $k$ groups $C_j \quad \forall j \in 1, 2, \ldots, k$
3: while The set of centers $\mathbf{c}_j$ changes each iteration do
4:    Find the distance matrix between the points and the centers: $d_{ij} =$ distance$(\mathbf{x}_i, \mathbf{c}_j)$
5:    Assign each object $\mathbf{x}_i \in X$ to the group $C_j$ with the closest center $\mathbf{c}_j$.
          $C_j \leftarrow C_j \cup \mathbf{x}_i : \text{argmin}_j \text{d}(\mathbf{x}_i, \mathbf{c}_j)$
6:    Recalculate the new positions of the centers of the groups:
          $\mathbf{c}_j \leftarrow \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$
7: end while
```

5.1 k-Means

The algorithm for k-means was proposed by MacQueen (1967), and it can be seen as an optimization problem where the goal is to find k clusters such that the quadratic error

$$\sum_{i=1}^{\infty} (\text{min}_j |\mathbf{x}_i - \mathbf{c}_j|^2) \tag{5.1}$$

is minimized, where $\mathbf{c}_j$ is the centroid of cluster j. For k-means, a centroid of a cluster is the *arithmetic mean* of all the points that belong to it.

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ be the points to group, assume these points $\mathbf{x}_i \in \mathbb{R}^N$ for $i = 1, 2, \ldots, m$ and let $k \geq 2$ be the number of clusters. Then Algorithm 1 selects as starting centroids k points in $\mathbb{R}^N$ denoted by $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$. An efficient method to initialize these centers will be discussed in Subsection 5.1.1. After the initial selection of centroids, each point $\mathbf{x}_i$ is assigned to the cluster that has the nearest centroid to it. After all the points are assigned, each cluster centroid is recalculated as the mean of all points in that cluster, hence the name k-means. Lastly, once all the centroids have been updated, the points $\mathbf{X}$ are reassigned, followed by recalculating the centroids and so on until a certain stopping condition is met. The process of k-means is shown in Algorithm 1, where the algorithm stops when it reaches a local optima, i.e., when the centers do not change between iterations.

The temporal complexity of the k-means algorithm is $O(mkNT)$, where T denotes the number of iterations. In general, since both the dimension N and the number of clusters k are smaller than the number of points m, it can be considered that each iteration of the k-means algorithm has lineal complexity on m. Among the limitations of k-means are that it can converge into a local optimum, it can only group *linearly* separable data points and since it uses an arithmetic mean as centroids, it cannot work with attributes such as labels, colors or shapes.

5.1.1 k-Means Initialization Algorithms

In this work, we consider two algorithms to initialize k-means. The first one, simply chooses uniformly and at random k points among the set of points $\mathbf{X}$ to be the initial centers.

The second one is a deterministic initialization that makes k-means a a 2-approximation algorithm on clustering where the goal is to minimize the diameter of each cluster (Dasgupta, Papadimitriou, and Vazirani, 2006) and it is shown in Algorithm 2. First the distance matrix is calculated and the two points that have the maximum distance are chosen as initial centers. Then, the next point chosen is the one that has the largest distance to the previously selected centers and this is repeated until k centers are found.

Two measures of distances between a point p to a set V were used in order to avoid poor choices of cluster centers for cases where there are ties. The first distance, used in step 5 of Algorithm 2 is the average distance of p to all the points in V, i.e.,

$$\text{distance}(p, V) = \frac{1}{\text{size}(V)} \sum_{v \in V} |p - v|$$

and in step 6, Algorithm 2 selects the point p that has the largest distance to the set V. In case of a tie, the second distance function is considered to break it. This second distance between a point p to a set V is:

$$\text{distance}(p, V) = \text{min}_{v \in V} \;\; |p, v|$$

which is the smallest distance between the point p to all the points in V. This tie breaker is considered because it is a better option than breaking it at random. For the instance shown on Figure 8, if a random tie breaker is used, poor initial choices of cluster centers can be chosen by the algorithm which lead to suboptimal clusters.

5.2 k-Medoids

The k-medoids algorithm was proposed by Kaufmann and Rousseeuw (1987) and it utilizes as the centroid of a cluster the most representative object in the group instead of using the arithmetic mean, therefore it does not require that the data points belong to $\mathbb{R}^N$, since the centroids will always be one of the points of $\mathbf{X}$.

Medoid Formally, a **medoid** of a cluster C_j is the object $\mathbf{x} \in C_j$ that has minimum average distance (or maximum average similarity) to the other objects. It can be

Algorithm 2: Algorithm to Initialize k-means

```
Require: Number of clusters $k$, data matrix $\mathbf{X}$ of $n$ times $d$ containing $n$ points $x_i$ (row vectors) of dimension $d$.
Ensure: Centroids Matrix $CM$ of $k$ times $d$.
1: Initialize SizeOfCM $\leftarrow 0$
2: Find the distance matrix $d_{ij} \leftarrow |\mathbf{x}_i - \mathbf{x}_j|$
3: Add to $CM$ the two points that have the largest $d_{ij} \quad \forall i = 1, \ldots, n \quad j = 1, \ldots, n$ and set SizeOfCM $\leftarrow 2$
4: while SizeOfCM $< k$ do
5:   Find distance vector of all points $\mathbf{x}_i$ to the set of selected clusters $CM$:
         $$\text{VecOfDistances}(i) \leftarrow \frac{1}{\text{SizeOfCM}} \sum_{v \in CM} |v - \mathbf{x}_i| \quad \forall i = 1, \ldots, n$$
6:   Find $\mathbf{x}_l$ such that max_val = max(VecOfDistances) = VecOfDistances($l$)
7:   if VecOfDistances has unique maximum then
8:     Add $\mathbf{x}_l$ to $CM$
9:     SizeOfCM $\leftarrow$ SizeOfCM $+1$
10:  else
11:    VecOfDistances($i$) $\leftarrow \min_{v \in CM} \; |\mathbf{x}_i - v|$
12:    Find $\mathbf{x}_l$ such that max_val = max(VecOfDistances) = VecOfDistances($l$)
13:    Add $\mathbf{x}_l$ to $CM$
14:    SizeOfCM $\leftarrow$ SizeOfCM $+1$
15:  end if
16: end while
```

seen as the most representative or central object of the group (Struyf, Hubert, and Rousseeuw, 1996). A medoid is illustrated in Figure 9.

Since a medoid is always a point of $\mathbf{X}$, other distance functions can be chosen and the objects can belong to spaces with non-numerical attributes, such as labels or letters. The procedure of k-medoids is shown in Algorithm 3.

5.3 Minimum Spanning Tree (MST) Clustering

Minimum Spanning Tree (MST) clustering utilizes a *distance graph* generated from the distance (or similarity) between the data points of $\mathbf{X}$.

Definition 1. *A **distance graph** of a set of objects $\mathbf{X}$ is created as follows:* ***Distance Graph***

- *Each object becomes a node.*
- *Create an edge between node $\mathbf{x}_i$ and node $\mathbf{x}_j$ with weight:*

$$w_{ij} = \text{distance}(\mathbf{x}_i, \mathbf{x}_j) \quad \forall i \neq j.$$

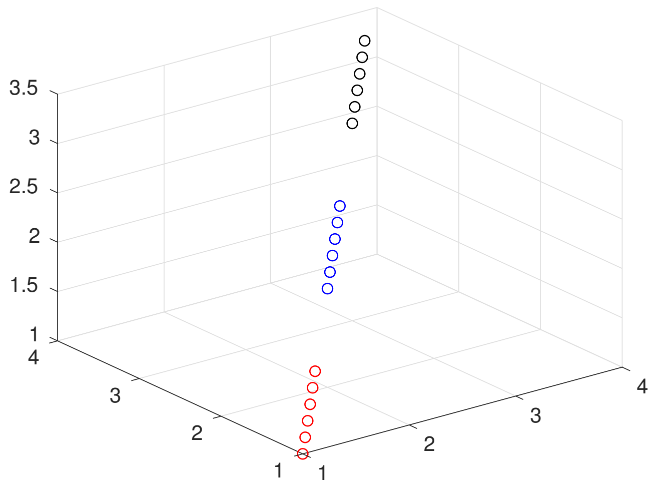


Figure 8: Troublesome instance for Algorithm 2. The first two centers chosen will be the far below red circle and the far above black circle. Now for the third choice of a center, without the tiebreaker there is a tie between *all* the remaining points, but with the tiebreaker distance, the third center chosen is one of the middle blue points.

Note that in Definition 1 a distance is needed, which will be assumed to be the Euclidean norm unless it is otherwise specified. If a similarity measure is used instead of a distance, then it is called a *similarity graph*. After this graph is generated, a MST is computed, and using the fact that removing $k-1$ edges of a tree creates k disconnected components, we remove the $k-1$ edges with higher weight to generate k clusters. For the case of a similarity graph, the algorithm removes the $k-1$ edges with smallest weight. MST clustering is shown in Algorithm 4.

5.4 Fuzzy C-means

Fuzzy C-means was proposed by Bezdek (1981) in order to take into account outliers or noisy data, and contrary of hard clustering in which each data point is assigned strictly to one cluster, in fuzzy C-means points are assigned to several clusters with a certain membership degree using fuzzy logic and fuzzy sets. This allow the clusters to overlap themselves instead of considering "defined borders" such as in k-means, providing more information about the data. Additionally, fuzzy C-means have a weighting exponent that controls the level of cluster fuzziness. The objective of Fuzzy C-means as an optimization problem is to minimize the generalized least-squared errors function:

$$J_f(U,v)=\sum_{k=1}^{m}\sum_{i=1}^{c}(u_{ik})^f\|\mathbf{x}_k-\mathbf{v}_i\|_A^2 \tag{5.2}$$

where $\mathbf{X}=\mathbf{x}_1,\ldots,\mathbf{x}_m\subset\mathbb{R}^n$ is the data, c is the number of clusters, $f\geq 1$ is a weighting exponent or **fuzzifier**, $\mathbf{v}_i=(v_{i1},\ldots,v_{in})$ is the center of cluster i,

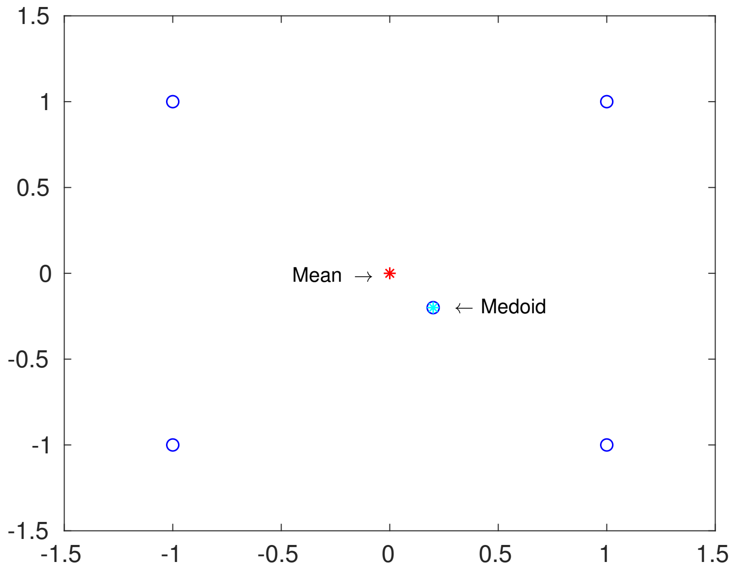


Figure 9: Mean and Medoid examples. In blue circles are the objects to cluster ($\mathbf{X}$). A red star (*) represents the mean of the blue points and a cyan star (*) represent the medoid of the points. Note that in this example the mean is not a point of $\mathbf{X}$, but the medoid always is.

$\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_c]$ is a matrix with the vector of centers $\mathbf{v}_i$ as its columns, $\| \cdot \|_A$ is the induced A-norm on $\mathbb{R}^n$, A is a positive definite $(n \times n)$ weight matrix and $U = \{u_{ik} \in [0,1], k = 1, \dots, m; i = 1, \dots, c\}$ is the fuzzy c-partition of $\mathbf{X}$; u_{ki} is the membership degree of point $\mathbf{x}_i$ to the kth cluster, and U must satisfy:

$$\sum_{k=1}^{c} u_{ki} = 1 \quad \forall i = 1, \dots, m$$

The fuzzifier f controls the level of cluster fuzziness; large values of f results in smaller membership degree values u_{ki}, i.e., fuzzier clusters, and in the limit where $f = 1$ the membership degrees converge to either 0 or 1, resulting in a hard clustering.

Local optimality of J_f in 5.2 can be reached only if

$$\mathbf{v}_i = \frac{\sum_{k=1}^{m} u_{ik}^f \mathbf{x}_k}{\sum_{k=1}^{m} u_{ik}^f} \qquad 1 \le i \le c \tag{5.3}$$

$$u_{ik} = \left(\sum_{j=1}^{c} \left(\frac{d_{ik}}{d_{jk}} \right)^{2/(f-1)} \right)^{-1} \qquad 1 \le k \le m, \;\; 1 \le i \le c \tag{5.4}$$

where $d_{ik} = \|\mathbf{x}_k - \mathbf{v}_i\|_A$ (Bezdek, 1981). Conditions of equations 5.3 and 5.4 are necessary, but not sufficient. However these equations give us an iterative method

Algorithm 3: k-medoids Algorithm

```
Require: Set of objects $\mathbf{X}$, number of clusters $k$.
Ensure: Set of medoids $\mathbf{c}_j$
1: Select $k$ initial centers $\mathbf{c}_j \in \mathbb{R}^n \quad \forall j \in 1, 2, \ldots, k$
2: Create $k$ groups $C_j \quad \forall j \in 1, 2, \ldots, k$
3: while The set of medoids $\mathbf{c}_j$ changes each iteration do
4:     Find the distance matrix between the points and the medoids: $d_{ij} = \text{distance}(\mathbf{x}_i, \mathbf{c}_j)$
5:     Assign each object $\mathbf{x}_i \in X$ to the group $C_j$ with the closest medoid $\mathbf{c}_j$.
           $C_j \leftarrow C_j \cup \mathbf{x}_i : \text{argmin}_j \text{d}(\mathbf{x}_i, \mathbf{c}_j)$
6:     Recalculate the new positions of the medoids of the groups:
           $\mathbf{c}_j \leftarrow \mathbf{x} : \text{argmax}_{\mathbf{x} \in C_j} \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \text{d}(\mathbf{x}, \mathbf{x_i})$
7: end while
```

Algorithm 4: Minimum Spanning Tree Clustering Algorithm

```
Require: Set of objects $\mathbf{X}$, number of clusters $k$.
Ensure: Partition of $\mathbf{X}$
1: Compute the distance graph $(DG)$ as shown in Definition 1.
2: Find a MST of the distance graph $DG$.
3: Remove the $k - 1$ edges with the highest weight of the MST.
4: Find the $k$ connected components. Each set of connected components becomes the partition of $\mathbf{X}$.
```

for optimizing J_m, that is, the fuzzy C-means algorithm: first, the values of U are initialized at random, then the values of v are computed using 5.3, and the values of U are corrected using 5.4 with the new values of V. This process is repeated until some stopping criteria is reached. The fuzzy C-means algorithm is shown in Algorithm 5.

Fuzzy C-means converges rapidly with a temporal complexity of $O(mNc^2T)$ (Ghosh and Dubey, 2013), where m is the number of data points with dimension N, c is the number of clusters and T the number of operations. However the algorithm strongly depends on the initial values and can return a local optimum clustering instead of the global optimum (Le and Altman, 2011).

Algorithm 5: Fuzzy C-Means Algorithm

```
Require: Number of clusters $c$, data matrix $\mathbf{X}$ of $n \times m$, where $N$ is the dimension
    of one object and $m$ is the number of objects.
Ensure: Centroid matrix $V$ of $n \times c$.
1: Initialize the center matrix $V$ at random, set the value of $f$ and initialize $U_{old}$
   with ones and a tolerance factor $tol$.
2: while $error > tol$ do
3:     Compute the distance matrix $D$ of $c \times m$, where $d_{ik}^2 = \|\mathbf{x}_k - \mathbf{v}_i\|^2$; $1 \leq i \leq c$
       and $1 \leq k \leq m$, $\mathbf{x}_k$ is the $k$th column of $\mathbf{X}$ and $\mathbf{v}_i$ is the $i$th column of $V$.
4:     Update $U$, where each component is given by:
       $$u_{ik} = \left( \sum_{j=1}^{c} \left( \frac{d_{ik}}{d_{jk}} \right)^{\frac{2}{f-1}} \right)^{-1} ; 1 \leq k \leq m ; 1 \leq i \leq c$$
5:     Update the centroids
       $$\mathbf{v}_i = \frac{\sum_{k=1}^{m} (u_{ik})^f \mathbf{x}_k}{\sum_{k=1}^{m} (u_{ik})^f} ; 1 \leq i \leq c$$
6:     $error \leftarrow \|U - U_{old}\|$
```

5.5 Spectral Clustering (SC)

There are some cases where the data points are not linearly separable, think for example of 2-dimensional points along two concentric circles with different radius. For cases like these it is more useful to group them based on another criteria or strategy than k-means for instance. Let $X = \{x_1, \ldots, x_n\}$ be the set of n points to cluster and let A be the similarity matrix of those points, i.e., A_{ij} is the similarity between point x_i and x_j. In Spectral Clustering (SC), the data points can be seen as nodes of a similarity graph where the edge weights between nodes i and j (representing x_i and x_j respectively) are the similarity values A_{ij} (Luxburg, 2007). Then, SC groups the data points using the eigenvectors of the Laplacian matrix of this similarity graph. A general version of SC can be seen in Algorithm 6.

There are different versions of SC depending on which Laplacian or similarity matrix is used. The algorithm used in this thesis is the SC of Ng, Jordan, and Weiss (2002) with a gaussian and a dotproduct similarity, shown in Algorithm 8.

Algorithm 6: General Spectral Clustering (SC) algorithm.

```
1: **Input:** Set of points $X$, number of clusters $k$
2: **Output:** $X_c$, a k partition of X
3: **Similarity Matrix.** Compute the similarity matrix if it is not given as data.
   Examples of similarities are Gaussian or dot product (Subsection 6.3.3).
4: Construct a similarity graph and let $W$ be its adjacency weighted matrix.
5: Compute the first $k$ eigenvectors $u_1, \dots, u_k$ corresponding to the largest $k$
   eigenvalues of $L$, the laplacian matrix of the similarity graph. Let $U \in \mathbb{R}^{n \times k}$
   be a matrix containing the vectors $u_i$ as columns.
6: Let $z_i \in \mathbb{R}^k$ denote the $i$-th row vector of $U$. Using the row vectors $z_i$, compute
   a partition $X_c$ of the data set $X$;
7: **return** $X_c$
```

Part II

Methods and Findings

CHAPTER 6

Max-Cut

Contents

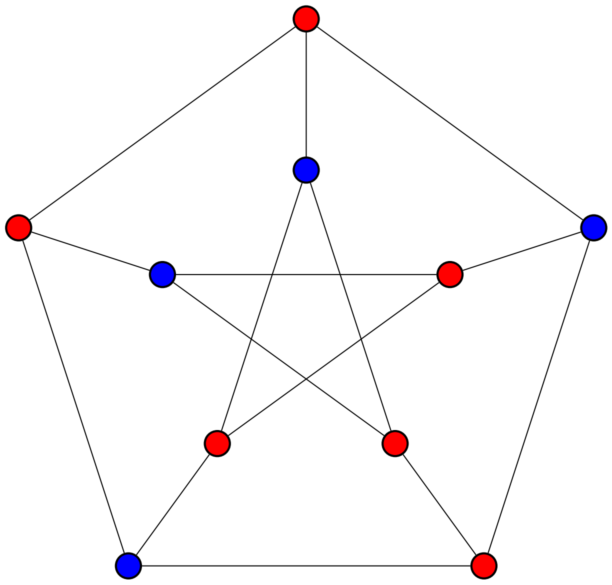


Figure 10: The Petersen graph together with its maximum cut. Nodes in red belong to partition A; and nodes in blue to partition B. The value of the maximum cut for this graph is 12.

The Max-Cut problem is one of the 21 NP-complete problems of Karp (1972). Apart from its theoretical importance, it has many applications in VLSI design (Barahona et al., 1988), network design (Deza and Laurent, 1994a,b), and in statistical physics. Additionally, finding the ground state of the Ising model of a spin glass, the integral error of linearity of an Digital-Analog convertor, and identifying vowels and consonants in cryptograms can be formulated as instances of the Max-Cut problem (Poljak and Tuza, 1993).

6.1 Definition

Let $G = (V, E)$ be an undirected weighted graph with node set $V = \{1, 2, \dots, n\}$ and edge set E where edges $(i, j) \in E$ have weight w_{ij}. A cut of graph G is a bipartition A, B of the node set V, and its associated **cut-value** is the sum of weights of edges with end nodes on different partitions, i.e., the sum of weights of edges (i, j) such that $i \in A$ and $j \in B$ or vice versa, $j \in A$ and $i \in B$. The **Max-Cut** of a graph is the cut or bipartition with maximum cut-value. An example of the maximum cut for the Petersen graph is depicted in Figure 10.

Cut-Value

Max-Cut

Using the fact that node i has only two options ($i \in A$ or $i \in B$), we can represent each node with a binary integer variable $x_i \in \{+1, -1\}$ such that $x_i = 1$ means $i \in A$ and $x_i = -1$ means $i \in B$. We can formulate the Max-Cut problem as the following integer quadratic optimization problem:

$$\begin{aligned} \text{Maximize} \quad & \frac{1}{2}\sum_{i<j} w_{ij}(1 - x_i x_j) \\ \text{subject to:} \quad & x_i \in \{+1, -1\} \qquad \forall i = 1, \dots, n \end{aligned} \tag{6.1}$$

The Max-Cut problem is an NP-Hard combinatorial optimization problem (Karp, 1972), therefore we can not hope to solve problem (6.1) in polynomial time unless $P = NP$. However, Goemans and Williamson (1995) gave an approximation algorithm that returns a solution that is guaranteed to be in the interval $[0.87\text{OPT}, \text{OPT}]$, where OPT is the maximum cut-value.

6.2 Relaxation

In general it is not directly possible to solve problem (6.1), instead, we need to transform it into a larger problem that we can solve which contains all of valid instances from the original problem. The main issue is working with integer variables, therefore the natural relaxation of 6.1 is to replace $x_i \in \{+1, -1\}$ with unitary vectors $v_i \in \mathbb{R}^n$. The Vector Programming (VP) relaxation obtained is:

$$\begin{aligned} \text{Maximize} \quad & \frac{1}{2}\sum_{i<j} w_{ij}(1 - v_i \cdot v_j) \\ \text{subject to:} \quad & |v_i| = 1 \qquad \forall i = 1, \dots, n \end{aligned} \tag{6.2}$$

Note that problem (6.2) reduces to (6.1) if we define the vectors $v_i = (x_i, 0, \dots, 0)$ for all $i \in V$. This implies that OPT$_{rel} \geq$ OPT, where OPT$_{rel}$ is the solution of problem (6.2) and OPT is the maximum cut-value.

VP and Semidefinite Programming (SDP) are equivalent (Williamson and Shmoys, 2011), but numerically it is better to work with a SDP problem. Let $X \in \mathbb{R}^{n \times n}$ be a symmetric matrix. We can convert problem (6.2) to the following SDP problem:

$$\begin{aligned} \text{Maximize} \quad & \frac{1}{2}\sum_{i<j} w_{ij}(1 - X_{ij}) \\ \text{subject to:} \quad & X_{ii} = 1 \qquad \forall i = 1, \dots, n \\ & X \succeq 0 \end{aligned} \tag{6.3}$$

As mentioned before, both problems are equivalent. In order to go from an instance of (6.2) to (6.3), let X be the **Gram Matrix** of the v_i vectors, i.e., each entry X_{ij} contains the dot product $X_{ij} = v_i \cdot v_j$. Vice versa, to transform an instance of (6.3) to (6.2), we obtain the Cholesky factorization of $X = V^T V$ with $V \in \mathbb{R}^{n \times n}$, and let v_i be the i-th column of matrix V. The Cholesky factorization of a SDP matrix always exists (Golub and Van Loan, 1996), and can be found in polynomial time (Watkins, 1991).

Gram Matrix

Algorithm 7: Randomized Rounding of Goemans and Williamson (1995)

```
1: Solve (6.2), obtaining an optimal set of vectors $v_i$
2: Pick a random vector $r = (r_1, \dots, r_n)$ by drawing each component from
   $N(0,1)$, the normal distribution with mean 0 and variance 1
3: Set $A = \{i | v_i \cdot r \geq 0\}$
```

6.3 Rounding Methods

In general, solutions of relaxed problems are not feasible in the original problem. For the particular case of Max-Cut, once a solution of problem (6.2) is found (typically by solving (6.3) and doing a Cholesky factorization), we need to convert the set of vectors v_i into the binary integer variables x_i. Here we present two methods, the randomized rounding of Goemans and Williamson (1995) and using clustering as in Rodriguez-Fernandez et al. (2020).

6.3.1 Randomized Rounding

Using a simple randomized rounding of the vectors v_i leads to the best approximation algorithm known for the Max-Cut problem. Additionally, if the unique games conjecture is true, no better approximation guarantee can be found (Khot et al., 2007). The rounding procedure of Goemans and Williamson (1995) is presented in Algorithm 7.

This rounding method is a 0.878 approximation algorithm and a sketch of the proof will be discussed in the next subsection. This proof is the main motivation for using clustering instead of randomized rounding. More in depth details of the proof can be seen in Goemans and Williamson (1995) and Williamson and Shmoys (2011).

Proof of 0.878 Approximation

Let W be the value of the cut obtained by algorithm 7. Then, the expected value $\mathbb{E}[W]$

$$\begin{aligned}
\mathbb{E}[W] &= \frac{1}{4}\sum_{i,j}^{n} w_{ij}\mathbb{E}\left[1 - \mathsf{sgn}\left(v_i \cdot r\right)\mathsf{sgn}\left(v_j \cdot r\right)\right] \\
&= \frac{1}{4}\sum_{i,j}^{n} w_{ij}\mathsf{Pr}\left(\{1 - \mathsf{sgn}\left(v_i \cdot r\right)\mathsf{sgn}\left(v_j \cdot r\right) = 2\}\right) \\
&= \frac{1}{2}\sum_{i,j}^{n} w_{ij}\mathsf{Pr}\left(\{\mathsf{sgn}(v_i \cdot r) \neq \mathsf{sgn}(v_j \cdot r)\}\right) \\
&= \frac{1}{2}\sum_{i,j}^{n} w_{ij}\frac{\arccos(v_i \cdot v_j)}{\pi}
\end{aligned}$$

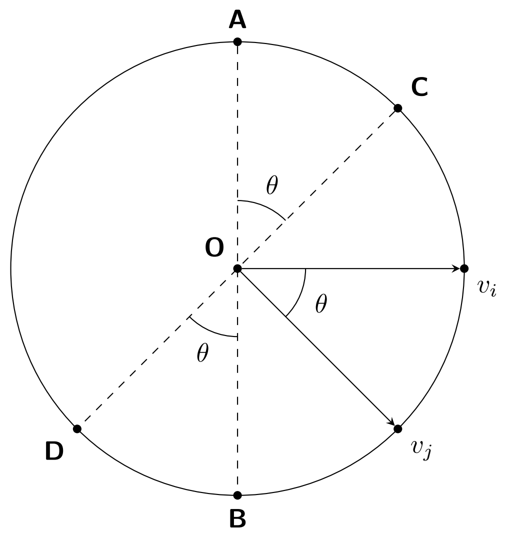


Figure 11: Cut of the hypersphere S^{n-1} by the hyperplane defined by v_i and v_j. θ is the angle between v_i and v_j, the line **AB** is the perpendicular line segment to v_i and line **CD** is the perpendicular line segment to v_j.

Where sgn(x) is 1 if $x \geq 0$ and -1 if $x < 0$, the second line follows from the definition of the expected value for discrete variables, and the last line will be explained in detail in lemma 1.

Lemma 1.

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

Proof. Let $r \in \mathbb{R}^n$ be a random vector in the unitary hypersphere S^{n-1}. Let $r = r' + r''$ where r' is the projection of r in plane defined by v_i and v_j and r'' is the perpendicular component of r to the same plane. It is important to note that r' is uniformly distributed on the unit circle where v_i and v_j are.

Now, $v_i \cdot r = v_i \cdot r'$ and also $v_j \cdot r = v_j \cdot r'$. Let θ be the angle between vectors v_i and v_j. The plane defined by v_i and v_j is illustrated in figure 11. The only cases where r' can be such that sgn$(v_i \cdot r) \neq$ sgn$(v_j \cdot r)$ are in the segments **AOC** and **DOB**.

Since r' is uniformly distributed in the unit circle, the probability that it lands in either **AOC** or **DOB** is $\frac{2\theta}{2\pi}$, since each segment represents an area of $\frac{\theta}{2\pi}$ from the whole unit circle. Finally, since the vectors are unitary, $\theta = \arccos(v_i \cdot v_j)$. $\square$

We need one more result before we can prove the approximation guarantee, which can be found in lemma 2.

Lemma 2. Let $\alpha = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1-\cos(\theta)} > 0.878$. Then $\frac{1}{\pi}\arccos(y) \geq \alpha \cdot \frac{1}{2}(1-y)$ for $y \in [-1, 1]$.

Proof. Using the change of variables $y = \cos\theta$, with $\theta \in [0, \pi]$ we have that the left side of the inequality is $\frac{1}{\pi}\theta$ and the right side becomes $\alpha \cdot \frac{1}{2}(1 - \cos\theta)$. For the case that $\theta = 0$, both sides are 0 so the equality is satisfied. For the case where $\theta \neq 0$, using the definition of α, we have that, for $\theta \in (0, \pi]$:

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

which is the result stated in the lemma. □

Summarizing the results from above, we have that the expected value of the cut found by algorithm 7 is:

$$\mathbb{E}[W] = \frac{1}{2}\sum_{i,j}^{n} w_{ij}\frac{\arccos(v_i \cdot v_j)}{\pi} \geq \alpha \cdot \frac{1}{4}\sum_{i,j}^{n} w_{ij}(1 - v_i \cdot v_j) = \alpha \cdot Z_{VP} \geq \alpha \cdot \text{OPT}$$

where Z_{VP} is the optimum value of the optimization problem (6.2) and OPT is the the value for the max cut, solution of the optimization problem (6.1). Note that the first inequality is only valid for **positive weights**. The last inequality follows from the fact that since problem (6.2) is a relaxation of the original problem (6.1), the solution $Z_{VP} \geq$ OPT, and therefore showing that algorithm 7 is a 0.878 approximation algorithm.

Proof for Negative Weights

As stated in the last paragraph of the last section, an alternative proof for negative weights has to be done. The trick is to add up all the negative weights in absolute value to the expected cut value and to the optimum, allowing us to compare the solutions. Before going to the proof with negative weights, we need to prove an auxiliary lemma similar to lemma 2.

Lemma 3. Let $\alpha = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi}\frac{\theta}{1 - \cos(\theta)} > 0.878$. Then $\left[1 - \frac{1}{\pi}\arccos(y)\right] \geq \alpha \cdot \frac{1}{2}(1 + y)$ for $y \in [-1, 1]$.

Proof. Using the change of variables $\theta = \arccos(-y)$, and the property that $\pi - \arccos(y) = \arccos(-y)$ and the definition of α, we have that for $\theta \neq 0$ (the case when $\theta = 0$ can be proved the same way as in lemma 2):

$$\alpha \cdot \frac{1}{2}(1 - \cos\theta) \leq \frac{\theta}{\pi} \implies \alpha \cdot \frac{1}{2}(1 + y) \leq \frac{\arccos(-y)}{\pi}$$

$$\implies \alpha \cdot \frac{1}{2}(1 + y) \leq \frac{\pi - \arccos(y)}{\pi} = 1 - \frac{1}{\pi}\arccos(y)$$

which is the result stated in the lemma. □

Finally, let us define the sum of all the negative weights $W_- = \sum_{i<j} w_{ij}^-$, where $x^- = min(0, x)$. Then the approximation factor α is defined in the following sense:

Theorem 1. $\mathbb{E}[W] - W_- = \alpha(\text{OPT} - W_-)$

Proof. First let us work with the quantities $\mathbb{E}[W] - W_-$ and VP $- W_-$.

$$\begin{aligned}
\mathbb{E}[W] - W_- &= \sum_{\substack{i<j \\ w_{ij}>0}} w_{ij} \frac{\arccos(v_i \cdot v_j)}{\pi} + \sum_{\substack{i<j \\ w_{ij}<0}} w_{ij} \frac{\arccos(v_i \cdot v_j)}{\pi} + \sum_{\substack{i<j \\ w_{ij}<0}} |w_{ij}| \\
&= \sum_{\substack{i<j \\ w_{ij}>0}} w_{ij} \frac{\arccos(v_i \cdot v_j)}{\pi} - \sum_{\substack{i<j \\ w_{ij}<0}} |w_{ij}| \frac{\arccos(v_i \cdot v_j)}{\pi} + \sum_{\substack{i<j \\ w_{ij}<0}} |w_{ij}| \\
&= \sum_{\substack{i<j \\ w_{ij}>0}} w_{ij} \frac{\arccos(v_i \cdot v_j)}{\pi} + \sum_{\substack{i<j \\ w_{ij}<0}} |w_{ij}| \left(1 - \frac{\arccos(v_i \cdot v_j)}{\pi}\right)
\end{aligned}$$

$$\begin{aligned}
\mathsf{VP} - W_- &= \frac{1}{2} \sum_{\substack{i<j \\ w_{ij}>0}} w_{ij}(1 - v_i \cdot v_j) - \frac{1}{2} \sum_{\substack{i<j \\ w_{ij}<0}} |w_{ij}|(1 - v_i \cdot v_j) + \sum_{\substack{i<j \\ w_{ij}<0}} |w_{ij}| \\
&= \frac{1}{2} \sum_{\substack{i<j \\ w_{ij}>0}} w_{ij}(1 - v_i \cdot v_j) + \frac{1}{2} \sum_{\substack{i<j \\ w_{ij}<0}} |w_{ij}|(1 + v_i \cdot v_j)
\end{aligned}$$

We need both lemma 2 and a similar result stated in lemma 3. Now, taking into account all this:

$$\begin{aligned}
\mathbb{E}[W] - W_- &= \sum_{\substack{i<j \\ w_{ij}>0}} w_{ij} \frac{\arccos(v_i \cdot v_j)}{\pi} + \sum_{\substack{i<j \\ w_{ij}<0}} |w_{ij}| \left(1 - \frac{\arccos(v_i \cdot v_j)}{\pi}\right) \\
&\geq \alpha \cdot \frac{1}{2} \sum_{\substack{i<j \\ w_{ij}>0}} w_{ij}(1 - v_i \cdot v_j) + \alpha \cdot \frac{1}{2} \sum_{\substack{i<j \\ w_{ij}<0}} |w_{ij}|(1 + v_i \cdot v_j) \\
&= \alpha(\mathsf{VP} - W_-) \geq \alpha(\mathsf{OPT} - W_-)
\end{aligned}$$

□

6.3.2 Clustering Improves Randomized Rounding

The randomized rounding 7 forms two partitions from a set of points, therefore it can be seen as a simple clustering algorithm that allows to prove the approximation guarantee of 0.878. This suggests that using another clustering method may improve the results obtained, at the possible cost of losing the approximation guarantee.

In addition to the randomized rounding 7, we consider the following clustering methods:

- k-**means** (MacQueen, 1967) adapted to the unitary sphere as in (Dhillon and Modha, 2001) for fixed $k = 2$.

- **k-medoids** (Kaufmann and Rousseeuw, 1987) for fixed $k = 2$.
- **Fuzzy c-means** (Bezdek, 1981) adapted to the unitary sphere as in Dhillon and Modha (2001) for fixed $c = 2$.
- **Minimum Spanning Tree** (MST) clustering, i.e., splitting the MST at the longest edge (Grygorash, Zhou, and Jorgensen, 2006).

In order to obtain the spherical variants of k-means and Fuzzy c-means, it is necessary to define the cluster centroids as points on the sphere. This can be achieved by rescaling the centroid to be unit length (Dhillon and Modha, 2001). On the sphere, cosine similarity is a more natural choice than Euclidean dissimilarity. It is not difficult to see that the two variants are actually equivalent:

Lemma 4. *Spherical k-means clustering with cosine similarity is equivalent to k-means clustering with Euclidean distances and normalizing of the centroid vectors in each step.*

Proof. For unitary vectors, the square of their Euclidean distance can be expressed as

$$\|x - y\|^2 = x \cdot x + y \cdot y - 2x \cdot y = 2(1 - \cos\theta)$$

in terms of the angle θ between x and y. The centroid c of a given cluster $\mathcal{C}$ minimizes $\sum_{i \in \mathcal{C}} \|x_i - c\|^2$ and thus maximizes the sum $\sum_{i \in \mathcal{C}} \cos\theta_i$. Analogous, each x_i is assigned to cluster $\mathcal{C}_j$ with minimal value of $\|x_i - c_j\|^2$ and thus maximal $\cos\theta_i$. Thus the squared Euclidean distance and the cosine distance optimize the same objective function. □

The same argument can be made for Fuzzy c-means clustering, since its cost function is also a linear combination of squared Euclidean distances and thus, equivalent, a linear combination of the corresponding cosines. For MST clustering no adjustment is necessary since relation between Euclidean distances and cosines is monotonic and thus the transformation does not affect the MST. By the same token, k-medoids clustering is unaffected by the restriction to the sphere since medoids by definition are always the unitary vectors v_i.

An important ingredient for the clustering procedures is the initialization. For k-means, k-medoids, and fuzzy c-means we consider both a deterministic and a non-deterministic version. In the deterministic version, the pair v_i^* and v_j^* with maximal distance from each other is chosen. In the non-deterministic variant, the two initial cluster centroids are selected from the solution vectors v_i at random. For k-means we observed that choosing the initial cluster centroids as vectors v_i does not work well since the optimization quickly get stuck in a local minimum. We therefore devised two alternative random initialization methods: (1) We generate a random unitary vector r and use $c_1 = r$ and $c_2 = -r$ as initial centroids. (2) We use two independently generated random unitary vectors r_1 and r_2 as initial centroids. The main advantage of method (1) is that this initialization is equivalent to starting k-means from solutions obtained by Goemans-Williamson rounding. To see this, assume (without loss of generality) that the random vector chosen as initial centroid points towards the north pole and therefore the negative of it will point towards the south pole. Then, any point in the lower hemisphere of the hypersphere will be clustered with the south pole vector since this is the closest centroid. The same will happen for points on the upper hemisphere and therefore this is equivalent to splitting the hypershpere into two hemispheres which is the Goemans-Williamson rounding.

In summary, we consider a total of nine clustering procedures:

i) Fuzzy c-means **(Fuzzy)**

ii) Randomized k-means initialized among vectors v_i **(K-MeansRand)**

iii) Deterministic k-means **(K-MeansDet)**

iv) Randomized k-medoids **(K-MedRand)**

v) Deterministic k-medoids **(K-MedDet)**

vi) Minimum Spanning Tree **(MST)**

vii) Randomized Rounding of Goemans-Williamson **(RR)**

viii) Randomized k-means initialized with 2 random vectors **(K-Means2N)**

ix) Randomized k-means initialized with a random vector and its negative **(K-MeansNM)**

Distortion

In order to quantify the quality of the clusters we use the **distortion** (Bishop, 2006; Hastie, Tibshirani, and Friedman, 2009), a measure of cluster coherence defined as

$$\mathcal{C} = \sum_{j=1}^{k} \sum_{x_i \in C_j} ||x_i - \mu_{C_j}|| \tag{6.4}$$

Here $\mu_{C_j} := \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$ is the centroid of the cluster $C_j \in \mathcal{C}$. We note that k-means clustering minimizes the distortion.

Benchmark Data

G-Set Benchmark

In order to assess the utility of clustering as rounding method we used the benchmark set of graphs generated using G. Rinaldi's machine-independent graph generator. Both the generator and the graphs can be downloaded from Ye's web page `http://web.stanford.edu/~yyye/yyye/Gset/`. These graphs vary from 800 to 20000 nodes and have edge weights of ± 1. The topology of the graph can be toroidal, almost planar or random. The first 21 G-set graphs are a standard benchmark for the MAX-CUT problem. The **G-set benchmark** consists of graphs G1 to G67, G70, G72, G77, and G81. The optimal cuts are not know for most of these graphs. We therefore use the best known cut-values compiled in Ma and Hao (2017) for comparison.

The relaxed problem (6.2) was solved using the `CVX` package in Matlab for graphs G1 to G21. For graphs G22 to G54, G57 to G59, G62 to G67, and G72 we used Mathematica's function `SemidefiniteOptimization`. For graphs G55, G56, G60, G61, G70, G77 and G81 neither `SemidefiniteOptimization` nor `CVX` were able to find a solution to the SDP problem. We therefore had to exclude these instances from further consideration. From the SDP solution of each instance we computed 50 iterations for the randomized clustering algorithms, including Goeamans and Williamson randomized clustering and reported the best solution for the seven best algorithms.

Cluster Quality Correlations with Solution Quality

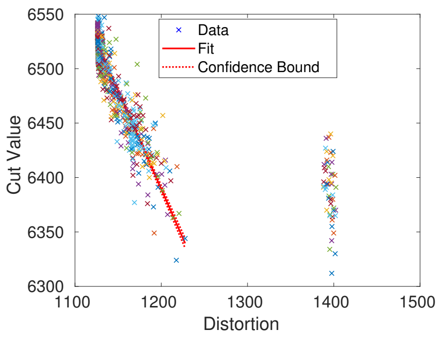
In a preliminary evaluation on the first 21 G-set graphs (see Section 6.3.2) we observed that clustering instead of random rounding yields systematically larger cuts. In order to better understand the reason for the beneficial effect of clustering we investigated the relationship between a quality measure for the clustering and the resulting weight of the maximal cut. Since k-means clustering minimizes distortion it serves as a natural measure of cluster quality, irrespective of the clustering method that is actually used. We chose K-MeansNM for this initial analysis because it uses RR solutions to initialize clusters and thus allows for a direct evaluation of the effect of clustering heuristics.

The RR solutions fall into a very narrow range of distortion values that is clearly separated from the near optimal range achievable by the clustering methods. The cut weights of the RR solutions do not appear to be correlated with the distortion of the corresponding clusters. However, after only a few clustering steps, k-means enters a regime in which distortion and cut weight are strongly correlated, see Figure 12.

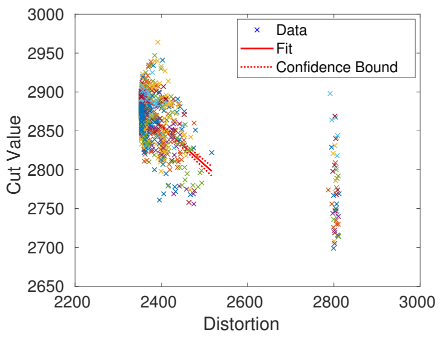
Fig. 12 provides a clear motivation to consider clustering as means of improving the Goemans-Williamson solution. We observe that there are two groups of graphs. In the first groups, exemplified by G43, Fig. 12a, we consistently observe lower RR values at the starting point of k-MeansNM (right) that at the endpoint (top left) and the cut values mostly increase monotonically with decreasing distortion. In the second group of graphs, exemplified by G31, Fig. 12b, this is still the case *on average*, however, the optimal cut weights are observed at sub-optimal distortions. This observation motivates us to record the cut weights for intermediary steps of the cluster procedures, not only at their endpoints. In the case of K-MeansNM this guarantees that we retain the performance guarantee of the Goemans-Williamson bound.

Local Search

In general, neither the Goemans-Williamson nor any of the clustering results are locally optimal. We therefore use adaptive walks as a simple way to further improve solutions. A natural definition of locality for cuts considers two cuts $A \neq A'$ *adjacent* if $A = A' \cup \{u\}$ for some $u \in V \setminus A'$ or $A' = A \cup \{u\}$ for some $u \in V \setminus A$ (Poljak and Rendl, 1995). In terms of the spin vectors x, this amounts to "flipping"



(a) Scatter plot of 50 runs of K-MeansNM for graph G43.



(b) Scatter plot of 50 runs of K-MeansNM for graph G31.

Figure 12: Exploration of the path of 50 iterations of K-MeansNM on the distortion weight of the corresponding cut. The diagram shows all values generated while running the k-means. Points at the bottom right of the plot are the starting points and thus the results of Goemans-Williamson rounding, meanwhile points at the top left are the end points. At each step of K-MeansNM, the points move to the left until the algorithm finds a local minima of distortion. The red line is a linear fit of all the points after the second step of k-means. These show a clear correlation between cluster distortion and cut weight.

(changing the sign of) exactly one spin x_i. An adaptive walk iteratively accepts a spin flip if the cut value $f(A)$ improves. By construction, therefore, an adaptive walk terminates in a locally optimal solution, i.e., in a cut A^* for which there is no adjacent cut with a strictly larger edge weight. We performed local improvement for each of the 50 repetitions of Goemans-Williamson randomized rounding, and the two best-performing clustering algorithms: K-MeansNM and K-Means2N.

An Instance-Specific Approximation Guarantee

Consider a fixed instance of Max-Cut, let $\{v_i\}$ be the solution of the relaxed problem (6.2) and let $\{A, V \setminus A\}$ be a discrete solution. Denote by $\partial A := \{(i,j) \in E(G) | i \in A,\ j \in V \setminus A\}$ the set of cut edges. The value S of the relaxed solution can be written as

$$\begin{aligned} S &= \frac{1}{2}\sum_{(i,j)\in\partial A} w_{ij}\,(1 - v_i \cdot v_j) + \frac{1}{2}\sum_{(i,j)\notin\partial A} w_{ij}\,(1 - v_i \cdot v_j) \\ &= \underbrace{\sum_{(i,j)\in\partial A} w_{ij}}_{f(A)} - \underbrace{\frac{1}{2}\sum_{(i,j)\in\partial A} w_{ij}\,(1 + v_i \cdot v_j)}_{g_{cut}} + \underbrace{\frac{1}{2}\sum_{(i,j)\notin\partial A} w_{ij}\,(1 - v_i \cdot v_j)}_{g_{in}} . \end{aligned}$$

Thus we have $f(A) = S + g_{cut} - g_{in}$. Writing f^* for the weight of the optimal cut, we know that the solution of the relaxed problem is an upper bound, i.e., $S \geq f^*$. We therefore have

$$f(A) \geq f^* - (g_{in} - g_{cut}) \tag{6.5}$$

Note that $g_{in} - g_{cut} \geq 0$ since by definition $f^* \geq f(A)$. First consider the case of positive edge-weights. Then $f(A) > 0$ and we can estimate the approximation ratio for the solution A as

$$\alpha(A) := \frac{f(A)}{f^*} \geq 1 - \frac{g_{in} - g_{cut}}{f^*} \geq 1 - \frac{g_{in} - g_{cut}}{f(A)} \tag{6.6}$$

If negative edge weights, we follow Goemans and Williamson (1995), define $W_- := \sum_{(i,j)} \min(w_{ij}, 0) \leq 0$, and make use of the fact that $f^* - W_- \geq 0$. From Eq.(6.5) we obtain immediately $f(A) - W_- \geq f^* - W_- - (g_{in} - g_{cut})$ and thus a generalized version of the approximation ration can be computed as

$$\alpha(A) := \frac{f(A) - W_-}{f^* - W_-} \geq 1 - \frac{g_{in} - g_{cut}}{f^* - W_-} \geq 1 - \frac{g_{in} - g_{cut}}{f(A) - W_-} \tag{6.7}$$

Instant-Specific Performance Bounds

The bounds in Eqns. (6.6) and (6.7) can be seen as instant-specific versions of the general approximation ratio derived in Goemans and Williamson (1995). Empirically, we found in benchmarking experiments (see below) that the **instance specific bound** $\alpha(A)$ substantially exceeds the uniform Goemans-Williamson bound of $\alpha \approx 0.878$. For the Goemans-Williamson algorithms, of course, we necessarily have $\mathbb{E}[\alpha(A)] \geq \alpha$.

Results

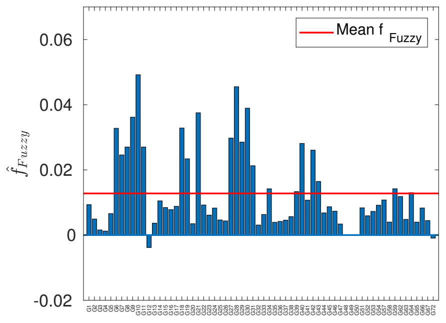
In order to compare the cut values of RR with each of the clustering algorithm we use the following relative measure of performance

$$\hat{f}_{cluster} = \frac{f_{cluster} - f_{RR}}{f_{RR}} = \frac{f_{cluster}}{f_{RR}} - 1 \tag{6.8}$$

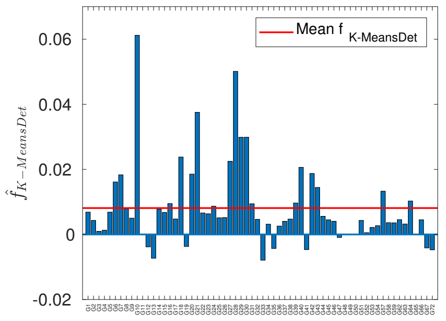
Fig. 13 shows that clustering on most instances yield significant improvement for most clustering approaches. K-MeansNM by construction always find a solution that is at least as good as RR, however both K-MeansNM and K-Means2N *never* give a lower solution than RR. K-meansRand improves for all graphs except G12, Fuzzy for all except G12 and G72, K-MedRand for all except G32, G39 and G72 and finally for K-MeansDet in 9 graphs a better solution was not found.

The gains in solution quality differ substantially between the test instances, and a few graphs, in particular G12 and G72, do not profit from the clustering approaches other than the randomized versions of K-means. Interestingly, these two graphs are toroidal.

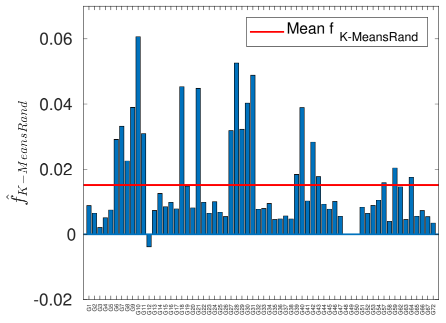
The same trend is also observed for the instance-specific performance bounds, see Fig. 14. The performance bounds for the individual solutions are well above 0.9, i.e., exceed the Goemans and Williamson also in those few cases where clustering is worse than RR. For bipartite graphs with non-negative edge weights the entire edge set forms a maximal cut (Delorme and Poljak, 1993). This is the case for the two unweighted graphs G48 and G49 and explains why for these two cases no difference between RR and clustering solutions is observed in Fig. 13.



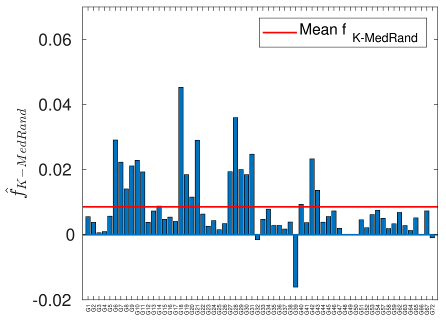
(a) Plot of $\hat{f}_{cluster}$ for Fuzzy c Means clustering.



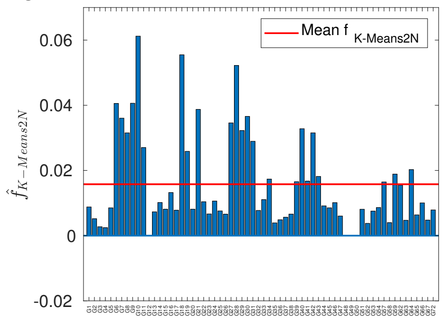
(b) Plot of $\hat{f}_{cluster}$ for K-MeansDet clustering.



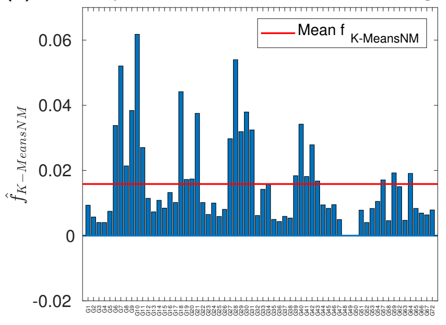
(c) Plot of $\hat{f}_{cluster}$ for K-MeansRand clustering.



(d) Plot of $\hat{f}_{cluster}$ for K-MedRand clustering.

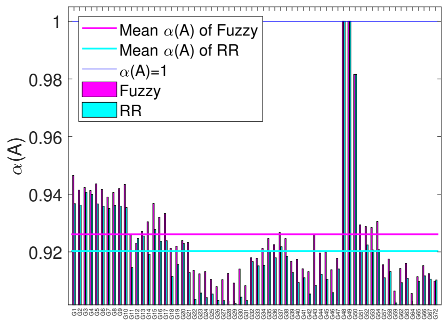


(e) Plot of $\hat{f}_{cluster}$ for K-Means2N clustering.

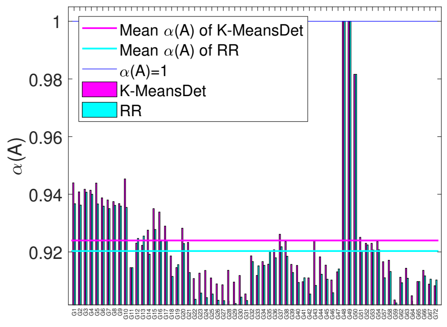


(f) Plot of $\hat{f}_{cluster}$ for K-MeansNM clustering.

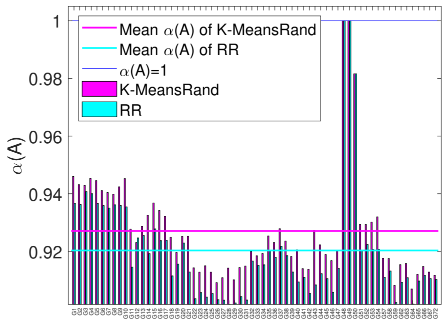
Figure 13: Comparison between the cut value found by clustering algorithms and RR, using $\hat{f}_{cluster}$ as the comparison. The horizontal axis represents the graph number, i.e., graph Gi is shown at position i. The red line indicates the average over the benchmark set. Positive values indicate that the clustering solutions are superior to the RR solutions.



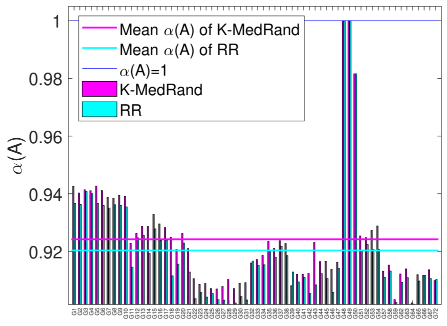
(a) Comparison of $\alpha(A)$ for Fuzzy clustering and RR.



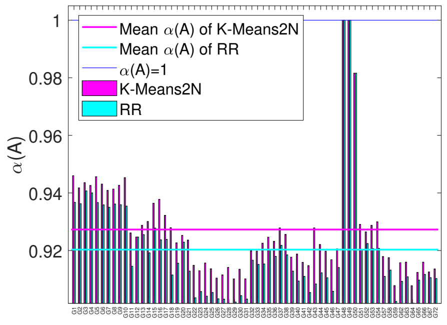
(b) Comparison of $\alpha(A)$ for K-MeansDet and RR.



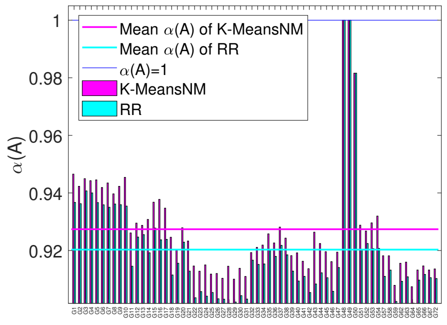
(c) Comparison of $\alpha(A)$ for K-MeansRand and RR.



(d) Comparison of $\alpha(A)$ for K-MedRand and RR.



(e) Comparison of $\alpha(A)$ for K-Means2N and RR.



(f) Comparison of $\alpha(A)$ for K-MeansNM and RR.

Figure 14: Comparison of instance-specific performance bounds $\alpha(A)$ between clustering algorithms and RR. The cyan line is the average of $\alpha(A)$ for the clustering methods and the magenta line is the average of RR for comparison.

On average all clustering algorithms yield improvements over RR. Interestingly, the average solution quality depends noticeably on the clustering algorithm. The clustering algorithm with largest average improvement was K-MeansNM, as expected. On the benchmark set, an average increase on the cut weight by $\hat{f}_{\text{K-MeansNM}} = 1.541\%$ compared with RR is obtained. Other variants of 2-means performed similarly well. We found $\hat{f}_{\text{K-Means2N}} = 1.533\%$ and $\hat{f}_{\text{K-MeansRand}} = 1.471\%$. For Fuzzy clustering we only observed an improvement of $\hat{f}_{\text{Fuzzy}} = 1.247\%$. With $\hat{f}_{\text{K-MedRand}} = 0.841\%$ and $\hat{f}_{\text{K-MeansDet}} = 0.789\%$, performance of medoids and deterministic methods was less encouraging. For individual instances the improvement was substantial. For example we obtain an improvement of 5.81% for the graph G10 and 5.12% for G28 with K-MeansNM.

Despite subtle differences between the clustering methods, the clusters are quite similar in their characteristics. One measure of interest is the *mean clustering angle*

$$\theta_A = \arccos\left(\left|\frac{1}{|A|}\sum_{v_i \in A} v_i\right|\right)$$

It measures the average angle between two unit vectors in the same cluster. We found that the cardinalities $|A|$ and $|X \setminus A|$ are nearly even and θ_A lies between 60 and 75 degrees for the graphs in the benchmark set. We observed not convincing trends connecting these parameters and the weight of maximum cut.

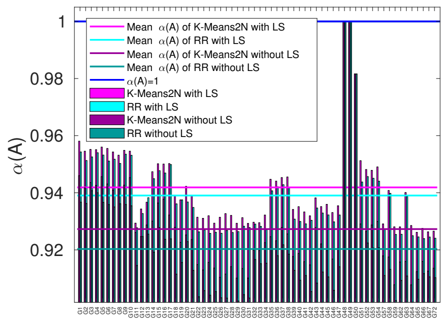
Local Search Improvement

A straightforward way to improve a given solution of $\textsc{Max-Cut}$ is to add a Local Search (LS) step. We use adaptive walks for this purpose and restrict ourselves to RR and the two best-performing clustering approaches, i.e., K-MeansNM and K-Means2N. The results presented in Figure 16 are the best solutions found in all of the 50 iterations for each algorithm. The same relative measure of performance, $\hat{f}_{cluster}$, is used as in Section 6.3.2. The corresponding instance-specific performance bound α can be found in Figure 15.

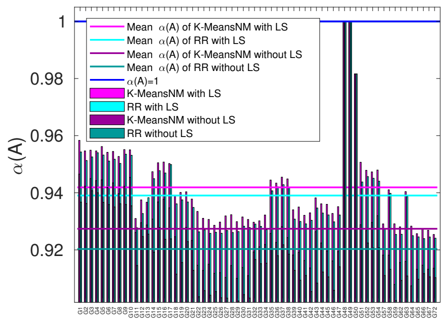
Figure 17 shows the ratio $f_{rounding}/f_{best}$, where $f_{rounding}$ is the best solution after local search found either by clustering or RR for that instance and f_{best} is the best solution known in the literature, taken from Ma and Hao (2017). The corresponding values are also available in tabular form as Additional Material. Graphs G11 to G13, G32 to G34, G57, G62, G65 to G67, and G72 have toroidal topology; both the clustering algorithm and Randomized Rounding show a comparably low approximation ratio for these instances. For graphs with other topologies, we obtain approximation ratios exceeding 0.96, sometimes closely approaching 1. The combination of the Goemans–Williamson relaxation with clustering thus at least comes close to the best solutions known in the literature.

6.3.3 Spectral Clustering

The success of k-means related clustering approaches suggests to extend this idea to other clustering methods, for instance, Spectral Clustering (SC). We

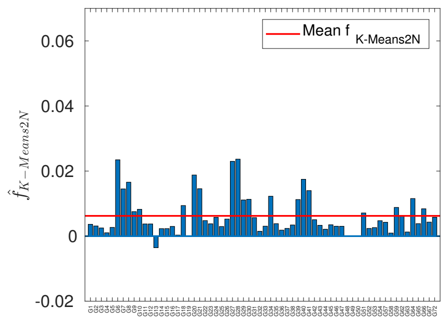


(a) Plot of $alpha$ for K-Means2N clustering after local search.

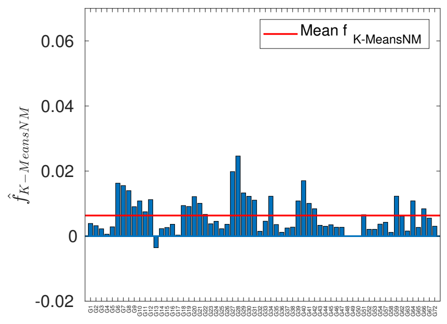


(b) Plot of $alpha$ for K-MeansNM clustering after local search.

Figure 15: Comparison of instance-specific performance bounds $\alpha(A)$ between clustering algorithms and RR after local search (LS). The horizontal axis represents the graph number, i.e., graph Gi is shown at position i. The cyan line is the average of $\alpha(A)$ for the clustering methods after LS and the magenta line is the average of RR after LS for comparison. Purple and Green lines are the average of the clustering methods and RR before LS respectively.



(a) Plot of $\hat{f}_{cluster}$ for K-Means2N clustering after local search.



(b) Plot of $\hat{f}_{cluster}$ for K-MeansNM clustering after local search.

Figure 16: Comparison between the cut value found by clustering algorithms and RR after local search (LS), using $\hat{f}_{cluster}$ as the comparison. The horizontal axis represents the graph number, i.e., graph Gi is shown at position i. The red line indicates the average over the benchmark set. Positive values indicate that the locally improved clustering solutions are superior to the locally improved RR solutions.

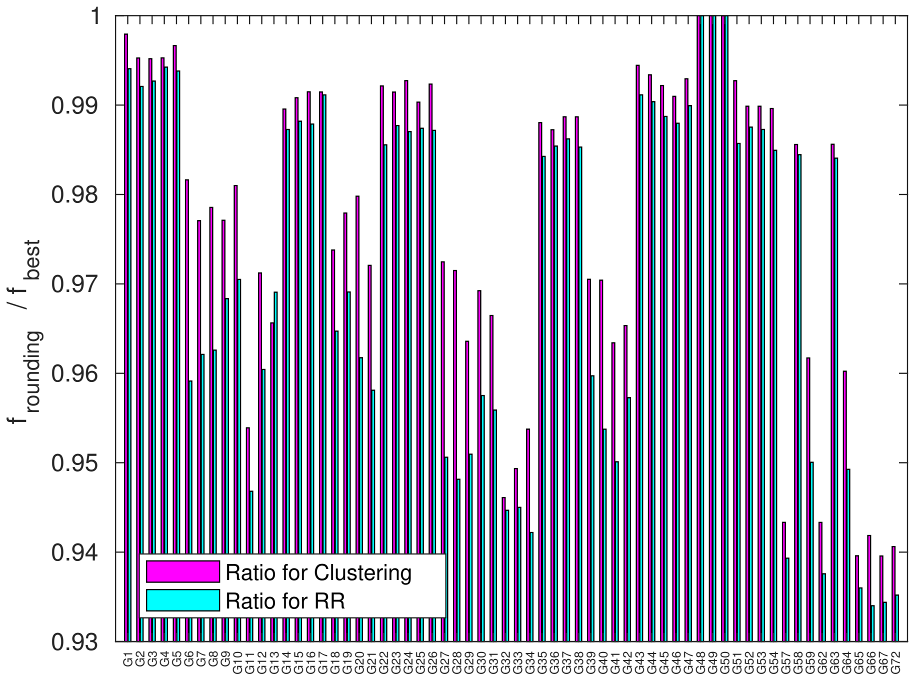


Figure 17: Comparison of the best cut value obtained with clustering (in magenta) and RR (in cyan) and subsequence local improvement with the best cut value (f_{best}) available in the literature.

implemented the graph version of the NJW-SC algorithm (Ng, Jordan, and Weiss, 2002) described in detail in Gonzalez-Torres (2020) and in Algorithm 8. Note that in step 3: **Similarity Matrix** of Algorithm 8, the similarity measure used is Gaussian, however it can be replaced for any suitable similarity measure.

In this case, we are using points that are in the unitary sphere, therefore the dot product between points can be used a similarity measure. If we have n data points $x_i \in X$, consider the following similarity matrix:

$$A_{ij} = 1 + x_i \cdot x_j \tag{6.9}$$

This similarity measure is always positive, and the closest a pair of points are, the higher the similarity will be. However it might be useful to fix a threshold β of values we want to consider, so we used the following **dot product similarity** measure:

Dot Product Similarity

$$A_{ij} = \begin{cases} 1 + x_i \cdot x_j & \text{if } 1 + x_i \cdot x_j \geq \beta \\ 0 & \text{otherwise} \end{cases} \tag{6.10}$$

The idea is that with the parameter β, points that are "sufficiently" far away between each other are essentially considered as minimum similarity which is then set to be 0. Note that by setting $\beta = 0$ we can recover the normal dot product similarity.

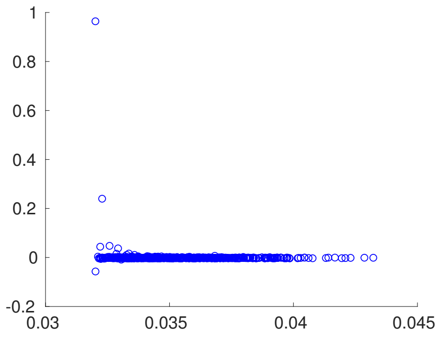
Algorithm 8: NJW-SC (Graph Version)

```
1: Input: Set of points $\mathbf{X}$, number of clusters $k$
2: Output: $X_c$, a k partition of X
3: Similarity Matrix. Set the similarity matrix $A$ as:
       $$A_{ij} \leftarrow \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right);$$
4: Compute the matrix of weights $D$:
       $$D \leftarrow \text{diag}(A1_n);$$
5: Compute $k$ eigenvectors corresponding to the largest $k$ eigenvalues:
       $$Z \leftarrow \text{topEig}(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}, k);$$
6: Using the points Z, compute a partition $X_c$ of the data set $X$;
7: return $X_c$
```

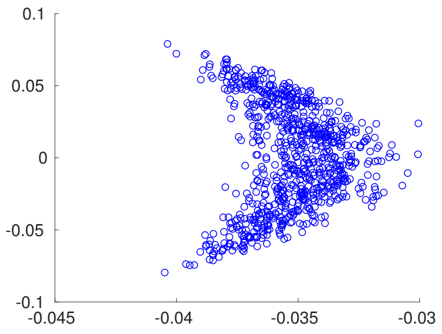
Note that both the Gaussian and the dot product similarities have parameters that can be modified. For the Gaussian similarity we have σ, which will increase the similarity between distant points, the larger σ is. For example, for the case where we have two anti parallel unitary vectors, a value of $\sigma = 1$ will give us a similarity of $\approx$ 0.135, meanwhile $\sigma = 2$ will give a similarity of $\approx$ 0.607. For the dot product similarity however, the parameter β is just a threshold of values to consider.

Also note that step 6 of Algorithm 8 requires to do clustering of the new set of data points Z. Therefore we can use the same six clustering algorithms used in Rodriguez-Fernandez et al. (2020) in order to do a direct comparison between both methods. For SC we will use the standard version of the algorithms instead of the spherical clustering version, since in general, data points Z obtained in step 5 of Algorithm 8 are not unitary.

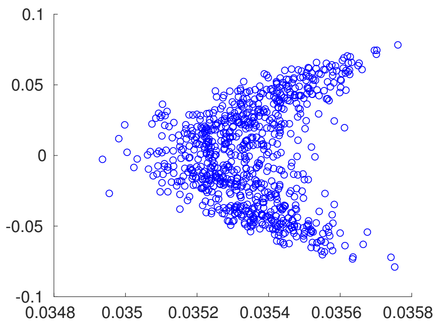
Additionally, for K-MeansNM and K-Means2N the initialization method had to be changed. Instead of generating each entry of the starting centroids using a normal distribution with 0 mean and standard deviation of 1, we generate the entries of the centroids using a standard deviation of 0.01. The reason for this, is illustrated in Figure 18, where we can see, specially for $\sigma = 1$, that the data points Z are close to 0 in one of the coordinates in the negative side. If we were to use centroids generated with standard deviation of 1, we would obtain two centroids that are, in general, with larger norm and far away from the set of data points Z. For some cases, this will result in a cluster with cut value 0, since all the data points Z were assigned to only one centroid.



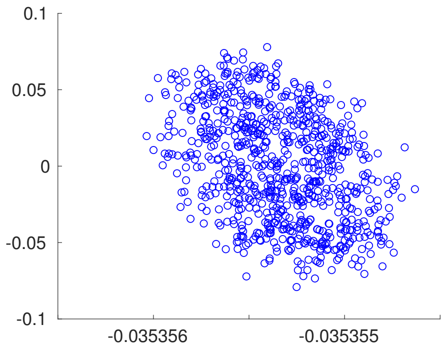
(a) Plot of Z (Algorithm 8) for graph G1 and $\sigma = 0.25$ using Gaussian similarity.



(b) Plot of Z (Algorithm 8) for graph G1 and $\sigma = 0.5$ using Gaussian similarity.



(c) Plot of Z (Algorithm 8) for graph G1 and $\sigma = 1$ using Gaussian similarity.

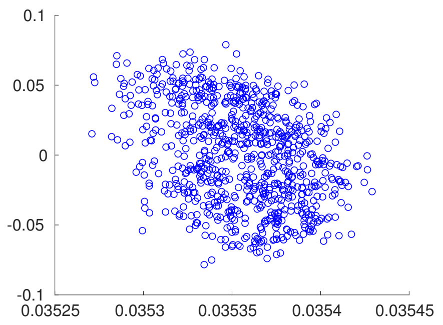


(d) Plot of Z (Algorithm 8) for graph G1 and $\sigma = 10$ using Gaussian similarity.

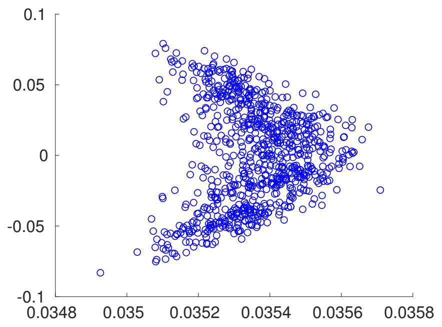
Figure 18: Comparison of the Z space of Algorithm 8 for graph G1 using different values of the parameter σ of the Gaussian similarity.

Experiments were made using Algorithm 8 with both the Gaussian and dot product similarity measures for step 3. For both similarity measures, a wide range of values for the parameters were used, however the results obtained were similar to the procedure of Rodriguez-Fernandez et al. (2020). A comparison between the results of SC and Rodriguez-Fernandez et al. (2020) (before LS) is presented in a separate table for each clustering algorithm and can be seen in Table 2 to Table 8 of the Appendix A.

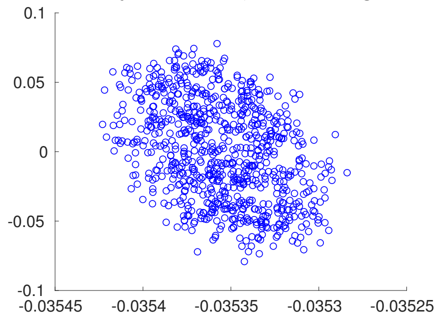
Looking at the tables in Appendix A, for *all clustering methods*, SC finds a higher cut value than Rodriguez-Fernandez et al. (2020) in 20 graphs. Those graphs are G5, G6, G9, G10, G14, G15, G17, G18, G22 - G24, G28 - G31, G40, G43, G51 and G63 - G64. Additionally, from both Figure 18 and Figure 19, we can notice that this similarity measure does not separate the data points into two clusters (except for graph G50 and the bipartite graphs G48, G49). Moreover, there are parameter values such as $\sigma = 0.25$ which result in points on the Z space with undesired geometry. There are two possible explanations for this; either there are better similarity measures and parameters, or the set of points are not separable to



(a) Plot of Z (Algorithm 8) for graph G1 and $\beta = 0.25$ using Dot Product similarity.



(b) Plot of Z (Algorithm 8) for graph G1 and $\beta = 0.5$ using Dot Product similarity.



(c) Plot of Z (Algorithm 8) for graph G1 and $\beta = 0$ using Dot Product similarity.

Figure 19: Comparison of the Z space of Algorithm 8 for graph G1 using different values of the parameter β of the Dot Product similarity.

begin with.

6.4 Laplacian and Gram Matrix Spectrum Analysis

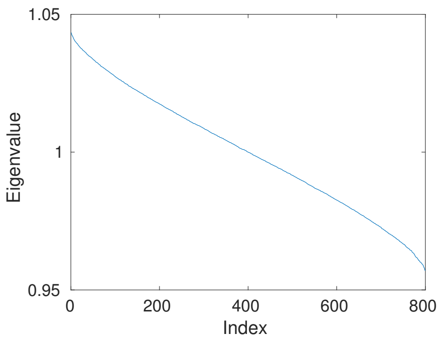
Using SC naturally motivates the analysis of the spectrum of the Gram matrix of the vectors v_i solution of (6.2) and also the spectrum itself of the benchmark graphs, since it contains important importation of properties and the structure of the problem.

The Gram matrix G of a set of n vectors v_i, is defined as $G_{ij} = v_i \cdot v_j$. The spectrum of a graph (Cvetkovic et al., 1988) is obtained by calculating the eigenvalues and eigenvectors of the **Symmetric Normalized Laplacian** (Chung, 1997) matrix L of a graph G, defined as

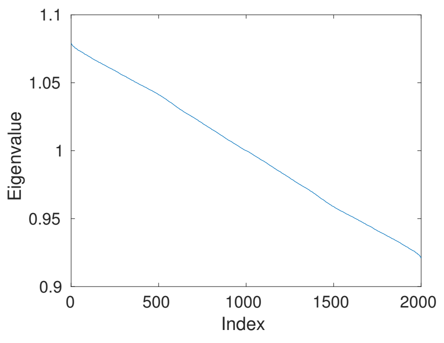
Symmetric Normalized Laplacian

$$L = I - D_W^{-\frac{1}{2}} W D_W^{-\frac{1}{2}} \tag{6.11}$$

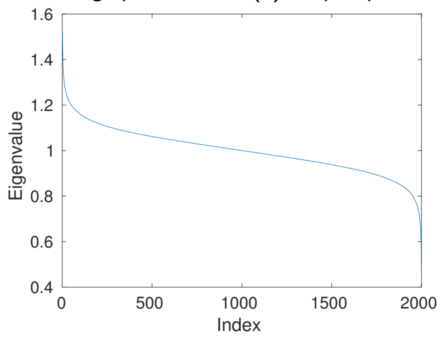
where D_W is the weighted degree matrix ($D = \text{diag}(D_1, \ldots, D_n)$ and $D_i = \sum_{j=1}^{n} w_{ij}$) and W the weighted adjacency matrix ($W_{ij} = w_{ij}$ for all $(i,j) \in$ E and 0 otherwise).



(a) Graph Spectrum for random graph G10.



(b) Graph Spectrum for toroidal graph G32.



(c) Graph Spectrum for planar graph G41.

Figure 20: Plots of the spectrum for different types of graphs.

We obtained the spectrum of all the benchmark graphs used in Section 6.3.2, and the results are shown in Figure 20 as a plot of the eigenvalues in decreasing order. There are only three plots since the behavior is well represented with these examples. For all planar graphs we see curves similar to Figure 20c, with rapid decrease of the largest eigenvalues, for random graphs we observe a slower decrease of the largest eigenvalues as depicted in Figure 20a, and finally we observe almost a straight line for the toroidal graphs (Figure 20b).

For the Gram matrix spectrum, the results are shown as a plot of the eigenvalues in decreasing order in Figure 21, where six representative examples were chosen. On average, almost all graphs present curves similar to Figure 21e, in the sense that around 15 eigenvalues are nonzero. There are some instances where the number of nonzero eigenvalues is larger than the average (Figure 21f) or smaller than the average (Figure 21a). There are three particular interesting graphs: G48 (Figure 21b), G49 (Figure 21c), and G50 (Figure 21d). As mentioned in Section 6.3.2, G48 and G49 are bipartite graphs, and the number of nonzero eigenvalues is exactly 2 for this case. For G50, which is not bipartite but all algorithms find the optimal solution, only the first 3 eigenvalues are nonzero. This suggests a *relation between the number of nonzero eigenvalues and the "difficulty"* of the instance.

B-set Benchmark

The **B-set benchmark**, consisting of 7 graphs and shown in Figure 28 of the Appendix B, was designed in order to explore the relation between difficulty and the number of nonzero eigenvalues. The design of the graphs in this benchmark was made so every new graph increases in structural complexity. The first 4 graphs are planar trees with only small changes in the edge weights, graph number 5 introduces a cycle and the last 2 graphs are two cliques joined by one edge. These simple graphs with a small number of nodes will provide good intuition about the MAX-CUT relaxation and the procedure of both Goemans and Williamson (1995) and Rodriguez-Fernandez et al. (2020).

A Principal Component Analysis (PCA) (Hotelling, 1933; Pearson, 1901) of the vectors v_i obtained from Equation 6.2 for the B-set, shown in Figure 22, will provide additional information about the relation between problem difficulty and graph structure. Additionally, the gram matrix and graph spectrum for these graphs are shown in Figure 29 and Figure 30 of the Appendix C.

For the first 4 graphs, which are trees, we see from Figure 22 that all the information needed to solve the problem is contained in only one Principal Component (PC). Moreover, from Figure 29 we see that the number of nonzero eigenvalues is 2, which agrees with the theory that this number is related to the difficulty of an instance.

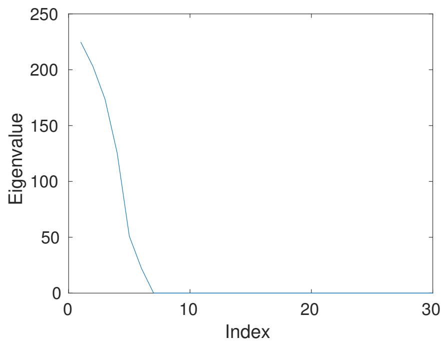
When we introduce a cycle to the trees, i.e., graph B5, the *necessary* information to solve the problem is still contained in the first PC (Figure 22e), however the variance is now contained in the first two PCs instead of only one. Additionally, the number of nonzero eigenvalues increases to three.

Finally, graph B6 consists of two cliques joined with a large positive weight edge, meanwhile for graph B7 the edge that joins the cliques has a large negative weight. The jump in structural complexity for these graphs comes together with a larger number of nonzero eigenvalues; 8 to be precise. Looking at the PC space in Figure 22f and Figure 22g, we see that there exists more than one optimal clustering (with respect to distortion). Also the variance is now contained in more than 3 PCs, and different distortion optimal clusters translate into different cut values, making the problem *difficult* to solve.

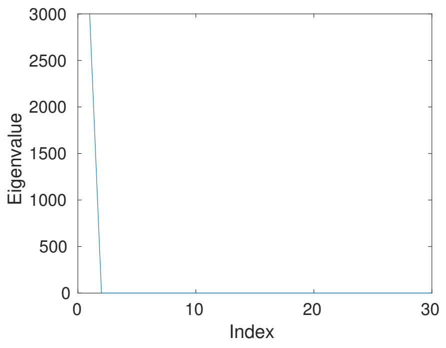
Note that we are using the vectors v_i that are the solution of an optimization problem. With the B-set benchmark we can see that this Semidefinite Programming (SDP) optimization problem is in fact trying to separate the nodes in two sets, using the less amount of dimensions possible. For structurally simple problems, such as trees or bipartite graphs, Equation 6.2 finds a solution that is close to the integer programming version, in the sense that two clearly separable sets are found using only *few dimensions*. There also appears to be a connection between the number of nonzero eigenvalues of the gram matrix and the *difficulty* of an instance.

6.5 RandomizedMinMax

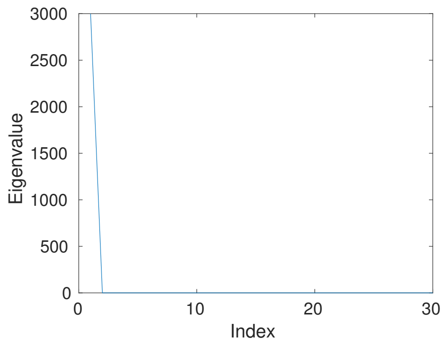
In summary, the procedure described in Rodriguez-Fernandez et al. (2020) consists of doing clustering to the solution vectors v_i of equation (6.2) to obtain a first



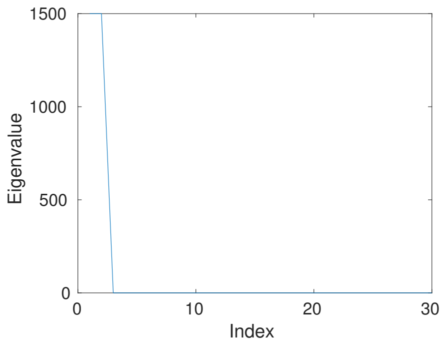
(a) Spectrum of the Gram matrix for graph G11.



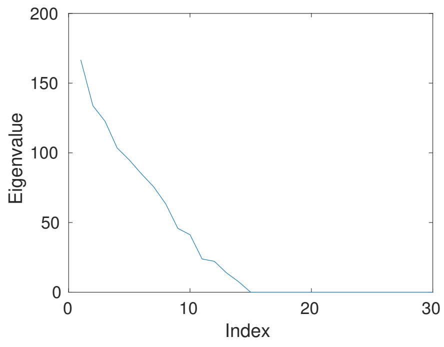
(b) Spectrum of the Gram matrix for graph G48.



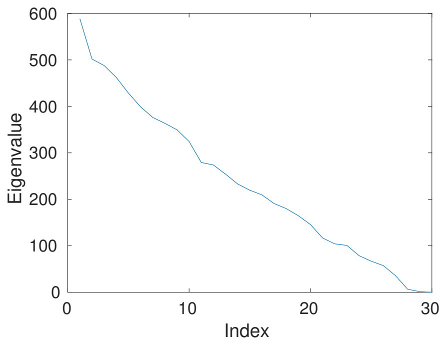
(c) Spectrum of the Gram matrix for graph G49.



(d) Spectrum of the Gram matrix for graph G50.

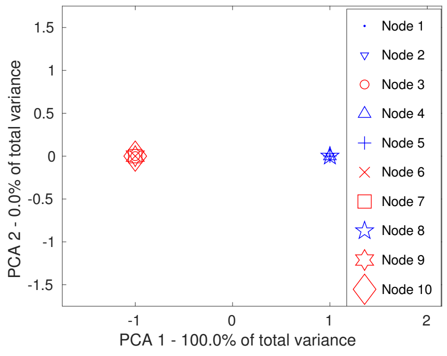


(e) Spectrum of the Gram matrix for graph G54.

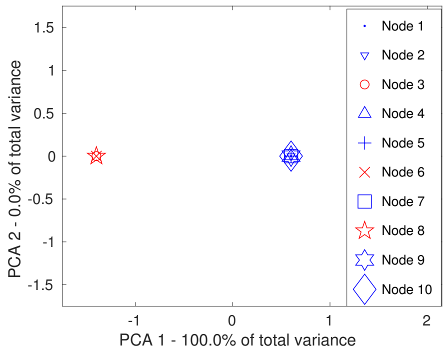


(f) Spectrum of the Gram matrix for graph G59.

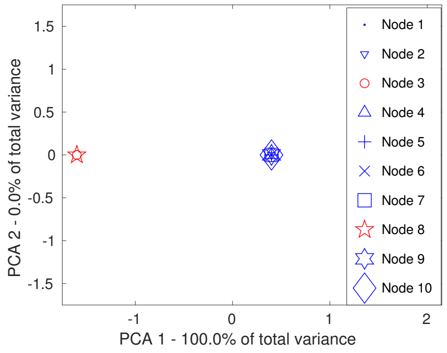
Figure 21: Spectrum plots of the first 30 Gram matrix eigenvalues in decreasing order for some representative examples.



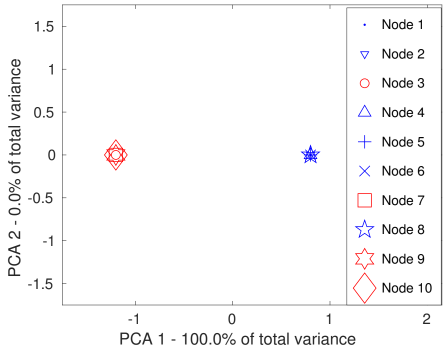
(a) Vectors v_i from Equation 6.2 after PCA for graph B1.



(b) Vectors v_i from Equation 6.2 after PCA for graph B2.



(c) Vectors v_i from Equation 6.2 after PCA for graph B3.

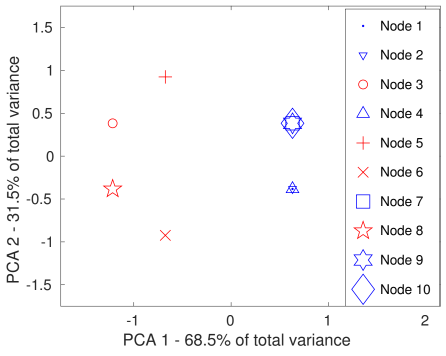


(d) Vectors v_i from Equation 6.2 after PCA for graph B4.

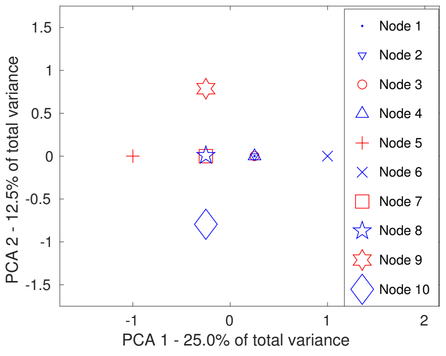
Figure 22: Plot of the vectors v_i from Equation 6.2 in the Principal Component space using the two components with most variance for the B-Set of graphs. The MAX-CUT partition is represented with two colors, red and blue, corresponding to the two partitions of the node set.

partition of the node set, and then, starting from this solution, a local search is performed maintaining only solutions that improve the cut value. Since k-means clustering minimizes distortion (see Subsection 6.3.2 for a definition), and Local Search (LS) stops when it finds a local maximum for the cut value, this can be seen, in very loose terms, as a randomized minimization maximization problem, which we will denote as **RandomizedMinMax (RMM)**. **RMM**

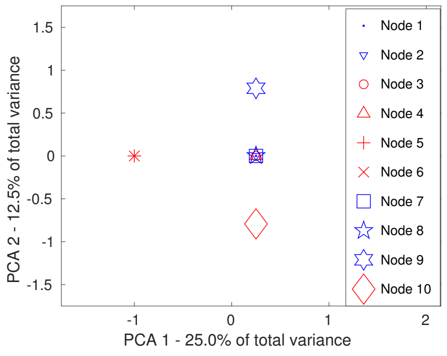
In Rodriguez-Fernandez et al. (2020) both k-means and LS are iterated only once, but other than computation time, there is no reason not to do both algorithms several times. This procedure is what we call RMM, and consists of repeating the process of Rodriguez-Fernandez et al. (2020), i.e., clustering using the solution vectors of (6.2) followed by a LS, and then from the solution obtained from LS, start again with clustering and continue, until no change is found or a maximum number of iterations is reached.



(e) Vectors v_i from Equation 6.2 after PCA for graph B5.



(f) Vectors v_i from Equation 6.2 after PCA for graph B6.



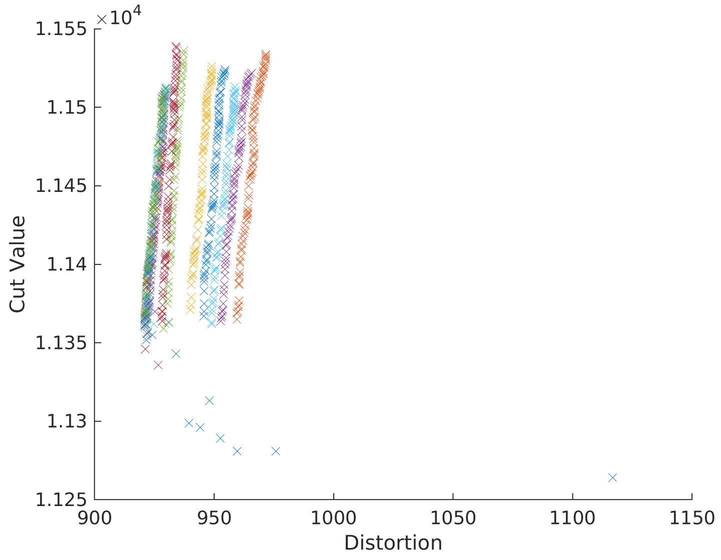
(g) Vectors v_i from Equation 6.2 after PCA for graph B7.

Figure 22: (Continued) Plot of the vectors v_i from Equation 6.2 in the Principal Component space using the two components with most variance for the B-Set of graphs. The MAX-CUT partition is represented with two colors, red and blue, corresponding to the two partitions of the node set.

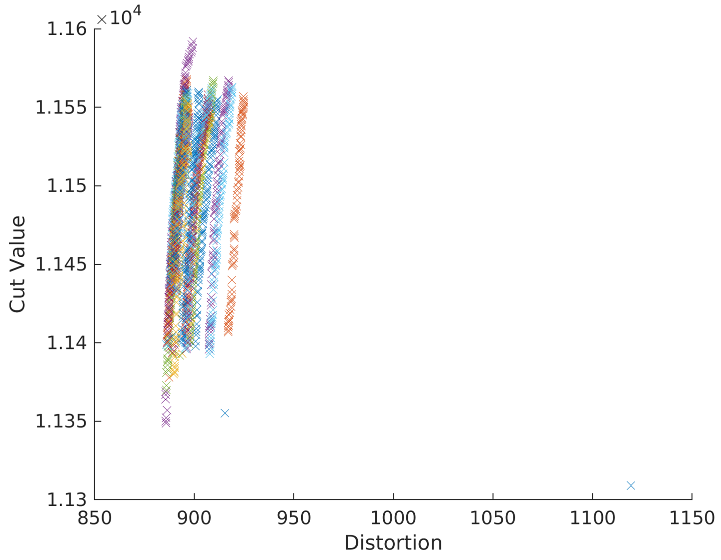
A summary of RMM can be seen in Algorithm 9. It has 3 stopping conditions, as seen in steps 6, 10 and 16. Respectively, the conditions are the following: In step 6, if a maximum number of iterations is reached, in step 10, if there is no change of both inertia and cut value comparing the partition obtained by clustering on the current iteration and the previous one, and in step 16, if the partition itself does not change between LS and clustering.

Ideally, in a plot of the trajectory of one particular solution in a cut value vs distortion plot, such as in Figure 12 but for RMM, one would expect to obtain points along some kind of saw function, going up during LS and going left during k-means clustering, but overall with an average tendency to the upper left of the plot, i.e., to low distortion with high cut values.

However, as seen in Figure 23, we obtain in most cases a contradictory behavior between LS and k-means. We see that LS optimizes almost in a straight vertical

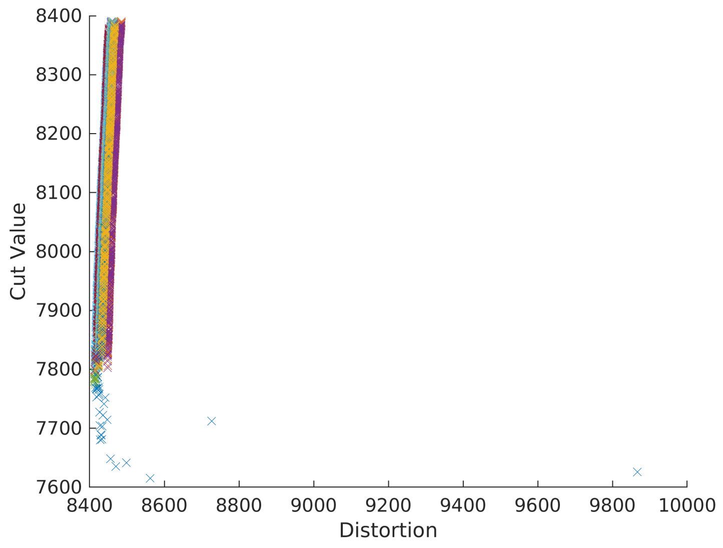


(a) 31st trajectory of RMM for graph G2.

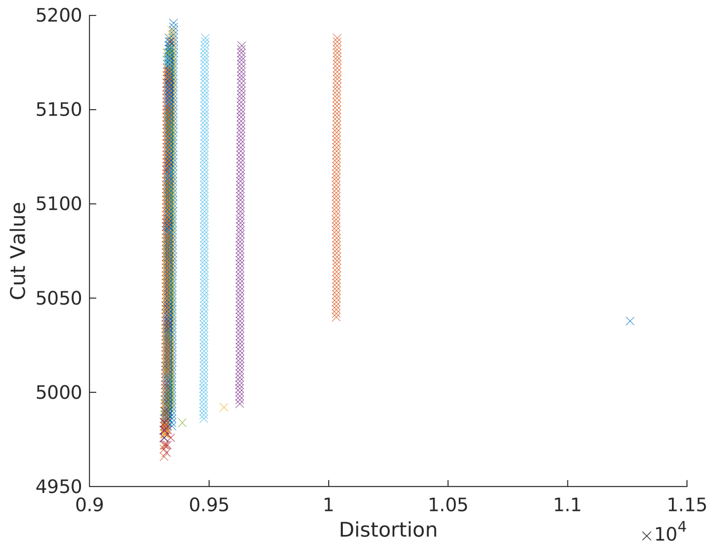


(b) 49th trajectory of RMM for graph G4.

Figure 23: Plot of trajectories obtained using Algorithm 9 for different graphs depicting some of the types of trajectories found. Points with the same color are obtained during a run of LS (or Clustering). Typically, the more vertical trajectories represent LS. The runs of clustering are difficult to observe since the number of steps to find a local minimum of inertia was in the order of $\sim 10^1$, and sometimes even less than 10. The starting point of the whole algorithm is the blue cross on the far bottom right.



(c) 45th trajectory of RMM for graph G60.



(d) 46th trajectory of RMM for graph G61.

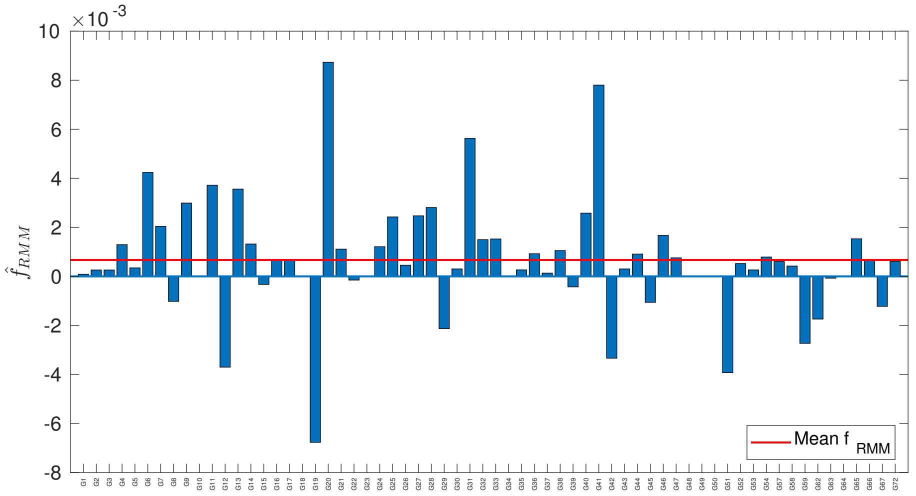
Figure 23: (Continued) Plot of trajectories obtained using Algorithm 9 for different graphs depicting some of the types of trajectories found. Points with the same color are obtained during a run of LS (or Clustering). Typically, the more vertical trajectories represent LS. The runs of clustering are difficult to observe since the number of steps to find a local minimum of inertia was in the order of $\sim 10^1$, and sometimes even less than 10. The starting point of the whole algorithm is the blue cross on the far bottom right.

Algorithm 9: Randomized Minimization Maximization (RMM)

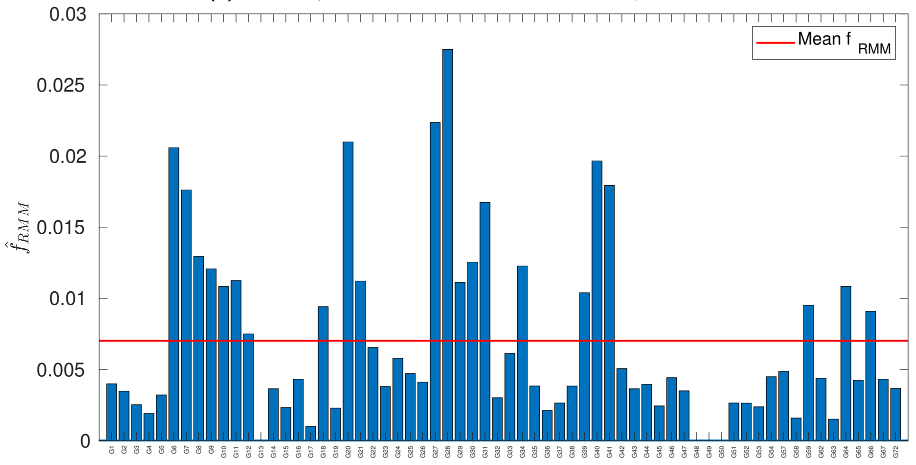
```
1: Solve (6.2), obtaining an optimal set of vectors $v_i$
2: Set $k \leftarrow 2$
3: Set number_iter $\leftarrow 0$
4: Set max_iter $\leftarrow 151$
5: **Initial Clustering.** Do k-means clustering on the set of $\{ v_i \ | i = 1, \dots, n \}$ to
   obtain two partitions $A$ and $B = A^c$, with cut value $A^{cv}$ and inertia $A^{inert}$
6: while number _iter $\leq$ max _iter do
7:     **LS step.** Do a LS starting from $A$ to obtain $A_{LS}$
8:     **Clustering step.** Starting from $A_{LS}$, do k-means clustering to obtain $A_{Clust}$,
       with cut value $A^{cv}_{Clust}$ and inertia $A^{inert}_{Clust}$
9:     $number\_iter + = 2$
10:    if $\left(A^{inert} == A^{inert}_{Clust}\right) \&\& \left(A^{cv} == A^{cv}_{Clust}\right)$ then
11:        BREAK
12:    else
13:        Set $A \leftarrow A_{Clust}$
14:        Set $A^{inert} \leftarrow A^{inert}_{Clust}$
15:        Set $A^{cv} \leftarrow A^{cv}_{Clust}$
16:    if $A_{Clust} == A_{LS}$ then
17:        BREAK
18: Return: $A_{Clust}$
```

line, i.e., with distortion almost constant. This may be to the fact that the local search we are using is only exchanging one pair of vectors at a time, therefore not affecting the overall quality of the cluster. After LS, k-means finds, already in the first step of the algorithm, a solution with lower distortion, however with cut value almost the same or lower than the starting point of LS. This contradictory behavior, however, let RMM explore a great part of the cut-inertia space.

We performed 50 iterations of RMM (Algorithm 9) for the same benchmark as in Section 6.3.2. On more than 78% of the instances, better or equal results than K-MeansNM from Rodriguez-Fernandez et al. (2020) were found, and a plot of the comparison between RMM, K-MeansNM and Randomized Rounding of Goemans and Williamson (1995) can be found on Figure 24. These results encourage other methods to explore the *distortion-cut value* space, for instance, random walks, genetic algorithms or a redesign of RMM that allows a wider search on the space.



(a) Plot of $\hat{f}_{RMM}$ with K-MeansNM as comparison.



(b) Plot of $\hat{f}_{RMM}$ with Randomized Rounding as comparison.

Figure 24: Comparison between the cut value found by RMM with K-MeansNM and Randomized Rounding, using the relative measure of performance $\hat{f}_{RMM}$ (see Equation 6.8). For comparing RMM and K-MeansNM in Figure 24a, f_{RR} has to be replaced by $f_{K-MeansNM}$ in Equation 6.8. The horizontal axis represents the graph number, i.e., graph Gi is shown at position i. The red line indicates the average over the benchmark set. Positive values indicate that the RMM solutions are superior to the clustering solutions.

CHAPTER 7

Max-k-Cut

Contents

The MAX-k-CUT problem is a generalization of the MAX-CUT problem and is closely related to graph coloring. It arises naturally when trying to find the ground states of the Potts model, a generalization of the Ising model for a spin glass (Welsh, 1993). This problem has applications in circuit layout design (Barahona et al., 1988), in wireless communication problems (Fairbrother, Letchford, and Briggs, 2018; Niu et al., 2017) and in statistical physics (Liers et al., 2004).

7.1 Problem Definition

Let $G = (V, E)$, be an undirected graph with $n = |V|$ nodes and $m = |E|$ edges. Each edge $(i, j) \in E$ going from node i to node j will have a weight of w_{ij}. The goal of MAX-k-CUT is to partition the node set V into k disjoint sets such that the sum of weights for edges going between nodes in different partitions is maximized. An example of the maximum k cut of the Petersen graph for $k = 3$ is depicted in Figure 25. Formally, we define a **k-cut** as a k partition $A = A_1, A_2, \dots, A_k$ of V with cut value

k-**Cut**

$$w(A) = \sum_{1 \le q < r \le k} \sum_{i \in A_q, j \in A_r} w_{ij}$$

Then, the Max k-Cut problem reads as follows:

$$\begin{aligned} \text{Maximize} \quad & w(A) \\ \text{subject to:} \quad & |A| = k \end{aligned} \tag{7.1}$$

This problem, however, is NP-Complete, so the best we can do is to find approximate solutions unless $P = NP$. For the MAX-CUT problem ($k = 2$), there are only two partitions and therefore each node either belongs to one partition or the other, which can be represented by assigning to each node i a binary integer $x_i \in \{+1, -1\}$ variable (as in Chapter 6). However, for the MAX-k-CUT problem simply assigning to each node i a k integer variable $y_i \in \{1, \dots, k\}$ does not make the problem easy to work with. Instead, let us analyze the reasoning behind the MAX-CUT problem. We wanted to have two points y_1 and y_2 to represent where each node belongs but in a way that $y_i^2 = 1$ and the angle between y_1 and y_2 is maximized, or in other words, that the points are as different or distant as possible. This is achieved by setting $y_1 = 1$ and $y_2 = -1$.

Therefore, for the MAX-k-CUT problem we want to find k points $y_1, \dots, y_k$ to represent each partition, that satisfy two things: $|y_i|^2 = 1$ for all i and $y_i \cdot y_j$ is minimum (or that the angle is maximized) for all $i \neq j$. The geometric set of points that satisfy the above conditions are the k vertices $V_1, \dots, V_k \in \mathbb{R}^{k-1}$ of the **unitary $k-1$ simplex** Σ_{k-1} centered at the origin. First we will define explicitly the coordinates of the vectors V_i and then prove that they satisfy all the properties. Let

Unitary $k-1$ Simplex

$$(V_i)_j = \begin{cases} -\sqrt{\frac{k-1}{k}} & \text{if } j = i \\ \frac{1}{\sqrt{k(k-1)}} & \text{if } j \neq i \end{cases} \tag{7.2}$$

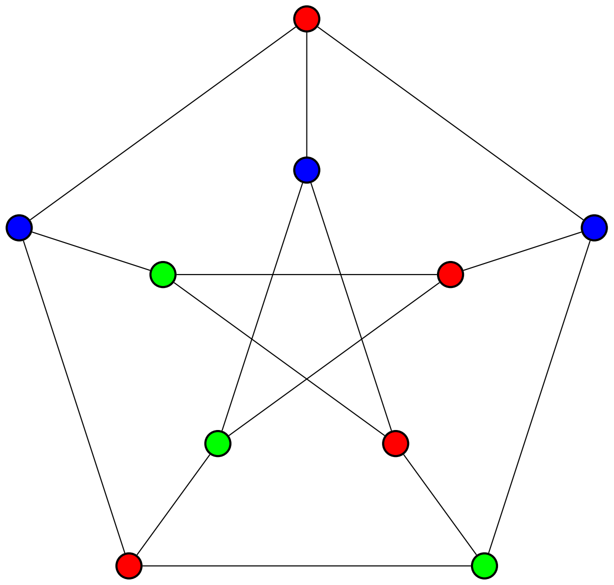


Figure 25: The Petersen graph together with its maximum 3-cut. Nodes in red belong to partition A; nodes in blue to partition B; and nodes in green to partition C. This graph can be colored using 3 colors, which means that the maximum 3-cut is 15, the total number of edges.

where $(V_i)_j$ is the jth entry of vector V_i.

Lemma 5. The k vectors $V_1, \dots, V_k \in \mathbb{R}^{k-1}$ from Equation 7.2 satisfy:

i) $|V_i|^2 = 1$ for all $i = 1, \dots, k$

ii) $V_i \cdot V_j = \frac{-1}{k-1}$ for all $i \neq j$

iii) $\frac{-1}{k-1}$ is the maximum angle of separation possible for k unitary vectors with equal dot product between all distinct pairs.

Proof. i)

$$|V_i|^2 = (k-1)\frac{1}{k(k-1)} + \frac{k-1}{k} = \frac{1}{k} + 1 - \frac{1}{k} = 1$$

ii)

$$V_i \cdot V_j = (k-2) \cdot \frac{1}{k(k-1)} + 2\sqrt{\frac{k-1}{k}} \cdot \frac{-1}{\sqrt{k(k-1)}} = (k-2) \cdot \frac{1}{k(k-1)} - \frac{2}{k}$$

$$= \frac{-k}{k(k-1)} = \frac{-1}{k-1}$$

iii) Assume all dot products $V_i \cdot V_j = \beta$. Then

$$0 \leq |V_1 + V_2 + \cdots + V_k|^2 = k|V_i|^2 + k(k-1)V_1 \cdot V_2 = k + k(k-1)\beta$$

Therefore $\beta \geq -1/(k-1)$

$\square$

Letting $\vec{1} = (1, 1, \ldots, 1)$, note that

$$V_i \cdot \vec{1} = \frac{k-1}{\sqrt{k(k-1)}} - \sqrt{\frac{k-1}{k}} = 0$$

Which means that the set of k points belong in a $k-1$ dimensional hyperplane perpendicular to the vector of all ones $\vec{1}$. Even though the *standard* k-simplex is usually defined in a k dimensional space with base vectors $\hat{e}_i$ as vertices, the k-simplex is actually a $k-1$ dimensional object. This can easily be seen for the familiar case of $k=3$, a triangle with vertices $\hat{e}_1 = (1,0,0)$, $\hat{e}_2 = (0,1,0)$ and $\hat{e}_3 = (0,0,1)$ which is a 2 dimensional object in a 3 dimensional space, or the tetrahedron where $k=4$, which is a 3 dimensional object but can be written in a 4 dimensional space.

We can now generalize the Max-Cut problem by letting the nodes of graph G be represented as one of the vertices V_i of the equilateral $k-1$ simplex Σ_{k-1} from Equation 7.2. This, according to Lemma 5, will give us k unitary vectors that are as separated as possible from each other, i.e., that have maximum possible angle (or minimum dot product) between each other. Note that for the $k=2$ case, we have the Σ_1 simplex, a line segment which end points are $y_1 = 1$ and $y_2 = -1$, which is the Max-Cut integer representation.

Max-k-Cut

Let Σ_{k-1} be the equilateral simplex with vertices $V_1, \ldots, V_k$ as defined in Equation 7.2, and $G = (V, E)$ an undirected weighted graph with n nodes and m edges with weights denoted by w_{ij} for each edge (i,j). Then the Max-k-Cut problem can be represented as the following optimization problem:

$$\begin{aligned} \text{Maximize} \quad & \frac{k-1}{k} \sum_{i<j} w_{ij}(1 - y_i \cdot y_j) \\ \text{subject to:} \quad & y_i \in \{V_1, \ldots, V_k\} \qquad \forall i = 1, \ldots, n \end{aligned} \tag{7.3}$$

7.2 SDP Relaxation

There are several Semidefinite Programming (SDP) problems that are valid relaxations to the Max-k-Cut problem, and most of them will be discussed in section 7.5. In this section we will discuss the relaxation to the optimization problem (7.3).

The fact that the vectors y_i have to be endpoints of Σ_{k-1} in (7.3) imply that $|y_i| = 1$ for all $i = 1, \ldots, k$, and $y_i \cdot y_j = -1/(k-1)$ for all $i \neq j$. It is not enough to simply relax $y_i \in \mathbb{R}^k$ to an unitary vector $v_i \in \mathbb{R}^n$ as in the Max-Cut problem, since it can happen that $v_i \cdot v_j = -1$ instead of $-1/(k-1)$, so we need to add the additional restriction $v_i \cdot v_j \geq -1/(k-1)$, obtaining the following Vector Programming (VP) problem:

$$\begin{aligned} \text{Maximize} \quad & \frac{k-1}{k} \sum_{i<j} w_{ij}(1 - v_i \cdot v_j) & & \\ \text{subject to:} \quad & |v_i| = 1 & \forall i = 1, \ldots, n & \\ & v_i \cdot v_j \geq \frac{-1}{k-1} & \forall i \neq j & \end{aligned} \tag{7.4}$$

This is an instance of VP, however if we want to solve it with a convex optimization tool it is convenient to have it in a SDP form. Let $X \in \mathbb{R}^{n \times n}$ be a symmetric matrix and let $X_{ij} = v_i \cdot v_j$. If we also ask X to be Positive Semidefinite (PSD), then problem (7.4) is equivalent to the following SDP problem:

$$\begin{aligned} \text{Maximize} \quad & \frac{k-1}{k} \sum_{i<j} w_{ij}(1 - X_{ij}) & & \\ \text{subject to:} \quad & X_{ii} = 1 & \forall i = 1, \ldots, n & \\ & X_{ij} \geq \frac{-1}{k-1} & \forall i \neq j & \\ & X \succeq 0 & & \end{aligned} \tag{7.5}$$

where $X \succeq 0$ means that X is PSD. This last constraint is what gives the bridge between VP and SDP. If matrix X is PSD, then there exists $V \in \mathbb{R}^{n \times n}$ such that $X = V^T V$ (Horn, 2012). Therefore, we can convert an instance of (7.5) to (7.4) in polynomial time by doing a Cholesky factorization $X = V^T V$ and setting the i-th column of V as the vector v_i.

In order to solve problem (7.5) numerically, the objective function of an SDP problem must have the form Tr$(C^T X)$ with $C \in \mathbb{R}^{n \times n}$ symmetric. We can achieve this using the **Laplacian matrix** L of the graph G. It is defined as

Laplacian Matrix

$$L = D_W - W$$

Where D_W is the weighted degree matrix ($D =$ diag$(D_1, \ldots, D_n)$ and $D_i = \sum_{j=1}^{n} w_{ij}$) and W the weighted adjacency matrix ($W_{ij} = w_{ij}$ for all $(i, j) \in$ E and 0 otherwise).

Starting from the objective function of (7.5), we see that

$$\begin{aligned}
\frac{k-1}{k}\sum_{i<j}^{n} w_{ij}(1-X_{ij}) &= \frac{k-1}{2k}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}X_{ii} - \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}X_{ij}\right) \\
&= \frac{k-1}{2k}\left(\sum_{i}^{n} X_{ii}\sum_{j}^{n} w_{ij} - \mathsf{Tr}(W^T X)\right) \\
&= \frac{k-1}{2k}\left(\mathsf{Tr}({D_W}^T X) - \mathsf{Tr}(W^T X)\right) \\
&= \frac{k-1}{2k}\mathsf{Tr}({D_W}^T X - W^T X) = \frac{k-1}{2k}\mathsf{Tr}(L^T X)
\end{aligned}$$

where the first equality follows using the fact that one of the restrictions set $X_{ii} = 1$. Therefore, the SDP relaxation problem of Max-k-Cut can be rewritten as:

$$\begin{aligned}
\text{Maximize} \quad & \frac{k-1}{2k}\mathsf{Tr}(L^T X) & \\
\text{subject to:} \quad & X_{ii} = 1 & \forall i = 1, \dots, n \\
& X_{ij} \geq \frac{-1}{k-1} & \forall i \neq j \\
& X \succeq 0 &
\end{aligned} \tag{7.6}$$

where L is the Laplacian matrix of G. As mentioned before, this is just one of the many SDP relaxations that represents the Max-k-Cut problem. SDP problem 7.5 can be seen as the natural generalization of the Max-Cut relaxation of Goemans and Williamson (1995) and it is the procedure of Frieze and Jerrum (1997). This will also provide a straightforward rounding with an approximation guarantee depending on k, which will be discussed in Section 7.3.

7.3 Rounding Methods

Contrary to the Max-Cut problem, there have been several approaches both to relax and round the solution for the Max-k-Cut problem. This subsection will contain some of the rounding methods in the literature and an original approach using clustering similar to the procedure of Rodriguez-Fernandez et al. (2020). Four different rounding methods will be discussed: one using complex SDP for $k = 3$ (Subsection 7.3.1), one using a SDP relaxation related to the Lovász ϑ-function (Subsection 7.3.2) and two based directly from solutions of Equation 7.5 (Subsection 7.3.3 and Subsection 7.3.4).

7.3.1 Goemans Williamson

The rounding method of Goemans and Williamson (2004) is based on a different relaxation to Equation 7.4. They used complex SDP to formulate a relaxation of the Max 2-Lin-Mon-3 problem, which is a more general problem than Max k-Cut.

The Max 2-Lin-Mon-3 problem consists of n integer variables $x_i \in \{0,1,2\}$ and m equations $x_i - x_j \equiv c \mod 3$ or inequations $x_i - x_j \not\equiv c \mod 3$, where $c \in \{0,1,2\}$. Each equation or inequation has a weight w_j assigned and the goal is to maximize the total weight of satisfied equations and inequations.

To model Max-k-Cut, create a variable x_i for each node $i \in V$, and for each edge (i,j) with weight w_{ij}, introduce the inequation $x_i - x_j \not\equiv 0 \mod 3$ with weight w_{ij}. The value of the integer variable x_i determine the partition where node i belongs, therefore $x_i - x_j \not\equiv 0$ means that edge (i,j) is in the cut since the end points are in different partitions.

The complex integer formulation for Max 2-Lin-Mon-3 is the following. Let $\mathbf{R}_3 \subset \mathbb{C}$ be the set that contains the three cubic roots of 1, and let $\omega = e^{i2\pi/3}$. The three cubic roots of 1 are $\mathbf{R}_3 = \{\omega^0, \omega^1, \omega^2\}$, and have the property that $\omega^i \cdot \omega^j = \omega^{i-j} \in \mathbf{R}_3$, where $\cdot$ is the dot product in $\mathbb{C}$. Introduce a variable $y_i \in \mathbf{R}_3$ for each x_i, such that $y_i = \omega^{x_i}$. Then, the contribution of the equation $x_i - x_j \equiv c \mod 3$ is

$$w_{ij}\left(\frac{1}{3} + \frac{1}{3}\omega^{-c} x_i \cdot x_j + \frac{1}{3}\omega^{c} x_j \cdot x_i\right)$$

and the contribution of the inequation $x_i - x_j \not\equiv c \mod 3$ is

$$w_{ij}\left(\frac{2}{3} - \frac{1}{3}\omega^{-c} x_i \cdot x_j - \frac{1}{3}\omega^{c} x_j \cdot x_i\right).$$

Note that with setting $c = 0$ we have the Max-k-Cut problem.

For the relaxation, the variable y_i can be inside of the triangle defined by $\mathbf{R}_3$, instead of only being the endpoints. Therefore the constraints for the y_i are $y_i \cdot y_i = 1$, and $\alpha y_i \cdot y_j + \bar{\alpha} y_j \cdot y_i \geq -1$ for all i,j and every $\alpha \in \mathbf{R}_3$. Letting Y be the Gram matrix $Y_{i,j} = y_i \cdot y_j$, we can formulate Max 2-Lin-Mon-3 as a complex SDP problem:

$$\begin{array}{rlr} \text{Maximize} & C \cdot Y & \\ \text{subject to:} & Y_{ii} = 1 & \forall i = 1, \dots, n \\ & \alpha Y_{ij} + \bar{\alpha} Y_{ji} \geq -1 & \forall i \neq j, \alpha \in \mathbf{R}_3 \\ & Y \succeq 0 & \end{array} \tag{7.7}$$

where $C \in \mathbf{H}_n$ is a Hermitian matrix that combines all the contributions of equations and inequations. Once we have the solution matrix Y, we can obtain the vectors y_i by doing a Cholesky decomposition $Y = LL^*$ and setting y_i as the i-th column of L. Then, generate a random vector $r \in \mathbb{C}$ and set

$$x_i = \begin{cases} 0 & \text{if } \arg(r \cdot y_i) \in [0, 2\pi/3) \\ 1 & \text{if } \arg(r \cdot y_i) \in [2\pi/3, 4\pi/3) \\ 2 & \text{if } \arg(r \cdot y_i) \in [4\pi/3, 2\pi) \end{cases}$$

where arg(z) is the argument of complex number z. This rounding gives an performance guarantee of $\alpha = 0.836$, however it is only for the case where $k = 3$.

7.3.2 De Klerk, Pasechnik and Warners

The procedure of Klerk, Pasechnik, and Warners (2004) uses the SDP formulation of the Lovász ϑ function instead of the direct SDP approach of the $\textsc{Max-}k\textsc{-Cut}$. Additionally, the main focus of the analysis is on graph coloring, which is a special case of the unweighted $\textsc{Max-}k\textsc{-Cut}$. Graph coloring assigns colors to each node such that the number of defective edges (edges with same color endpoints) is minimized. This is the equivalent of maximizing the number of edges with end nodes in different sets, which is the goal of the unweighted $\textsc{Max-}k\textsc{-Cut}$.

In Klerk, Pasechnik, and Warners (2004), first they formulate graph coloring as a boolean quadratic feasibility problem of size $kn \times kn$, then they relax it to an instance of SDP of size $(kn + 1) \times (kn + 1)$, show that this problem is equivalent to the Lovász ϑ representation of Karger, Motwani, and Sudan (1998) and finally use the Lovász ϑ solution to do the rounding.

For the boolean quadratic feasibility problem, let the binary $\{-1, 1\}$ variables x_i^j be:

$$x_i^j = \begin{cases} 1 & \text{if node } i \text{ has color } j \\ -1 & \text{otherwise} \end{cases}$$

with $i = 1, \ldots, n$ and $j = 1, \ldots, k$. Then, the feasibility problem for graph k-coloring is:

$$\begin{aligned} \text{Find} \quad & \{x_i^j \mid \ i = 1, \ldots, n \ \wedge \ j = 1, \ldots, k\} \\ \text{such that:} \quad & \left(x_i^p + x_j^p + 1\right)^2 = 1 \qquad \text{if } (i, j) \in E \\ & \sum_{j=1}^{k} x_i^j = -k + 2 \qquad \forall i = 1, \ldots, n \end{aligned} \tag{7.8}$$

where the first constraint forces adjacent vertices to have different colors and the second constraint makes each node to have only one color. The SDP relaxation of Equation 7.8 involves a matrix $X \in \mathbb{R}^{kn+1 \times kn+1}$ which includes all the possible products of variables x_i^j, and can be seen in detail in Klerk, Pasechnik, and Warners (2004). They show that this problem is equivalent to the following formulation of the Lovász ϑ:

$$\begin{aligned} & \vartheta(\bar{G}) = \min_{U,k} k \\ \text{subject to:} \quad & U_{i,j} = \frac{-1}{k-1} \qquad \text{if } (i, j) \in E \\ & U_{ii} = 1 \qquad \forall i = 1, \ldots, n \\ & U \succeq 0, \ k \geq 2. \end{aligned} \tag{7.9}$$

Algorithm 10: Randomized Rounding of Klerk, Pasechnik, and Warners (2004)

```
1: Solve (7.9), obtaining an optimal Positive Semidefinite (PSD) matrix $U$.
2: Let
       $$Y = U \otimes \frac{k}{k-1}\left(I_k - \frac{1}{k}e_k e_k^T\right)$$
   where $\otimes$ is the Kronecker product, $I_k$ is the identity matrix of size $k \times k$ and
   $e_k$ is the all ones vector of size $k$.
3: Do the Cholesky factorization of $Y = V^T V$ and the columns of $V$ as the
   vectors $v_1^1, v_1^2, \ldots, v_1^k, \ldots, v_n^k$.
4: Pick a random unitary vector $r \in \mathbb{R}^{kn}$ from the uniform distribution.
5: Set $x_i^j = 1$ if $r \cdot v_i^j = \max_{l=1,\ldots,k}(r \cdot v_i^l)$, or set $x_i^j = -1$ otherwise.
```

where $\bar{G}$ is the complementary graph of G. For more details of the meaning of the Lovász ϑ and its relation to the coloring problem see Section 7.5. The SDP problem 7.9 is used for rounding the solution and the procedure is shown in Algorithm 10.

Note that the problem of 7.8 is a feasibility problem, therefore for smaller k integers the rounding may not be possible, in particular, for k values smaller than the optimal $\vartheta(\bar{G})$.

This rounding method works for unweighted graphs, possibly not for small k, and the rounding method lift the dimensions from $n \times n$ to $kn \times kn$. However the analysis of the performance guarantee allows to get larger values than the other methods which work with maximization problems instead of feasibility, such as the rounding of Frieze and Jerrum (1997). Additionally Klerk, Pasechnik, and Warners (2004) proved the equivalence of their method to the rounding of Frieze and Jerrum (1997) and obtained the performance bounds of 0.836 for $k = 3$ and 0.8575 for $k = 4$, and a way to calculate the value of the performance bound for other k values.

7.3.3 Frieze and Jerrum

This is the natural generalization to the procedure of Max-Cut and it is described in Frieze and Jerrum (1997). Once the vectors $v_1, \ldots, v_k$ solution of Equation 7.4 are obtained, generate k random unitary vectors which will represent the k partitions. To round the vectors to an instance of Max-k-Cut, assign node j to partition A_i if the vector r_i is the closest among all the r vectors to v_j, i.e., if $v_j \cdot r_i \geq v_j \cdot r_{i'}$ for all $i \neq i'$. This algorithm is summarized in Algorithm 11.

Let A^* denote the optimum partition and A_{JF} the partition obtained with Algorithm 11. Then, Frieze and Jerrum (1997) proved that $\mathbf{E}(w(A_{JF})) \geq \alpha_k w(A^*)$, where the performance guarantees α_k satisfy:

i) $\alpha_k > 1 - k^{-1}$

ii) $\alpha_k - (1 - k^{-1}) \sim 2k^{-2} \log k$

Algorithm 11: Randomized Rounding of Frieze and Jerrum (1997)

```
1: Solve (7.4), obtaining an optimal set of vectors $v_1, \dots, v_n$
2: Pick $k$ random vectors $r_i = (r_{i,1}, \dots, r_{i,n})$ by drawing each component $r_{i,j}$
   independently from $N(0,1)$, the normal distribution with mean 0 and variance
   1.
3: Set $A_i = \{j | v_j \cdot r_i \geq v_j \cdot r_{i'} \quad \forall i' \neq i\}$. Break ties arbitrarily.
```

Algorithm 12: Rounding using k-Clustering

```
1: Solve (7.4), obtaining an optimal set of vectors $v_1, \dots, v_n$
2: Do clustering to the set of vectors $v_1, \dots, v_n$ to obtain $k$ clusters $C_1, \dots, C_k$
3: Set $A_i = \{i | v_i \in C_i\}$.
```

iii) $\alpha_2 \geq 0.878567, \alpha_3 \geq 0.800217, \alpha_4 \geq 0.850304, \alpha_5 \geq 0.874243, \alpha_{10} \geq 0.926642, \alpha_{100} \geq 0.990625$

where $\sim$ indicates that the ratio of the two expressions tends to 1 in the limit where $k \to \infty$.

7.3.4 Clustering

Let $v_1, \dots, v_n$ be the solution vectors of Equation 7.4. If the weight w_{ij} is large, the SDP problem will try to put vectors v_i and v_j as far as possible, i.e., making $(1 - v_i \cdot v_j)$ as large as possible in order to add that large weight value to the objective function. This motivates the use of clustering for the rounding of the vectors v_i instead of a randomized approach, so that we can take better advantage of the geometrical properties of these solution vectors. The rounding algorithm, described in Algorithm 12, consists on clustering the set of vectors $v_i, \dots, v_n$ into k clusters $C_1, C_2, \dots, C_k$, and set the partitions $A_i = \{i | v_i \in C_i\}$.

As we will see in Section 7.4, using clustering gives better solutions than the randomized rounding, however a performance guarantee is yet to be found.

7.4 Numerical Experiments

Test Graphs

In order to test the performance of Algorithm 12 and Algorithm 11, four types of **test graphs** as in Goemans and Williamson (1995) were used:

i) Random Graph Type A. Each possible edge is included with probability 1/2. All edges have weight 1 (Erdős–Rényi of type $G(n,p)$ with $p = 1/2$).

ii) Random Graph Type B. Each possible edge is included with probability 1/2. Edge weights are integers taken randomly from the interval [-50,50] uniformly.

iii) Random Graph Type C. Each possible edge is included with probability $10/n$ where n is the number of nodes. This way each node has constant expected degree. All edges have weight 1.

iv) Random Graph Type D. Edge (i, j) is included with probability 0.1 if $i \leq n/2$ and $j > n/2$ or with probability 0.05 otherwise. This favors large 2-cuts between the first $n/2$ nodes and the remaining $n/2$ ones. All edges have weight 1.

For all the test graphs, 50 instances of 50 nodes, 20 instances of 100 nodes and 5 instances of 200 nodes were used, having a total of 300 graphs to use as **benchmark**. For notation, $X_{m,i}$ will be used to refer a graph of type $X \in \{A, B, C, D\}$, with m nodes and instance number i. For example, the 30th graph of type C, with 100 nodes is denoted as $C_{100,30}$. A table summarizing the Random Graphs Benchmark and its properties is shown in Table 1.

Random Graphs Benchmark

Table 1: Random Graphs Benchmark. For these graphs, the number of nodes is fixed, the number of edges are in expected value and ρ is the graph density.

Instance	Nodes n	Edges ($\mathbb{E}$)	$\mathbb{E}[m]$	Density ρ ($\mathbb{E}$)	$\mathbb{E}[\rho]$	Weight	Type of Graph
A_{50}	50	$\frac{n(n-1)}{4}$	612.5	$\frac{1}{2}$	50%	1	Erdős–Rényi
A_{100}	100		2475		50%		type $G(n,p)$
A_{200}	200		9950		50%		with $p = 0.5$
B_{50}	50	$\frac{n(n-1)}{4}$	612.5	$\frac{1}{2}$	50%	[-50,50]	Erdős–Rényi
B_{100}	100		2475		50%		type $G(n,p)$
B_{200}	200		9950		50%		with $p = 0.5$
C_{50}	50	$\frac{10n}{2}$	250	$\frac{10}{n-1}$	20.41%	1	(Expected)
C_{100}	100		500		10.10%		Constant
C_{200}	200		1000		5.03%		Degree of 10
D_{50}	50	$\frac{n(3n-2)}{80}$	92.5	$\frac{3n-2}{40(n-1)}$	7.55%	1	Favors 2-Cut:
D_{100}	100		372.5		7.53%		$A = [1, n/2]$
D_{200}	200		1495		7.51%		$B = [n/2+1, n]$

In order to obtain the vectors $v_1, \ldots, v_n$ from Equation 7.4, first we used the function *SemidefiniteOptimization* of Mathematica to solve Equation 7.6 obtaining the matrix $X \succeq 0$, then use a Cholesky decomposition to factorize $X = V^T V$, and finally, set v_i to be the i-th column of V.

We used k-means (MacQueen, 1967) adapted to the unitary sphere as the clustering method for Algorithm 12. The centroids that initialize k-means are k random unitary vectors: each centroid entry is independently drawn from a normal distribution with mean 0 and variance 1, and then properly scaled to make them unitary.

Finally, we use both Algorithm 11 and Algorithm 12 to round the set of vectors $v_1, \ldots, v_n$ and the results are discussed in Subsection 7.4.1.

7.4.1 Results

In this section, we will refer to Algorithm 11 as *RandRound* and to Algorithm 12 as *K-Means*. Let OPT be the optimal value of Equation 7.4 and f_* the cut found by any of the rounding algorithms. Then, following the definition of Goemans and Williamson, 1995,the **Average Integer Gap** AIG is the quotient

Average Integer Gap

$$AIG = \frac{f_*}{OPT} \tag{7.10}$$

and for graphs with negative weights, we need to use the alternative quotient defined in Section 6.3.1

$$AIG = \frac{f_* - W_-}{OPT - W_-}$$

where $W_- = \sum_{i<j} w_{ij}^-$ is the sum of all the negative weights with the function $x^- = min(0, x)$. Additionally, we define the **Cut Gap** CG as the quotient of the cut value found by any of the rounding algorithms f_* and the largest possible cut, i.e., the sum of all weights:

Cut Gap

$$CG = \frac{f_*}{\sum_{i<j} w_{ij}}$$

and for graphs with negative weights, we need to use the alternative quotient

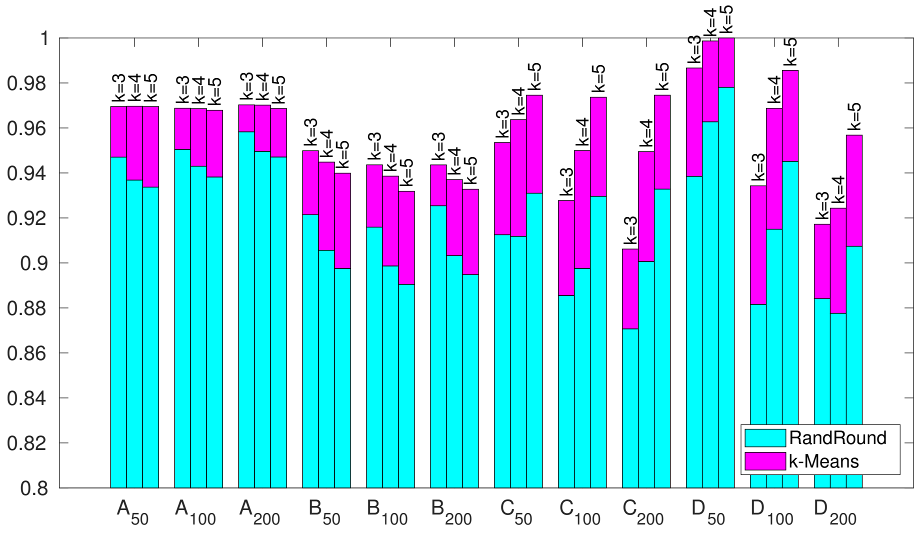
$$CG = \frac{f_* - W_-}{W_+ - W_-}$$

where $W_+ = \sum_{i<j} w_{ij}^+$ is the sum of all the positive weights with the function $x^+ = max(0, x)$. We use the sum of all weights since it is the highest possible value a cut can have.

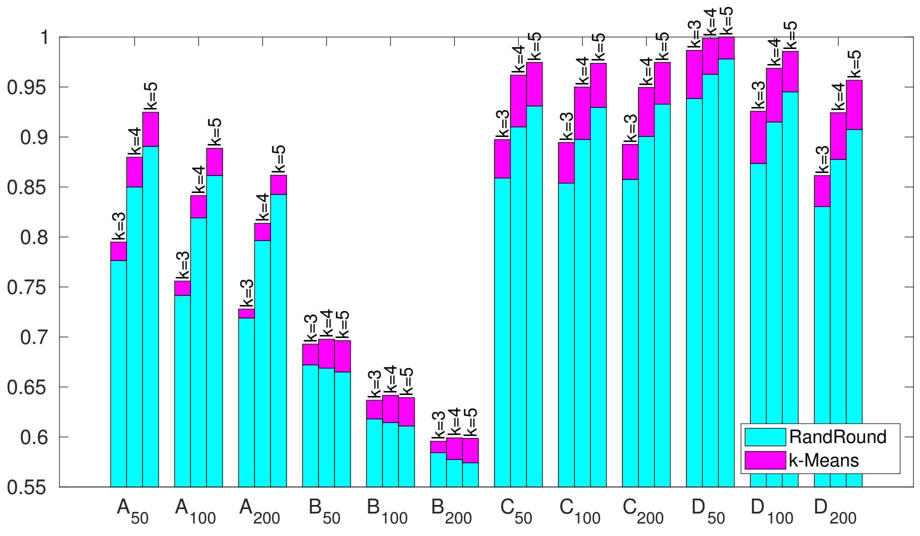
A comparison of the average integer gap and the cut gap obtained for the Random Graphs benchmark is shown in Figure 26. Looking at graphs with positive weights, i.e., type A, C and D, we see in Figure 26b that the cut value increase with larger k as expected. For the Erdős–Rényi graphs A with high density, the cut gaps are lower than for graphs C and D with lower density. Moreover, graphs of type A have no additional structure compared with graphs C (with constant expected degree) and graphs D (favors a specific two cut), which may make those instances *easier* to solve. For the type B graphs, the negative weights have an impact and the cut gaps are the lowest among all of the graph types. Even having $k = n$ will not make the problem *easier* since isolating each node as a partition will not give the optimal cut as in the case for graphs with only positive weights. However looking at the integer gap, we see that the solutions are still very close to the relaxed optimum OPT, therefore the cut gap is not a good measure for graphs with negative edges.

In Figure 26a the average integer gap obtained for each algorithm and some values of k is depicted. The information obtained from the average integer gap as k increases, can be interpreted as the *difficulty* of the rounding. Let us analyze each type of graph separately:

For type A graphs, the average integer gap is reaching almost 0.97 with clustering as the rounding method and independently of the value of k. This suggests that, even though the cut value obtained increases, the effectiveness of the rounding remains the same and around 0.97. As for the randomized rounding, the higher the k, the more *difficult* is to find better cuts, since the average integer gap is lower.



(a) Average Integer Gap values.



(b) Cut Gap values.

Figure 26: Values of the average integer gap (26a) and the cut gap (26b) obtained by both Algorithm 11 (RandRound in cyan) and Algorithm 12 (k-Means in magenta) for the random graphs A, B, C and D, and $k = 3$, $k = 4$ and $k = 5$.

Not to be confused, the cut values found are higher with higher k as seen in the cut gap plots; a lower average integer gap means that the relaxed solution can not be rounded as effectively with larger k.

For type B graphs, the graphs with negative weights, we find that increasing k decreases the integer gap, however the value of the cut remains approximately constant (as seen in the cut gap). This suggests that the relaxation for larger k is *harder*.

For type C graphs, with expected constant degree of 10, we observe that a larger k gives larger average integer gap values, and for $k = 5$ already a value of almost 0.98 is reached (for clustering) and the same one independently of the number of nodes. If we were to continue until $k = 10$, the average integer gap should be 1 for almost all instances (using clustering as rounding), since the expected degree is 10 and the density is low, therefore avoiding lots of cliques of size 10 (the worst possible case). The behavior for randomized rounding is the same as in clustering, but around 0.04 lower. Lastly, for $k = 3$, increasing the number of nodes decreases the integer gap, but the value of the cut is maintained regardless of the size, which is to be expected for a graph with expected constant degree.

For type D graphs, and in particular for graphs with 50 nodes, clustering finds solutions with almost the same value as the relaxed solution OPT, and for $k = 5$ the average integer gap is closed. However, as the number of nodes increases, the average integer gap decreases, both for randomized rounding and clustering, and for $k = 5$ the average integer gap is not closed anymore.

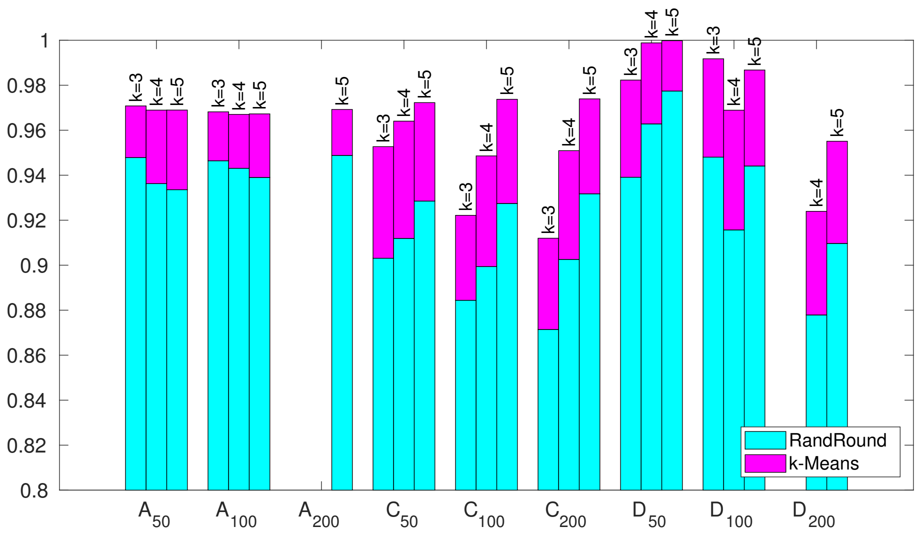
Finally, we observe that both for the cut values (as seen in the cut gap) and the integer gap, and for all cases, *clustering yields better results than the randomized rounding algorithm*.

7.4.2 Correcting Numerical Issues

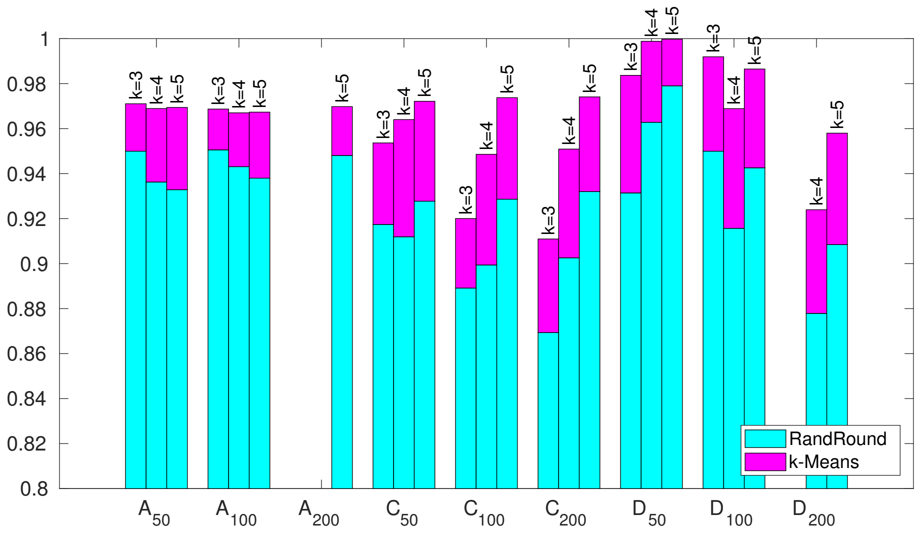
Matrix X obtained by solving Equation 7.6 with Mathematica can have numerical precision problems and have small ($\approx 10^{-9}$) negative eigenvalues, although the function *SemidefiniteOptimization* is supposed to return a Positive Semidefinite (PSD) matrix. Therefore a Cholesky decomposition in those cases is not possible. However, since we know that X is supposed to be PSD, we can slightly modify it to make it PSD. We follow the procedure of Higham (1988) to compute $\bar{X}$, the nearest PSD matrix to X, and do the Cholesky decomposition of $\bar{X}$ instead of X.

In order to see how using $\bar{X}$ may affect the results, we solved using both X and $\bar{X}$ for all the instances where the original matrix X was PSD, which are: 8 graphs of type A_{50}, 1 A_{100}, 5 C_{50}, 2 C_{100}, 1 C_{200}, 37 D_{50} and 6 D_{100} for $k = 3$; 23 A_{50}, 8 A_{100}, 46 C_{50}, 18 C_{100}, 4 C_{200}, 50 D_{50}, 18 D_{100} and 5 D_{200} for $k = 4$; and 34 A_{50}, 14 A_{100}, 2 A_{200}, 50 C_{50}, 20 C_{100}, 5 C_{200}, 50 D_{50}, 20 D_{100} and 5 of type D_{200} for $k = 5$.

This comparison is shown in Figure 27, and from the figure we see that using the corrected matrix $\bar{X}$ the overall behavior is maintained except for the case of C_{50} with $k = 3$, where the average integer gap increased. Therefore using $\bar{X}$ is justified.



(a) Values of the average integer gap for the original matrix X.



(b) Values of the average integer gap for the corrected matrix $\bar{X}$.

Figure 27: Values of the average integer gap obtained by RandRound and K-Means for the random graphs A,C and D where the original matrix X was PSD, for $k = 3$, $k = 4$ and $k = 5$, and for both X and $\bar{X}$.

7.5 Relation with k-Coloring and the Lovász ϑ Function

In this section we will discuss the relation between the Max-k-Cut problem, the k-coloring problem and the Lovász ϑ function.

The Lovász ϑ can be computed efficiently and is "sandwiched" by two NP-hard graph numbers, the chromatic $\chi(G)$ and the clique number $\omega(G)$. Lovasz (1979) proved the so called "sandwich" theorem:

$$\omega(G) \leq \vartheta(\bar{G}) \leq \chi(G)$$

where $\bar{G}$ is the complementary graph of G. We can already see that it is closely related to the graph coloring problem.

In Karger, Motwani, and Sudan (1998), they define *vector k-coloring*, a relaxation of the coloring problem. The idea is to represent each node as a vector, and make the vectors of adjacent nodes to be as different as possible, i.e., as far away from each other as possible. For a graph $G = (V, E)$, a **vector k-coloring** is a set of n unitary vectors v_i such that satisfy:

Vector k-Coloring

$$v_i \cdot v_j \leq \frac{-1}{k-1} \quad \forall (i,j) \in E, \tag{7.11}$$

and the set of vectors is a *strict* vector k-coloring if it satisfies the equality. The vector chromatic number and the strict vector chromatic number are the smaller values of k such that a vector k-coloring and a strict vector k-coloring exists. The strict vector chromatic number can be formulated as a Semidefinite Programming (SDP) problem, and the dual of this problem is one of the formulations for the Lovász ϑ (Karger, Motwani, and Sudan, 1998).

All three problems have similar relaxed formulations related to the k vertices of the $k-1$ simplex Σ_{k-1}. A difference in the nature of the problems is that for k-coloring we are interested in finding the smallest k such that a k-coloring of the graph exists, meanwhile for Max-k-Cut, k can be smaller than the chromatic number (if we are interested in fixed k). However, one can ask for the smallest k such that an unweighted graph has a perfect k-cut, and that will be equivalent to finding the smallest k for which a coloring exists.

Part III

Conclusions

CHAPTER 8

Conclusions and Future Work

As seen from the results of Chapter 6, clustering improves the results of a randomized rounding for the MAX-CUT problem, however the proof of a performance guarantee has yet to be found. Moreover, if the unique games conjecture (Khot, 2002) is true, the performance guarantee for the MAX-CUT problem cannot be better than $\alpha \approx 0.878$ and in general cannot be better than $16/17$ unless P=NP. For application purposes using clustering is recommended since it obtains larger cut values. This is a clear example that using the structure of the problem helps design algorithms that find better solutions possibly at the cost of a lower (or none at all) approximation guarantee, such as Garcia-Diaz et al. (2017) for the k-center problem.

Using Spectral Clustering (SC) as the clustering method for the MAX-CUT problem, the results obtained although competitive, were on average lower than using k-means. This is due to the fact that the similarity measures utilized were not able to adequately separate the vectors as seen in Figure 18 and Figure 19. There might be another similarity measure that separates better the vectors and therefore give better results than k-means, however it has yet to be found.

The spectrum analysis of the Gram Matrix of the solution vectors v_i, obtained from the relaxed problem for MAX-CUT (Equation 6.2), give an insight into the difficulty of the problem and it seems to be directly related to the number of non-zero eigenvectors. From the Principal Component Analysis (PCA) shown in Section 6.4 we can observe that even though the relaxation *enlarges* the original 1-dimensional integer problem into an n-dimensional space (n being the number of nodes in the graph), the Vector Programming (VP) problem utilizes as few dimensions as possible. For instance, the vectors v_i are 1-dimensional and integer for the simple planar graphs (B1-B4) of the B-set benchmark, i.e., all entries are 0 except for the first, for which the values are ± 1. Adding a cycle to the trees (graph B5) forces the VP problem to use 2 dimensions, but still manages to perfectly separate the vectors. Adding cliques (graphs B6 and B7) raises the number of dimensions utilized even more, and the points are not perfectly separable anymore.

If the topology of a graph is not known in advance, a spectral analysis of its Laplacian matrix can be computed. Using this information, the graph can be classified as similar to a toroidal, planar or random graph by comparing the spectrum to the ones in Figure 20. Then, for this instance, select the clustering method that performs the best for that particular graph topology.

Naturally, RandomizedMinMax (RMM) obtained higher cut values since it is an iterative process consisting of a clustering rounding step followed by a Local Search (LS) until a *local optima* of this algorithm is found, i.e., until the solution after clustering and LS is the same as in the previous iteration. As seen in the trajectories on the cut vs inertia space (Figure 23), there is a large portion of the space which was not explored and most of the trajectories are in a reduced area. This suggests to design other guided heuristics that explore a larger region of the space.

Using clustering instead of randomized rounding improves the results for the MAX-k-CUT problem as well, as seen in the results from Chapter 7. The benchmark utilized was originally designed for the MAX-CUT problem (Goemans and Williamson, 1995), therefore additional types of graphs can be designed to test and compare the algorithms. For example, in the Random Graph Benchmark (Table 1), graphs of type D were designed to favor a bipartition for the MAX-CUT problem, therefore designing additional graphs that favor a k partition for some values of k can give additional information about the problem. Additionally, the data strongly suggests that the approximation guarantee of the randomized rounding is maintained, however a proof has yet to be found.

The results shown in Figure 26 suggest how to expand and continue the analysis of the $\textsc{Max-}k\textsc{-Cut}$ problem. For instance, for graphs of type A (Erdős-Rényi graphs), the average integer gap (Equation 7.10) of clustering have similar values independent of the value of k, meanwhile for the randomized rounding, the larger the k the more *difficult* it seems to obtain a good solution parting from the relaxation. This means that, although the cut values obtained in the relaxed problem increases with larger k, the rounding increases in the same manner when using clustering, but slower for the randomized rounding. A natural continuation for this analysis will be testing with larger k values. For types C and D, both clustering and randomized rounding obtained better rounded solutions as k increases, in contrast to type B graphs where the opposite behavior is observed. Graphs of type B allow edges to have negative weights, which can be the main reason for being the only types of graph for which larger k values obtain worst average integer gap values for the clustering rounding.

The software used to solve the Semidefinite Programming (SDP) relaxation of $\textsc{Max-}k\textsc{-Cut}$ stops working for instances of $n \geq 200$ nodes. Since the relaxation is an SDP problem, an efficient interior point method specifically programmed for the $\textsc{Max-}k\textsc{-Cut}$ that solves larger instances can be designed. With this, more benchmarks, such as the G-set used for the $\textsc{Max-Cut}$ problem can be solved.

As discussed in Section 7.3, there are several rounding methods for the $\textsc{Max-}k\textsc{-Cut}$ problem, however using a direct relaxation of the integer formulation of the problem (as used for the results of clustering as rounding) allows us to find a solution for any integer $k \geq 2$, meanwhile using the formulation of the Lovász ϑ-function allows only for $k \geq \vartheta(\bar{G})$, i.e., greater or equal than the solution of Equation 7.9.

For the Sherrington-Kirkpatrick, the p-spin and the NK models of fitness landscapes, an attempt was made to find a relation between the covariance matrix and the number of local optima. However for these models and neighborhoods used, a relation has yet to be found. The procedure in order to find this relation in future work can be summarized as follows: First, find simple models or neighborhoods for which both the covariance matrix and the number of local optima is known, then find a relation in these simple cases and lastly search if there is a general relation for more complex problems which the number of local optima and the covariance matrix still can be found.

Appendices

APPENDIX A
Spectral Clustering (SC) Results

Table 2: Cut values obtained with Spectral Clustering using K-MeansDet. The last column is the result obtained in Rodriguez-Fernandez et al. (2020). For each graph, i.e., for each row, a white-green color scale was used. A white color indicate higher cut values while a green color indicate lower cut values. Small cut values were replaced with a N/A to make the white-green color differences larger.

Graph	$\beta = 0$	$\beta = 0.5$	$\beta = 0.25$	$\sigma = 0.5$	$\sigma = 0.25$	$\sigma = 1$	$\sigma = 10$	kMeansDet
G1	11414	11427	11414	11408	N/A	11417	11414	11470
G2	11310	11315	11310	11286	11245	11284	11310	11365
G3	11393	11402	11397	11398	N/A	11398	11398	11408
G4	11353	11350	11353	11356	53	11356	11353	11426
G5	11432	11424	11432	11455	45	11432	11432	11379
G6	1936	1941	1936	1931	1912	1934	1934	1923
G7	1714	1719	1713	1709	N/A	1732	1714	1748
G8	1758	1751	1759	1746	N/A	1759	1767	1776
G9	1819	1806	1816	1787	N/A	1815	1819	1797
G10	1765	1756	1765	1765	N/A	1759	1753	1748
G11	512	510	510	510	430	514	512	518
G12	516	514	516	512	512	516	516	524
G13	530	534	530	530	528	532	532	548
G14	2951	2952	2951	2918	N/A	2941	2939	2953
G15	2965	2965	2965	2955	2938	2965	2965	2958
G16	2946	2951	2947	2951	N/A	2951	2946	2950
G17	2959	2961	2959	2953	N/A	2958	2960	2947
G18	897	895	897	896	N/A	897	897	883
G19	792	794	792	772	N/A	791	793	812
G20	844	839	844	851	838	842	844	863
G21	812	814	816	817	810	810	812	826
G22	12935	12926	12935	12942	N/A	12935	12935	12893
G23	12987	12991	12987	12985	N/A	12991	12987	12924
G24	12983	12976	12983	12978	12927	12983	12983	12904
G25	12870	12870	12868	12876	N/A	12866	12868	12922
G26	12867	12877	12867	12889	N/A	12866	12867	12886
G27	2892	2890	2892	2866	N/A	2890	2892	2891
G28	2852	2836	2852	2850	N/A	2847	2848	2834
G29	2973	2972	2973	2975	N/A	2971	2973	2946
G30	3007	3001	3007	3002	N/A	3003	3007	2979
G31	2884	2880	2881	2842	N/A	2875	2880	2867
G32	1266	1262	1266	1242	1204	1258	1266	1294
G33	1222	1222	1222	1226	N/A	1220	1222	1266
G34	1244	1248	1244	1254	1254	1246	1244	1268
G35	7412	7407	7412	7401	N/A	7408	7412	7424
G36	7383	7375	7383	7333	N/A	7382	7383	7399
G37	7425	7424	7424	7439	N/A	7430	7428	7437
G38	7358	7362	7359	7342	7330	7355	7360	7411
G39	2136	2134	2134	2117	N/A	2136	2140	2176
G40	2153	2138	2155	2150	N/A	2156	2154	2134
G41	2116	2114	2116	2102	N/A	2111	2114	2149
G42	2171	2162	2171	2162	2118	2176	2171	2188
G43	6510	6505	6513	6515	N/A	6513	6510	6442
G44	6455	6458	6455	6414	N/A	6461	6455	6461
G45	6449	6426	6449	6448	N/A	6447	6449	6448
G46	6423	6420	6422	6415	N/A	6419	6423	6425
G47	6456	6452	6460	6442	N/A	6457	6456	6480
G48	6000	6000	6000	6000	6000	6000	6000	6000
G49	6000	6000	6000	6000	6000	6000	6000	6000
G50	5880	5880	5880	5880	5880	5880	5880	5880
G51	3726	3737	3726	3714	N/A	3725	3726	3711
G52	3711	3713	3711	3705	N/A	3710	3711	3721
G53	3695	3694	3695	3694	N/A	3688	3692	3715
G54	3691	3689	3691	3600	N/A	3668	3691	3712
G57	3152	3152	3150	3152	N/A	3146	3148	3158
G58	18557	18565	18557	18516	N/A	18557	18560	18528
G59	5312	5310	5312	5300	N/A	5313	5309	5350
G62	4378	4384	4380	4378	N/A	4364	4372	4394
G63	25988	25994	25988	25988	N/A	25983	25984	25933
G64	7748	7742	7751	7747	N/A	7742	7748	7698
G65	4970	4962	4972	4966	N/A	4970	4972	5026
G66	5672	5668	5670	5646	N/A	5666	5672	5776
G67	6232	6238	6230	6236	N/A	6226	6228	6282
G72	6248	6252	6246	6236	N/A	6238	6250	6342

Table 3: Cut values obtained with Spectral Clustering using K-MeansNM. The last column is the result obtained in Rodriguez-Fernandez et al. (2020). For each graph, i.e., for each row, a white-green color scale was used. A white color indicate higher cut values while a green color indicate lower cut values. Small cut values were replaced with a N/A to make the white-green color differences larger.

Graph	$\beta = 0$	$\beta = 0.5$	$\beta = 0.25$	$\sigma = 0.5$	$\sigma = 0.25$	$\sigma = 1$	$\sigma = 10$	kMeansNM
G1	11414	11427	11414	11419	N/A	11417	11414	11470
G2	11310	11331	11310	11286	N/A	11297	11310	11365
G3	11398	11402	11398	11398	N/A	11398	11398	11408
G4	11353	11350	11353	11356	N/A	11356	11353	11426
G5	11432	11424	11432	11455	N/A	11432	11432	11379
G6	1936	1942	1936	1931	1912	1934	1934	1923
G7	1722	1728	1722	1709	N/A	1732	1722	1748
G8	1758	1751	1759	1746	N/A	1759	1767	1776
G9	1819	1806	1816	1787	N/A	1815	1819	1797
G10	1765	1758	1765	1765	1680	1759	1759	1748
G11	512	510	510	510	N/A	514	512	518
G12	516	514	516	512	512	516	516	524
G13	532	534	532	530	534	532	532	548
G14	2956	2958	2954	2918	2901	2943	2952	2953
G15	2965	2965	2965	2955	2940	2967	2965	2958
G16	2946	2951	2947	2951	N/A	2951	2946	2950
G17	2960	2961	2960	2953	N/A	2958	2960	2947
G18	897	895	897	896	850	897	897	883
G19	792	794	792	776	799	793	793	812
G20	844	840	844	854	838	842	844	863
G21	812	814	816	817	817	810	812	826
G22	12935	12931	12935	12942	N/A	12935	12935	12893
G23	12987	12991	12987	12985	N/A	12991	12987	12924
G24	12983	12976	12983	12978	12927	12983	12983	12904
G25	12870	12878	12868	12881	N/A	12866	12868	12922
G26	12873	12877	12873	12889	N/A	12866	12873	12886
G27	2892	2890	2892	2868	N/A	2891	2892	2891
G28	2852	2840	2852	2850	2636	2847	2848	2834
G29	2978	2972	2979	2979	2903	2980	2981	2946
G30	3007	3001	3007	3002	N/A	3003	3007	2979
G31	2884	2880	2881	2842	N/A	2875	2880	2867
G32	1266	1262	1266	1244	1204	1258	1266	1294
G33	1222	1222	1222	1226	716	1220	1222	1266
G34	1246	1248	1248	1254	1254	1246	1244	1268
G35	7412	7409	7412	7401	N/A	7411	7412	7424
G36	7383	7375	7383	7333	7316	7382	7383	7399
G37	7441	7443	7440	7439	N/A	7439	7442	7437
G38	7359	7364	7359	7342	7330	7356	7360	7411
G39	2136	2134	2134	2117	2046	2136	2140	2176
G40	2153	2138	2155	2150	N/A	2156	2154	2134
G41	2118	2114	2118	2102	N/A	2113	2114	2149
G42	2171	2165	2171	2169	2118	2176	2171	2188
G43	6510	6505	6513	6515	N/A	6513	6510	6442
G44	6456	6465	6456	6414	N/A	6461	6456	6461
G45	6449	6426	6449	6448	N/A	6447	6449	6448
G46	6423	6422	6422	6415	N/A	6420	6423	6425
G47	6456	6452	6460	6442	N/A	6457	6456	6480
G48	6000	6000	6000	6000	6000	6000	6000	6000
G49	6000	6000	6000	6000	6000	6000	6000	6000
G50	5760	5760	5880	5880	5880	5880	5880	5880
G51	3736	3738	3736	3719	N/A	3725	3740	3711
G52	3711	3713	3711	3707	N/A	3710	3711	3721
G53	3695	3694	3695	3694	3288	3688	3692	3715
G54	3691	3689	3691	3600	3574	3668	3691	3712
G57	3152	3152	3150	3152	3084	3146	3148	3158
G58	18557	18565	18557	18516	N/A	18557	18560	18528
G59	5312	5310	5314	5300	5302	5313	5309	5350
G62	4378	4390	4380	4378	N/A	4364	4376	4394
G63	25988	25994	25988	25988	N/A	25985	25984	25933
G64	7748	7743	7751	7747	N/A	7743	7748	7698
G65	4972	4962	4972	4966	N/A	4970	4972	5026
G66	5672	5668	5672	5662	N/A	5666	5672	5776
G67	6232	6238	6230	6236	N/A	6226	6228	6282
G72	6248	6252	6246	6238	N/A	6238	6250	6342

Table 4: Cut values obtained with Spectral Clustering using K-Means2N. The last column is the result obtained in Rodriguez-Fernandez et al. (2020). For each graph, i.e., for each row, a white-green color scale was used. A white color indicate higher cut values while a green color indicate lower cut values. Small cut values were replaced with a N/A to make the white-green color differences larger.

Graph	$\beta = 0$	$\beta = 0.5$	$\beta = 0.25$	$\sigma = 0.5$	$\sigma = 0.25$	$\sigma = 1$	$\sigma = 10$	kMeans2N
G1	11414	11427	11414	11419	N/A	11417	11414	11470
G2	11310	11331	11310	11286	11245	11297	11310	11365
G3	11398	11402	11398	11398	N/A	11398	11398	11408
G4	11353	11350	11353	11356	N/A	11356	11353	11426
G5	11432	11424	11432	11455	N/A	11432	11432	11379
G6	1936	1942	1936	1931	1912	1934	1934	1923
G7	1722	1728	1722	1709	N/A	1732	1722	1748
G8	1758	1751	1759	1746	N/A	1759	1767	1776
G9	1819	1806	1816	1787	N/A	1815	1819	1797
G10	1765	1758	1765	1765	1680	1759	1759	1748
G11	512	510	510	510	430	514	512	518
G12	516	514	516	512	512	516	516	524
G13	532	534	532	530	534	532	532	548
G14	2956	2958	2954	2918	2901	2943	2952	2953
G15	2965	2965	2965	2955	2940	2967	2965	2958
G16	2946	2951	2947	2951	N/A	2951	2946	2950
G17	2960	2961	2960	2953	N/A	2958	2960	2947
G18	897	895	897	896	850	897	897	883
G19	792	794	792	776	799	793	793	812
G20	844	840	844	854	838	842	844	863
G21	812	814	816	817	817	810	812	826
G22	12935	12931	12935	12942	N/A	12935	12935	12893
G23	12987	12991	12987	12985	N/A	12991	12987	12924
G24	12983	12976	12983	12978	12927	12983	12983	12904
G25	12870	12878	12868	12881	N/A	12866	12868	12922
G26	12873	12877	12873	12889	N/A	12866	12873	12886
G27	2892	2890	2892	2868	N/A	2891	2892	2891
G28	2852	2840	2852	2850	2636	2847	2848	2834
G29	2978	2972	2979	2979	2903	2980	2981	2946
G30	3007	3001	3007	3002	N/A	3003	3007	2979
G31	2884	2880	2881	2842	N/A	2875	2880	2867
G32	1266	1262	1266	1244	1204	1258	1266	1294
G33	1222	1222	1222	1226	716	1220	1222	1266
G34	1246	1248	1248	1254	1254	1246	1244	1268
G35	7412	7409	7412	7401	N/A	7411	7412	7424
G36	7383	7375	7383	7333	7316	7382	7383	7399
G37	7441	7443	7440	7439	N/A	7439	7442	7437
G38	7359	7364	7359	7342	7330	7356	7360	7411
G39	2136	2134	2134	2117	2046	2136	2140	2176
G40	2153	2138	2155	2150	N/A	2156	2154	2134
G41	2118	2114	2118	2102	N/A	2113	2114	2149
G42	2171	2165	2171	2169	2118	2176	2171	2188
G43	6510	6505	6513	6515	N/A	6513	6510	6442
G44	6456	6465	6456	6414	N/A	6461	6456	6461
G45	6449	6426	6449	6448	N/A	6447	6449	6448
G46	6423	6422	6422	6415	N/A	6420	6423	6425
G47	6456	6452	6460	6442	N/A	6457	6456	6480
G48	6000	6000	6000	6000	6000	6000	6000	6000
G49	6000	6000	6000	6000	6000	6000	6000	6000
G50	5760	5760	5880	5880	5760	5880	5880	5880
G51	3736	3738	3736	3719	N/A	3725	3740	3711
G52	3711	3713	3711	3707	N/A	3710	3711	3721
G53	3695	3694	3695	3694	3288	3688	3692	3715
G54	3691	3689	3691	3600	3574	3668	3691	3712
G57	3152	3152	3150	3152	3084	3146	3148	3158
G58	18557	18565	18558	18516	N/A	18557	18560	18528
G59	5312	5310	5314	5300	5302	5313	5309	5350
G62	4378	4390	4380	4378	3644	4364	4376	4394
G63	25988	25994	25988	25988	N/A	25985	25984	25933
G64	7748	7743	7751	7747	N/A	7743	7748	7698
G65	4972	4962	4972	4966	N/A	4970	4972	5026
G66	5672	5668	5672	5662	N/A	5666	5672	5776
G67	6232	6238	6230	6236	N/A	6226	6228	6282
G72	6248	6252	6246	6238	N/A	6238	6250	6342

Table 5: Cut values obtained with Spectral Clustering using K-MeansRand. The last column is the result obtained in Rodriguez-Fernandez et al. (2020). For each graph, i.e., for each row, a white-green color scale was used. A white color indicate higher cut values while a green color indicate lower cut values. Small cut values were replaced with a N/A to make the white-green color differences larger.

Graph	$\beta = 0$	$\beta = 0.5$	$\beta = 0.25$	$\sigma = 0.5$	$\sigma = 0.25$	$\sigma = 1$	$\sigma = 10$	kMeansRand
G1	11414	11427	11414	11419	N/A	11417	11414	11470
G2	11310	11331	11310	11286	11245	11297	11310	11365
G3	11398	11402	11398	11398	N/A	11398	11398	11408
G4	11353	11350	11353	11356	N/A	11356	11353	11426
G5	11432	11424	11432	11455	N/A	11432	11432	11379
G6	1936	1942	1936	1931	1912	1934	1934	1923
G7	1722	1728	1722	1709	N/A	1732	1722	1748
G8	1758	1751	1759	1746	N/A	1759	1767	1776
G9	1819	1806	1816	1787	N/A	1815	1819	1797
G10	1765	1758	1765	1765	1680	1759	1759	1748
G11	512	510	510	510	430	516	512	518
G12	516	514	516	512	512	516	516	524
G13	532	534	532	530	534	532	532	548
G14	2956	2958	2954	2918	2901	2943	2952	2953
G15	2965	2965	2965	2955	2940	2967	2965	2958
G16	2946	2951	2947	2951	N/A	2951	2946	2950
G17	2960	2961	2960	2953	N/A	2958	2960	2947
G18	897	895	897	896	850	897	897	883
G19	792	794	792	776	799	793	793	812
G20	844	840	844	854	838	842	844	863
G21	812	814	816	817	817	810	812	826
G22	12935	12931	12935	12942	N/A	12935	12935	12893
G23	12987	12991	12987	12985	N/A	12991	12987	12924
G24	12983	12976	12983	12978	12927	12983	12983	12904
G25	12870	12878	12868	12881	N/A	12866	12868	12922
G26	12873	12877	12873	12889	N/A	12866	12873	12886
G27	2892	2890	2892	2868	N/A	2891	2892	2891
G28	2852	2840	2852	2850	2636	2847	2848	2834
G29	2978	2972	2979	2979	2903	2980	2981	2946
G30	3007	3001	3007	3002	N/A	3003	3007	2979
G31	2884	2880	2881	2842	N/A	2875	2880	2867
G32	1266	1262	1266	1244	1204	1258	1266	1294
G33	1222	1222	1222	1226	N/A	1220	1222	1266
G34	1246	1248	1248	1254	1254	1246	1244	1268
G35	7412	7409	7412	7401	N/A	7411	7412	7424
G36	7383	7375	7383	7333	7316	7382	7383	7399
G37	7441	7443	7440	7439	N/A	7439	7442	7437
G38	7359	7364	7359	7342	7330	7356	7360	7411
G39	2136	2134	2134	2117	2046	2136	2140	2176
G40	2153	2138	2155	2150	N/A	2156	2154	2134
G41	2118	2114	2118	2102	N/A	2113	2114	2149
G42	2171	2165	2171	2169	2118	2176	2171	2188
G43	6510	6505	6513	6515	N/A	6513	6510	6442
G44	6456	6465	6456	6414	N/A	6461	6456	6461
G45	6449	6426	6449	6448	N/A	6447	6449	6448
G46	6423	6422	6422	6415	N/A	6420	6423	6425
G47	6456	6452	6460	6442	N/A	6457	6456	6480
G48	6000	6000	6000	6000	6000	6000	6000	6000
G49	6000	6000	6000	6000	6000	6000	6000	6000
G50	5880	5760	5880	5880	5880	5880	5880	5880
G51	3736	3738	3736	3719	N/A	3725	3740	3711
G52	3711	3713	3711	3707	N/A	3710	3711	3721
G53	3695	3694	3695	3694	3288	3688	3692	3715
G54	3691	3689	3691	3600	3574	3668	3691	3712
G57	3152	3152	3150	3152	3084	3146	3148	3158
G58	18558	18565	18557	18516	N/A	18557	18560	18528
G59	5312	5310	5314	5300	5302	5313	5309	5350
G62	4378	4390	4380	4378	3634	4364	4376	4394
G63	25988	25994	25988	25988	N/A	25985	25984	25933
G64	7748	7743	7751	7747	N/A	7743	7748	7698
G65	4972	4962	4972	4966	N/A	4970	4972	5026
G66	5672	5668	5672	5662	N/A	5666	5672	5776
G67	6232	6238	6230	6236	N/A	6226	6228	6282
G72	6248	6252	6246	6238	N/A	6238	6250	6342

Table 6: Cut values obtained with Spectral Clustering using K-MedDet. The last column is the result obtained in Rodriguez-Fernandez et al. (2020). For each graph, i.e., for each row, a white-green color scale was used. A white color indicate higher cut values while a green color indicate lower cut values. Small cut values were replaced with a N/A to make the white-green color differences larger.

Graph	$\beta = 0$	$\beta = 0.5$	$\beta = 0.25$	$\sigma = 0.5$	$\sigma = 0.25$	$\sigma = 1$	$\sigma = 10$	kMedDet
G1	11417	11417	11417	11417	11417	11417	11417	11470
G2	11284	11284	11284	11284	11284	11284	11284	11365
G3	11393	11393	11393	11393	11393	11393	11393	11408
G4	11353	11353	11353	11353	11353	11353	11353	11426
G5	11440	11440	11440	11440	11440	11440	11440	11379
G6	1934	1934	1934	1934	1934	1934	1934	1923
G7	1709	1709	1709	1709	1709	1709	1709	1748
G8	1759	1759	1759	1759	1759	1759	1759	1776
G9	1815	1815	1815	1815	1815	1815	1815	1797
G10	1759	1759	1759	1759	1759	1759	1759	1748
G11	512	512	512	512	512	512	512	518
G12	516	516	516	516	516	516	516	524
G13	532	532	532	532	532	532	532	548
G14	2955	2955	2955	2955	2955	2955	2955	2953
G15	2966	2966	2966	2966	2966	2966	2966	2958
G16	2941	2941	2941	2941	2941	2941	2941	2950
G17	2961	2961	2961	2961	2961	2961	2961	2947
G18	897	897	897	897	897	897	897	883
G19	793	793	793	793	793	793	793	812
G20	845	845	845	845	845	845	845	863
G21	812	812	812	812	812	812	812	826
G22	12951	12951	12951	12951	12951	12951	12951	12893
G23	12987	12987	12987	12987	12987	12987	12987	12924
G24	12984	12984	12984	12984	12984	12984	12984	12904
G25	12868	12868	12868	12868	12868	12868	12868	12922
G26	12870	12870	12870	12870	12870	12870	12870	12886
G27	2891	2891	2891	2891	2891	2891	2891	2891
G28	2845	2845	2845	2845	2845	2845	2845	2834
G29	2982	2982	2982	2982	2982	2982	2982	2946
G30	3008	3008	3008	3008	3008	3008	3008	2979
G31	2883	2883	2883	2883	2883	2883	2883	2867
G32	1260	1260	1260	1260	1260	1260	1260	1294
G33	1208	1208	1208	1208	1208	1208	1208	1266
G34	1242	1242	1242	1242	1242	1242	1242	1268
G35	7400	7400	7400	7400	7400	7400	7400	7424
G36	7382	7382	7382	7382	7382	7382	7382	7399
G37	7432	7432	7432	7432	7432	7432	7432	7437
G38	7355	7355	7355	7355	7355	7355	7355	7411
G39	2133	2133	2133	2133	2133	2133	2133	2176
G40	2152	2152	2152	2152	2152	2152	2152	2134
G41	2111	2111	2111	2111	2111	2111	2111	2149
G42	2178	2178	2178	2178	2178	2178	2178	2188
G43	6513	6513	6513	6513	6513	6513	6513	6442
G44	6444	6444	6444	6444	6444	6444	6444	6461
G45	6316	6316	6316	6316	6316	6316	6316	6448
G46	6394	6394	6394	6394	6394	6394	6394	6425
G47	6439	6439	6439	6439	6439	6439	6439	6480
G48	6000	6000	6000	6000	6000	6000	6000	6000
G49	6000	6000	6000	6000	6000	6000	6000	6000
G50	5880	5880	5880	5880	5880	5880	5880	5880
G51	3725	3725	3725	3725	3725	3725	3725	3711
G52	3710	3710	3710	3710	3710	3710	3710	3721
G53	3696	3696	3696	3696	3696	3696	3696	3715
G54	3689	3689	3689	3689	3689	3689	3689	3712
G57	3148	3148	3148	3148	3148	3148	3148	3158
G58	18449	18449	18449	18449	18449	18449	18449	18528
G59	5312	5312	5312	5312	5312	5312	5312	5350
G62	4350	4350	4350	4350	4350	4350	4350	4394
G63	25977	25977	25977	25977	25977	25977	25977	25933
G64	7747	7747	7747	7747	7747	7747	7747	7698
G65	4964	4964	4964	4964	4964	4964	4964	5026
G66	5664	5664	5664	5664	5664	5664	5664	5776
G67	6226	6226	6226	6226	6226	6226	6226	6282
G72	6242	6242	6242	6242	6242	6242	6242	6342

Table 7: Cut values obtained with Spectral Clustering using K-MedRand. The last column is the result obtained in Rodriguez-Fernandez et al. (2020). For each graph, i.e., for each row, a white-green color scale was used. A white color indicate higher cut values while a green color indicate lower cut values. Small cut values were replaced with a N/A to make the white-green color differences larger.

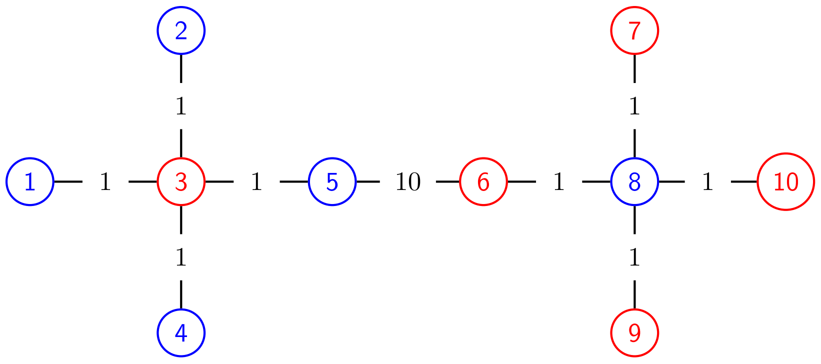
Graph	$\beta = 0$	$\beta = 0.5$	$\beta = 0.25$	$\sigma = 0.5$	$\sigma = 0.25$	$\sigma = 1$	$\sigma = 10$	kMedRand
G1	11417	11417	11417	11417	11417	11417	11417	11470
G2	11284	11284	11284	11284	11284	11284	11284	11365
G3	11393	11393	11393	11393	11393	11393	11393	11408
G4	11353	11353	11353	11353	11353	11353	11353	11426
G5	11440	11440	11440	11440	11440	11440	11440	11379
G6	1934	1934	1934	1934	1934	1934	1934	1923
G7	1715	1715	1715	1715	1715	1715	1715	1748
G8	1759	1759	1759	1759	1759	1759	1759	1776
G9	1815	1815	1815	1815	1815	1815	1815	1797
G10	1766	1766	1766	1766	1766	1766	1766	1748
G11	512	512	512	512	512	512	512	518
G12	516	516	516	516	516	516	516	524
G13	532	532	532	532	532	532	532	548
G14	2955	2955	2955	2955	2955	2955	2955	2953
G15	2966	2966	2966	2966	2966	2966	2966	2958
G16	2941	2941	2941	2941	2941	2941	2941	2950
G17	2961	2961	2961	2961	2961	2961	2961	2947
G18	897	897	897	897	897	897	897	883
G19	793	793	793	793	793	793	793	812
G20	845	845	845	845	845	845	845	863
G21	814	814	814	814	814	814	814	826
G22	12955	12955	12955	12955	12955	12955	12955	12893
G23	12988	12988	12988	12988	12988	12988	12988	12924
G24	12984	12984	12984	12984	12984	12984	12984	12904
G25	12868	12868	12868	12868	12868	12868	12868	12922
G26	12870	12870	12870	12870	12870	12870	12870	12886
G27	2891	2891	2891	2891	2891	2891	2891	2891
G28	2845	2845	2845	2845	2845	2845	2845	2834
G29	2982	2982	2982	2982	2982	2982	2982	2946
G30	3008	3008	3008	3008	3008	3008	3008	2979
G31	2885	2885	2885	2885	2885	2885	2885	2867
G32	1260	1260	1260	1260	1260	1260	1260	1294
G33	1212	1212	1212	1212	1212	1212	1212	1266
G34	1246	1246	1246	1246	1246	1246	1246	1268
G35	7412	7412	7412	7412	7412	7412	7412	7424
G36	7382	7382	7382	7382	7382	7382	7382	7399
G37	7436	7436	7439	7436	7436	7439	7439	7437
G38	7355	7355	7355	7355	7355	7355	7355	7411
G39	2133	2133	2133	2133	2133	2133	2133	2176
G40	2156	2152	2156	2156	2152	2152	2152	2134
G41	2113	2113	2113	2113	2113	2113	2113	2149
G42	2178	2178	2178	2178	2178	2178	2178	2188
G43	6513	6513	6513	6513	6513	6513	6513	6442
G44	6444	6444	6444	6444	6444	6444	6444	6461
G45	6316	6316	6316	6316	6316	6316	6316	6448
G46	6411	6411	6411	6411	6411	6411	6411	6425
G47	6439	6439	6439	6439	6439	6439	6439	6480
G48	6000	6000	6000	6000	6000	6000	6000	6000
G49	6000	6000	6000	6000	6000	6000	6000	6000
G50	5880	5880	5880	5880	5880	5880	5880	5880
G51	3725	3725	3725	3725	3725	3725	3725	3711
G52	3710	3710	3710	3710	3710	3710	3710	3721
G53	3696	3696	3696	3696	3696	3696	3696	3715
G54	3689	3689	3689	3689	3689	3689	3689	3712
G57	3148	3148	3148	3148	3148	3148	3148	3158
G58	18449	18449	18449	18449	18449	18449	18449	18528
G59	5312	5312	5312	5312	5312	5312	5312	5350
G62	4350	4350	4350	4350	4350	4350	4350	4394
G63	25981	25981	25981	25981	25981	25981	25981	25933
G64	7747	7747	7747	7747	7747	7747	7747	7698
G65	4964	4964	4964	4964	4964	4964	4964	5026
G66	5668	5668	5668	5668	5668	5668	5668	5776
G67	6226	6226	6226	6226	6226	6226	6226	6282
G72	6242	6242	6242	6242	6242	6242	6242	6342

Table 8: Cut values obtained with Spectral Clustering using Fuzzy c-means. The last column is the result obtained in Rodriguez-Fernandez et al. (2020). For each graph, i.e., for each row, a white-green color scale was used. A white color indicate higher cut values while a green color indicate lower cut values. Small cut values were replaced with a N/A to make the white-green color differences larger.

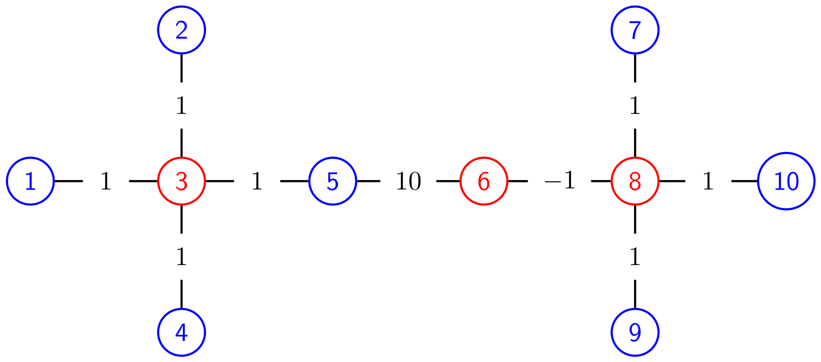
Graph	$\beta = 0$	$\beta = 0.5$	$\beta = 0.25$	$\sigma = 0.5$	$\sigma = 0.25$	$\sigma = 1$	$\sigma = 10$	Fuzzy
G1	11417	11418	11417	11425	N/A	11422	11417	11470
G2	11328	11328	11328	11321	11266	11329	11328	11365
G3	11392	11402	11398	11397	N/A	11392	11392	11408
G4	11349	11344	11349	11356	N/A	11349	11349	11426
G5	11432	11432	11428	11454	10261	11440	11432	11379
G6	1948	1947	1948	1931	1912	1934	1936	1923
G7	1729	1734	1722	1712	N/A	1715	1722	1748
G8	1758	1744	1761	1745	N/A	1760	1763	1776
G9	1800	1810	1804	1784	N/A	1796	1803	1797
G10	1765	1756	1765	1765	1694	1759	1759	1748
G11	510	510	508	510	492	514	510	518
G12	516	514	516	514	510	516	514	524
G13	530	534	530	530	526	532	532	548
G14	2956	2960	2955	2923	2902	2943	2953	2953
G15	2965	2966	2965	2955	2941	2965	2965	2958
G16	2952	2951	2947	2953	2458	2951	2946	2950
G17	2960	2962	2960	2961	N/A	2958	2960	2947
G18	896	893	901	896	862	894	896	883
G19	792	793	792	776	798	793	793	812
G20	837	837	837	851	831	841	844	863
G21	812	814	813	818	812	809	812	826
G22	12958	12962	12956	12944	N/A	12958	12958	12893
G23	12987	12997	12987	12985	N/A	12991	12987	12924
G24	12983	12982	12985	12981	12931	12985	12985	12904
G25	12866	12868	12867	12876	N/A	12876	12868	12922
G26	12866	12868	12866	12880	N/A	12874	12866	12886
G27	2892	2896	2893	2868	2385	2890	2892	2891
G28	2852	2836	2852	2852	2806	2847	2848	2834
G29	2973	2975	2973	2975	2903	2971	2973	2946
G30	3008	3002	3004	3008	N/A	3004	3006	2979
G31	2877	2874	2877	2837	N/A	2881	2880	2867
G32	1262	1262	1262	1242	1204	1260	1260	1294
G33	1222	1226	1228	1226	882	1222	1222	1266
G34	1246	1248	1248	1254	1248	1246	1244	1268
G35	7416	7407	7412	7401	2230	7411	7410	7424
G36	7386	7398	7385	7340	7334	7382	7383	7399
G37	7438	7439	7439	7438	N/A	7436	7439	7437
G38	7366	7387	7364	7354	7330	7355	7359	7411
G39	2137	2138	2135	2130	2042	2133	2140	2176
G40	2151	2133	2147	2154	N/A	2160	2151	2134
G41	2115	2109	2115	2103	1498	2116	2109	2149
G42	2173	2161	2173	2169	2135	2178	2173	2188
G43	6520	6521	6520	6515	N/A	6516	6520	6442
G44	6461	6465	6461	6457	N/A	6461	6461	6461
G45	6441	6430	6439	6441	N/A	6447	6441	6448
G46	6419	6422	6419	6415	6069	6420	6419	6425
G47	6449	6456	6446	6441	N/A	6450	6452	6480
G48	6000	6000	6000	6000	6000	6000	6000	6000
G49	6000	6000	6000	6000	6000	6000	6000	6000
G50	5880	5880	5880	5880	5880	5880	5880	5880
G51	3740	3738	3740	3724	N/A	3738	3740	3711
G52	3711	3715	3711	3705	N/A	3710	3711	3721
G53	3695	3694	3695	3694	N/A	3696	3692	3715
G54	3691	3713	3691	3672	N/A	3676	3691	3712
G57	3146	3154	3148	3152	N/A	3148	3148	3158
G58	18550	18548	18550	18530	N/A	18541	18547	18528
G59	5314	5310	5311	5305	5301	5313	5314	5350
G62	4376	4388	4380	4376	N/A	4370	4378	4394
G63	25991	25990	25985	25983	N/A	25982	25992	25933
G64	7745	7751	7745	7752	N/A	7745	7747	7698
G65	4970	4960	4972	4964	N/A	4966	4968	5026
G66	5674	5668	5668	5660	5604	5670	5674	5776
G67	6228	6236	6230	6238	N/A	6226	6224	6282
G72	6248	6254	6246	6238	N/A	6242	6248	6342

APPENDIX B

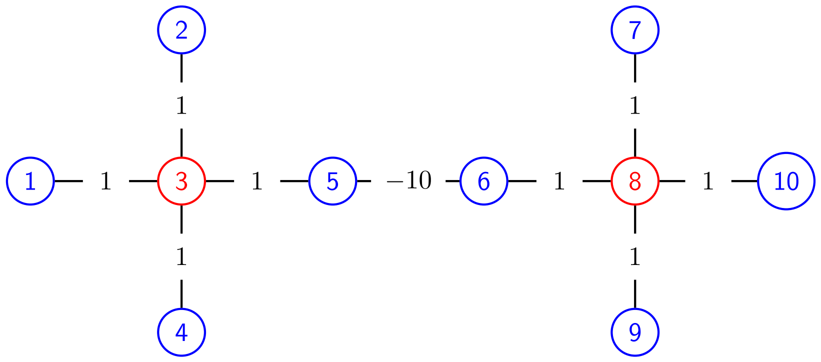
B-Set Graphs



(a) Graph B1 with Max-Cut of 18.

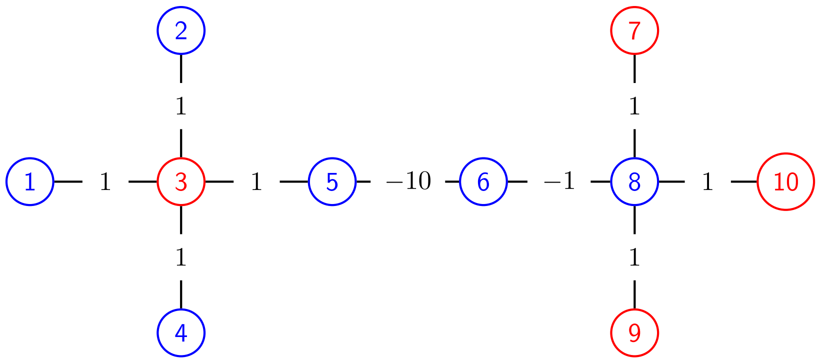


(b) Graph B2 with Max-Cut of 17.

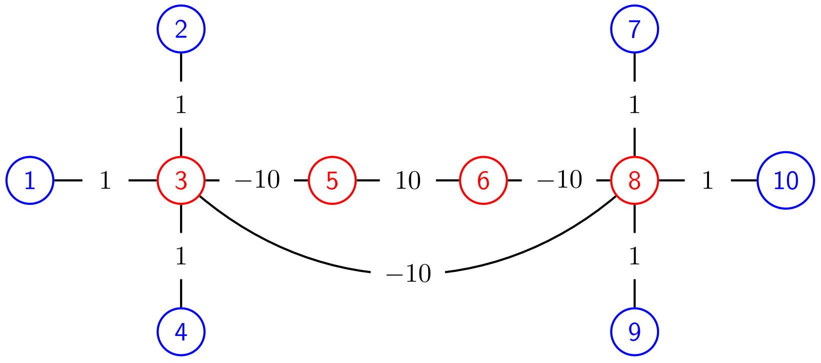


(c) Graph B3 with Max-Cut of 8.

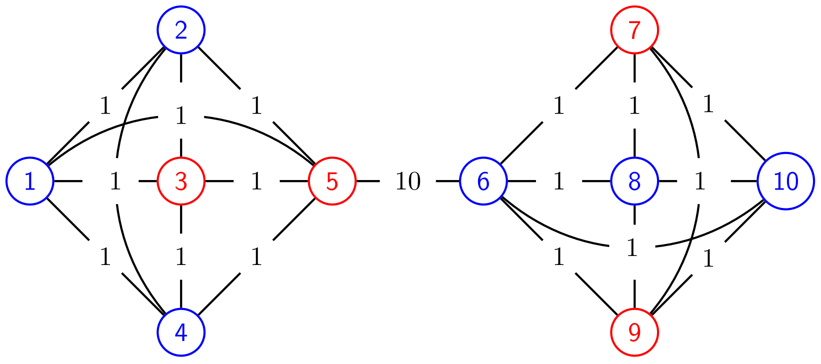
Figure 28: B-set of graphs and their Max-Cut solution. Nodes with the same color belong to the same partition.



(d) Graph B4 with Max-Cut of 7.

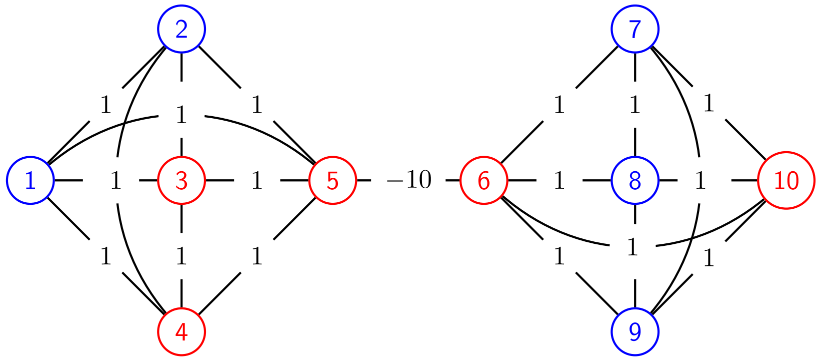


(e) Graph B5 with Max-Cut of 6.



(f) Graph B6 with Max-Cut of 22.

Figure 28: (Continued) B-set of graphs and their Max-Cut solution. Nodes with the same color belong to the same partition.

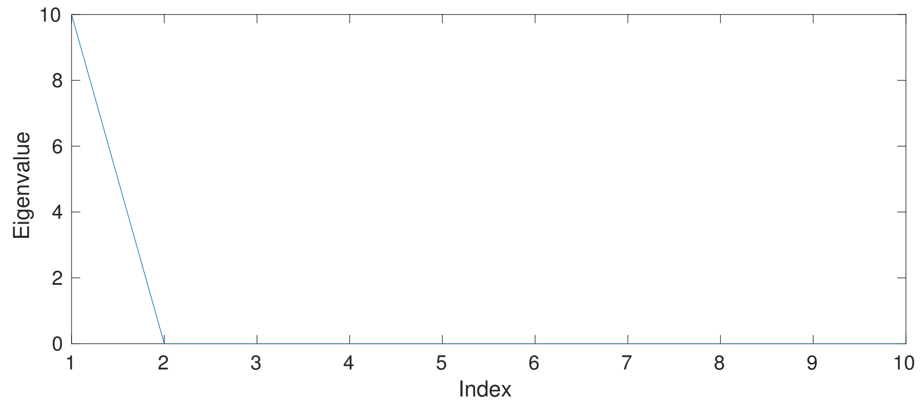


(g) Graph B7 with Max-Cut of 12.

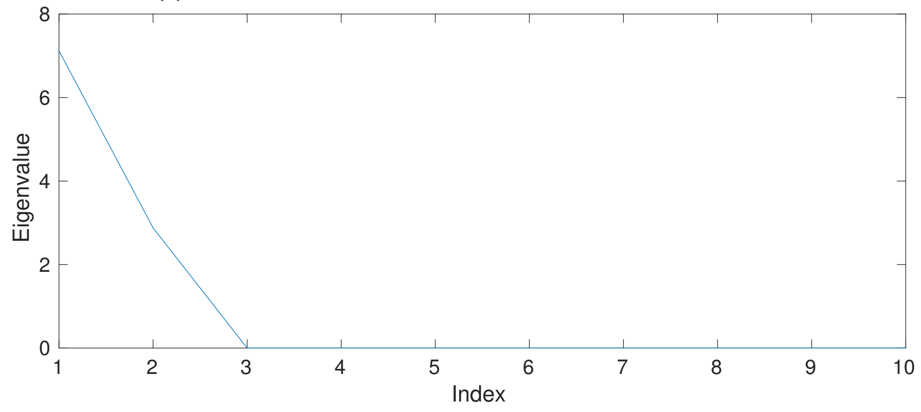
Figure 28: (Continued) B-set of graphs and their Max-Cut solution. Nodes with the same color belong to the same partition.

APPENDIX C

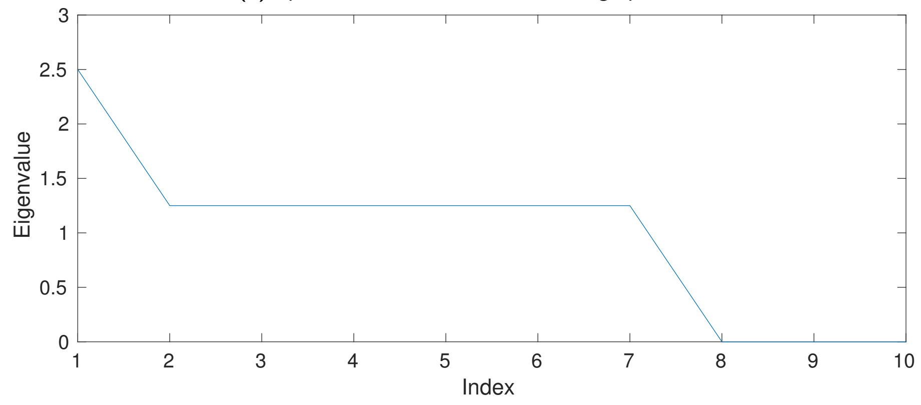
Spectrum of the Gram and Laplacian Matrix for the B-Set graphs



(a) Spectrum of the Gram Matrix for graphs B1, B2, B3 and B4.

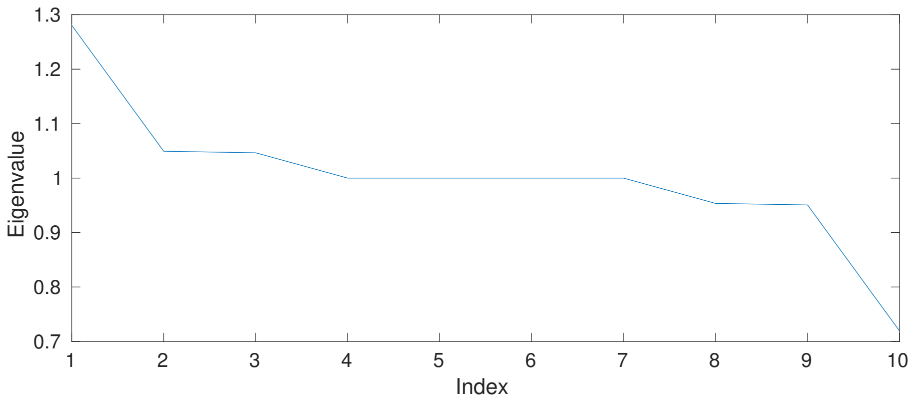


(b) Spectrum of the Gram Matrix for graph B5.

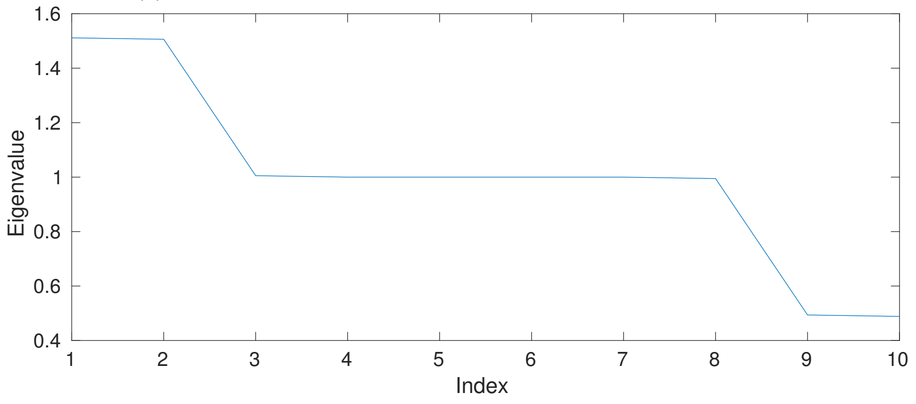


(c) Spectrum of the Gram Matrix for graphs B6 and B7.

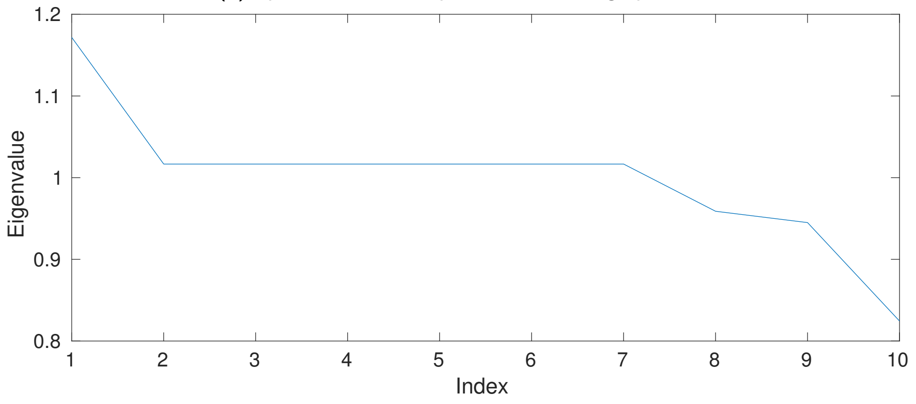
Figure 29: Spectrum of the Gram Matrix for the B-Set of graphs. Some graphs present the same spectrum.



(a) Spectrum of the Laplacian matrix of graphs B1, B2, B3 and B4.



(b) Spectrum of the Laplacian matrix of graph B5.



(c) Spectrum of the Laplacian matrix of graphs B6 and B7.

Figure 30: Spectrum of the Laplacian Matrix for the B-Set graphs. Some graphs present the same spectrum.

List of Abbreviations

CSDP	Complex Semidefinite Programming.
HoC	House of Cards.
IP	Integer Programming.
IR	Infinite Range.
LP	Linear Programming.
LS	Local Search.
MAX SAT	Maximum Satisfiability.
MST	Minimum Spanning Tree.
PC	Principal Component.
PCA	Principal Component Analysis.
PSD	Positive Semidefinite.
RMM	RandomizedMinMax.
SC	Spectral Clustering.
SDP	Semidefinite Programming.
SR	Short Range.
TSP	Traveling Salesman Problem.
VP	Vector Programming.

Definition Index

Bibliography

Addario-Berry, L., L. Devroye, G. Lugosi, and R. I. Oliveira (Apr. 2019). "Local optima of the Sherrington-Kirkpatrick Hamiltonian". In: *Journal of Mathematical Physics* 60.4, p. 043301. *DOI*: `10.1063/1.5020662`. *URL*: `https://doi.org/10.1063/1.5020662`.

Aita, T. and Y. Husimi (Sept. 2000). "Adaptive walks by the fittest among finite random mutants on a Mt. Fuji-type fitness landscape". In: *Journal of Mathematical Biology* 41.3, pp. 207–231. *DOI*: `10.1007/s002850000046`. *URL*: `https://doi.org/10.1007/s002850000046`.

Amitrano, C., L. Peliti, and M. Saber (Dec. 1989). "Population dynamics in a spin-glass model of chemical evolution". In: *Journal of Molecular Evolution* 29.6, pp. 513–525. *DOI*: `10.1007/bf02602923`. *URL*: `https://doi.org/10.1007/bf02602923`.

Bader, D. A., W. E. Hart, and C. A. Phillips (2005). "Parallel Algorithm Design for Branch and Bound". In: *Tutorials on Emerging Methodologies and Applications in Operations Research: Presented at Informs 2004, Denver, CO*. Ed. by H. J. G. New York, NY: Springer New York, pp. 5-1–5-44. *ISBN*: 978-0-387-22827-3. *DOI*: `10.1007/0-387-22827-6_5`. *URL*: `https://doi.org/10.1007/0-387-22827-6_5`.

Barahona, F., M. Grötschel, M. Jünger, and G. Reinelt (1988). "An application of combinatorial optimization to statistical physics and circuit layout design". In: *Operations Research* 36.3, pp. 493–513.

Benaïchouche, M., V.-D. Cung, S. Dowaji, B. Le Cun, T. Mautor, and C. Roucairol (1996). "Building a parallel branch and bound library". In: *Solving Combinatorial Optimization Problems in Parallel: Methods and Techniques*. Ed. by A. Ferreira and P. Pardalos. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 201–231. *ISBN*: 978-3-540-49875-9. *DOI*: `10.1007/BFb0027123`. *URL*: `https://doi.org/10.1007/BFb0027123`.

Bezdek, J. C. (1981). *Pattern Recognition With Fuzzy Objective Function Algorithms*. Boston, MA: Springer US. *DOI*: `10.1007/978-1-4757-0450-1`.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag. *ISBN*: 0387310738.

Bray, A. J. and M. A. Moore (Aug. 1980). "Replica theory of quantum spin glasses". In: *Journal of Physics C: Solid State Physics* 13.24, pp. L655–L660. *DOI*: `10.1088/0022-3719/13/24/005`. *URL*: `https://doi.org/10.1088/0022-3719/13/24/005`.

Byrka, J., F. Grandoni, T. Rothvoß, and L. Sanità (2010). "An Improved LP-Based Approximation for Steiner Tree". In: *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*. STOC '10. Cambridge, Massachusetts, USA: Association for Computing Machinery, pp. 583–592. *ISBN*: 9781450300506. *DOI*: `10.1145/1806689.1806769`. *URL*: `https://doi.org/10.1145/1806689.1806769`.

Christofides, N. (1976). *Worst-case analysis of a new heuristic for the travelling salesman problem*. Tech. rep. Carnegie-Mellon University.

Chung, F. R. K. (1997). *Spectral Graph Theory*. American Mathematical Society.

Cvetkovic, D., M. Doob, I. Gutman, and A. Torgasev (1988). *Recent results in the theory of graph spectra*. Annals of Discrete Mathematics. Burlington, MA: Elsevier. *URL*: `https://cds.cern.ch/record/1252661`.

Dasgupta, S., C. H. Papadimitriou, and U. Vazirani (2006). *Algorithms*. 1st ed. USA: McGraw-Hill, Inc. *ISBN*: 0073523402.

Delorme, C. and S. Poljak (1993). "Laplacian eigenvalues and the maximum cut problem". In: *Mathematical Programing* 62, pp. 557–574. *DOI*: `10.1007/BF01585184`.

Derrida, B. (Sept. 1981). "Random-energy model: An exactly solvable model of disordered systems". In: *Phys. Rev. B* 24 (5), pp. 2613–2626. *DOI*: `10.1103/PhysRevB.24.2613`. *URL*: `https://link.aps.org/doi/10.1103/PhysRevB.24.2613`.

Deza, M. and M. Laurent (1994a). "Applications of cut polyhedra — I". In: *Journal of Computational and Applied Mathematics* 55.2, pp. 191–216. *ISSN*: 0377-0427. *DOI*: `https://doi.org/10.1016/0377-0427(94)90020-5`. *URL*: `https://www.sciencedirect.com/science/article/pii/0377042794900205`.

Deza, M. and M. Laurent (1994b). "Applications of cut polyhedra — II". In: *Journal of Computational and Applied Mathematics* 55.2, pp. 217–247. *ISSN*: 0377-0427. *DOI*: `https://doi.org/10.1016/0377-0427(94)90021-3`. *URL*: `https://www.sciencedirect.com/science/article/pii/0377042794900213`.

Dhillon, I. S. and D. S. Modha (2001). "Concept Decompositions for Large Sparse Text Data Using Clustering". In: *Machine Learning* 42, pp. 143–175. *DOI*: `10.1023/A:1007612920971`.

Dósa, G. (2007). "The Tight Bound of First Fit Decreasing Bin-Packing Algorithm Is FFD(I) $\leq 11/9$OPT(I) $+ 6/9$". In: *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Ed. by B. Chen, M. Paterson, and G. Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–11. *ISBN*: 978-3-540-74450-4.

Evans, S. N. and D. Steinsaltz (2002). "Estimating some features of NK fitness landscapes". In: *The Annals of Applied Probability* 12.4, pp. 1299–1321. *DOI*: `10.1214/aoap/1037125864`. *URL*: `https://doi.org/10.1214/aoap/1037125864`.

Fairbrother, J., A. N. Letchford, and K. Briggs (2018). "A two-level graph partitioning problem arising in mobile wireless communications". In: *Computational Optimization and Applications* 69.3, pp. 653–676.

Frieze, A. and M. Jerrum (May 1997). "Improved approximation algorithms for MAXk-CUT and MAX BISECTION". In: *Algorithmica* 18.1, pp. 67–81. *DOI*: `10.1007/bf02523688`. *URL*: `https://doi.org/10.1007/bf02523688`.

Garcia-Diaz, J., J. Sanchez-Hernandez, R. Menchaca-Mendez, and R. Menchaca-Mendez (July 2017). "When a worse approximation factor gives better performance: a 3-approximation algorithm for the vertex k-center problem". In: *Journal of Heuristics* 23.5, pp. 349–366. *DOI*: 10.1007/s10732-017-9345-x. *URL*: https://doi.org/10.1007/s10732-017-9345-x.

Ghosh, S. and S. K. Dubey (2013). "Comparative Analysis of K-Means and Fuzzy C-Means Algorithms". In: *International Journal of Advanced Computer Science and Applications* 4.4. *DOI*: 10.14569/IJACSA.2013.040406. *URL*: http://dx.doi.org/10.14569/IJACSA.2013.040406.

Goemans, M. X. and D. P. Williamson (Nov. 1995). "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming". In: *J. ACM* 42.6, pp. 1115–1145. *ISSN*: 0004-5411. *DOI*: 10.1145/227683.227684. *URL*: https://doi.org/10.1145/227683.227684.

Goemans, M. X. and D. P. Williamson (2004). "Approximation algorithms for Max-3-Cut and other problems via complex semidefinite programming". In: *Journal of Computer and System Sciences* 68.2. Special Issue on STOC 2001, pp. 442–470. *ISSN*: 0022-0000. *DOI*: https://doi.org/10.1016/j.jcss.2003.07.012. *URL*: https://www.sciencedirect.com/science/article/pii/S0022000003001454.

Golub, G. H. and C. F. Van Loan (1996). *Matrix Computations (3rd Ed.)* USA: Johns Hopkins University Press. *ISBN*: 0801854148.

Gonzalez-Torres, B. A. (2020). *An Algorithmic Introduction to Clustering*. arXiv: 2006.04916 [cs.LG].

Gross, D. and M. Mezard (1984). "The simplest spin glass". In: *Nuclear Physics B* 240.4, pp. 431–452. *ISSN*: 0550-3213. *DOI*: https://doi.org/10.1016/0550-3213(84)90237-2. *URL*: https://www.sciencedirect.com/science/article/pii/0550321384902372.

Grygorash, O., Y. Z. Zhou, and Z. Jorgensen (2006). "Minimum Spanning Tree Based Clustering Algorithms". In: *18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*. Ed. by C.-T. L. Lu and N. G. B. Bourbakis. Los Alamitos, CA: IEEE Computer Society, pp. 73–81. *DOI*: 10.1109/ICTAI.2006.83.

Hadlock, F. (Sept. 1975). "Finding a Maximum Cut of a Planar Graph in Polynomial Time". In: *SIAM Journal on Computing* 4.3, pp. 221–225. *DOI*: 10.1137/0204019. *URL*: https://doi.org/10.1137/0204019.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Berlin, Heidelberg: Springer. *ISBN*: 9780387848846.

Helmberg, C. (2000). *Semidefinite programming for combinatorial optimization*. Konrad-Zuse-Zentrum für Informationstechnik Berlin.

Higham, N. J. (1988). "Computing a nearest symmetric positive semidefinite matrix". In: *Linear Algebra and its Applications* 103, pp. 103–118. *ISSN*: 0024-3795. *DOI*: https://doi.org/10.1016/0024-3795(88)90223-6. *URL*: https://www.sciencedirect.com/science/article/pii/0024379588902236.

Horn, R. A. (Oct. 2012). *Matrix Analysis: Second Edition*. Cambridge University Press. *ISBN*: 0521548233. *URL*: `https://www.xarg.org/ref/a/0521548233/`.

Hotelling, H. (1933). "Analysis of a complex of statistical variables into principal components." In: *Journal of Educational Psychology* 24.6, pp. 417–441. *DOI*: `10.1037/h0071325`. *URL*: `https://doi.org/10.1037%2Fh0071325`.

Hwang, S., B. Schmiegelt, L. Ferretti, and J. Krug (Feb. 2018). "Universality Classes of Interaction Structures for NK Fitness Landscapes". In: *Journal of Statistical Physics* 172.1, pp. 226–278. *ISSN*: 1572-9613. *DOI*: `10.1007/s10955-018-1979-z`. *URL*: `http://dx.doi.org/10.1007/s10955-018-1979-z`.

Karger, D., R. Motwani, and M. Sudan (Mar. 1998). "Approximate Graph Coloring by Semidefinite Programming". In: *J. ACM* 45.2, pp. 246–265. *ISSN*: 0004-5411. *DOI*: `10.1145/274787.274791`. *URL*: `https://doi.org/10.1145/274787.274791`.

Karlin, A. R., N. Klein, and S. O. Gharan (2020). *A (Slightly) Improved Approximation Algorithm for Metric TSP*. arXiv: `2007.01409 [cs.DS]`.

Karmarkar, N. (1984). "A New Polynomial-Time Algorithm for Linear Programming". In: *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*. STOC '84. New York, NY, USA: Association for Computing Machinery, pp. 302–311. *ISBN*: 0897911334. *DOI*: `10.1145/800057.808695`. *URL*: `https://doi.org/10.1145/800057.808695`.

Karp, R. M. (1972). "Reducibility among combinatorial problems". In: *Complexity of Computer Computation*. Ed. by R. E. Miller and J. W. Thacher. Plenum Press, pp. 85–103.

Kauffman, S. and S. Levin (1987). "Towards a general theory of adaptive walks on rugged landscapes". In: *Journal of Theoretical Biology* 128.1, pp. 11–45. *ISSN*: 0022-5193. *DOI*: `https://doi.org/10.1016/S0022-5193(87)80029-2`. *URL*: `https://www.sciencedirect.com/science/article/pii/S0022519387800292`.

Kaufmann, L. and P. Rousseeuw (1987). "Clustering by Means of Medoids". In: *Data Analysis based on the L_1-Norm and Related Methods*. Ed. by Y. Dodge. Amsterdam: North-Holland, pp. 405–416.

Khot, S. (2002). "On the Power of Unique 2-Prover 1-Round Games". In: *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*. STOC '02. Montreal, Quebec, Canada: Association for Computing Machinery, pp. 767–775. *ISBN*: 1581134959. *DOI*: `10.1145/509907.510017`. *URL*: `https://doi.org/10.1145/509907.510017`.

Khot, S., G. Kindler, E. Mossel, and R. O'Donnell (2007). "Optimal inapproximability results for MAX-CUT and other 2-variable CSPs?" In: *SIAM J. Computing* 37, pp. 319–357. *DOI*: `10.1137/S0097539705447372`.

Kingman, J. F. C. (1978). "A simple model for the balance between selection and mutation". In: *Journal of Applied Probability* 15.1, pp. 1–12. *DOI*: `10.2307/3213231`.

Klemm, K., A. Mehta, and P. F. Stadler (Apr. 2012). "Landscape Encodings Enhance Optimization". In: *PLoS ONE* 7.4. Ed. by S. Gómez, e34780. *ISSN*:

1932-6203. *DOI*: `10.1371/journal.pone.0034780`. *URL*: `http://dx.doi.org/10.1371/journal.pone.0034780`.

Klerk, E. de, D. Pasechnik, and J. Warners (Sept. 2004). "On Approximate Graph Colouring and MAX-k-CUT Algorithms Based on the ϑ-Function". In: *Journal of Combinatorial Optimization* 8.3, pp. 267–294. *DOI*: `10.1023/b:joco.0000038911.67280.3f`. *URL*: `https://doi.org/10.1023/b:joco.0000038911.67280.3f`.

Krug, J. and B. Schmiegelt (2013). "Evolutionary Accessibility of Modular Fitness Landscapes". In: *Journal of Statistical Physics* 154, pp. 334–355. *ISSN*: 0022-4715. *DOI*: `10.1007/s10955-013-0868-8`.

Krząkała, F. and L. Zdeborová (Jan. 2008). "Phase transitions and computational difficulty in random constraint satisfaction problems". In: *Journal of Physics: Conference Series* 95, p. 012012. *ISSN*: 1742-6596. *DOI*: `10.1088/1742-6596/95/1/012012`. *URL*: `http://dx.doi.org/10.1088/1742-6596/95/1/012012`.

Krząkała, F., A. Montanari, F. Ricci-Tersenghi, G. Semerjian, and L. Zdeborová (June 2007). "Gibbs states and the set of solutions of random constraint satisfaction problems". In: *Proceedings of the National Academy of Sciences* 104.25, pp. 10318–10323. *ISSN*: 1091-6490. *DOI*: `10.1073/pnas.0703685104`. *URL*: `http://dx.doi.org/10.1073/pnas.0703685104`.

Land, A. H. and A. G. Doig (1960). "An Automatic Method of Solving Discrete Programming Problems". In: *Econometrica* 28.3, pp. 497–520. *ISSN*: 00129682, 14680262. *URL*: `http://www.jstor.org/stable/1910129`.

Le, T. and T. Altman (2011). "A new initialization method for the Fuzzy C-Means Algorithm using Fuzzy Subtractive Clustering". In: *Proc. Intl'Conf. on Information and Knowledge Engineering, Las Vegas USA*. Vol. 1, pp. 144–150.

Liers, F., M. Jünger, G. Reinelt, and G. Rinaldi (2004). "Computing Exact Ground States of Hard Ising Spin Glass Problems by Branch-and-Cut". In: *New Optimization Algorithms in Physics*. Ed. by H. R. Alexander K. Hartmann. Wiley-VCH, pp. 47–68. *DOI*: `10.1002/3527603794.ch4`.

Lovasz, L. (Jan. 1979). "On the Shannon capacity of a graph". In: *IEEE Transactions on Information Theory* 25.1, pp. 1–7. *DOI*: `10.1109/tit.1979.1055985`. *URL*: `https://doi.org/10.1109/tit.1979.1055985`.

Luxburg, U. von (Aug. 2007). "A tutorial on spectral clustering". In: *Statistics and Computing* 17.4, pp. 395–416. *DOI*: `10.1007/s11222-007-9033-z`. *URL*: `https://doi.org/10.1007/s11222-007-9033-z`.

Ma, F. and J.-K. Hao (2017). "A multiple search operator heuristic for the max-k-cut problem". In: *Ann. Oper. Res.* 248, pp. 365–403. *DOI*: `10.1007/s10479-016-2234-0`.

MacQueen, J. (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, CA: Univ. California Press, pp. 281–297. *URL*: `https://projecteuclid.org/euclid.bsmsp/1200512992`.

Neidhart, J., I. G. Szendro, and J. Krug (2013). "Exact results for amplitude spectra of fitness landscapes". In: *Journal of Theoretical Biology* 332, pp. 218–227.

ISSN: 0022-5193. *DOI*: `https://doi.org/10.1016/j.jtbi.2013.05.002`. *URL*: `https://www.sciencedirect.com/science/article/pii/S0022519313002178`.

Neidhart, J., I. G. Szendro, and J. Krug (2014). "Adaptation in Tunably Rugged Fitness Landscapes: The Rough Mount Fuji Model". In: *Genetics* 198.2, pp. 699–721. *ISSN*: 0016-6731. *DOI*: `10.1534/genetics.114.167668`. eprint: `https://www.genetics.org/content/198/2/699.full.pdf`. *URL*: `https://www.genetics.org/content/198/2/699`.

Newman, A. (2018). "Complex Semidefinite Programming and Max-k-Cut". In: *1st Symposium on Simplicity in Algorithms (SOSA 2018)*. Ed. by R. Seidel. Vol. 61. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 13:1–13:11. *ISBN*: 978-3-95977-064-4. *DOI*: `10.4230/OASIcs.SOSA.2018.13`. *URL*: `http://drops.dagstuhl.de/opus/volltexte/2018/8309`.

Ng, A., M. Jordan, and Y. Weiss (2002). "On Spectral Clustering: Analysis and an algorithm". In: *Advances in Neural Information Processing Systems*. Ed. by T. Dietterich, S. Becker, and Z. Ghahramani. Vol. 14. MIT Press, pp. 849–856. *URL*: `https://proceedings.neurips.cc/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf`.

Niu, C., Y. Li, R. Q. Hu, and F. Ye (2017). "Femtocell-enhanced multi-target spectrum allocation strategy in LTE-A HetNets". In: *IET communications* 11.6, pp. 887–896.

Nowak, S. and J. Krug (June 2015). "Analysis of adaptive walks on NK fitness landscapes with different interaction schemes". In: *Journal of Statistical Mechanics: Theory and Experiment* 2015.6, P06014. *DOI*: `10.1088/1742-5468/2015/06/p06014`. *URL*: `https://doi.org/10.1088/1742-5468/2015/06/p06014`.

Oliveira, V. M. de and J. F. Fontanari (Dec. 1997). "Landscape statistics of the p-spin Ising model". In: *Journal of Physics A: Mathematical and General* 30.24, pp. 8445–8457. *DOI*: `10.1088/0305-4470/30/24/010`. *URL*: `https://doi.org/10.1088/0305-4470/30/24/010`.

Oliveira, V. M. d., J. F. Fontanari, and P. F. Stadler (Dec. 1999). "Metastable states in short-ranged p-spin glasses". In: *Journal of Physics A: Mathematical and General* 32.50, pp. 8793–8802. *ISSN*: 1361-6447. *DOI*: `10.1088/0305-4470/32/50/302`. *URL*: `http://dx.doi.org/10.1088/0305-4470/32/50/302`.

Pearson, K. (Nov. 1901). *LIII. On lines and planes of closest fit to systems of points in space*. *DOI*: `10.1080/14786440109462720`. *URL*: `https://doi.org/10.1080/14786440109462720`.

Poljak, S. and Z. Tuza (1993). "Maximum cuts and largest bipartite subgraphs". In: *Combinatorial Optimization*.

Poljak, S. and F. Rendl (1995). "Solving the max-cut problem using eigenvalues". In: *Discr. Appl. Math.* 62, pp. 249–278. *DOI*: `10.1016/0166-218X(94)00155-7`.

Rieger, H. (Dec. 1992). "The number of solutions of the Thouless-Anderson-Palmer equations for p-spin-interaction spin glasses". In: *Phys. Rev. B* 46 (22), pp. 14655–14661. *DOI*: `10.1103/PhysRevB.46.14655`. *URL*: `https://link.aps.org/doi/10.1103/PhysRevB.46.14655`.

Rodriguez-Fernandez, A. E., B. Gonzalez-Torres, R. Menchaca-Mendez, and P. F. Stadler (Aug. 2020). "Clustering Improves the Goemans–Williamson Approximation for the Max-Cut Problem". In: *Computation* 8.3, p. 75. *DOI*: `10.3390/computation8030075`. *URL*: `https://doi.org/10.3390/computation8030075`.

Sherrington, D. and S. Kirkpatrick (Dec. 1975). "Solvable Model of a Spin-Glass". In: *Phys. Rev. Lett.* 35 (26), pp. 1792–1796. *DOI*: `10.1103/PhysRevLett.35.1792`. *URL*: `https://link.aps.org/doi/10.1103/PhysRevLett.35.1792`.

Slavík, P. (1996). "A Tight Analysis of the Greedy Algorithm for Set Cover". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, pp. 435–441. *ISBN*: 0897917855. *DOI*: `10.1145/237814.237991`. *URL*: `https://doi.org/10.1145/237814.237991`.

Stadler, P. F. (1995). "Towards a theory of landscapes". In: *Complex Systems and Binary Networks*. Ed. by R. López-Peña, H. Waelbroeck, R. Capovilla, R. García-Pelayo, and F. Zertuche. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 78–163. *ISBN*: 978-3-540-44937-9.

Stadler, P. F. (1996). "Landscapes and their correlation functions". In: *Journal of Mathematical Chemistry* 20.1, pp. 1–45. *DOI*: `10.1007/bf01165154`. *URL*: `https://doi.org/10.1007/bf01165154`.

Stadler, P. F. (2002). "Fitness landscapes". In: *Biological Evolution and Statistical Physics*. Ed. by M. Lässig and A. Valleriani. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 183–204. *ISBN*: 978-3-540-45692-6. *DOI*: `10.1007/3-540-45692-9_10`. *URL*: `https://doi.org/10.1007/3-540-45692-9_10`.

Stadler, P. F. and R. Happel (May 1999). "Random field models for fitness landscapes". In: *Journal of Mathematical Biology* 38.5, pp. 435–478. *DOI*: `10.1007/s002850050156`. *URL*: `https://doi.org/10.1007/s002850050156`.

Struyf, A., M. Hubert, and P. Rousseeuw (1996). "Clustering in an Object-Oriented Environment". In: *Journal of Statistical Software* 1.4. *DOI*: `10.18637/jss.v001.i04`. *URL*: `https://doi.org/10.18637/jss.v001.i04`.

Thouless, D. J., P. W. Anderson, and R. G. Palmer (1977). "Solution of 'Solvable model of a spin glass'". In: *The Philosophical Magazine: A Journal of Theoretical Experimental and Applied Physics* 35.3, pp. 593–601. *DOI*: `10.1080/14786437708235992`. eprint: `https://doi.org/10.1080/14786437708235992`. *URL*: `https://doi.org/10.1080/14786437708235992`.

Watkins, D. S. (1991). *Fundamentals of Matrix Computations*. USA: Wiley. *ISBN*: 0471614149.

Welsh, D. (1993). *Complexity: Knots, Colourings and Countings*. London Mathematical Society Lecture Note Series. Cambridge University Press. *DOI*: `10.1017/CBO9780511752506`.

Williamson, D. P. and D. B. Shmoys (2011). *The Design of Approximation Algorithms*. 1st. USA: Cambridge University Press. *ISBN*: 0521195276.

Zdeborová, L. and F. Krząkała (Sept. 2007). "Phase transitions in the coloring of random graphs". In: *Phys. Rev. E* 76 (3), p. 031131. *DOI*: `10.1103/PhysRevE.76.031131`. *URL*: `https://link.aps.org/doi/10.1103/PhysRevE.76.031131`.

Curriculum Scientiae

Personal Information

Name	Angel E. Rodriguez Fernandez
Birth	July 18, 1991
Birthplace	Puebla, Mexico

Education

since 5/2017	PhD Student Bioinformatics Group Universität Leipzig
2014–2016	MSc. Computer Science Computer Research Center (CIC) of the National Polytechnic Institute (IPN). • Thesis: “Structure driven SDP based algorithms for the Max-Cut problem”. • GPA: 98/100.
2009–/2014	BSc. Physics National Autonomous University of Mexico (UNAM). • GPA: 95.2 / 100.

Academic Achievements

2013	International Internship At the University of California San Diego (UCSD) from January to July.
2012-2014	SNI Researcher Assistant CONACyT Research grant on acoustic properties of helicoidal elastic materials.
2015	National School of Optimization and Numerical Analysis (ENOAN) Participation with the lecture “Characterization of Semidefinite Programming Problems”.

Academic Achievements (continued)

2016	Research Assistant for UC MEXUS - CONACYT CN15-1451 Work in the project “Semantic routing for the Internet of Everything”.

Working Experience

01/2015–07/2015	Teaching Assistant CIC-IPN • Lecture: Randomized Algorithms
01/2016–07/2016	Teaching Assistant CIC-IPN • Lecture: Randomized Algorithms

IT-Knowledge

Programming:	Fortran, Python, Java, Matlab

Languages

Spanish:	Native speaker
English:	Fluent
German:	Conversational

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialen oder erbrachten Dienstleistungen als solche gekennzeichnet.

(Ort, Datum)

(Unterschrift)