MASTER'S DEGREE THESIS

MSc in Applied Mathematics

Title: Overlapping Community Search in Very Large Graphs

Author: Arnau Padrol Sureda

Advisors: Julian Pfeifle & Victor Muntés Mulero

Departments: Matemàtica Aplicada II
Arquitectura de Computadors

Facultat de Matemàtiques i Estadística

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Contents

Acknowledgements 5

Chapter 1. Introduction 7
1. Our world is made of graphs 7
2. Communities in graphs 9
3. What does community mean? 10
4. Size matters 12
5. Objectives and Contributions 12
6. The structure of this Master Thesis 13

Chapter 2. State of the art 15
1. Partitions 15
2. Overlapping 18

Chapter 3. Vector Representations 21
1. The search space and community structures 21
2. The sinks strategy 22
3. Why could vectors help us? 24
4. (c, d)-Vector representations 25
5. On the possible graph representations 30
6. The smallest eigenvalue 33

Chapter 4. Objective functions 37
1. Norms are not enough 37
2. Towards a good objective function 40
3. Projected vector representations 41
4. Sinks from Increment Orientations 46
5. The directed Laplacian 54
6. The conductance 59

Chapter 5. Analysis of the sinks 61
1. Cartesian product of cliques 61
2. The d-cube 63
3. A random graph 67
4. Qualitative differences between $\mathcal{L}$ and ϕ 73

Chapter 6. Community detection: the algorithm 75
1. LOCA 76
2. OCA: Global community search 77
3. Implementation details 79

Chapter 7. The results 83

1. The gallery of graphs tested 83
2. Quality Analysis 84
3. Local Search Analysis 93

Chapter 8. Conclusions and Future Work 95

Bibliography 97

Acknowledgements

First of all I must thank my co-directors, Julian and Victor, who guided me through this long journey, and worked hard to make this thesis possible. They are both great researchers but above all they are even greater people, and had enough patience to teach me not only what to do, but how to do it.

Then, I am happy to particulary thank Guillem Perarnau, who not only worked joint with me to get some of the results presented in the thesis, but was a really pleasant company for the bunch of hours we spent together.

I am also very grateful to "the other" Victor and the rest of members of AETFM(E), without whom I would not have survived to all the paperwork and LaTeX fighting. And even if I will not mention them all because there are too many and I would surelly forget somebody inexcusable, I must thank all the friends at *fme* who gave me support and brought me coffees when I needed it the most.

Thanks to my flatmates and my family, who feeded me when I could only concentrate in typing.

Finally, I am specially grateful to Laura, who deserved a much bigger portion of me and my time.

CHAPTER 1

Introduction

1. Our world is made of graphs

A graph consists of a set of elements called vertices and a set of binary relations between its vertices called edges. Back in 1736, Leonard Euler published *Solutio problematis ad geometriam situs pertinentis* solving the famous problem of the *Seven Bridges of Königsberg* (see Figure 1.1). It is considered as the first apparition of the concept of graph and the birth of graph theory. Since then, graphs have become a key piece in mathematics because their natural appearance in many different fields of study. We will try to show some of these situations by travelling through graph theory story.

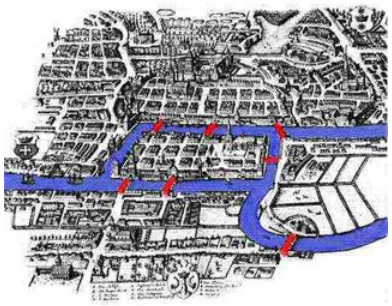


Figure 1.1: Is there a walk through the city that would cross each bridge once and only once?

Euler did not use the term graph when he published his article. But even if graphs did not have a name, they were used anyway, and graph theory evolved. For example, in 1845 the physicist Gustav Kirchhoff combined algebraic techniques with graph theory and published his circuit laws, which are now basic in the study of electrical circuits.

It is believed that the first apparition of the word *graph* in this context is in a paper in 1878 called *Chemistry and Algebra* [**49**] by James Sylvester, who was interested in the diagrams representing molecules. And it is also from chemistry the origin of the actual dots and lines diagrams we use to represent graphs. It is not a coincidence, because graphs are widely used in chemistry with atoms as vertices and chemical bonds representing edges. Graph theory concepts as planarity have important chemical consequences.

To present more graphs, we use one of the most famous graph-theory problems: the *Four Color Theorem*. It states that every map can be colored with four colours without having two adjacent regions with the same colour.

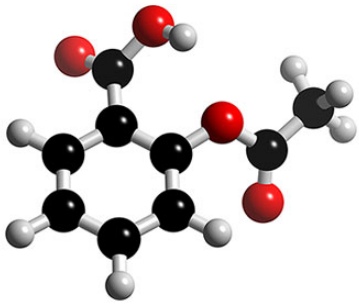


Figure 1.2: Molecule of Acetylsalicylic Acid(C9 H8 O4), a.k.a. Aspirin

This can be easily restated in graph terms: if nodes are regions and there is an edge whenever they share a border, we are asking if we can colour any planar graph with adjacent nodes having different colors. It was formulated in 1852 and remained an open problem until 1976, when it was proved by Kenneth Appel and Wolfgang Haken [**3**]. It was the first important proof that was based on computer calculations.

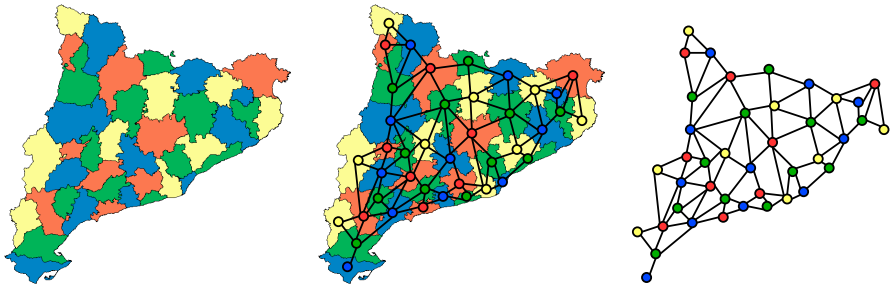


Figure 1.3: The map of Catalonia is 4-coloreable

And another interesting graph can be introduced with the *Erdős number*. Paul Erdős was a Hungarian mathematician who was specially prolific in terms of publishing papers and made key contributions in graph theory. He published around 1,475 articles with 511 different collaborators. Some of his friends created the *Erdős number*: Paul Erdős had Erdős number 0, and his co-authors have an Erdős number of 1. The people who have collaborated with them (but not with Erdős himself) have an Erdős number of 2, and those who have collaborated with people who have an Erdős number of 2 (but not with Erdős or anyone with an Erdős number of 1) have an Erdős number of 3, and so forth. A person with no such coauthorship chain connecting to Erdős has an Erdős number of infinity (or an undefined one). In fact, this is just the usual definition of graph distance in the co-authorship graph!

So, in these short highlights from graph-theory history, we have already seen graphs that have just emerged from some specific examples, but the graphs we will really be interested in did not exist yet! And that is because

in the history of graphs there had to be a new revolution that regained its interest: the birth of computers.

In fact it was not a revolution but two. First of all, graphs became larger, much larger. In fact, graphs did not become larger, but treatable graphs did. Before, we could only work with graphs consisting of few tens of nodes that one could draw by hand, while nowadays computers can efficiently handle graphs of billions of nodes.

But there was a second cause for the increase of the interest in graphs: new graphs were born. Because with computers, internet was born. Internet is nothing but a large graph, with computers and routers acting as nodes. And the world wide web is also a large graph, with webpages as nodes and hyperlinks as edges. For example, Google's sorting is based in some spectral graph theory properties [**45**]. Another example are social networks, as Facebook or Twitter, created with a clear graph-structure.

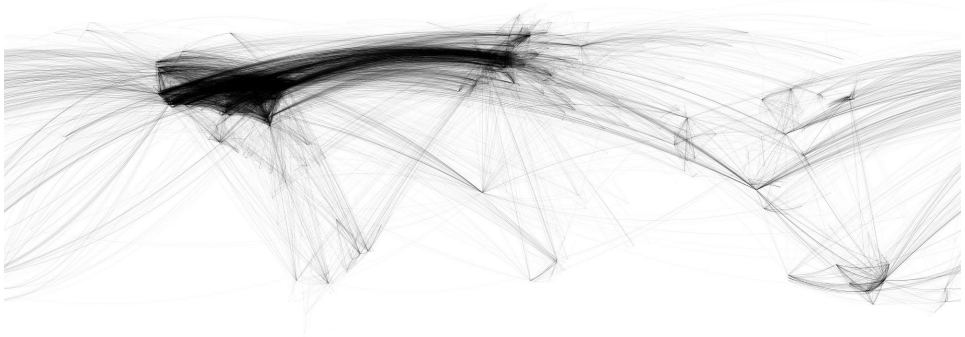


Figure 1.4: Representation of the Internet graph from *ChrisHarrison.net*

2. Communities in graphs

We have seen some graphs that naturally arise from our environment, and whose study is of great importance. But when we focus on their structure, we observe that they do not behave as a uniform random graph. In 1959, Erdős and Rényi proposed a model of random graphs [**17**], in which each pair of nodes have the same probability of having an edge between them. This leads to a uniform structure of all the graph. For example, their degree distributions follow binomial laws.

However, real networks do not behave like that, they do not show that regularity. Instead, it has been proven that many real networks have a *scale-free* structure, with a degree distribution that follows a power law: there are few vertices of huge degree and plenty of vertices of very small degree [**2, 4, 26**][1]. And not every pair of vertices are equally connected: there

[1]Despite the common assumption based in these publications that internet is a scale-free network, a very recent paper [**52**] states that this claim is false and presents an alternative model

are groups of nodes with higher density of edges among them, and without many connections within each other. This is the *community structure* of the graph.

Communities are groups of nodes that share some relation. For example, in the World Wide Web, communities will group web pages about similar topics. And in the coautorship network they would reflect different fields of study. In a social network they would show groups of related people. And so on.

The ability to understand the relationship among the different entities in the network is crucial in order to comprehend the underlying structure, extract patterns, summarize the structural information, etc. In large graphs containing millions of nodes, knowing the community structure can help to understand the real structure of the graph probably better than the graph itself, and is a starting point for graph summarization and visualization. Individuals in a social network may be grouped to better understand their interests, authors in a bibliographic graph database containing authors, papers and references may be grouped to automatically identify research areas.

And even the topological information of the node within its community is important: central vertices (nodes having a lot of connections within the group) might be important elements for the control and stability of the whole group; and boundary vertices (nodes with connections to many communities) might have an important role in mediation and exchange.

This leads to the need of finding algorithms that unveil the community structure of graphs. And these algorithms have to be efficient enough to deal with huge graphs, as nowadays graph databases contain billions of nodes.

To obtain such an algorithm will be our goal.

3. What does community mean?

One of the main problems when approaching community detection is the difficulty of giving a specific definition of what a community is. We all have some abstract intuition of what a community might be, at least in some particular examples. But the problem appears when trying to generalize these examples to a specific definition valid any graph. Several proposals have been done so far, with none really established. This situation has lead to a state where the most usual approach is to propose some algorithm and to define communities as the outcome of the given algorithm.

Even if we will not give a particular definition of community yet, we will bound it by providing some of the properties we think communities should have. While we will treat some of them as prerequisites that will determine the whole approach of the project, others will just be some secondary goals used in the analysis of the results.

Density. A community is a subset of the vertices that has high edge density. That is, nodes within a community must be well connected within each other. This is a clear imposition, as, along with isolation, it is the basis of the abstract concept of community we all have in our minds. The probability of having an edge between two nodes of a community is expected to be much higher than the probability of having an arbitrary edge.

Isolation. Communities are isolated sets, in the sense that they do not have as many links towards the outside world as they have in the inside. Although it is a quite normal parameter to think about, one must consider it with caution.

The reason resides at the possible contradiction between the concept of isolation and of overlapped sets (discussed at next point). Although edges within different communities are not common, if a node belongs to two communities, it must have edges towards both of them, what implies the apparition of outgoing edges for both communities! Thus, if we priorize isolation, we might obtain that those nodes that belong to several communities might not be taken by any of them.

Overlapping. The fact that communities overlap with each other is the key point of all this section, and the main motivation of all the project. So we need to put some emphasis into motivating the fact that communities from real networks do overlap in a natural way, and therefore it is important to be able to detect overlapped communities.

Main approaches from literature give methods for partitioning the vertices of a graph, and there are several different proposals of how to evaluate a partition and how to find it. However, particular examples of real networks show quite clearly the presence of nodes belonging to many communities, and where not dealing with this overlap implies a loss of quality. This is why we impose to our algorithm to be able to deal with overlapping communities.

For example, if we examine the structure of communities around a single person, we will probably find out his family, then his job, his friends, etc. and he will belong to all these communities (and not including him in them would be an error).

Another example would be any scientific collaboration network, where communities could be used to classify papers into fields of study. Interdisciplinary papers belong to more than a community just by definition. Take this master thesis, it belongs to mathematics, but it can as well be considered as a computer science article. The point is that it does not belong to only one of those, but both.

Hierarchy. Communites form a hierarchical structure: small communities group up to form larger communities, which also merge to create larger communities, and so on. There is a full organization of communities nested into other communities.

Locality. A community is a local concept in the sense that it should just take into consideration small neighborhoods. A node shares community with nodes that are at short distance of it.

This has two implications in the algorithms that have to detect communities. First, parameters that determine whereas a set is a community or not should be local, they should not depend on the full graph, neither on the other communities. In a friendship network this is reflected as follows. Every group of friends has a different degree of attachedness, but that does not imply that less attached groups are not groups: a group of teenagers surely call each other more than I do with my friends, but that does not mean that I don't have friends!

The second consequence is that in sparse small-world graphs (as those we expect to study), the complexity of finding a sole community should depend only on the properties of the community itself, independently of the size of the whole graph.

Naturalness. Finally, we requested that the algorithms that unveiled the community structure of the graph, they should not rely on prefixed parameters or constraints as the number of communities or their size. Communities should arise naturally from the data of the graph, and we should not guess a parameter a priory, but get them as result of our algorithm.

4. Size matters

Before getting deeper in the matter, we want to stress still one more time that our object of interest are very large graphs. The larger the better. And that has important consequences in the decisions we will have to make all over the thesis.

We will not be able to afford non efficient algorithms, even if they provided better results. Working with graphs of millions of nodes, even quadratic time algorithms are out of mind.

This need for efficiency will translate not only in terms of time, but of memory: probably the graph will not even fit in the computer's memory! We will not be able to store sophisticated data structures if they are not small.

5. Objectives and Contributions

The main objective of the thesis is the creation of an algorithm to detect the community structure of large graphs, allowing for nestings and overlappings.

Although it has been shown that communities are usually overlapping and hierarchical, we must stress that most of the literature related to community search has focused on finding partitions of the graph. In addition, given the size of modern data sets, most of them typically rely on prohibitively expensive computations.

We will propose the algorithm OCA, an algorithm for community detection with nestings and overlaps. It has been able to run in the larger datasets of which we are aware. Our algorithm neither requires the user to set non-intuitive parameters in order to get good results, nor to preassume a certain size or number for the communities, since they are found naturally from the graph structure.

The core of the algorithm relies on the definition of a new fitness function that allows to evaluate the quality of a community naturally including those nodes shared by other communities.

First, we implicitly map the graph into a higher dimentional real space. Then, we apply the *Directed Laplacian*, an new operator that acts on directed graphs and that is presented in this thesis. The study of the vectorial representation is another of the objectives, as well as the theoretical study of the fitness function.

Some of the results of this thesis will be found in a paper writen jointly by the directors, Guillem Perarnau and myself [**38**]. It is not published yet.

6. The structure of this Master Thesis

What do we mean for community and why it is important to find them are issues we have just discussed in this introductory chapter. From now on we will explore different paths to reach our objective of finding a community detection algorithm. In Chapter 2, we introduce the main solutions given so far in the area, and expose why a new method is needed.

The general framework of our proposal is based in *objective functions*. And it is in this context that one of the pillars where the thesis is supported appears: *Vector representations*. Because we will try to build objective functions on vector representations. Both concepts are presented in Chapter 3.

The following two chapters are devoted to the process of obtaining objective functions from vector representations. At the end of Chapter 4 we present our objective function the *Directed Laplacian*. The previous sections in the chapter are devoted to other possible definitions that did not work. However, they are relevant for many reasons. First, they provide the intuition necessary to reach the final function. Moreover, they answer the compulsory question of why simpler approaches do not work. And finally, they provide ideas that, even if they do not work for us, they may be applicable in other contexts.

A theoretical analysis of the *Directed Laplacian* in front of a well known objective function, the *conductance*, can be found in Chapter 5.

Finally, all the ideas developed to that point materialize in Chapter 6, where the algorithm itself is presented. The results of the application of the algorithm on real networks are given in the consecutive chapter. Leaving Chapter 8 to state the conclusions and the future lines of research that can be derived from this work.

The idea of this project is to have a tree-like structure: along the way towards the objectives, many branches will appear. Some of them will be followed to the end and some will lead to a dead-end, some will be left hanging in the air until they are recovered later on, and finally some will just open some doors for future exploration. To reinforce this idea, we will introduce a new kind of observation, called *Follow up*, and is just used to present conjectures or ideas that may be interesting but have not been dealt with yet, and that do not have room in this master thesis.

CHAPTER 2

State of the art

Most of the approaches to community detection are focused on obtaining complete partitions of the vertex set. While first partition algorithms where proposed in 1970's, Newman and Girvan proposed in [**19**] a new approach, which renewed the interest into graph partitioning to became a hot research theme at the current moment. Their proposal is still one of the most established methods in graph partitioning. Despite the amount of different proposals this article triggered, the great majority of them do not admit the presence of overlappings between the communities. We will do an overview of the principal existing partitioning methods as well as the existing methods that do allow for overlaps.

An interesting survey on community detection can be found at [**18**].

1. Partitions

1.1. Classical approaches.

Cut Size. The first approaches of graph partitioning stated the problem into the following way: Partition a given graph G in a predefined number of groups g of predefined size, such that the number of edges between groups, the *cut-size* is minimal. With this approach, it is unavoidable to prefix the number of communities, as otherwise only the trivial partition putting all nodes in a sole group would be the one to be returned.

In general, graph partitioning problems are NP-hard, even if there are algorithms that give good approximations [**41**]. Most of the algorithms focus into bisecting the graphs, and then apply it recursively to obtain partitions into more groups.

Hierarchical clustering. Many of the classical clustering approximations, principally from Social Science, followed the structure of Hierarchical clustering. This structure is based in the definition of a similarity criterion between vertices that gets encoded in a *similarity matrix* M. Then, from an initial partition in which each node is a community, iteratively, two communities get merged at each step based in the information of M. Specifically, from the empty graph, the edge with greatest similarity is added until two disconnected components get connected.

The output gets summarized in a *dendogram*, a tree that has the nodes of the graph at the lowest level and where at each level it is represented a step of the algorithm in which two communities are merged. Then, each slice of the dendogram is a partition of the nodes of the graph (Figure 2.1).

1.2. Divisive algorithms: Newman and Girvan. Instead of iteratively adding edges in order to merge communities as in hierarchical clustering, divisive algorithms do the opposite: they iteratively remove edges

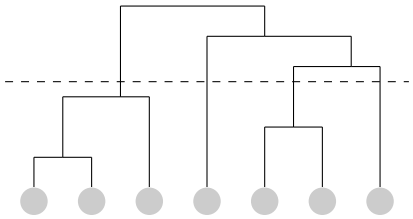


Figure 2.1: Example of dendogram

to split communities. As with hierarchical clustering, the result of these algorithms is a collection of partitions from a dendogram.

The most popular of these algorithms is the one proposed by Newman and Girvan at [**19**], and which is based in the concept of *edge betweenness*. We define the betweenness of an edge as the number of shortest paths between vertices of the graph that go through it.

Algorithm 1 Girvan-Newman algorithm

```
1: procedure NEWMAN-GIRVAN(G)
2:     G: Input graph
3:     while G has edges do
4:         Calculate the betweenness for all edges in the net- work.
5:         Remove the edge with the highest betweenness.
6:         Recalculate betweennesses for all edges affected by the removal.
7:     end while
8: end procedure
```

The betweenness of all edges of the graph can be calculated in time $\mathcal{O}(nm)$, where n and m are the number of nodes and edges of the graph respectively. Therefore, the algorithm has a cost of $\mathcal{O}(nm^2)$ which make it applicable to sparse graphs of at most 10000 nodes.

1.3. Modularity optimization. One of the major drawbacks of previous methods was that they provided a large number of hierarchical partitions without having no established method to discriminate between them and give out the best one. In [**34**], a solution to this is proposed: select the best partition in terms of *Modularity*. The modularity $\mathcal{Q}$ of a given partition of a graph is

$$\mathcal{Q} = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(C_i, Cj) \tag{2.1}$$

where A is the adjacency matrix of the graph, $k_i = \deg(i)$ is the degree of the vertex i, C_i is the community to which the vertex i belongs and δ is the usual Kronecker delta, that is one if the two communities are the same and zero otherwise. Observe that $\frac{k_i k_j}{2m}$ is the expectancy that nodes i and j were connected if edges were distributed randomly respecting the degrees of the nodes. Thus, modularity indicates how many inter-community edges exceed the expected number.

Modularity is considered a good criterion to indicate the quality of a partition, therefore many approaches have been done in order to find the partition with maximal modularity in a graph, which is indeed an NP-hard problem [**7**].

The first of these approaches, in [**35**], consists in a greedy algorithm that, in the style of hierarchical clustering, iteratively merged the communities such that the resulting partition had maximal modularity. Their first algorithm was $\mathcal{O}((m+n)n)$, significantly faster that previous proposals. In [**13**] the algorithm is improved to achieve a complexity of $\mathcal{O}(md\log n)$, where d is the depth of the community dendogram (which usually is of order $\mathcal{O}(\log n)$).

Other heuristic search methods have been used for the same purpose, as *Simulated Annealing* or *Extremal optimization* in [**16, 22**].

We remark that all these methods just work with partitions and do not allow for overlappings.

1.4. Spectral Algorithms. Many algorithms for graph partitioning are based in spectral properties of the graph.

The first technique is *spectral bisection method* [**41**], which is based in the properties of the *Laplacian matrix* of the graph. We define the Laplacian matrix L as follows:

$$L = \begin{cases} \deg(i) & \text{if } \ i = j \\ -1 & \text{if } i \neq j \text{ and } \{i,j\} \in E \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

The smallest eigenvalue of the Laplacian matrix is always 0, corresponding to the eigenvector $\mathbf{1} = (1, \ldots, 1)$.

Given a partition of $G = A \bigsqcup B$, let s be the vector that has $s_i = 1$ if $i \in A$ and $s_i = -1$ if $i \in B$. And let R be the number of edges between the two groups, then we have the expression

$$R = \sum_{\substack{i,j \text{ in} \\ \text{different} \\ \text{groups}}} A_{ij} = \frac{1}{4} s^T L s$$

If we now express s in an eigenvector orthogonal basis of L $v_1, \ldots, v_i$, and we normalize it so that $ss^T = |V|$, then $s = \sum_i a_i v_i$, and we observe that $R = \sum_i a_i^2 \lambda_i$, where λ_i is the eigenvalue corresponding to v_i. This observation has lead to a bunch of partitioning algorithms.

The spectral bisection method consists in computing the first non-trivial eigenvector, the *Fiedler vector* v_2, and, fixed the sizes of the groups, n_1 and n_2 ($n_1 + n_2 = |V|$), taking the n_1 smaller components of v_2 into a group and the rest into the other.

This method has the big inconvenient of having to predefine the number of communities and their size. However, there are recent spectral algorithms more sophisticated that do not need this input. In concrete, in [**15**], nodes are mapped into a Euclidean space via the coordinates of the Laplacian eigenvectors.

Other spectral algorithms have been proposed using the spectrum of other alternative matrices [**8, 36**].

Their main problem is their complexity.

1.5. And many more. There are other techniques like dynamic algorithms [**40**] based on random walks, Markov clustering or maximum likelihood algorithms [**37**], applied to graphs of size 10^6.

Also, attempts to reduce the complexity of the graph clustering problem and improve performance have been made [**14, 42**], but only for disjoint communities where each node is assigned to a single community.

2. Overlapping

There have also been some recent approaches into searching for community structures with overlaps. We present them now.

2.1. Clique Percolation. The first attempt to uncover the overlapping community structure of a graph of which we are aware is at the article by *Palla et. al* published at *Nature* in 2005 entitled *"Uncovering the overlapping community structure of complex networks in nature and society"* [**39**].

Their assumption is that a community is usually formed by the union of several complete subgraphs. Then they define a k-clique community as a union of all k-cliques (complete subgraphs of size k) that can be reached from each other through a series of adjacent k-cliques (where adjacency means sharing $k-1$ nodes). This "sliding" of cliques is what they call *clique percolation*. This procedure naturally allows for overlaps.

The computational basis consist in searching first for the maximal cliques. This is a NP-complete problem. However, from their computational experience, the authors found out that they could indeed run their algorithm in real networks up to 30000 nodes.

2.2. Local optimization. The second approach to overlapping community finding is due to *Lancichinetti, Fortunato and Kertesz*. In [**28**] the authors present an efficient way to find overlapped communities based on a randomized approach.

Their method is based in locally maximizing the function $f_{\mathcal{S}} = \frac{k_{in}^{\mathcal{S}}}{(k_{in}^{\mathcal{S}}+k_{out}^{\mathcal{S}})^{\alpha}}$, where $k_{in}^{\mathcal{S}}$ and $k_{out}^{\mathcal{S}}$ are the total internal and external degrees of the nodes of module $\mathcal{S}$, and α is a positive real valued parameter, controlling the size of the communities.

The dependence on the parameter α is the main drawback of this algorithm. The choice of α must be done a priory, and may have a significant impact on the quality of communities.

2.3. EAGLE. In [**44**], the authors propose a new algorithm: EAGLE.

As in clique percolation, the first step consists in finding all the maximal cliques. This are they initial communities. Now, iteratively at each step they merge the pair of communities that have maximal *similarity* until only one community remains. Here similarity is an adapted version of the modularity at (2.1) that allows for overlapping communities.

As the initial nodes do have overlaps, these are maintained all over the process giving up to overlapping communities.

Once again, the computational cost is an insuperable drawback for this proposal, which after searching all the maximal cliques, apply two stages of cost $\mathcal{O}(n^2 + (h+n)s)$ and $\mathcal{O}(n^2 s)$, where n is the number of nodes, s the

number of maximal cliques and h the number of maximal cliques that are neighbours.

CHAPTER 3

Vector Representations

To recover the communities, we will go through an intermediate stage where we will map our graph into some vector space, substituting nodes by vectors. But first of all we will spend two sections to introduce some basic definitions and to settle down the basis of our proposal.

1. The search space and community structures

We want to find algorithms that retrieve the community structure of a given graph, but we do not know yet what a community structure really is! So we will first of all proceed to formalize this concept. We will as well provide the framework for the community search, based on a "generic" community definition.

DEFINITION 3.1 (Community structure). *A* community structure *on a graph $G = (V, E)$ is a collection $\mathcal{C} \subset 2^V$ of subsets $S \subset V$. Each of these will be called a* community.

REMARK 3.1. *We do not require that all nodes belong to a community, i.e., we do not ask to the community structure to cover all nodes. There might be nodes not relevant that are not taken, and if required can be considered as communities of a single element.*

We also allow that some community is contained inside another one. Hence, they do not need to form an antichain. Allowing nestings is coherent with the concept of the hierarchical structure.

To obtain the community structure, we will proceed to search every community individually, and return the collection of all these resulting communities as the community structure.

Every community belongs to the lattice of subsets of V, which is indeed the search space. We capture this concept in the following definitions.

DEFINITION 3.2. *We define Γ_G ,the search space associated to a graph $G = (V, E)$, as a graph with a vertex for every subset of $|V|$ and a link from A to B if and only if there exists a node $v \in V$ such that $B = A \cup v$ or $A = B \cup v$.*

Whenever G is obvious from context, we will just use Γ to denote Γ_G.

We observe that, independently of the graph G, Γ_G is $Q_{|V|}$, the hypercube graph of dimension $|V|$.

DEFINITION 3.3. *Given a graph G the directed graph $\Gamma_G^{\uparrow}$ corresponds to Γ_G where edges are directed from smallest to larger subsets.*

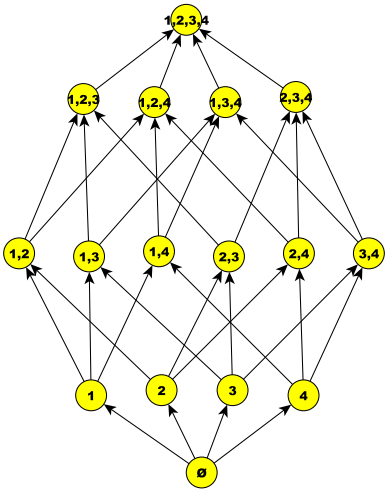


Figure 3.1: $\Gamma^{\uparrow}_{G} = Q_4$ of any graph G with 4 vertices

2. The sinks strategy

The strategy we will adopt to allow for community overlappings is based on objective functions. The objective function will be used to evaluate the "community goodness" of a group of nodes. A community will be a group such that neither adding or removing a node increases the value of the objective function, hence we will define communities as local maxima of "community goodness". Now, the definition community is strongly lied to the definition of the objective function. This approach will provide the general framework, and leads to the seek of the best objective function.

Definition 3.4 (Objective function). *Given a graph $G = (V, E)$, an objective (or fitness) function $\phi : 2^V \to \mathbb{R}$ is a map from subsets of vertices to real numbers.*

Once we have an objective function ϕ, the search space becomes concrete and can be seen as a directed graph.

Definition 3.5. *Given a graph G, and an objective function ϕ, the graph $\Gamma_G(\phi)$ is the directed graph induced from ϕ. Namely, an edge $\{A, B\}$ is directed from A to B if $\phi(A) < \phi(B)$. In the case $\phi(A) = \phi(B)$, the edge is directed from smaller subset to the larger.*

Observation 3.1. *When we use an objective function, we are dealing with* two *distinct orientations of Γ: the underlying orientation $\Gamma^{\uparrow}$ where the fitness function ϕ is defined on, and $\Gamma(\phi)$, the one induced by this objective function.*

To avoid confusion, we will refer to ascending and descending edges when we talk about the orientation at $\Gamma^{\uparrow}$, projecting the idea of a layered representation of $\Gamma^{\uparrow}$ as in Figure 3.1. And we will refer to increasing and decreasing edges when dealing with $\Gamma(\phi)$, reinforcing the definition that orients edges depending on the increases in the fitness function.

Now, a sink in $\Gamma_G(\phi)$ is a node with outdegree 0. In Γ_G, a node representing a set A is connected to all the nodes of the form $A \cup v$ or $A \setminus v$ with $v \in V$, so if A is a sink we have that $\phi(A) \geq \phi(A \cup v)$ and $\phi(A) \geq (A \setminus v)$ for all $v \in V$, and therefore A is a community, following the local maxima definition.

DEFINITION 3.6. *Given a directed graph D, a node v is a* sink *if its outdegree is zero. If the direction on D is induced by a function ϕ, then sinks are local maxima of ϕ.*

OBSERVATION 3.2. *As the orientation in $\Gamma_G(\phi)$ is induced by a real function, that means that it is acyclic. Therefore, there exists at least one sink.*

PROOF. First, we see that the orientation is acyclic. Let $a_1 \to a_2 \to \cdots \to a_{n-1} \to a_1$ be a cycle in $\Gamma_G(\phi)$, then by definition $\phi(a_1) \leq \phi(a_2) \leq \cdots \leq \phi(a_{n-1}) \leq \phi(a_1)$ and all inequalities become equalities.

Now we have $\phi(a_1) = \phi(a_2) = \cdots = \phi(a_{n-1}) = \phi(a_1)$, let a_i be the smallest set in the cycle, and let $k = |a_i|$. By the construction of Γ, all neighbors of a_i have size $k \pm 1$. As a_i is the smallest in the cycle, that means that in concrete a_{i-1} has size $k + 1$. But now $\phi(a_i) = \phi(a_{i-1})$ and $|a_i| < |a_{i-1}|$, so the direction of the link is $a_{i-1} \leftarrow a_i$, that contradicts the definition of the cycle.

To see that we have at least one sink, we use the following algorithmic proof: We start in a random node. If the node is a sink we are done. Otherwise there is at least an outgoing edge leading to a new node, we move to that node. We keep moving until we reach a sink.

The algorithm above will end, because as the graph is acyclic, we will not repeat any node, so we must end our walk due to the finiteness of the number of nodes of Γ.

□

DEFINITION 3.7 (ϕ-community). *A ϕ-community is a subset S of V corresponding to a sink in $\Gamma_G(\phi)$.*

OBSERVATION 3.3 (Implications of this approach). *Looking for local optima in the full lattice of subsets carries naturally some implications, which obviously will finally get reflected with different strength depending on the concrete objective function used.*

The first of this implications is the allowance for overlaps. Regarding each possible subset on its own permits the apparition of sinks that represent communities that share nodes. Depending on the objective function, this situation will be more or less frequent, but in any case the framework will not obstruct its apparition.

Analogously, it is also feasible that two nested subsets are both sinks, bringing into place hierarchically included communities. In fact, hierarchies and nested subsets are just regarded as an overlap of an entire community.

Another aspect that is worth commenting is the tendency of considering a community all the unions of smaller communities. Although it strongly depends on the objective function, it is a normal tendency. If A and B are sinks, that means that adding or removing any node to A or B just makes the community worse. Then, in general adding a node or removing it from

$A \cup B$ will generally make this community worse. Notice that this can mean the apparition of even disconnected sinks!

So now we finally have all the tools and definitions we need to start the community search. Our quest will consist in finding good objective functions, that both give good results (in the sense that returns communities that match our intuitive community definition) and are computationally affordable.

We will derive our own objective functions from vector representations (introduced in the following sections) and study the already established function of *conductance* [**25**].

3. Why could vectors help us?

To define a fitness function for recovering the community structure of a graph $G = (V, E)$, we will go through an intermediate step in which we will embed it into a real vector space $\mathbb{R}^d$. We will identify each vertex $v \in V$ with a vector in $\mathbb{R}^d$. These vectors will be the vector representation $\mathcal{V}$ of G. To emphasize this identification of nodes with vectors, from now on we will deliberately abuse notation and use the same symbol to designate a node $v \in V$ and its corresponding vector $v \in \mathcal{V}$.

We will then expand a vector representation to the full collection of subsets of vertices 2^V. We will identify each set of nodes $S \subset V$ with the sum of the vectors representing its nodes, i.e. with $s = \sum_{v \in S} v$. We will often also abuse notation here, and identify the set with the vector. For example we might use $\|S\|$ instead of $\|s\|$, or $|s|$ instead of $|S|$.

First of all, we will introduce some hints to motivate why using vector representations seems a reasonable idea.

The key will consist in finding representations that somehow reflect the topology of the graph. In concrete, we will look for representations where adjacent nodes form more acute angles than disconnected nodes. The following examples will illustrate their power.

EXAMPLE 3.1. *Let $\mathcal{V}_1$ and $\mathcal{V}_2$ be two sets of k unitary vectors, such that vectors in $\mathcal{V}_1$ are pairwise orthogonal and in $\mathcal{V}_2$ all pairs of vectors form an acute angle α. And consider the vectors $s_1 = \sum_{v \in \mathcal{V}_1} v$ and $s_2 = \sum_{v \in \mathcal{V}_2} v$. Later on we will see that $\mathcal{V}_1$ will correspond to an independent set of size k and $\mathcal{V}_2$ to a k-clique, and thus s_1 and s_2 will be their vector representations. Just study the norm of these two vectors.*

The first one is easily seen to have norm

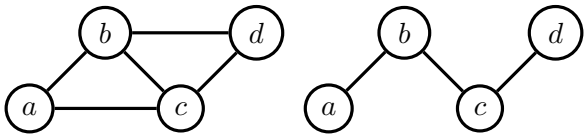
$$\|s_1\|^2 = k.$$

On the other hand, if we set $c = \cos\alpha$, some calculus allows us to see that

$$\|s_2\|^2 = c\,k^2 + (1 - c)\,k = \Theta(k^2).$$

The sum of vector sets that form more acute angles within each other have longer norm! In particular, the different behavior becomes more and more striking as c grows, that is, as α becomes acuter.

EXAMPLE 3.2. *We exemplify the concept of vector representations with two graphs: H_1 and H_2, presented in Figure 3.2.*



(a) H_1 (b) H_2

Figure 3.2: Graphs H_1 and H_2

Figure 3.3 shows a vector representation of the graph H_1. Since b and c are connected in H_1, but a and d are not, the angle $\angle(b, c)$ between b and c is smaller than $\angle(a, d) = \frac{\pi}{2}$, so $b + c$ is longer (further away from 0) than $a + d$.

We now compare H_1 with H_2. In this concrete representation, two vectors are orthogonal iff the corresponding nodes are not connected. As H_1 is more connected than H_2, the vectors representing it are closer together than those of H_2. Thus, if H_1 and H_2 are subgraphs in a larger graph G, the sum of all vectors corresponding to nodes in H_1 will be longer than the sum for H_2.

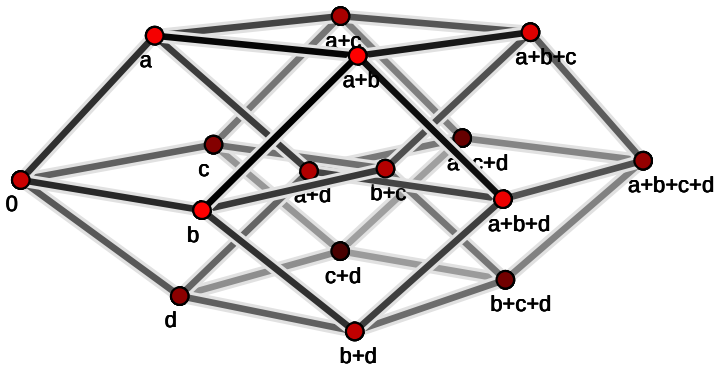


Figure 3.3: A vector representation $\{a, b, c, d\}$ of H_1, together with the search space formed by all sum vectors.

These examples show which is the track we want to follow: if we represent sets of vertices with the sum of the corresponding vectors, denser sets will get longer vectors. We will hence be able to establish a relation between having large norm and being a community.

But there is plenty to do yet. First, we need to formalize the concept of vector representations, and then try to find any (if they exist!). And then we will still have to see how to define a proper fitness function from them, because we will see that the norm will not be good enough.

4. (c, d)-Vector representations

We will work with a particular kind of vector representations, (c, d)-vector representations, that have the particular property that vectors can

have only two different angles between them, and where the angle between adjacent nodes is acuter that between disconnected nodes. It gets formalized in the following definition.

DEFINITION 3.8. *Given a graph $G = (V, E)$ we define a (c, d)-vector representation of G as a family of vectors $\mathcal{V} \subset \mathbb{S}^{d-1} \subset \mathbb{R}^d$ as well as a map $\nu : V \to \mathcal{V}$, that fulfill:*

(a) $-1 \leq d < c \leq 1$

(b) For each pair of vertices $u, v \in V$,

$$\langle \nu(u), \nu(v) \rangle = \begin{cases} c & \text{if } \{u, v\} \in E \\ d & \text{if } \{u, v\} \notin E \end{cases}$$

We will often skip the map ν, because as we have already said, we will often directly identify v and $\nu(v)$.

Definition 3.8 can be restated in the following equivalent way: given a graph G, $\mathcal{V}$ is a (c, d)-Vector representation of G if and only if its vectors can be reordered so that their gram matrix is

$$G(c, d) = I_n + cA + d\bar{A} \tag{3.1}$$

where I_n is the $n \times n$ identity matrix, A the adjacency matrix of G and $\bar{A}$ the adjacency matrix of its complementary graph $\bar{G}$.

OBSERVATION 3.4 (Why do we restrict to only two possible values?). *There is the obvious part that the symmetry of this configuration will surely simplify the calculus and operations. However, it is not the most important reason. We have to keep in mind our final objective, which is the creation of an algorithm that computes the community structure of the graph. If we allowed having a different angle for each pair of vectors, we would have to store them, so we would need an storage of quadratic space on the number of nodes. Considering that we will apply the algorithm to graphs that may even not fit into the computer's core memory, it is out of question to store so much data.*

In contrast, if we use only two values, there is still the possibility that we can find the communities storing only the graph properties and not the vector representation. This will be indeed the actual case: we will never construct the vector representation explicitly, but we will regard it as a "virtual" embedding, which will serve only to deduce a fitness function in graph terms.

We formalize the extension of the vector representation to the full structure of subsets of vertices using sum vectors, namely

DEFINITION 3.9. *Let $\mathcal{V} \subset \mathbb{S}^{d-1} \subset \mathbb{R}^d$ with $\nu : V \to \mathcal{V}$ be a (c, d)-vector representation of a graph $G = (V, E)$, we define an extended vector representation as a set $\bar{\mathcal{V}}$ and a map $\bar{\nu} : 2^V \to \mathbb{R}^d$, such that $\bar{\nu}(s) = \sum_{v \in s} \nu(v)$ and $\bar{\mathcal{V}} = \bar{\nu}(2^V)$.*

Observe that this extension induces a mapping of the search space graph Γ into $\mathbb{R}^d$, even if it is not injective. This mapping can be clearly seen comparing Figures 3.1 and 3.3.

And finally, for completeness, we make the following definition.

DEFINITION 3.10. *The* sum polytope *of $\mathcal{V}$ is the convex hull*

$$Q \;=\; Q(\mathcal{V}) \;=\; \operatorname{conv}\left\{\bar{\nu}(T) : T \in 2^{\mathcal{V}}\right\}$$

of all sums of vectors in $\mathcal{V}$.

This is a linear projection $Q = \pi(Q_n) \subset \mathbb{R}^d$ of the vertices of an n-dimensional unit cube Q_n to d-space; one vertex of Q lies at the origin, while the n edges incident to that vertex are just $v_1, \ldots, v_n$. In particular, Q is a zonotope, and centrally symmetric about its barycenter. Moreover, Q is the *Minkowski sum* of the vectors in $\mathcal{V}$.

After these first definitions, it remains open the question of the existence of such vector representations. The following theorem by Lovász in 1979 [**31**] will provide a construction for a particular vector representation.

But first, a small lemma that will simplify proofs.

LEMMA 3.1. *A $n \times n$ matrix M is the Gram matrix of a set of vectors if and only if it is symmetric and positive semidefinite.*

PROOF. If M is a Gram matrix of a set of vectors, from the definition of inner product it has to be symmetric and positive semidefinite.

It is left to proof the reciprocal. Let M be a symmetric matrix, by the spectral theorem it diagonalizes in an orthonormal basis W, so that $M = W^T D W$ with $D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ a diagonal matrix with entries the eigenvalues of M. M being positive semidefinite implies that $\lambda_i \geq 0$ for all i. So now if we set $\sqrt{D} = \operatorname{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n})$, we have that $M = W^T \sqrt{D}^T \sqrt{D} W$. Therefore, the columns of $\sqrt{D}W$ span a family of vectors whose Gram matrix is M. □

THEOREM 3.1. *For any graph G, there exist $(c, 0)$-representations.*

PROOF. If G consists in n isolated nodes, then we can represent them by the canonical basis of $\mathbb{R}^n$. So assume that G has an edge. Let A be its adjacency matrix. As A is a real symmetric matrix, it has all its eigenvalues real: $\lambda_n \leq \cdots \leq \lambda_1$. As A has zero trace, its least eigenvalue λ_n is negative (because we assumed not all eigenvalues where zero). Now, let $B = I_n - \frac{1}{\lambda_n} A$. Observe that the eigenvalues of B are of the form $1 - \frac{\lambda_i}{\lambda_n}$ which are all non-negative. Therefore B is symmetric and positive semidefinite, and by Lemma 3.1 is the Gram matrix of a set of vectors $\mathcal{V}$.

If we set $c = -\frac{1}{\lambda_n} > 0$, we have that $\mathcal{V}$ is the desired representation. We will call it the *Lovász representation.* □

COROLLARY 3.1. *$c = -\frac{1}{\lambda_n} > 0$ is the maximum possible value among all possible $(c, 0)$-vector representations of G.*

PROOF. It suffices to study which are the possible values of c such that $G(c, 0) = I_n + cA$ is positive semidefinite.

If A has eigenvalues $\lambda_n \leq \cdots \leq \lambda_1$, then we know that $G(c, 0)$ has eigenvalues $\mu_i = 1 + c\lambda_i$.

Therefore 1 is the only eigenvalue of $G(0, 0) = I_n$, and it is clear that $c = 0$ is a feasible value. It corresponds to a configuration of unit vectors,

each of them orthogonal to all the others—in other words, an orthonormal basis.

As c grows away from 0, the influence of the adjacency matrix A will become greater, and its eigenvalues will become more dominant. As we have seen, some eigenvalues of A are negative. Since the eigenvalues of $G(c,0)$ depend continuously on the entries of A as monotone functions, while c increases there exists a moment in which some eigenvalue of $G(c,0)$ becomes negative for the first time. By what we have said above, $G(c,0)$ will then no longer represent a vector configuration.

It only remains to find the first $c > 0$ such that any $\mu_i = 1 + c\lambda_i$ vanishes, which clearly is $c = -\frac{1}{\lambda_n}$. □

OBSERVATION 3.5. *If we recall Example 3.1, we observe that we are interested in the larger possible c. As the largest c is, the clearer communities are.*

REMARK 3.2. *If $\bar{\lambda}_1 \geq \cdots \geq \bar{\lambda}_n$ are the eigenvalues of $\bar{A}$, then an analogous reasoning shows that there exists a $(0, -\bar{\lambda_1}^{-1})$-vector representation, and that $d = -\bar{\lambda_1}^{-1} < 0$ is minimal among all $(0,d)$-vector representations.*

Now that the existence of vector representations is proved, and we are sure that we will be able to go further, it is the moment to make a small pause before studying other representations different from Lovász'. Because it is a crucial step to be able to see how the structure of the original graph shows up in the vectors. Our objective is to translate some useful vector operations to properties of the original graph.

First we need to define some notation we will extensively use. Let $G = (V, E)$ be a graph, $v \in V$ and $S \subset V$. We define $\mathrm{nb}_S(v)$ the set of neighbors of v that belong to S, $\mathrm{E_{in}}(S)$ the internal edges of a set S and $\mathrm{E_{out}}$ the edges connecting S to the rest of the graph:

DEFINITION 3.11. *We define the following sets*

- $\mathrm{nb}_S(v) = \{w \in S \mid \{v,w\} \in E\}$
- $\mathrm{E_{in}}(S) = \{\{v,w\} \in E \mid v, w \in S\}$
- $\mathrm{E_{out}}(S) = \{\{v,w\} \in E \mid v \in S \ w \notin S\}$

So if we take $S \subset V$, $u \in V$ and name $s = \sum_{v \in S} v$, we can deduce the following formula for the inner product of a set with a node

PROPOSITION 3.1. *If S is a set of nodes, $s = \sum_{v \in S} v$ its vector representation, then for any node v the following equations hold*

(a) The scalar product between a set and a node is

$$\langle s, u \rangle = \begin{cases} 1 + c\,|\,\mathrm{nb}_S(u)| + d\,|S \setminus (u \cup \mathrm{nb}_S(u))| & \text{if } u \in S \\ c\,|\,\mathrm{nb}_S(u)| + d\,|S \setminus \mathrm{nb}_S(u)| & \text{if } u \notin S \end{cases} \tag{3.2}$$

(b) The norm of a set is

$$\|s\|^2 = |S| + 2c\,|\,\mathrm{E_{in}}(S)| + 2d\left(\binom{|S|}{2} - |\,\mathrm{E_{in}}(S)|\right) \tag{3.3}$$

(c) When $v \notin S$, we have

$$\|S \cup v\|^2 = \|s + v\|^2 = \|s\|^2 + 1 + 2c\,|\,\mathrm{nb}_S(v)| + 2d\,(|S| - |\,\mathrm{nb}_S(v)|) \tag{3.4}$$

(d) If $v \in S$,

$$\|S \setminus v\|^2 = \|s - v\|^2 = \|s\|^2 - 1 - 2c\,|\,\mathrm{nb}_S(v)| + 2d\,(|S| - 1 - |\,\mathrm{nb}_I(v)|) \tag{3.5}$$

PROOF. (a) Just from the definition we have If u belongs to S, we develop

$$\begin{aligned}\langle s, u\rangle &= \langle \sum_{v\in S} v, u\rangle = \sum_{v\in S}\langle v, u\rangle \\ &= \langle u, u\rangle + \sum_{v\in \mathrm{nb}_S(u)}\langle v, u\rangle + \sum_{v\in S\setminus \mathrm{nb}_S(u)}\langle v, u\rangle \\ &= 1 + c\,|\,\mathrm{nb}_S(u)| + d\,|S \setminus (u \cup \mathrm{nb}_S(u))|\end{aligned}$$

Otherwise, if u does not belong to S, an analogous reasoning yields the claimed formula.

(b) We prove it using the relation above

$$\begin{aligned}\|s\|^2 &= \langle s, s\rangle = \langle \sum_{v\in S} v, s\rangle = \sum_{v\in S}\langle v, s\rangle \\ &= \sum_{v\in S} 1 + c\,|\,\mathrm{nb}_S(v)| + d\,|S \setminus (v \cup \mathrm{nb}_S(v))| \\ &= |S| + c\sum_{v\in S}|\,\mathrm{nb}_S(v)| + d\sum_{v\in S}|S \setminus (v \cup \mathrm{nb}_S(v))| \\ &= |S| + 2c\,|\,\mathrm{E}_{\mathrm{in}}(S)| + 2d\left(\binom{|S|}{2} - |\,\mathrm{E}_{\mathrm{in}}(S)|\right)\end{aligned}$$

(c) Direct consequence of the Law of Cosines, $\|a + b\|^2 = \|a\|^2 + \|b\|^2 + 2\,\langle a, b\rangle$.

(d) Analogous to previous item.

□

OBSERVATION 3.6. *Equations* (3.2), (3.3), (3.4) *and* (3.5) *from Proposition 3.1 make it computationally feasible to define the squared norm of sum vectors as fitness function, as they do not need to store all the vector configuration but only use graph properties.*

Basically, as the Gram matrix of the representation depends on the adjacency matrix, all those operations based in inner products translate naturally into graph terms. Therefore, they do not depend on the vectors themselves. Note that this does not represent any problem to our objectives, quite the opposite, as the cause of why vector representations can be useful to us is only their relative position, which is captured by the gram matrix, not the exact vectors. Any isometry of the vector representation would be equally useful to our purposes.

As a consequence of Proposition 3.1, from now on we will represent the vector representations by their Gram matrices $G(c, d) = I_n + cA + d\bar{A}$. For

example, the Gram matrices corresponding to the graphs in Example 3.2 are

$$H_1(c) = \begin{pmatrix} 1 & c & c & d \\ c & 1 & c & c \\ c & c & 1 & c \\ d & c & c & 1 \end{pmatrix}, \quad H_2(c) = \begin{pmatrix} 1 & c & d & d \\ c & 1 & c & d \\ d & c & 1 & c \\ d & d & c & 1 \end{pmatrix}.$$

5. On the possible graph representations

We face the question: for which pairs (c, d) there exists a vector representation? This can only happen when $G(c, d)$ is a Gram matrix. Due to the symmetry of $G(c, d)$, from Lemma 3.1 this is when $G(c, d)$ is positive semidefinite, with all its eigenvalues non-negative. The set of valid values of c, d is thus the intersection of the two-dimensional *matrix pencil* $M = \{G(c, d) : c, d \in \mathbb{R}\}$ with C, the cone of real positive semidefinite $n \times n$-matrices. Therefore, it is a convex set.

To study the range of permissible scalar products, observe that the intersection $M \cap \partial C$ satisfies the equation $p(c, d) = 0$, where

$$p(c, d) \;=\; \det(I_n + cA + d\bar{A}),$$

because this vanishing locus traces the points where eigenvalues of $G(c, d)$ change signs. More precisely, $M \cap \partial C$ is the innermost oval of the real algebraic curve defined by p. See [**23, 30**].

We can explicitly determine four points of $M \cap \partial C$. For this, let $\lambda_n \leq \cdots \leq \lambda_1$ and $\bar{\lambda}_n \leq \cdots \leq \bar{\lambda}_1$ be the eigenvalues of A, respectively $\bar{A}$; since the trace of a symmetric matrix equals the sum of the eigenvalues, and A and $\bar{A}$ are traceless, it follows that $\lambda_n < 0 < \lambda_1$ and $\bar{\lambda}_n < 0 < \bar{\lambda}_1$ if G is neither empty nor complete. Now set $d = 0$ and $\lambda = -\frac{1}{c}$. Then

$$\det(I_n + cA) \;=\; c^n \det\left(\tfrac{1}{c} I_n + A\right) \;=\; \left(\tfrac{1}{\lambda}\right)^n \det(\lambda I_n - A),$$

so that the zeros of $p(c, 0)$ closest to the origin are $-\frac{1}{\lambda_1} < 0 < -\frac{1}{\lambda_n}$. Similarly, for $c = 0$ we obtain the roots $(0, -\frac{1}{\bar{\lambda}_1})$ and $(0, -\frac{1}{\bar{\lambda}_n})$.

Example 3.3. *The root* $(-\frac{1}{\lambda_n}, 0)$ *of* p *corresponds to the well-known* Lovász construction [**31**] *stated in Theorem 3.1 : vectors whose corresponding nodes are adjacent in* G *have scalar product* $-\frac{1}{\lambda_n} > 0$*, and non-adjacent ones are orthogonal.*

Observation 3.7. *The set* $M \cap C$ *is the intersection of the cone of semidefinite matrices with an affine space, thus we can use semidefinite programming on it. If we found a linear function whose optimum represented the best pair* (c, d)*, we could compute it efficiently with semidefinite programming techniques* [**6, 21, 51**].

5.1. Linear inequalities for permissible scalar products. In this section we try to understand the shape of $M \cap C$, by bounding it with semispaces.

To derive linear inequalities on the set of permissible scalar products, we recall that the $n \times n$ matrix $G(c, d)$ is positive semidefinite if and only if

$$\boldsymbol{\lambda}^T G(c, d)\, \boldsymbol{\lambda} \;\geq\; 0 \tag{3.6}$$

for all real *test vectors* $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^n$.

We will use test vectors $\boldsymbol{\lambda}$ that are incidence vectors of subsets of the vertices of G, i.e., $\lambda_i = 1$ if $i \in S$ and $\lambda_i = 0$ otherwise, where $S \subset \{1, \ldots, n\} = V$ is any nonempty subset.

We observe that with this kind of test vectors $\boldsymbol{\lambda}^T G(c,d)\,\boldsymbol{\lambda}$ corresponds to our definition of $\|s\|^2$. Hence, we can use (3.3) from Lemma 3.1, and we translate (3.6) to

$$|S| + 2c\,|\,\mathrm{E_{in}}(S)| + 2d\left(\binom{|S|}{2} - |\,\mathrm{E_{in}}(S)|\right) \;\geq\; 0,$$

Using $k = |S|$ and $m = |\,\mathrm{E_{in}}(S)|$, we rewrite this as

$$\frac{2m}{k}\,c + \left(k - 1 - \frac{2m}{k}\right) d \;\geq\; -1. \tag{3.7}$$

Denote by $\ell(k,m)$ the affine line in the (c,d)-plane where this inequality is sharp. For fixed $k \geq 2$, all such lines pass through the point $(c,d) = (-\frac{1}{k-1}, -\frac{1}{k-1})$, and each normal vector to $\ell(k,m)$ has nonnegative or infinite slope $\frac{1}{m}\binom{k}{2} - 1$. Putting $m = \binom{k}{2}$, we deduce that

$$c \;\geq\; -\frac{1}{\omega(G) - 1} \tag{3.8}$$

must hold, where $\omega(G)$ is the size of a largest clique in G; similarly, the relation

$$d \;\geq\; -\frac{1}{\alpha(G) - 1} \tag{3.9}$$

follows from the case $m = 0$, where the *independence number* $\alpha(G)$ measures the size of a largest vertex-induced subgraph without edges.

Observation 3.8. *The same reasoning in Corollary 3.1 can be used to prove that* $\frac{-1}{\lambda_1}$ *is the smallest possible* c *we can use in a* $(c,0)$ *representation, where* λ_1 *is the largest eigenvalue of* A*. Then equation* (3.8) *can be used to deduce the bound* $\omega(G) - 1 \leq \lambda_1$*.*[1]

To determine the lowest point on the d-axis that satisfies all relations of the form (3.7), we introduce a parameter $\bar{\rho}(S) = |\binom{|S|}{2} - \mathrm{E_{in}}(S)|/|S| = \frac{m'}{k}$ to each vertex-induced subgraph of G that measures the ratio of non-edges to vertices in the subgraph; here $m' = \binom{k}{2} - m$.

A point $(0,d)$ will satisfy all inequalities (3.7) written as $d \geq -\frac{1}{2\bar{\rho}(I)}$ if and only if

$$d \;\geq\; -\frac{1}{2\bar{\rho}_{\max}}, \tag{3.10}$$

where

$$\bar{\rho}_{\max} \;=\; \max\left\{ \frac{\binom{|S|}{2} - |\,\mathrm{E_{in}}(S)|}{|S|} \;:\; S \subseteq \{1, \ldots, n\} \right\}.$$

[1]This is not an original result, as can be deduced from interlacing theorem [**20**], but yet it provides a different method to deduce it

OBSERVATION 3.9. *By* (3.10) *and what we said above about* $\det G(c,d)$, *we obtain* $-\frac{1}{2\bar{\rho}_{\max}} \le -\frac{1}{\bar{\lambda}_1}$. *We can similarly define*

$$\rho_{\max} \;=\; \max\left\{ \frac{|\operatorname{E}(I)|}{|I|} \;:\; I \subseteq \{1,\dots,n\} \right\},$$

so that a point $(c,0)$ *satisfies all inequalities* (3.7) *if and only if* $c \ge -\frac{1}{2\rho_{\max}}$. *In particular,*

$$2\rho_{\max} \;\le\; \lambda_1 \qquad \textit{and} \qquad 2\bar{\rho}_{\max} \;\le\; \bar{\lambda}_1.$$

Which is indeed a generalization of the result presented at Observation 3.8.

EXAMPLE 3.4. *Let* $G = H(n,D)$ *be a* circulant *graph on* n *vertices, which means that each vertex* i *is adjacent to* $(i+d) \bmod n$ *and* $(i-d) \bmod n$, *for each element* d *in some list* $D \subset \{1,\dots,n\}$. *(Note that by definition, the symmetry group acts transitively on vertices, so that all vertices have the same degree, say* e.*) Equivalently, the adjacency matrix* A *of* G *is* circulant*: the rows of* A *are cyclic shifts of the first row, say* $(x_0, x_1, \dots, x_{n-1})$, *in such a way that all diagonal entries are* x_0. *In this case, the matrix* $G(c,d)$ *is also circulant with* $x_0 = 1$ *and* $x_i \in \{c,d\}$ *for* $i \ne 0$, *and it is known (*[**5**]*) that its determinant factors as*

$$\det G(c,d) \;=\; \prod_{k=0}^{n-1} P_x(\zeta^k),$$

where $\zeta = \exp(2\pi i/n)$ *and*

$$P_x(\zeta^k) \;=\; \sum_{j=0}^{n-1} x_j \zeta^{kj} \;=\; 1 + c \sum_{j \in \operatorname{nb}(v_0)} \zeta^{kj} + d \sum_{j \notin \operatorname{nb}(v_0)} \zeta^{kj}.$$

In particular, all factors are linear, $P_x(\zeta^0) = 1 + ec + (n-1-e)d$, *and all* $P_x(\zeta^k)$ *with* $k \ne 0$ *evaluate to zero at* $(c,d) = (1,1)$. *The last statement is true because for all nonzero* k *the sequence* $(\zeta^{kj} : j = 0,1,\dots,n-1)$ *is just some permutation of all* n*-th roots of unity, and these sum to zero. Moreover, since the graph is circulant,*

$$\sum_{j \in \operatorname{nb}(v_0)} \zeta^{kj} \;=\; \tfrac{1}{2} \sum_{j \in \operatorname{nb}(v_0)} \left(\zeta^{kj} + \zeta^{-kj} \right) \;=\; \sum_{j \in \operatorname{nb}(v_0)} \cos \frac{2\pi kj}{n},$$

and analogously for the non-neighbors of v_0. *Thus, the coefficients of* c *and* d *in each* $P_x(\zeta^k)$ *are real, so the vanishing locus of* $\det G(c,d)$ *is a union of at most* n *lines, and the ones closest to the origin correspond to* $k = 0$ *and the two values of* k *with the ratio*

$$\frac{\sum_{j \in \operatorname{nb}(v_0)} \cos \frac{2\pi kj}{n}}{\sum_{j \notin \operatorname{nb}(v_0)} \cos \frac{2\pi kj}{n}}$$

closest to -1. *Using* $\sum_{j=0}^{n-1} \cos \frac{2\pi kj}{n} = 0$, *this amounts to finding* k *such that the absolute value of* $\sum_{j \in \operatorname{nb}(v_0)} \cos \frac{2\pi kj}{n}$ *is as large as possible.*

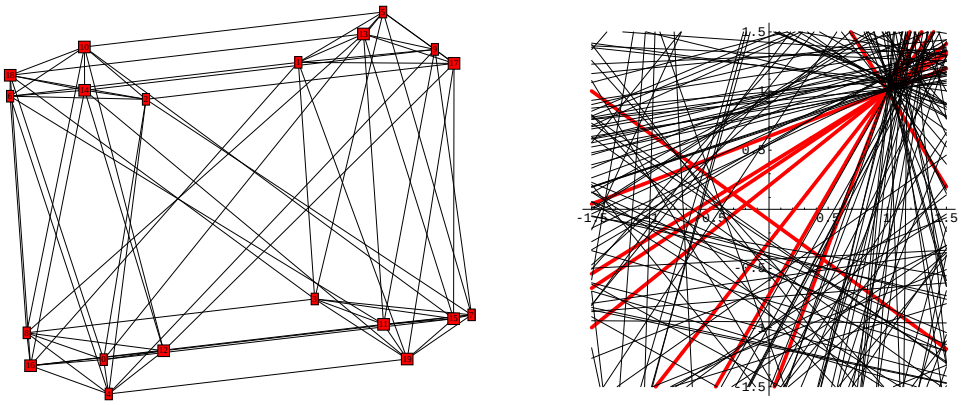


Figure 3.4: *Left:* The circulant graph $G = H(20, \{2, 4, 8, 16, 5, 10\})$. *Right:* The thin lines correspond to 200 random linear inequalities bounding $G(c, d)$ for this graph, which were generated by plugging random vectors $\boldsymbol{\lambda}$ into (3.6). The thick red lines are the vanishing locus of $\det G(c, d)$. The innermost triangle bounded by red lines is $M \cap C$.

6. The smallest eigenvalue

In last section, we have seen the strong relation between possible (c, d)-vector representations and the eigenvalues of the graph and its complementary. At Observation 3.9 we have used this relation to derive results on the eigenvectors.

Among all possible vector representations, a very important role will be played by the Lovász vector representation, where $c = -\frac{1}{\lambda_n}$ and $d = 0$. In this representation, the smallest eigenvalue of the adjacency matrix plays an essential role, and is the maximum value c can take when $d = 0$.

In this section, we will study what this eigenvalue means and which relation it has with the graph. We will present some classical bounds as well as a small theorem that relates it to bipartiteness. We must notice that any lower bound on λ_n will provide a valid vector representation. Therefore, if it was too expensive to compute λ_n, we could always use the sharper bound instead.

These first observations reinforce the idea that we can derive results on the eigenvalues from the geometry of the representations.

Observation 3.10. *If $S \subset V(G)$ and $H = G[S]$ is the corresponding induced subgraph, then the smallest eigenvalue of G is smaller than the smallest eigenvalue of H.*[2]

Proof. Any vector representation of G induces a valid vector representation of S, not necessarily tight. So c_G induced by the smallest eigenvalue of G is a valid constant that is not greater than c_S, and our statement holds. □

Observation 3.11. *For all non empty graph, the smallest eigenvalue of the adjacency matrix of the graph is smaller or equal than -1.*

[2]Again, this is just a consequence of the interlacing theorem [**20**].

PROOF. We know from vector representations that we have a vector configurations in which adjacent vectors have scalar product $-\frac{1}{\lambda_n}$. Scalar products are always smaller or equal than one, and that proves our hypothesis. □

6.1. Bounds from the literature. Let G be a graph, and $\lambda_1 \geq \cdots \geq \lambda_n$ the eigenvalues of its adjacency matrix A. An easy consequence of the Perron-Frobenius Theorem is that $\lambda_n \geq -\lambda_1$, an equality that holds if and only if G is bipartite [**20**]. This implies that in particular, any bound on λ_1 gives a bound on λ_n.

In [**24**], λ_n is bounded by

$$\lambda_n \leq -\frac{\lambda_1}{\chi(G)-1}, \tag{3.11}$$

where $\chi(G)$ is the chromatic number of a non-bipartite graph G.

Now, we present a classical result from Stanley in 1987, a bound of the spectral radius in terms of the number of edges of the graph [**47**]. It states that if G is a graph with e edges, then its maximal eigenvalue λ_1 is bounded by

$$\lambda_1 \leq \frac{\sqrt{1+8\,e}-1}{2}.$$

Then in year 2000, Alon and Sudakov prove in [**1**] the bound

$$\lambda_n \geq -\Delta + \frac{1}{n(D+1)}$$

for non-bipartite graphs, and where Δ denotes the maximum degree and D the diameter.

The article [**11**] culminates a series of publications of bounds on the largest eigenvalue for non-regular graphs [**12, 48, 53**], each one improving the previous. The resulting bound is

$$\lambda_1 \leq \Delta - \frac{1}{nD}. \tag{3.12}$$

6.2. Graphs with the smallest smallest eigenvalue. Here we present a small result that characterises the graphs that have the smallest smallest eigenvalue among all the graphs with the same number of nodes. These are the bipartite complete graphs. Recall that the graph that has the largest largest eigenvalue is the complete graph, but it has the smallest eigenvalue equal to -1, which from Observation 3.11 is a maximal value. However, graphs with few edges also have large smallest eigenvalues, with the extreme at the independent graph, which has the smallest eigenvalue equal to 0.

THEOREM 3.2. *Among all the graphs with n nodes, the one that has its smallest eigenvalue the smallest is the complete bipartite graph $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$.*

PROOF. The proof takes 3 steps. First, we will see that for every graph that is not bipartite, there exists a different graph with smaller λ_n. That proves that the graph with smaller λ_n must be bipartite. Second step consists in seeing that if we have a bipartite graph which is not complete, we can add an edge and have a graph with smaller λ_n. That proves that our

graphs are complete bipartite. Finally we just need to see that among all complete bipartite graphs, the one with smaller λ_n is $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$.

Our proof relies on the following result from Rayleigh quotient that states that if A is a Hermitian matrix, then the minimum of the operator $\frac{x^* A x}{x^* x}$ is λ_n and is reached if and only if $x = v_n$ is a corresponding eigenvector (see [**5**]).

Step one: Let G be a non-bipartite graph, A its adjacency matrix and let λ_n and v_n be the smallest eigenvalue of A and a corresponding unitary eigenvector.

It is well known that there exists an eigenvector v_1 corresponding to the largest eigenvalue which has all its entries positive (Perron-Frobenius theorem). As adjacency matrices diagonalize in an orthonormal basis, v_n is orthogonal to v_1. Therefore v_n must have at least a positive and a negative entry. We will use the notation $v(i)$ to refer to the i-th entry of v.

We define the following partition of the set of vertices $V_+ = \{i \in V \mid v_n(i) > 0\}$, $V_- = \{i \in V \mid v_n(i) < 0\}$ and $V_0 = \{i \in V \mid v_n(i) = 0\}$. As G is not bipartite, there is an edge $\{u, w\}$ between two nodes either in $V_+ \cup V_0$ or V_-, we assume that $u, w \in V_+ \cup V_0$ w.l.o.g. (otherwise we change the sign of the eigenvector). We may also assume that u, w correspond to the first two vertices.

Now let $G' = \{V(G), E(G) \setminus \{u, w\}\}$ and A' its adjacency matrix with eigenvalues $\lambda'_n \leq \cdots \leq \lambda'_1$. Observe that for every vector v we have

$$A' \cdot v = A \cdot v - v(1) \cdot e_2 - v(2) \cdot e_1$$

and therefore

$$\lambda'_n = \min \frac{v^T A' v}{v^T v} \leq \frac{v_n^T A' v_n}{v_n^T v_n} = \lambda_n - 2v(1)v(2) \leq \lambda_n \tag{3.13}$$

as $v(1)$ and $v(2)$ are both positive. This proves that we can recursively remove edges from our graph without increasing the smallest eigenvalue until we get a bipartite graph.

Step two:
We will need the following auxiliary lemma

LEMMA 3.2. *If $G(S, T)$ is a bipartite graph, there exists an eigenvector of λ_n such that $V_+ \subset S$ and $V_- \subset T$.*

PROOF. Let v_n be an unitary eigenvector of λ_n. It minimizes the Rayleigh quotient. We define $\tilde{v}_n$ as

$$\tilde{v}_n(i) = \begin{cases} +|v_n(i)| & \text{if } i \in S \\ -|v_n(i)| & \text{otherwise} \end{cases}$$

We will show that $\tilde{v}_n$ is an eigenvector of λ_n because it will reach the minimum of the Rayleigh quotient.

$$\begin{aligned} v_n^T A v_n &= \sum A_{ij} v_n(i) v_n(j) = \sum_{i\in A, j\in B} A_{ij} v_n(i) v_n(j) \geq \\ &\geq \sum_{i\in A, j\in B} -A_{ij} |v_n(i)||v_n(j)| = \sum_{i\in A, j\in B} A_{ij} \tilde{v}_n(i) \tilde{v}_n(j) = \\ &= \tilde{v}_n^T A \tilde{v}_n \end{aligned}$$

□

Let G be a non-complete bipartite graph, A its adjacency matrix and let λ_n and v_n the smallest eigenvalue of A and the corresponding unitary eigenvector from Lemma 3.2. As G is non-complete there exist $u \in (V_+ \cup V_0)$ and $w \in (V_- \cup V_0)$ such that $\{u, w\} \notin E(G)$. We can assume that u and w are the first two vertices.

Now, if we define $G' = (V(G), E(G) \cup \{u, w\})$ and A' its adjacency matrix, we can see as in the previous step that

$$\lambda_n' = \min \frac{v^T A' v}{v^T v} \leq \frac{v_n^T A' v_n}{v_n^T v_n} = \lambda_n + 2v(1)v(2) \leq \lambda_n \tag{3.14}$$

Because now $v(1)$ and $v(2)$ have different sign.

Therefore we can add new edges without increasing the smallest eigenvalue until we reach a complete bipartite graph.

Step three: Finally, we must just observe that when we have a complete bipartite graph $K_{r,s}$ then its eigenvalues are $\pm\sqrt{rs}$ with both multiplicity one and zero with multiplicity $r + s - 2$. The minimum we can obtain corresponds to the graph $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$ and is

$$\lambda_n = \begin{cases} \frac{n}{2} & \text{if } n \text{ even} \\ \frac{\sqrt{n(n-1)}}{2} & \text{if } n \text{ odd} \end{cases}$$

□

This immediately gives us the following bound:

COROLLARY 3.2. *The smallest eigenvalue of the adjacency matrix of a graph is at least $-\frac{n}{2}$. This bound is sharp as $K_{\frac{n}{2},\frac{n}{2}}$ attains it.*

REMARK 3.3. *Observing equations* (3.13) *and* (3.14)*, we deduce that adding or removing an edge can at most change the value of λ_n in a unity. We have somehow that λ_n changes continuously respect to the edges.*

FOLLOW UP 1. *Can Lemma 3.2 be extended to be used in the Max-cut problem as in* [**50**]*?*

CHAPTER 4

Objective functions

It is now the moment to derive an objective function from vector representations. In Chapter 3 we already gave hints that indicated that the norm could be a good objective function. Unfortunately, soon we will see that the norm is *not* the correct choice. However, not everything is lost, as the norm still carries some valuable information from where we can derive better objective functions.

In this chapter we will first illustrate why the norm is not a good choice. Then, we present some failed attempts to derive alternative objective functions from the norm. We will explain them, show the initial intuition why they could work, and develop them a bit to finally show why they do not work. Afterwards, we give what will be the chosen fitness function: the *Directed Laplacian*. In the end we will present an objective function from the literature that we will also test and compare to our proposal.

1. Norms are not enough

1.1. Lovász construction. First, we focus in Lovász vector representation, and set $\phi(s) = \|s\|^2$ as the objective function (notice that it is equivalent to the norm). To study which sinks are arising from this configuration, we concentrate in the orientation of each edge of Γ. We need to study $\|S \cup v\|^2 - \|S\|^2$, that from equation (3.4), we have

$$\|S \cup v\|^2 - \|S\|^2 = 1 + 2c\,|\operatorname{nb}_S(v)| > 0$$

and all edges point to the largest set! Therefore, the only sink in $\Gamma(\phi)$ is the one corresponding to the entire graph.

This was clear all along because all our vectors lie in an acute cone, and even adding a perpendicular vector increases the resulting length.

1.2. (c, d)-Vector representations. It is still conceivable that we can overcome this problem when we allow the full family of (c, d)-vector representations, because when we allow obtuse angles for non connected nodes norms of sum vectors can indeed decrease.

Again, we must compare $\|S \cup v\|^2$ and $\|S\|$. We are interested in which cases $S \cup v$ is smaller than S. From (3.4), $\|s + v\|^2 \leq \|s\|^2$ if and only if

$$1 + 2c\,|\operatorname{nb}_s(v)| + 2d\,(|s| - |\operatorname{nb}_s(v)|) \leq 0. \tag{4.1}$$

From this equation, we will see with a particular example that there are some graphs which are clearly communitized, but that there exists no pair (c, d) in the set of admissible scalar products for which our clear communities are sinks.

EXAMPLE 4.1. *Let $G = K_6 \square C_5$ be the cartesian product of a complete graph of 6 nodes with a cycle of 5 nodes. That is the graph whose vertices are $V(G) = \{(u, v) | u \in V(K_6) \quad v \in (C_5)\}$, and where (u_1, v_1) is adjacent to (u_2, v_2) if either $u_1 = u_2$ and $\{v_1, v_2\} \in E(C_5)$ or $v_1 = v_2$ and $\{u_1, u_2\} \in E(K_6)$.*

It is a graph that consists of five clear clusters– that are the 6-cliques – cyclically connected.

If we focus on one of these cliques, and study the inequality (4.1)*, where the set S is one of the K_6 and v one of the nodes connected to it, which can at most have a neighbor in the set. This translates into the equation*

$$1 + 2\,c + 14\,d \leq 0 \tag{4.2}$$

On the other hand, we can explicitly find the set of valid values for pairs (c, d) by studying the points that vanish $det(G(c, d))$, as we did in Section 5. In figure 4.1 we plot the acceptance region as well as the the boundary of the region delimited by (4.2)*.*

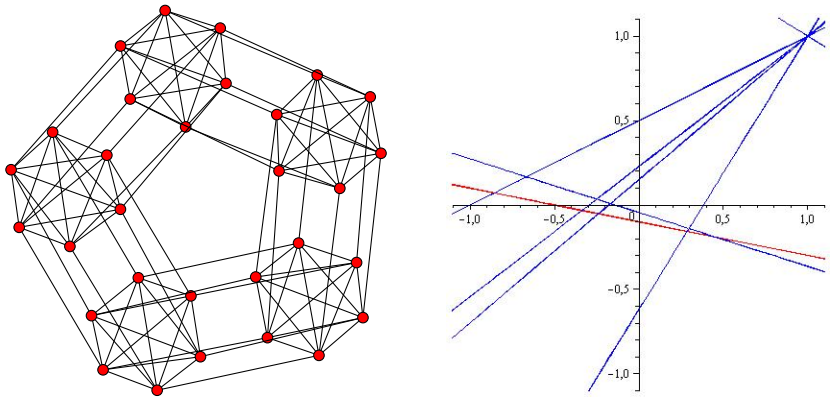


Figure 4.1: $K_6 \square C_5$: The blue lines are the vanishing points of $det(G(c, d))$, while the red line plots the equation $1 + 2\,c + 14\,d = 0$.

We can see that all acceptable pairs (c, d) accomplish that

$$1 + 2\,c + 14\,d > 0;$$

so none of the clear communities would ever be a sink, even if we found the best admissible pair (c, d)!

So even when we admit all possible (c, d)-vector configurations, the method would still not be good enough to recognise communities.

1.3. There is still hope. We have just seen that the length of the sum vector is not a good choice as an objective function. However, at Section 3 of previous chapter we saw that the length of vectors encoded some information about the connectivity of a community, and we resist to lose this information. This is why we will proceed to study the behaviour of the norm of communities on some particular graphs.

Just take the family of graphs $K_{k\times l} = K_k \square K_l$, the cartesian product of two complete graphs of sizes k and l. When $k \gg l$, then this graph consists of a collection of l complete graphs K_k connected within each other. These copies of the complete graph are likely to be communities. In Figure 4.2, we show the concrete example of the graph $K_{20}\square K_5$, the one that we will proceed to analyse.

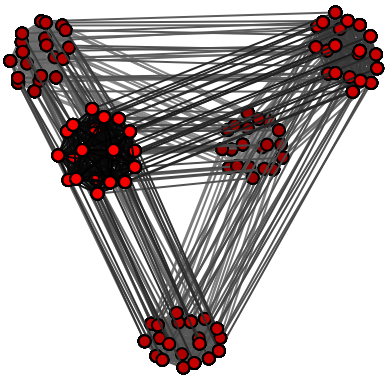


Figure 4.2: Representation of $K_{20}\square K_5$ using a Spring embedder. We can see that the K_{20}'s form natural communities

Now, with the Lovász representation, we do the following steepest increment walk in $\Gamma(\|\cdot\|^2)$. We start at $s = \emptyset$ the empty set, and then at each step we add to s a node v that maximizes $\|s \cup v\|^2$. At Figure 4.3 we plot the value of the squared norm of the set belonging to each step of the path.

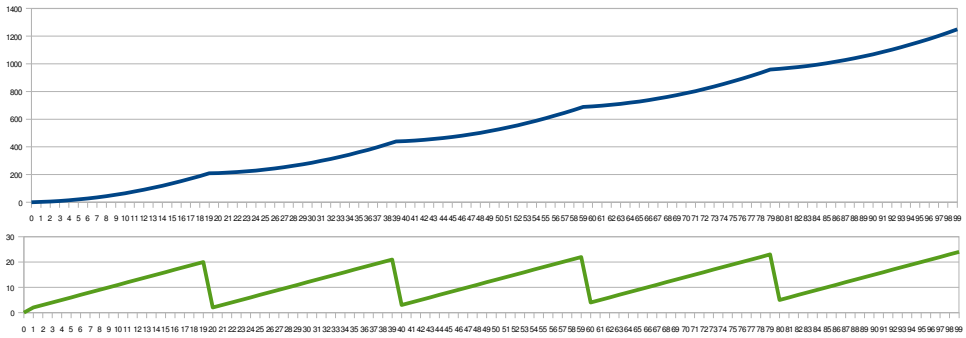


Figure 4.3: In the blue plot we can see the squared norm of the subsets of the form $\{0, \ldots, i\}$, while in the green plot there are the partial differences, that is $\|s + v\|^2 - \|s\|^2$

And now, if we observe Figure 4.3, we can detect the following tendence, that will be later on crutial to obtain a fitness function: Although the norm always increases, it does not do it steadily. The points where the increasing velocity changes are those points where we had predicted there would be

a community (the cliques)![1] This is even clearer when we plot the partial differences between step i and $i-1$, where we can see that there communities do correspond to maxima. So we can in fact detect comunities from vector representations. If only we could get rid of the "always increasing" component...

2. Towards a good objective function

The fact that the addition of nodes that do not belong to a community still contributes to the norm makes that considering the length of sum vectors is not enough to detect the community structure. In a way, having in mind Figure 4.3, we want to detect which part of the length increment comes from being part of the community and which comes just from the inherent norm increasing. Here we present some brainstorming of ways to effectively derive the community structure from vector representations. Some of them where discarded from the beginning. Others seem promising but we did not truly follow them, while some where followed until they turned out not to be worth it. Finally we kept the one that gave us the best results.

We present here the main approaches we came up with to resolve it. We will afterwards in next sections expand with some results the branches where we dug further and mention why did we finally discard them. And obviously we will finally make a detailed study of our chosen solution.

Subtracting the random component. The first solution we can imagine consists in subtracting "by hand" some function to the norm. This function must be able to tell which part of the norm is relevant to the community structure and which just comes from the fact of working with vectors. For example, in the Lovász representation, the part that forces the addition of an unconnected node to a community is the one we would wish to subtract.

That means we have to define a function $g : 2^V \to \mathbb{R}$ and then set the objective function $\phi(S) = \|S\|^2 - g(S)$. Some proposals of g have been tested, but none worked well enough and we focused on other approaches

Subtract the norm of an independent set of the same size: This is not enough, as we want to prevent the addition of nodes not well connected to the set, not only the addition of disconnected nodes. For example, in the last example of $K_{20}\square K_5$, even when we subtract the size of an independent set the only community is the total vector.

Subtract the norm of a complete graph of the same size: This one is too strong. Our communities are not expected to be as dense as a clique, and thus the subtracted formula will dominate; leading to a situation where the empty set is the only sink.

Subtract the norm of a "expected" community: We could also substract the expected length of a community that followed the Erdős-Rényi model $G(n,p)$. But there arises a problem in the choice of p: which probability should we use? If we used the density of the whole graph, that would imply

[1]Recall that from Observation 3.3, we expect to consider the union of communities as a community.

a loss in the locality: if there are two communities, one denser that another, probably only the denser one will be detected.

And if we used the density of the set itself, then the resulting objective function would be always zero!

Projecting vector representations. Again, we focus on Lovász' vector representation. We saw that the only sink we can get using the objective function $\phi = \|\cdot\|$ is the full set that has the associated vector $t = \sum_{v \in V} v$.

This proposal consists in looking for vectors of maximal length in the projection $\pi(\mathcal{V})$ obtained by projecting to the orthogonal complement of the maximal sum vector t.

In this way the full set, which used to be the longest vector, gets associated to the vector 0, and there arise different sinks in Γ.

Differential approach. In Figure 4.3 we plotted the partial differences on a maximum ascending path at Γ. If we consider this path to be one-dimensional, then the partial differences are an approximation of the derivative of the norm along the path.

We saw that indeed the maxima of this derivative were the sets we expected to be communities, what indicates that it could be a good objective function. However, so far we only know how to differentiate along a concrete path, and we would wish to extend this notion of derivative to "more dimensions". In the sense that we would like to make a definition of derivative on all the graph Γ, such that it has a unique value for each node (and hence does not depend on the specific path used to reach the node).

With this purpose in mind, we will try to define some extensions of the notion of derivative, and will study their behaviour, keeping the one that gives best results.

3. Projected vector representations

To prevent communities being too large and to prevent the tendency of having the total set as the only community, we will modify a given vector representation $\mathcal{V}$ to obtain the projected vector representation $\pi(\mathcal{V})$, the projection to the orthogonal complement of the maximal sum vector $t = \sum_{v \in \mathcal{V}} v$.

Concretely, for each vector $v \in \mathcal{V}$, we will define

$$\pi(v) = v - \frac{\langle v, t \rangle}{\langle t, t \rangle}\, t.$$

In the spirit of Observation 3.6, we need to find equivalent equations that substitute (3.2), (3.3), (3.4) and (3.5) for the projected vector representation. Otherwise, we should already discard this option because we would not be able to compute it! We need to be able to find locally longest sum vectors without explicitly constructing $\pi(\mathcal{V})$.

The work-horse for this will be the relation

$$\|s + v\|^2 - \|s\|^2 \;=\; 1 + 2\langle s, v \rangle, \tag{4.3}$$

which holds for any pair of real vectors s, v. Here s will be (the projection of) a sum vector built from $\mathcal{V}$, and $v \in \mathcal{V}$ a new (projected) vector we want to add to the sum.

LEMMA 4.1. *Let $G = (V, E)$ be a graph on n vertices with m edges. Pick inner products $c, d \in \mathbb{R}$ corresponding to a vector representation $\mathcal{V} = \{v_1, \ldots, v_n\}$ of G. Let $s = \sum_{v_i \in S} v_i$ be a sum of vectors from $\mathcal{V}$, and set $t = \sum_{v_i \in \mathcal{V}} v_i$. Then,*

(a) $\langle t, t\rangle = n + 2(c-d)m + 2d\binom{n}{2}$.
(b) $\langle v, t\rangle = 1 + d(n-1) + (c-d)\deg v$.
(c) $\langle s, t\rangle = |S|(1 + d(n-1)) + (c-d)(2\,\mathrm{E}_{\mathrm{in}}(S) + \mathrm{E}_{\mathrm{out}}(S))$.
(d) $\|\pi(s)\|^2 = \langle \pi(s), \pi(s)\rangle = \|s\|^2 - \langle s, t\rangle^2 / \langle t, t\rangle$.
(e) $\|\pi(v)\|^2 = \langle \pi(v), \pi(v)\rangle = 1 - \langle v, t\rangle^2 / \langle t, t\rangle$.
(f) $\langle \pi(s), \pi(v)\rangle = [\![v \in S]\!] + c\,|\,\mathrm{nb}_S(v)| + d\,|S \setminus \mathrm{nb}(v) \setminus v| - \frac{\langle v,t\rangle}{\langle t,t\rangle} \sum_{i \in S} \langle v_i, t\rangle$.

Here $[\![x]\!]$ is Iverson's notation*: it evaluates to 1 if x is true, and to 0 otherwise.*

PROOF. (a) This follows directly from equation (3.3).
(b) It is also direct, now from equation (3.2).
(c) From entry (b) and observing that $\sum_{v_i \in S} \deg(v_i) = 2\ \mathrm{E}_{\mathrm{in}}(S) + \mathrm{E}_{\mathrm{out}}(S)$, we develop

$$\begin{aligned}
\langle s, t\rangle &= \langle \sum_{v_i \in S} v_i, t\rangle = \sum_{v_i \in S} \langle v_i, t\rangle \\
&= \sum_{v_i \in S} 1 + d(n-1) + (c-d)\deg v_i \\
&= |S|(1 + d(n-1)) + (c-d) \sum_{v_i \in S} \deg v_i \\
&= |S|(1 + d(n-1)) + (c-d)(2\,\mathrm{E}_{\mathrm{in}}(S) + \mathrm{E}_{\mathrm{out}}(S))
\end{aligned}$$

(d) By the definition of projection we know that $\pi(s) = s - \frac{\langle s,t\rangle}{\langle t,t\rangle} t$, therefore

$$\begin{aligned}
\|\pi(s)\|^2 &= \|s - \frac{\langle s, t\rangle}{\langle t, t\rangle} t\|^2 \\
&= \|s\|^2 + \|\frac{\langle s, t\rangle}{\langle t, t\rangle} t\|^2 - 2\langle s, \frac{\langle s, t\rangle}{\langle t, t\rangle} t\rangle \\
&= \|s\|^2 + \left(\frac{\langle s, t\rangle}{\langle t, t\rangle}\right)^2 \|t\|^2 - 2\frac{\langle s, t\rangle}{\langle t, t\rangle}\langle s, t\rangle \\
&= \|s\|^2 - \langle s, t\rangle^2 / \langle t, t\rangle
\end{aligned}$$

(e) Direct from (d)
(f) This is true because

$$\begin{aligned}
\langle \pi(s), \pi(v)\rangle &= \sum_{v_i \in S} \langle \pi(v_i), \pi(v)\rangle \\
&= \sum_{v_i \in S} \langle v_i - \tfrac{\langle v_i,t\rangle}{\langle t,t\rangle} t,\ v - \tfrac{\langle v,t\rangle}{\langle t,t\rangle} t\rangle \\
&= \sum_{v_i \in S} \langle v_i, v\rangle - \tfrac{\langle v,t\rangle}{\langle t,t\rangle} \sum_{v_i \in S} \langle v_i, t\rangle,
\end{aligned}$$

and $\sum_{v_i \in S} \langle v_i, v\rangle = [\![v \in S]\!] + c\,|\operatorname{nb}_S(v)| + d\,|S \setminus \operatorname{nb}(v) \setminus v|$.

□

This lemma gives us the power of expressing $\|\pi(s)\|$ in graph terms. As a consequence, it is computationally feasible to work with projected representations to find the communities, and to define $\|\pi(s)\|$ as the objective function.

To go deeper in what this representation is, we give some concrete results for the Lovász representation:

PROPOSITION 4.1. *Let $G = (V, E)$ be a graph on $n = |V|$ vertices, let $p = -\lambda_n = |\lambda_n|$ be the absolute value of the smallest eigenvalue of the adjacency matrix of G, set $K = 2|E| + pn > 0$, and let $\pi(\mathcal{V})$ be the projection of Lovász' representation of G (that is, the $(\frac{1}{p}, 0)$-vector representation).*

(a) *The length of a projected vector decreases with the degree of the corresponding node. In particular, $\pi(\mathcal{V})$ does not consist of unit vectors, and high-degree nodes do not have an extremely large influence on the community.*

(b) *Projections of pairs of orthogonal vectors are obtuse, so their sum becomes shorter. On the other hand, whenever the nodes corresponding to v and w are adjacent, and*

$$(p + \deg v)(p + \deg w) \;<\; K,$$

the angle between the corresponding projected vectors is acute. In the case $\deg v = \deg w$, this condition reads

$$\deg v \;<\; -p + \sqrt{K}.$$

(c) *Let $s = \sum_{i \in S} v_i$ be a sum of vectors in $\pi(\mathcal{V} =$ corresponding to a community of size $|S|$, and let $v \in \mathcal{V}$ be a vector representing a node not belonging to S. Then $\|\pi(s + v)\| > \|\pi(s)\|$ if and only if*

$$\frac{K}{p + \deg v}\left(\frac{|\operatorname{nb}_S(v)|}{p} + \frac{1}{2}\right) \;>\; |S| + \frac{1}{2p}\left(\deg v + 2\sum_{i \in S} \deg v_i\right) + \frac{1}{2}. \tag{4.4}$$

Roughly, adding v will profit a community if the corresponding node has lots of neighbors in it, but the number of vertices and the total degree of the new community do not grow too much.

PROOF. (a) The image of $v \in \mathcal{V}$ is $\pi(v) = v - \lambda t \in \pi(\mathcal{V})$, with $\lambda = \frac{\langle v,t\rangle}{\langle t,t\rangle}$. By the properties of Lovász' representation, these scalar products are

$$\begin{aligned} \langle v, t\rangle &= \left\langle v, v + \textstyle\sum_{i \in \{1,\dots,n\}: v_i \neq v} v_i \right\rangle \\ &= \langle v, v\rangle + \textstyle\sum_{i \in \{1,\dots,n\}: \langle v, v_i\rangle \neq 0} \langle v, v_i\rangle \\ &= 1 + \tfrac{1}{p}\deg v, \end{aligned}$$

and

$$\langle t, t\rangle \;=\; n + \tfrac{2}{p}|E| \;=\; \tfrac{K}{p}.$$

The scalar product of the image of any pair of vectors $v, w \in \mathcal{V}$ turns out to be

$$\begin{aligned} \langle \pi(v), \pi(w) \rangle &= \langle v, w \rangle - \frac{\langle v, t \rangle \langle w, t \rangle}{\langle t, t \rangle} \\ &= \langle v, w \rangle - \frac{(p + \deg v)(p + \deg w)}{pK}. \end{aligned} \tag{4.5}$$

In particular,

$$\langle \pi(v), \pi(v) \rangle = 1 - \frac{(p + \deg v)^2}{pK} < 1.$$

(b) follows from (4.5).

(c) Consider the projection of a sum $s = \sum_{v_i \in S} v_i$. If v does not belong to S, then

$$\|\pi(s) + \pi(v)\|^2 = \|\pi(s)\|^2 + \|\pi(v)\|^2 + 2\langle \pi(v), \pi(s) \rangle,$$

and

$$\begin{aligned} \langle \pi(v), \pi(s) \rangle &= \sum_{v_i \in S} \langle \pi(v), \pi(v_i) \rangle \\ &= \sum_{v_i \in S} \langle v, v_i \rangle - \frac{p + \deg v}{K} \sum_{i \in S} (1 + \tfrac{1}{p} \deg v_i) \\ &= \tfrac{1}{p} |\operatorname{nb}_S(v)| - \frac{p + \deg v}{K} \left(|S| + \tfrac{1}{p} \sum_{i \in S} \deg v_i \right), \end{aligned}$$

so that

$$\begin{aligned} \|\pi(s) + \pi(v)\|^2 - \|\pi(s)\|^2 &= 1 + \tfrac{2}{p} |\operatorname{nb}_S(v)| \\ &\quad - \frac{p + \deg v}{K} \left(1 + \tfrac{1}{p} \deg v + 2|S| + \tfrac{2}{p} \sum_{i \in S} \deg v_i \right). \end{aligned}$$

The claim now follows from requiring the right-hand side to be positive.

This concludes the proof. □

Results in this proposition show some agreeable tendencies. First, we observe that it plays down the power of nodes with really high degree, and enhances that of nodes with small degree. This could be a good symptom if we were interested in assigning "weak" nodes to communities, without losing them. Roughly, if a node has few neighbors, but all of them belong to a community, then it will be part of the community.

The second point helps us to understand the redistribution of vector positions after the projection. And finally we get both the computational tools to set the orientation at $\Gamma(\phi)$ and the intuition of how the sinks are going to be.

However, to cool off the euphoria, the following Remark 4.1 together with Example 4.2 will give us some intuition of the main drawback of this approach, and suggests one of main reasons why we did not follow this path further.

REMARK 4.1. *If G is e-regular (all node degrees are e), it is well known* [**1**] *that $p = e$ if and only if G contains a bipartite connected component, that is, if and only if there is a set S and partition of S into disjoint sets L, R, such that all edges incident on S have one endpoint in L and one endpoint in R. In that case, the condition* (4.4) *reads*

$$|S| \ < \ n\left(\frac{1}{4} + \frac{|\operatorname{nb}_S(v)|}{2e}\right) - \frac{1}{2}.$$

Thus, all communities have size between $\frac{n}{4}$ and $\frac{3n}{4}$. The ϕ-best ones are those that consistently add vertices with lots of neighbors already inside.

This remark shows the tendency of the projected vector representations to have communities of size about $\frac{n}{2}$. When projecting, vectors representing small sets are those which are the most perpendicular to t (and therefore conserve almost all of their length), and vectors representing very large subsets are almost lined up with t and therefore get projected to very short vectors. This enhancement of small sets and the reduction of large sets gives as a result that the communities more likely to appear are those of size $\frac{n}{2}$! This is **not** a natural tendency: we expect communities to be much smaller!

EXAMPLE 4.2. *We take for this example a graph G that is a cartesian product of a circulant graph and a cycle*[2]*. In concrete, we will use the circulant graph $H = H(15, \{1, 2, 3, 4, 5\})$ and we will set $G = H \square C_{10}$. In Figure 4.2 we can see these graphs. In concrete, we can see that H is a pretty dense graph, each node is connected to $\frac{2}{3}$ of all the possible neighbors. In G, each copy of H should stand out as a community.*

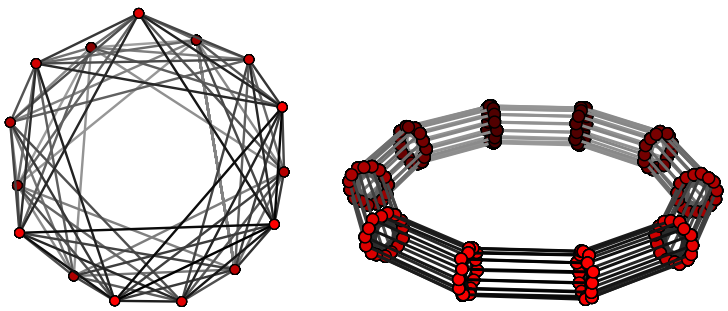


Figure 4.4: $H(15, \{1, 2, 3, 4, 5\})$ and $H(15, \{1, 2, 3, 4, 5\}) \square C_{10}$

To see whether this happens, we will do a procedure analogous to that in Section 1.3. We plot $\|\pi(s)\|^2$, for a set s that goes from the empty set to V adding at each step a node that maximizes the fitness function. We must stress that if a set is not a maximum in this path, then it can not be a sink of Γ.

We can see the result at Figure 4.5. The objective function does order the nodes correctly, however it is not good enough to identify the sinks we

[2]The definition of circulant graph is at Example 3.4, and that of cartesian product is at Example 4.1

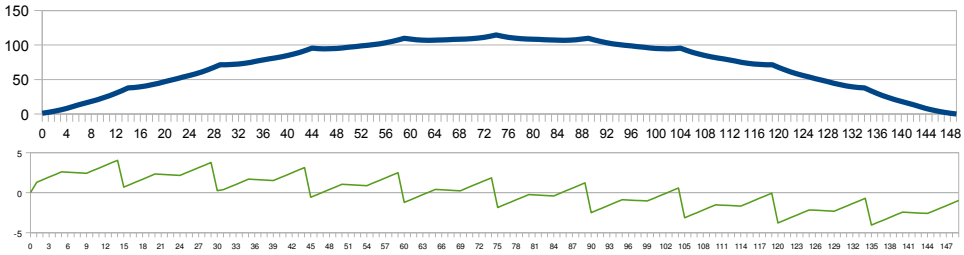


Figure 4.5: First plot represents for each i, $\|\sum_{k=0}^{i} \pi(v_k)\|^2$. The second one represents $\|\sum_{k=0}^{i} \pi(v_k)\|^2 - \|\sum_{k=0}^{i-1} \pi(v_k)\|^2$

would like: neither the first copy of H nor the union of the first two copies are local maxima. When the path goes through them, there is a change in the curvature, but it is yet not enough to become a maximum. In contrast, the union of three copies of H is a local maximum.

Follow Up 2 (Iterated projections). *An interesting idea that should at least be considered further are* Iterated projections. *It consist in exploiting the idea of projecting the vector representation much deeper, by successively projecting against different vectors. In brief, each time a community was found, the vector representation would be projected along the vector representing that community.*

Would this be computationally feasible? What kind of communities would it recover?

4. Sinks from Increment Orientations

We will now detail our first approach to a "differential" fitness function. We present here our first results, that proved to be not good enough and lead to the birth of new differential objective functions in Section 5.

Recall that if we imagined an ascending path from the empty set to the total set, adding an extra node almost always incremented the value of the objective function. Nevertheless, the increment was not always the same: it was greater when the added node fit better the community structure. This situation lead us to wonder how to reflect these changes in the increments that we interpreted as a derivative.

In each node there are several paths that go through it, giving each one a different increment, so which one should we choose? A max-min approach tells us to choose the one that gives us the least increment.

Therefore, as a first attempt, we will use the following objective function

$$\begin{aligned} (4.6)\ f : 2^V &\longrightarrow \mathbb{R} \\ s &\longmapsto f(s) = \min_{v \in S}\{\|s\| - \|s - v\|\} = \|s\| - \max_{v \in S}\{\|s - v_i\|\} \end{aligned}$$

That is, f is the minimal difference between the norm of a set and the norm of a set without a node, the steadiest increment we can use in the last step to reach the set.

First of all, we will study which is the $v \in S$ that maximizes $\|s - v_i\|$. Noticing that $\operatorname{argmax}_{v\in S}\{\|s - v\|\} = \operatorname{argmin}_{v\in S}\{\langle s, v\rangle\}$, we are looking for the node v whose vector forms the largest angle with the community vector, i. e. the node that fits worst in the community so far.

PROPOSITION 4.2. *Given a (c, d)- vector representation. Then the node $v \in S$ that minimizes $\langle s, v\rangle$ is $\operatorname{argmin}_{v\in S}\{|\operatorname{nb}_S(v)|\}$. In particular, it does not depend on the values c and d.*

PROOF. Let u and v be two members of the community S. Then we have that $\langle s, u\rangle < \langle s, v\rangle$ if and only if

$$1 + c\,|\operatorname{nb}_S(u)| + d\,|S \setminus (u \cup \operatorname{nb}_S(u))| < 1 + c\,|\operatorname{nb}_S(v)| + d\,|S \setminus (v \cup \operatorname{nb}_S(v))|$$

Since $|S-(u\cup\operatorname{nb}_S(u))| = |S|-1-|\operatorname{nb}_S(u)|$ and all vector representations fulfill $c - d > 0$, this simplifies to $|\operatorname{nb}_S(u)| < |\operatorname{nb}_S(v)|$. □

It is important to remark that although the node that reaches the minimum is independent of the scalar product (and thus of the vector representation), that does not mean that the sinks are. Later on we will see in examples that the sinks do depend on the scalar product chosen.

Until now, we have been bouncing between norms and squared norms of vectors. So far, that did not make any difference, as the function x^2 is monotonous and thus maxima of the norm are the same as maxima of the squared norm. However, from now on it will make a difference, as it is not the same to differentiate x^2 or x.

Consequently, we proceed to compare f to another objective function

$$\begin{aligned} g : 2^V &\longrightarrow \mathbb{R} \\ s &\longmapsto g(s) = \|s\|^2 - \max_{v\in S}\{\|s - v\|^2\} \end{aligned}$$

which represents the derivative of the squared norm instead of that of the norm.

Mainly, if v and w are the nodes that reach the minimum number of neighbors at S and $S \setminus v$ respectively, then using the relation

$$\|s\|^2 - \|s - v\|^2 = (\|s\| - \|s - v\|)(\|s\| + \|s - v\|),$$

we have that $g(s) > g(s - v)$ if and only if

$$(\|s\| - \|s-v\|)(\|s\| + \|s-v\|) > (\|s-v\| - \|s-v-w\|)(\|s-v\| + \|s-v-w\|),$$

which is equivalent to

$$\frac{(\|s\| + \|s - v\|)}{(\|s - v\| + \|s - v - w\|)} \cdot f(s) > f(s - v).$$

In the case $\|s\| > \|s - v - w\|$ (recall that the particular case of $d = 0$ we always have that $\|s\| > \|s - v - w\|$), when we work with g as compared to f, we are prioritizing the larger communities.

The following example will motivate the choice of g over f.

EXAMPLE 4.3. *We use a concrete example that will explain the reason why it may be a better option to use the function g instead of f. Let S be a (maximal) clique. We would expect that all its sub-cliques point towards it at $\Gamma(\phi)$, as we understand that a large clique has a better community structure than any of its sub-cliques.*

In this case we have all the corresponding internal edges, so in its length we only have to use the value of c in the scalar product $G(c,d)$. Denoting $n = |S|$ and assuming $n \geq 3$, we have, for any $v, w \in S$,

$$\begin{aligned} \|s\|^2 &= n + cn(n-1) \\ \|s-v\|^2 &= n - 1 + c(n-1)(n-2) \\ \|s-v-w\|^2 &= n - 2 + c(n-2)(n-3) \end{aligned}$$

and the difference in the objectives values simplifies to

$$g(s) - g(s-v) = 2c \geq 0.$$

So for g, the community will grow at least until the maximal clique.

However when we study the same case, but using as objective function $f(s) = \|s\| - \max \|s-v\|$, after some calculations we obtain that

$$f(s) - f(s-v) \leq 0$$

if and only if

$$-4 + 8\,c - 4\,c^2 \leq 0.$$

So, we are left with a parabola $p(c)$ that has its maximum at $c = 1$, with value $p(c) = 0$. The only case in which we could have $c = 1$ and yet $G(c,d)$ still positive semidefinite is when our graph is a complete graph (and then a community structure does not make much sense). Otherwise our inequality is not sharp, and $f(s) - f(s-v)$ is strictly negative. In this case, cliques will point towards their subcliques, and the sinks will be the smaller communities. This is not the desired behaviour of our objective function, so we decide to use the objective function g.

4.1. Graph characterization of the orientations. Having fixed the objective function, we follow our intention of expressing the results in terms of our original graph, so that we can orient Γ without having to explicitly construct any vector representation.

Observe that $g(S) = \min_{v \in S}\{\|s\|^2 - \|s-v\|^2\} = \min_{v \in S}\{2\langle s, v\rangle - 1\}$. However, as we are only interested in sinks instead of the actual value, we can just simplify to $\tilde{g}(S) = \min_{v \in S}\{\langle s, v\rangle\}$ (to simplify notation, we will continue using the symbol g instead of $\tilde{g}$). So we are led to analyze the behaviour of

$$\langle s, v\rangle = 1 + (c-d)|\operatorname{nb}_S(v)| + d(|S| - 1).$$

A characterization of how edges are directed at Γ with function g is at the following lemma.

LEMMA 4.2. *Let $G(c,d)$ be the Gram matrix of a vector representation, then*

- *If $d = 0$, the edge between S and $(S \cup u)$ is oriented towards $S \cup u$ if and only if $|\operatorname{nb}_S(u)| \geq \min_{v \in S}\{\operatorname{nb}_S(v)\}$*
- *If $d < 0$, the edge between S and $(S \cup u)$ is oriented towards $S \cup u$ if and only if $|\operatorname{nb}_S(u)| \geq \min_{v \in S}\{\operatorname{nb}_S(v)\}$ and u is adjacent to all the nodes that reach this minimum,*

PROOF. If $c > d = 0$, then $\langle s, v\rangle = 1 + c|\operatorname{nb}_S(v)|$, so the only important feature to determine the quality of a set of nodes is the number of neighbors of the least communicated node in the set. Concretely, given $s = \sum_{v\in S} v$ and $u \notin S$, the edge between S and $(S \cup u)$ is oriented towards $S \cup u$ if and only if $|\operatorname{nb}_S(u)| \geq \min_{v\in S}\{|\operatorname{nb}_S(v)|\}$. Roughly speaking, a community will absorb any new node that at least has the same number of neighbours as the least connected node already in the community.

Now consider the case when $d < 0$. Again $s = \sum_{v\in S} v$ and now we name $w \in S$ to be the node that achieves the minimum of $\langle w, s\rangle$. First, we assume there exists a $u \notin S$ such that u is not connected to w. Observe that $\min_{v\in S\cup u}\{\langle s + v, v\rangle\} = \langle s, w\rangle + d$, because

$$\langle s + u, v\rangle = \langle s, v\rangle + x$$

where $x \in \{1, c, d\}$, and choosing $v = w$ simultaneously minimizes both summands. Thus,

$$g(s + u) = \langle s, w\rangle + d < \langle s, w\rangle = g(s)$$

and the edge at Γ between S and $S \cup u$ will be oriented from $S \cup u$ towards S.

So, if we call $K \subseteq S$ the set of $w \in S$ that fulfill $\langle s, w\rangle = \min_{v\in S}\langle s, v\rangle$ (that is, the set of all the nodes that form a maximum angle with the sum vector) we will suppose from now on that $u \notin S$ is connected to at least all $w \in K$.

Now, if $|\operatorname{nb}_S(u)| \leq |\operatorname{nb}_S(w)|$ then

$$\begin{aligned}
g(S \cup u) &= min_{v\in S\cup u}\{\langle s + u, v\rangle\} \\
&\leq \langle s + u, u\rangle \\
&= 1 + (c - d)(|\operatorname{nb}_S(u)|) + d(|S| - 1) + d \\
&\leq 1 + (c - d)(|\operatorname{nb}_S(w)|) + d(|S| - 1) + d \\
&= min_{v\in S}\{\langle s, v\rangle\} + d \\
&< min_{v\in S}\{\langle s, v\rangle = \tilde{g}(s)
\end{aligned}$$

the minimum in $S \cup u$ is less or equal than the particular value in u, which is strictly smaller than $g(s)$. Thus, the orientation is again from $S \cup v$ to S.

Finally, we study the case when $|\operatorname{nb}_S(u)| > |\operatorname{nb}_S(w)|$. We need to spot the node that reaches the minimum of $\langle \cdot, s\rangle$ in $S \cup u$. It can be reached either a node in S or u. From the hypothesis $|\operatorname{nb}_S(u)| > |\operatorname{nb}_S(w)|$, u cannot reach this minimum. And, for all $v \in S \setminus K$ we have that

$$|\operatorname{nb}_S(v)| \geq |\operatorname{nb}_S(w)| + 1$$

and therefore the new minimum must be reached by a node in K. Hence, we have for all $v \in S \cup u$

$$\begin{aligned}
\langle s + u, v\rangle &= 1 + (c - d)(|\operatorname{nb}_S(v)| + |\operatorname{nb}_{\{u\}}(v)|) + d|S| \\
&\geq 1 + (c - d)(|\operatorname{nb}_S(w)| + 1) + d|S| \\
&= \langle s + u, w\rangle > g(S).
\end{aligned}$$

In particular $g(S \cup u) = \min\langle s + u, v\rangle > g(S)$. Therefore, in this case the orientation is from S to $S \cup u$.

In conclusion, if $d < 0$, then the orientation is towards $S \cup u$ if and only if u is adjacent to all the nodes w that reach $min_{v \in S}\{\langle s, v\rangle\}$ and $|\operatorname{nb}_S(u)| > |\operatorname{nb}_S(w)|$.

□

OBSERVATION 4.1. *As a direct consequence, we have that when $d < 0$, all paths that start at a singleton set lead to a sink that is one of the maximal cliques. And all maximal cliques are sinks.*

4.2. Study of paths. In order to determine the nature of sinks and understand the resulting communities, we start studying how paths are at the orientated graph $\Gamma(g)$.

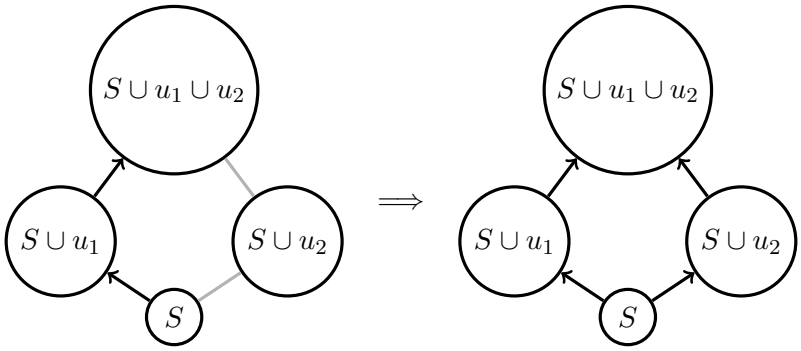
LEMMA 4.3. *When $d < 0$. If there is an ascending incrementing path*

$$S \longrightarrow S \cup u_1 \longrightarrow S \cup u_1 \cup u_2$$

at $\Gamma(g)$, then there is also the ascending incrementing path

$$S \longrightarrow S \cup u_2 \longrightarrow S \cup u_1 \cup u_2,$$

and therefore we can commute the order of node addition in the path.



PROOF. Let be $w = \operatorname{argmin}_{v \in S}\{\langle s, v\rangle\}$. Using the characterization of Lemma 4.2, if we have the orientation from S to $(S \cup u_1)$, then the edge w, u_1 must exist and $\operatorname{nb}_S(w) \leq \operatorname{nb}_S(u_1)$.

Moreover if we are increasing our community from $S \cup u_1$ to $(S \cup u_1 \cup u_2)$, then we must also have the edge w, u_2 (because w is still the minimum at $S \cup u_1$), and

$$\begin{aligned} |\operatorname{nb}_S(w)| + 1 &= |\operatorname{nb}_{S \cup u_1}(w)| \\ &\leq |\operatorname{nb}_{S \cup u_1}(u_2)| \\ &= |\operatorname{nb}_S(u_2)| + |\operatorname{nb}_{\{u_1\}}(u_2)| \leq |\operatorname{nb}_S(u_2)| + 1 \end{aligned}$$

So the edge w, u_2 exists and $|\operatorname{nb}_S(w)| \leq |\operatorname{nb}_S(u_2)|$, and therefore S points to $(S \cup u_2)$. It only remains to prove that we have the orientation $S \cup u_2 \longrightarrow S \cup u_1 \cup u_2$. But we must only notice that the minimum in $S \cup u_2$ is still at w, as in $S \cup u_1 \cup u_2$. And, as $|nb_{S \cup u_1 \cup u_2}(w)| = |nb_{S \cup u_2}(w)| + 1$, we have the expected orientation.

□

Some easy consequences:

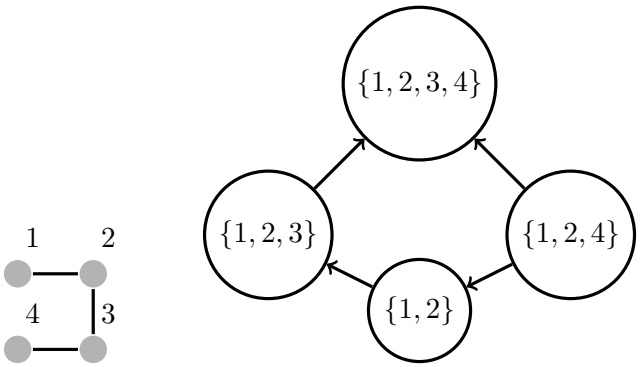
COROLLARY 4.1. *Let $A \subseteq S$ and $\{b_1, \ldots, b_n\} \in S$. For any permutation $\sigma \in \mathcal{S}_n$, denote $A_k = A \cup \{b_1, \ldots, b_k\}$ and $A_{\sigma(k)} = A \cup \{b_{\sigma(1)}, \ldots, b_{\sigma(k)}\}$. If $A_k \longrightarrow A_{k+1}$ for any $1 \leq k \leq n$, then, $A_{\sigma(k)} \longrightarrow A_{\sigma(k+1)}$ for all $\sigma \in \mathcal{S}_n$*

COROLLARY 4.2.
In an increasing path $a_0 \to a_1 \to \cdots \to a_n$, if we have a maximal ascending subpath $a_i \to a_{i+1} \to \cdots \to a_j$, then we cannot have a descent in $a_{i-1} \to a_i$ or $a_j \to a_{j+1}$ that consists in removing any element added at the ascending subpath.

However, this behaviour of paths does not hold with weaker hypothesis, which is proved in the following observations.

OBSERVATION 4.2. *Lemma 4.3 is not true anymore when $d = 0$.*

PROOF. As a counterexample, take the graph $G_1 = (V_1, E_1)$ with nodes $V_1 = \{1, 2, 3, 4\}$ and edges $E_1 = \{\{1, 2\}, \{2, 3\}, \{3, 4\}\}$. Then at $\Gamma_{G_1}(g)$ there is a path $\{1, 2\} \longrightarrow \{1, 2, 3\} \longrightarrow \{1, 2, 3, 4\}$ but $\{1, 2\} \longrightarrow \{1, 2, 4\} \longrightarrow \{1, 2, 3, 4\}$ is not a path because the orientations are $\{1, 2\} \longleftarrow \{1, 2, 4\} \longrightarrow \{1, 2, 3, 4\}$ (see Figure 4.6). □



(a) G_1 (b) Portion of $\Gamma_{G_1}(g)$

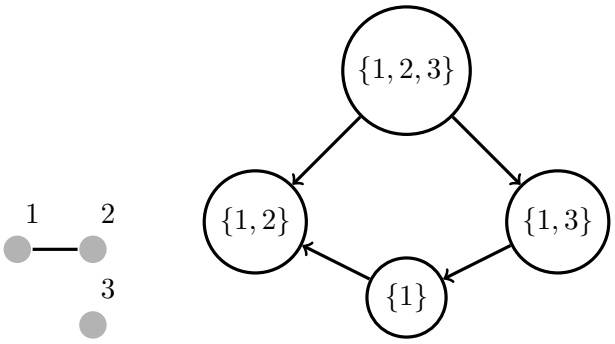
Figure 4.6: G_1 and some of the nodes and orientations of $\Gamma_{G_1}(g)$ when $c = 0$

OBSERVATION 4.3. *Lemma 4.3 is neither true for descending paths, i.e. for paths whose orientation is inverse to inclusion.*

PROOF. Now we can use the following counterexample, the graph $G_2 = (V_2, E_2)$ with nodes $V_2 = \{1, 2, 3\}$ and only one edge $E_2 = \{\{1, 2\}\}$. There is a path $\{1\} \longleftarrow \{1, 3\} \longleftarrow \{1, 2, 3\}$, but $\{1\} \longrightarrow \{1, 2\} \longleftarrow \{1, 2, 3\}$, so the descending moves do not commute (see Figure 4.7).

□

With what we have seen so far we have already enough to decide that this objective function is **not** a correct choice: it is way too fussy. We are looking only at the weakest node in the community, not at the overall quality of the community. Among other things, this implies that there are too many sinks. During the proof of Lemma 4.2 we saw that the weakest node w of



(a) G_2 (b) Portion of $\Gamma_{G_2}(g)$

Figure 4.7: G_2 and some of the nodes and orientations of $\Gamma_{G_2}(g)$

a set was preserved on the addition of new nodes that increased the fitness function. Moreover, it states that in order to increase the objective value, the added nodes had to be connected to this weakest node w. In conclusion, on an ascending increasing path[3], at each step we added a node adjacent to w, and therefore any increasing path could not be longer than the degree of the node w. In particular, it could not be longer than Δ, the maximum degree of the graph. This gives a hint on the great number of sinks.

Moreover, Observation 4.1 does not encourage us neither. We will not be able to find sinks via growing communities from a node, because this way we would only find the maximal cliques, that are too restrictive to portray the communities of a graph.

4.2.1. *Other possible definitions of an Increment Orientation.* We want to adapt the former fitness function so that it gives us a more general picture of the whole. Another way of defining the increment of a community could be to use the average of the increments of all paths that go through it. This takes all the members of the community into account.

So we define

$$h : 2^V \longrightarrow \mathbb{R} \tag{4.7}$$

$$s \longmapsto h(s) = \|s\|^2 - \frac{1}{|S|}\sum_{v\in S}\|s - v\|^2 \tag{4.8}$$

If we study this function in more detail, we see that

$$h(s) = 1 + \frac{2}{|S|}\sum_{v\in S}\langle v, s\rangle = 1 + \frac{2}{|S|}\langle\sum_{v\in S} v, s\rangle = 1 + 2\frac{\|s\|^2}{|S|}$$

and basically consists in scaling the norm of the set according to its size. To this formula in terms of our graph, we get the following proposition.

[3]Recall from Observation3.1 that here ascending refers to cardinals and increasing to objective function values

PROPOSITION 4.3. *$h(S) < h(S \cup u)$ if and only if*

$$(c-d)\left(|\operatorname{nb}_S(u)| - \frac{|\operatorname{E_{in}}(S)|}{|S|}\right) \geq -d\frac{|S|+1}{2}$$

PROOF. We have that the expression of this difference is

$$h(S \cup u) - h(S) = 2\left(\frac{\|s\|^2 + \|u\|^2 + 2\langle u, s\rangle}{|S|+1} - \frac{\|s\|^2}{|S|}\right) \geq 0$$

which translates into

$$1 + 2\left((c-d)\operatorname{nb}_S(u) + d|S|\right) - \left(1 + 2(c-d)\frac{|\operatorname{E_{in}}(S)|}{|S|} + d(|S|-1)\right) \geq 0$$

and gets finally simplified to

$$(c-d)\left(|\operatorname{nb}_S(v)| - \frac{|\operatorname{E_{in}}(S)|}{|S|}\right) \geq -d\frac{|S|+1}{2}$$

□

To determine the quality of this proposal, we will use the concrete example of a k-regular graph, and study in which cases we will get an orientation of $\Gamma(h)$ that would provide a path from the empty set to it. In concrete, we will see in which cases the addition of the last node results in an increase of the objective function.

EXAMPLE 4.4. *Let $S \cup u$ be a k-regular graph, $|S| = n$, then using the formula we just got in Proposition 4.3, we have the condition*

$$(c-d)\left(|\operatorname{nb}_S(u)| - \frac{|\operatorname{E_{in}}(S)|}{|S|}\right) \geq -d\frac{|S|+1}{2}$$

When applying the k-regularity, we get

$$(c-d)\left(k - \frac{nk-k}{2n}\right) \geq -d\frac{n+1}{2}$$

that simplifies into

$$\frac{(n-1)k}{(n+1)n} \geq \frac{-d}{c-d}$$

Therefore, the degree of the nodes must be

$$k \geq \frac{-d}{c-d}\frac{(n+1)n}{n-1} = \theta(n)$$

So in order to have an increasing last orientation, k must be $\mathcal{O}(n)$.

Recall that many of the real world graphs are *scale-free*, that means that their degree distribution follows a power law [**2, 4, 26**]. In concrete, the average degree is very small in comparison to the number of nodes, and it can not be $\mathcal{O}(n)$.

5. The directed Laplacian

After the failure of later attempts to define a good derivative on Γ, we make a new and final proposal: the *directed Laplacian*, which experimentally leads to the best results.

Moreover, we will henceforth focus in the Lovász' vector representation, and unless we state otherwise, our vector representations will be with $0 < c = -\frac{1}{\lambda_n} < 1$ and $d = 0$.

A classical tool of graph theory is the discrete Laplacian operator (or the normalized discrete Laplacian operator) which are deeply studied in [**9**]. Given an undirected graph, and a real valued function f on a graph, they are defined as

- Laplacian:

$$L_f(v) = \deg(v) \cdot f(v) - \sum_{u \sim v} f(u) \tag{4.9}$$

- Normalized Laplacian:

$$\mathcal{L}_f(v) = f(v) - \sum_{u \sim v} \frac{f(u)}{\sqrt{\deg(u)\deg(v)}} \tag{4.10}$$

This operator is heavily related to the Laplace-Beltrami operator for Riemmanian manifolds ($\triangle(f) = -\operatorname{div}(\bigtriangledown(f))$) [**9**]. In concrete, if we cover the manifold by a fine enough grid, then the discrete Laplacian estimates $\triangle$.

The core of this estimation lies at the approximation of the second order derivative

$$f''(x) \sim \frac{f(x+h) + f(x-h) - 2f(x)}{2h}, \tag{4.11}$$

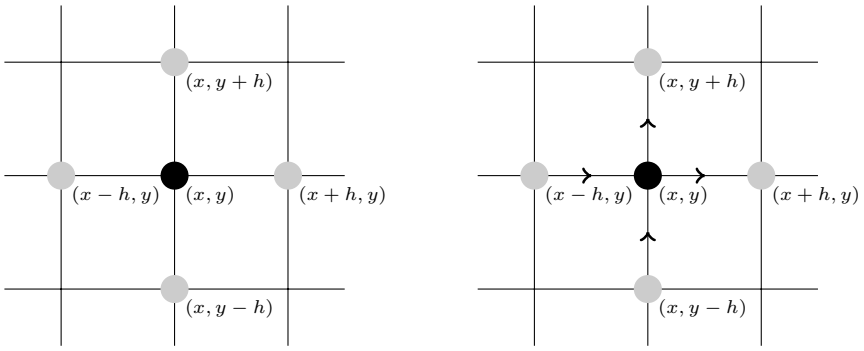
as if we take each edge as a direction, then the neighbors play the role of points at distance h of the vertex. However, we do not need second order derivatives but first, so we would like to use an approximation on the shape of

$$f'(x) \sim \frac{f(x) - f(x-h)}{h}. \tag{4.12}$$

But how can we tell which neighbors represent $f(x+h)$ and which $f(x-h)$? Here the orientation of $\Gamma_G^{\uparrow}$ comes to rescue us, as we can distinguish between in-neighbors and out-neighbors. The next observation clarifies this.

Observation 4.4. *If we restrict to the Euclidean plane and draw a grid around a point (see Figure 4.8a), and consider the neighbors of a node in the grid, then we can approximate the Laplacian at the node $v = (x, y)$ using* (4.11) *as*

$$\begin{aligned} \triangle f(x,y) &= \tfrac{\partial^2 f}{\partial x^2}(x,y) + \tfrac{\partial^2 f}{\partial y^2}(x,y) \\ &\sim \tfrac{f(x+h,y)+f(x-h,y)-2f(x,y)}{2h} + \tfrac{f(x,y+h)+f(x,y-h)-2f(x,y)}{2h} \\ &= \tfrac{-1}{2h}\left(\deg(u)\, f(u) - \textstyle\sum_{v \sim u} f(v)\right) \end{aligned}$$



(a) undirected grid *(b) directed grid*

Figure 4.8

from which we can see the strong relation to the graph Laplacian. However, we are interested in the first order derivative, not the second. In particular, if we try to estimate the divergence of the function, we need to use a directed grid as in Figure 4.8b. If the approximating method uses backward differences approximation, we orient the edges in the same direction as the canonical basis.

$$\begin{aligned}\operatorname{div} f(x,y) = \bigtriangledown f(x,y) &= \tfrac{\partial f}{\partial x}(x,y) + \tfrac{\partial f}{\partial y}(x,y) \\ &\sim \tfrac{f(x,y)-f(x-h,y)}{h} + \tfrac{f(x,y)-f(x,y-h)}{h} \\ &= \tfrac{1}{h}\left(\operatorname{indeg}(u)\, f(u) - \textstyle\sum_{v\to u} f(v)\right)\end{aligned}$$

Therefore, changing neighbors for incoming neighbors and degrees for indegrees in the discrete Laplacian definition, relates the Laplacian to the divergence.

This observation leads us to the following definition:

Definition 4.1. *The value at v of the* directed Laplacian *of a function f on the directed graph $\Gamma^{\uparrow}$ is*

$$\mathcal{L}_{\Gamma^{\uparrow},f}(v) \;=\; f(v) - \frac{1}{\sqrt{\operatorname{indeg}(v)}} \sum_{u:u\to v} \frac{f(u)}{\sqrt{\operatorname{indeg}(u)}}.$$

Wherever it is defined. That is, where there is no in-neighbor that has $\operatorname{indeg}(u) = 0$. *In this case, we set* $\mathcal{L}_{\Gamma^{\uparrow},f}(v) = 0$.

Now the next step is clear: we want to apply this operator on the squared norm, and use it as the fitness function. We want to set $\phi = \mathcal{L}_{\Gamma^{\uparrow},\|\cdot\|^2}$. But before, we should make several observations:

Observation 4.5. *By the definition of the orientation on $\Gamma^{\uparrow}$, the indegree* $\operatorname{indeg}(v)$ *of any node $v \in \Gamma^{\uparrow}$ is $|S|$, the cardinality of the subset S modeled by v in Γ. Notice that the only cases when we divide by zero are when $|S| = 0$ or $|S| = 1$, that is the empty set and the singletons. Otherwise, it is well defined.*

OBSERVATION 4.6. *In the community search, we can ask whether the usual Laplacian operators (those defined at* (4.9) *and* (4.10)*) could be useful. They are not. If we use the first definition of the Laplacian* (4.9) *on the squared length, we get after some calculations:*

$$\begin{aligned} L_{\|\cdot\|^2}(s) &= (n-1)\cdot\|s\|^2 - \sum_{v\in s}\|s\setminus v\|^2 - \sum_{v\notin s}\|s\cup v\|^2 \\ &= 2\,|s| - n + 1 + 4\,c|\,\mathrm{E}_{\mathrm{in}}(s)| - 2\,c\,\mathrm{E}_{\mathrm{out}}(s) \end{aligned}$$

Why is this not good enough? If we specify the growing condition, we get

$$L_{\|\cdot\|^2}(S\cup v) - L_{\|\cdot\|^2}(S) = 2 + 4c|\,\mathrm{nb}_S(v)| - 2c(\deg(v) - 2\,|\,\mathrm{nb}_S(v)|)$$

For example we can easily see that it is possible to add new nodes v to a community S even if they are in a different connected component (which happens for example for isolated nodes or nodes of degree 1*). Moreover, the general behaviour of this method is that for each new edge added to the community, the node is allowed to have more edges towards outside and still increase the objective function.*

We can easily see that in our case, as Γ is regular, if we apply the normalized Laplacian (4.10) *we obtain the same fitness function values as $L_{\|\cdot\|^2}$ multiplied by a constant term depending only on the size of the graph, and thus inducing the same orientation in Γ (and giving as a result the same communities).*

OBSERVATION 4.7. *We can also wonder why we do not use an adapted version of the Laplacian* (4.9) *instead of a version of the normalized Laplacian*(4.10)*. That is: is $L_f(v) = \deg_{in}(v)\cdot f(v) - \sum_{u\to v} f(u)$ a suitable choice? Let's analyse it. We get the following:*

$$\begin{aligned} \mathcal{L}_{\|\cdot\|^2}(s) &= |s|\cdot\|s\|^2 - \sum_{v\in s}\|s\setminus v\|^2 = \\ &= |s|\cdot\|s\|^2 - \sum_{v\in s}\|s\|^2 - 1 - 2c\,\mathrm{nb}_s(v) = \\ &= |s| + 4c|\,\mathrm{E}_{\mathrm{in}}(s)| \end{aligned}$$

Which has the only maximum in the connected components! The point is at the fact that nodes of $\Gamma^{\uparrow}$ have a different in-degree at each node, and thus the dimension of the space where we are deriving is continuously changing. We need to normalize this situation somehow, otherwise nodes with greatest indegree are reinforced.

OBSERVATION 4.8. *There are already some definitions at the literature of Laplacian operators on directed graphs. One of the most extended is first proposed by Chung at* [**10**]*, based on Perron vectors. However, it requires the directed graph to be strongly connected. We must remark that $\Gamma^{\uparrow}$ is* ***not*** *strongly connected. Moreover, on strongly connected graphs the directed Laplacian is also always well defined.*

OBSERVATION 4.9. *When working with the Laplacian, it is very important to keep in mind Observation 3.1, and clearly differentiate between the*

two orientations we have provided to Γ. *The cardinality orientation* $\Gamma^{\uparrow}$ *is used to define the objective function* $\phi = \mathcal{L}_{\Gamma^{\uparrow},\|\cdot\|^2}$. *And then this function is used to induce a new orientation* $\Gamma(\phi)$.

After these appreciations, we return to our concrete problem. We will proceed to apply the directed Laplacian on the squared length to define a new objective function. Here and in the following we set $s := |S|$, and assume $s > 1$.

Definition 4.2. *The directed squared-length Laplacian of a subset of nodes* $S \subset V$ *of a graph* $G = (V, E)$ *is*

$$\mathcal{L}_{\Gamma^{\uparrow},\|\cdot\|^2}(S) = \|S\|^2 - \frac{1}{\sqrt{s(s-1)}} \sum_{v \in S} \|S \setminus v\|^2 \tag{4.13}$$

where $\|\cdot\|$ *refers to the norm of its Lovász representation.*

From now on, without risk of confusion, we will simplify notation and we will denote as $\mathcal{L}$ the directed Laplacian of the squared norm on $\Gamma^{\uparrow}$ ($\mathcal{L}_{\Gamma^{\uparrow},\|\cdot\|^2}$), and we will refer to it as the directed Laplacian or simply the Laplacian.

To better understand the behaviour of the directed Laplacian as objective function, we present some results. First of all, we will translate the formula in graph terms, in the following small lemma.

Lemma 4.4. *The directed Laplacian evaluates to*

$$\mathcal{L}_{\Gamma^{\uparrow},\|\cdot\|^2}(S) = s - \sqrt{s(s-1)} + 2c\,\mathrm{E}_{\mathrm{in}}(S)\left(1 - \frac{s-2}{\sqrt{s(s-1)}}\right).$$

where $s = |S|$.

Proof. Using (3.5) and (3.3) to expand the norms, and recalling that $\sum_{u \in S} |\,\mathrm{nb}_S(u)| = 2\,\mathrm{E}_{\mathrm{in}}(S)$, we obtain

$$\begin{aligned}
\mathcal{L}_{\Gamma^{\uparrow},\|\cdot\|^2}(S) &= \|S\|^2 - \frac{1}{\sqrt{s(s-1)}} \sum_{v \in S} \|S \setminus v\|^2 \\
&= \|S\|^2 - \frac{1}{\sqrt{s(s-1)}} \sum_{v \in S} \|S\|^2 - 1 - 2c\,\mathrm{nb}_S(v) \\
&= \|S\|^2 \left(1 - \frac{s}{\sqrt{s(s-1)}}\right) + \frac{s}{\sqrt{s(s-1)}} + \frac{4c|\,\mathrm{E}_{\mathrm{in}}(S)|}{\sqrt{s(s-1)}} \\
&= (s + 2c\,|\,\mathrm{E}_{\mathrm{in}}(S)|)\left(1 - \frac{s}{\sqrt{s(s-1)}}\right) + \frac{s}{\sqrt{s(s-1)}} + \frac{4c|\,\mathrm{E}_{\mathrm{in}}(S)|}{\sqrt{s(s-1)}} \\
&= s - \sqrt{s(s-1)} + 2c|\,\mathrm{E}_{\mathrm{in}}(S)|\left(1 - \frac{s-2}{\sqrt{s(s-1)}}\right).
\end{aligned}$$

□

An important question is: what makes communities grow for this Laplacian?

LEMMA 4.5. *The difference of the directed Laplacian of a set S and the directed Laplacian after adding a node u is*

$$\begin{aligned}(4.14)\ \mathcal{L}(S\cup u)-\mathcal{L}(S) &= 2c\,\mathrm{nb}_S(u)\left(1-\frac{s-1}{\sqrt{s(s+1)}}\right)\\ &\quad+2c\,\mathrm{E}_{\mathrm{in}}(S)\left(\frac{s-2}{\sqrt{s(s-1)}}-\frac{s-1}{\sqrt{s(s+1)}}\right)\\ &\quad+1-\sqrt{s}\big(\sqrt{s+1}-\sqrt{s-1}\,\big)\end{aligned}$$

PROOF. First, we need the objective value of $S\cup u$,

$$\mathcal{L}(S\cup u)=s+1-\sqrt{s(s+1)}+2c|\,\mathrm{E}_{\mathrm{in}}(S\cup v)|\left(1-\frac{s-1}{\sqrt{s(s+1)}}\right)$$

and now, using the relation $\mathrm{E}_{\mathrm{in}}(S\cup u)=\mathrm{E}_{\mathrm{in}}(S)+\mathrm{nb}_S(u)$, the result is straightforward

$$\begin{aligned}\mathcal{L}(S\cup u)-\mathcal{L}(S) &= 2c\,\mathrm{nb}_S(u)\left(1-\frac{s-1}{\sqrt{s(s+1)}}\right)\\ &\quad+2c\,\mathrm{E}_{\mathrm{in}}(S)\left(\frac{s-2}{\sqrt{s(s-1)}}-\frac{s-1}{\sqrt{s(s+1)}}\right)\\ &\quad+1-\sqrt{s}\big(\sqrt{s+1}-\sqrt{s-1}\,\big)\end{aligned}$$

□

Despite the apparent complexity of this expression, we can easily understand it asymptotically

THEOREM 4.1. *Using the objective function $\phi=\mathcal{L}_{\Gamma^{\uparrow},\|\cdot\|^2}$, a node u of G gets added to a community S of large enough size $|S|\gg 0$ if and only if*

$$(4.15)\qquad \mathrm{nb}_S(u)\ >\ \mathrm{avdeg}(S),$$

i.e., the number of neighbors of u in S is greater than the average degree of the nodes in S. In particular,

PROOF. To estimate the difference $\mathcal{L}(S\cup u)-\mathcal{L}(S)$ asymptotically for large $s=|S|$, we start at the formula in Lemma 4.5, and apply some Taylor expansions at infinity

$$\begin{aligned}\mathcal{L}(S\cup u)-\mathcal{L}(S) &= 2c\,\mathrm{nb}_S(u)\left(1-\frac{s-1}{\sqrt{s(s+1)}}\right)\\ &\quad+2c\,\mathrm{E}_{\mathrm{in}}(S)\left(\frac{s-2}{\sqrt{s(s-1)}}-\frac{s-1}{\sqrt{s(s+1)}}\right)\\ &\quad+1-\sqrt{s}\big(\sqrt{s+1}-\sqrt{s-1}\,\big)\\ &= c\,\mathrm{nb}_S(u)\left(\frac{3}{s}+O(s^{-2})\right)\\ &\quad+c\,\mathrm{E}_{\mathrm{in}}(S)\left(-\frac{3}{s^2}+O(s^{-3})\right)+O(s^{-2}),\end{aligned}$$

and this is nonnegative for $s \gg 0$ iff $\mathrm{nb}_S(u) > \mathrm{E}_{\mathrm{in}}(S)/s$. □

COROLLARY 4.3. *The asymptotic behavior of $\mathcal{L}$ does not depend on the value of c.*

COROLLARY 4.4. *"Only trees grow hairs": Setting $\mathrm{nb}_S(u) = 1$ we get that nodes with one neighbor in S get accepted only if S is a tree.*

REMARK 4.2. *Theorem 4.1 tells us that for large enough communities, the behavior of $\mathcal{L}$ is indistinguishable from the simple heuristic* (4.15)*. On the one hand, this is an argument in favor of the naturalness of $\mathcal{L}$; on the other, it says that for small communities, the squared-length Laplacian takes the local structure much more into account than this simple rule can. This is all the more important as our computational experience suggests that most communities in real-world graphs are in fact small, cf. Figure 7.7 (in Chapter 7).*

In view of this remark, we turn to small communities. For constant $s \geq 2$, the right-hand side of (4.14) is positive iff $\mathrm{nb}_S(u) > \kappa(c, \mathrm{E}_{\mathrm{in}}) = \frac{\alpha}{c} + \beta\,\mathrm{E}_{\mathrm{in}}$, where α and β are explicit constants that depend on s. This tells us, for any fixed community of size s with E_{in} internal edges, how many neighbors in S a node u must have in order to get accepted.

REMARK 4.3. *Notice how the local growth of communities is governed by the choice of c, a parameter determined globally by the entire graph. We also see again the naturalness of using the largest possible value of c, this time because it imposes the fewest restrictions on the growth of communities.*

For example, paths of length 2 cannot grow at all for values of c in the range $[0, 0.02275)$, can grow only to triangles for $c \in (0.02275, 0.09612)$, and to triangles or paths of length three for $c \in (0.09612, 1]$. Analogous computations can be carried out for larger communities.

6. The conductance

We will also present another objective function well used in the literature, the *conductance* [**25, 27**]. From now on, the directed Laplacian and the conductance will be compared in order to determine the goodness of our proposal against established ones.

DEFINITION 4.3. *The* conductance *of a subset $S \subseteq V$ in a weighted graph with edge weights a_{ij} is* [**25**]

$$\varphi(S) \;=\; \frac{\sum_{i\in S,\, j\notin S} a_{ij}}{\min\big\{a(S), a(V\setminus S)\big\}}\,,$$

where $a(S) = \sum_{i\in S,\, j\in V} a_{ij}$. This specializes to

$$\varphi(S) \;=\; \frac{\mathrm{E}_{\mathrm{out}}(S)}{\mathrm{E}_{\mathrm{out}}(S) + \min\big\{\,\mathrm{E}_{\mathrm{in}}(S),\, \mathrm{E}_{\mathrm{in}}(V\setminus S)\big\}}.$$

where $\mathrm{E}_{\mathrm{out}}$ counts the edges in G with exactly one end in S.

OBSERVATION 4.10. *As we do not expect communities to be large enough to contain more than half the edges of the graph, we will often be able to*

simplify the minimum at the definition $\min\{\mathrm{E}_{\mathrm{in}}(S), \mathrm{E}_{\mathrm{in}}(V \setminus S)\} \sim \mathrm{E}_{\mathrm{in}}(S)$, *and use instead as definition of conductance*

$$\varphi(S) \;=\; \frac{\mathrm{E}_{\mathrm{out}}(S)}{\mathrm{E}_{\mathrm{out}}(S) + \mathrm{E}_{\mathrm{in}}(S)}$$

Since the conductance measures the ratio of outgoing edges to the total number of edges in the set or its complement (whichever is smaller), we will seek to minimize it (instead of maximizing it as we used to with general fitness functions).

THEOREM 4.2. *Suppose that* $\mathrm{E}_{\mathrm{in}}(S) \le \mathrm{E}_{\mathrm{in}}(V \setminus S) - \Delta$, *where* $\Delta = \mathrm{maxdeg}(G)$, *and set* $R = \mathrm{E}_{\mathrm{out}}(S)/\mathrm{E}_{\mathrm{in}}(S)$. *Then a node* $u \in V \setminus S$ *improves the conductance* ϕ *if and only if*

$$\mathrm{nb}_S(u) \;>\; \frac{\deg(u)}{R+2}\,.$$

In particular, nodes with one neighbor in S *can get accepted to* S *only if* $\deg(u) < R + 2$.

PROOF. Using

$$\begin{aligned}
\mathrm{E}_{\mathrm{out}}(S \cup u) &= \mathrm{E}_{\mathrm{out}}(S) + \deg(u) - 2\,\mathrm{nb}_S(u),\\
\mathrm{E}_{\mathrm{in}}(S \cup u) &= \mathrm{E}_{\mathrm{in}}(S) + \mathrm{nb}_S(u),\\
\mathrm{E}_{\mathrm{in}}\big(V \setminus (S \cup u)\big) &= \mathrm{E}_{\mathrm{in}}(V \setminus S) - \deg(u) + \mathrm{nb}_S(u)
\end{aligned}$$

we find

$$\phi(S \cup u) \;=\; \frac{\mathrm{E}_{\mathrm{out}}(S) - 2\,\mathrm{nb}_S(u) + \deg(u)}{\mathrm{E}_{\mathrm{out}}(S) - \mathrm{nb}_S(u) + \deg(u) + m},$$

where we abbreviate $m = \min\{\mathrm{E}_{\mathrm{in}}(S), \mathrm{E}_{\mathrm{in}}(V \setminus S) - \deg(u)\}$. By assumption, $\mathrm{E}_{\mathrm{in}}(S) \le \mathrm{E}_{\mathrm{in}}(V \setminus S) - \deg(u)$, so that

$$\phi(S \cup u) - \phi(S) \;=\; \frac{\deg(u) - (R+2)\,\mathrm{nb}_S(u)}{(R+1)\big(T + \mathrm{nb}_{V\setminus S}(u)\big)} \tag{4.16}$$

with $T = \mathrm{E}_{\mathrm{in}}(S) + \mathrm{E}_{\mathrm{out}}(S)$. The claim follows. □

CHAPTER 5

Analysis of the sinks

After presenting our proposals of objective functions, in this chapter we will study how they act. We do this by looking at the orientation of edges in $\Gamma(\phi)$ and the structure of sinks in particular cases.

We stress that here we undertake this study from a theoretical point of view. Experimental analyses are performed in Chapter 7.

OBSERVATION 5.1 (Hairs). *Before concentrating on special graphs, our first object of study are "hairs", nodes of degree 1. Notice that "hairs" get treated differently by the Laplacian and the conductance.*

Suppose that S is a large enough set, and that u is a node with total degree $\deg(u)$*, and exactly one neighbor in S. From Theorem 4.1, we know that u gets accepted into S by the Laplacian if and only if S is a tree, irrespective of* $\deg(u)$*.*

The conductance, however, always rejects u if $\deg(u) \geq 2$*, and accepts nodes of total degree* 1 *only if* $\mathrm{E}_{\mathrm{out}}(S) < \mathrm{E}_{\mathrm{in}}(S)$*.*

REMARK 5.1. *Observe that not all nodes are always taken in some community. There are nodes that do not belong to any community!*

1. Cartesian product of cliques

We study now a family of graphs we already know from last chapter, the cartesian products of a pair of cliques $G = K_r \square K_s$. G can be seen as s copies of K_r and/or r copies of K_s. Specifically if $r \gg s$, their inherent structure makes this graph look like a collection of s copies of K_r (see Figure 4.2).

Our first step must consist in calculating the smallest eigenvalue of G. To achieve this, we recall a known result about cartesian products [**20**]:

LEMMA 5.1. *If G and H are graphs with eigenvalues $\lambda_1, \ldots, \lambda_n$ and $\mu_1, \ldots, \mu_n$ respectively, then the eigenvalues of $G \square H$ are $\lambda_i + \mu_j$ where $1 \leq i \leq n$ and $1 \leq j \leq m$.*

Moreover, for any complete graph the smallest eigenvalue is always -1, so the smallest eigenvalue of $K_r \square K_s$ is -2 and hence $c = \frac{1}{2}$.

It is reasonable that the copies of K_r and K_s should be sinks in $K_r \square K_s$, and our objective will be to to prove or disprove this, for each objective function.

1.1. Directed Laplacian. Given a copy H of K_r included in G, we will see that H is a sink if $r \geq 3$. We prove this in two steps, using the following notation: $H^+ = H \cup u$ denotes the union of H with one of its neighbors $u \in V$, and $H^- = H \setminus u$ for any $u \in H$.

First we see that we cannot add a node without decreasing the objective value.

PROPOSITION 5.1. *For $r \geq 3$ we have $\mathcal{L}(K_r) - \mathcal{L}(K_r^+) > 0$.*

PROOF. We observe that $\mathrm{E}_{\mathrm{in}}(K_r) = \frac{r(r-1)}{2}$ and $\mathrm{E}_{\mathrm{in}}(K_r^+) = \frac{r(r-1)}{2} + 1$. This leads to

$$\mathcal{L}(K_r) - \mathcal{L}(K_r^+) = \frac{\left(-r^5 + r^3\right)\sqrt{r\,(r-1)} + \left(r^5 - r^4 + 5\,r^3 - 7\,r^2 + 2\,r\right)\sqrt{(r+1)\,r} + 4\,r^3 - 4\,r}{2\,(r^4 - r^2)},$$

which is always positive if $k \geq 3$. The proof needs some cumbersome calculations, that will be skipped[1]. Instead we will provide some asymptotic indication. From the Taylor expansion,

$$\mathcal{L}(K_r) - \mathcal{L}(K_r^+) = \frac{24\,r^2 - 76\,r + 51 + 32\,O\left(r^{-1}\right)}{32r^2},$$

and both roots of the parabola $24\,r^2 - 76\,r + 51$ are smaller than 3. □

Next, we study the possibility of removing a node from the clique.

PROPOSITION 5.2. *For all $r > 1$ we have $\mathcal{L}(K_r) - \mathcal{L}(K_r^-) > 0$.*

PROOF. Now we will have to use $\mathrm{E}_{\mathrm{in}}(K_r^-) = \frac{(r-2)(r-1)}{2}$.

$$\mathcal{L}(K_r) - \mathcal{L}(K_r^-) = \frac{2 - \sqrt{(r-1)\,r} + \sqrt{r^2 - 3\,r + 2} - \sqrt{-3\,r^{-1} + 2\,r^{-2} + 1}}{2\,r}$$

And in this case the calculations are much nicer. Basically, both $\sqrt{r^2 - 3\,r + 2} - \sqrt{(r-1)\,r}$ and $-\sqrt{-3\,r^{-1} + 2\,r^{-2} + 1}$ are functions that decrease to -1, but from $r = 2$ are both strictly greater than -1, which proves our claim. □

Therefore, we can conclude that for $r, s > 2$, the maximal cliques in $K_r \square K_s$ are always sinks, as there is no improvement neither adding nor removing a node.

1.2. Conductance. Now we face the study of the same case with the conductance, which results in a much simpler analysis.

PROPOSITION 5.3. *If $r \geq 3$ then K_r is a sink.*

PROOF. For the conductance we also need the number of outgoing edges. We can see that $\mathrm{E}_{\mathrm{out}}(K_r) = k(l-1)$, $\mathrm{E}_{\mathrm{out}}(K_r^+) = kl + l - 4$ and $\mathrm{E}_{\mathrm{out}}(K_r^-) = kl - l$. Then,

$$\mathcal{L}(K_r) - \mathcal{L}(K_r^+) = -2\,\frac{(r-3)\,(r-2+s)}{(r-3+2\,s)\,(r^2 - r - 6 + 2\,rs + 2\,s)}$$

$$\mathcal{L}(K_r) - \mathcal{L}(K_r^-) = -2\,\frac{r-2+s}{(r-2+2\,s)\,(r-3+2\,s)}$$

It is straightforward to see that for $r \geq 3$ both expressions are positive. □

Therefore, with this family of graphs, both functions find clear communities as sinks.

[1]Using twice the fact $a - b \geq 0 \Leftrightarrow a^2 - b^2 \geq 0$ when $a, b \geq 0$, we can get rid of the square roots and obtain a polynomial of degree 24, where some easy coefficient manipulations show the result

2. The d-cube

Now we proceed to study the next case, the graph of the d-cube Q_d. It is defined recursively by the following procedure: the graph of the 2-cube corresponds to K_2 (an edge), while the graph of the k-cube is the cartesian product of an edge with the $k-1$-cube, $Q_k = Q_{k-1} \square K_2$.

As in the previous case, this is a very well known graph. Although it does not have a clear community structure, it does have some differentiated subsets, the faces (induced subgraphs which are isomorphic to a smaller dimensional cube), which are the only reasonable bet to be sinks. Our initial goal was to prove that the sinks are the k-faces of Q_d, but what we will see is that almost all faces are sinks, but not all sinks are faces.

First, we compute the smallest eigenvalue using Lemma 5.1. Since $\mathrm{Spec}(K_2) = \{-1, 1\}$, the smallest eigenvalue of Q_d is $-d$, and hence $c = \frac{1}{d}$.

2.1. Directed Laplacian. First we study whether faces are sinks. It is easy to see that if F is the graph of a k-face, it has 2^k vertices and $\mathrm{E}_{\mathrm{in}}(F) = k2^{k-1}$. So the Laplacian evaluates to,

$$\mathcal{L}(F) = 2^k - \sqrt{2^k(2^k-1)} + \frac{k2^k}{d}\left(1 - \frac{2^k-2}{\sqrt{2^k(2^k-1)}}\right)$$

Again, we will use the notation $F^+ = F \cup v$, where v is adjacent to a node in F, and $F^- = F \setminus u$ where u is any of the vertices in F. And we try to study whether the faces F of a cube are sinks. The answer will be that "almost" all the faces of the cube are sinks, with quite a precise expression of what "almost" means. The main results get reduced to the following two propositions.

PROPOSITION 5.4. *For all $k \geq 2$, $\mathcal{L}(F) > \mathcal{L}(F^+)$.*

PROOF. We know that $|F^+| = 2^k + 1$ and $\mathrm{E}_{\mathrm{in}}(F^+) = k2^{k-1} + 1$, and we want to check that $\mathcal{L}(F) - \mathcal{L}(F^+) > 0$. To see that this inequality holds for every $2 \leq k \leq d$, we expand this expression to

$$\begin{aligned}(5.1)\quad \mathcal{L}(F) - \mathcal{L}(F^+) &= -\sqrt{2^k\,(2^k-1)} + k2^k\left(1 + \frac{-2^k+2}{\sqrt{2^k\,(2^k-1)}}\right)d^{-1} \\ &\quad -1 + \sqrt{(2^k+1)\,2^k} \\ &\quad + \left(-k2^k - 2\right)\left(1 - \frac{2^k-1}{\sqrt{(2^k+1)\,2^k}}\right)d^{-1}.\end{aligned}$$

To study its sign, we examine its vanishing locus. We will see that its vanishing locus only contains negative d's.

If we assume that there are only negative d's, then our result then follows from this appreciation: The vanishing locus divides the plane between regions where $\mathcal{L}(F) > \mathcal{L}(F^+)$ and where $\mathcal{L}(F) < \mathcal{L}(F^+)$. If the entire vanishing locus has negative d's, then all points with a positive d are in the same region. To see that this region is the positive one, we observe that the limit of $\mathcal{L}(F) - \mathcal{L}(F^+)$ when d tends to infinity is

$$\lim_{d\to\infty}\left(\mathcal{L}(F) - \mathcal{L}(F^+)\right) = \sqrt{4^k + 2^k} - \sqrt{4^k - 2^k} - 1,$$

and some elementary square root operations show that this expression is always positive. Therefore all pairs (k, d) with $k \geq 2$ and $d > 0$ verify our original claim.

So is still left to study the points where $\mathcal{L}(F) - \mathcal{L}(F^+) = 0$. In particular, it is a linear equation in d, and we are able to express d as a function of k:

$$d = -\frac{\left(-1 + \frac{2^k - 1}{\sqrt{4^k + 2^k}}\right)\left(k2^k + 2\right) + 2^k k\left(1 - \frac{2^k - 2}{\sqrt{4^k - 2^k}}\right)}{-1 + \sqrt{4^k + 2^k} - \sqrt{4^k - 2^k}}. \tag{5.2}$$

After some calculations, we can express d as a quotient $d = \frac{a(k)}{b(k)}$, where

$$\begin{aligned} a(k) &= k\left(4^k - 2^{1+k}\right)\sqrt{4^k + 2^k} + 2\sqrt{4^{2\,k} - 2^{2\,k}} + \\ &\quad \left(k\left(-4^k + 2^k\right) + 2 - 2^{1+k}\right)\sqrt{4^k - 2^k} \end{aligned} \tag{5.3}$$

and

$$\begin{aligned} b(k) &= \left(-4^k + 2^k\right)\sqrt{4^k + 2^k} - \sqrt{4^{2\,k} - 2^{2\,k}} + \\ &\quad \left(4^k + 2^k\right)\sqrt{4^k - 2^k}. \end{aligned} \tag{5.4}$$

We will see that the numerator is negative and the denominator is positive. Therefore there will be no d-dimensional cube with $\mathcal{L}(F) < \mathcal{L}(F^+)$. First we prove that the denominator is positive. Some elementary calculus shows us that

$$\frac{b(k)}{\sqrt{16^k - 4^k}} = \sqrt{4^k + 2^k} - \sqrt{4^k - 2^k} - 1 > 0.$$

And concerning the numerator, we get

$$\begin{aligned} a(k)\sqrt{16^k - 4^k} &= \left(4^k k - 2^{1+k}k\right)\left(4^k + 2^k\right)\sqrt{4^k - 2^k} \\ &\quad + 2\left(4^k - 2^k\right)\left(4^k + 2^k\right) \\ &\quad - \left(-2^k k + 4^k k + 2^{1+k} - 2\right)\left(4^k - 2^k\right)\sqrt{4^k + 2^k} \\ &= -16^k k\left(\sqrt{4^k + 2^k} - \sqrt{4^k - 2^k} - \frac{2}{k}\right) \\ &\quad - k\sqrt{4^k - 2^k}8^k \\ &\quad - 4^k k\left(2\sqrt{4^k - 2^k} + \sqrt{4^k + 2^k} + 2\,\frac{\sqrt{4^k + 2^k}}{k} - \frac{2}{k}\right) \\ &\quad - 2^{1+k}\sqrt{4^k + 2^k}. \end{aligned}$$

Studying this expression componentwise, we see that it is clearly smaller than 0 for all $k \geq 2$. This concludes the proof. □

Proposition 5.5. *Asymptotically,*

$$\operatorname{sign}(\mathcal{L}(F) - \mathcal{L}(F^-)) = \operatorname{sign}\left(k - \frac{1}{\ln 2}\, W\left(\frac{d \ln 2}{12}\right)\right)$$

where $W(z) \sim \ln z - \ln\ln z$ *is the Lambert W-Function*[2].

PROOF. We have now $|F^-| = 2^k - 1$ and $\mathrm{E}_{\mathrm{in}}(F^-) = k2^k - k$. Thus,

$$\begin{aligned}\mathcal{L}(F) - \mathcal{L}(F^-) &= -\sqrt{2^k\,(2^k-1)} \\ &\quad + \frac{2\,2^{k-1}k\left(1 - \frac{2^k-2}{\sqrt{2^k\left(2^k-1\right)}}\right)}{d} \\ &\quad + 1 + \sqrt{(2^k-1)\,(2^k-2)} \\ &\quad - \frac{2\,\left(2^{k-1}k-k\right)\left(1 - \frac{2^k-3}{\sqrt{\left(2^k-1\right)\left(2^k-2\right)}}\right)}{d}\end{aligned} \tag{5.5}$$

Again, we will use the strategy of studying the vanishing locus of $\mathcal{L}(F) - \mathcal{L}(F^-) = 0$. And now we get

$$d = \frac{4^k k\,(B-A) + 2^k k\,(5\,A - 2\,B) - 2\,k\,(AB + 3\,A)}{(A-B)\,4^k + (B - 3\,A)\,2^k + AB + 2\,A} = \alpha(k) \tag{5.6}$$

where $A = \sqrt{2^k\,(2^k-1)}$ and $B = \sqrt{(2^k-1)\,(2^k-2)}$.

If we focus on the asymptotic case ($k \gg 0$), we observe that

$$\alpha(k) \sim 12\,k2^k, \tag{5.7}$$

that is, $\lim_{k\to\infty} \frac{\alpha(k)}{12\,k2^k} = 1$; therefore, the boundary of the acceptance region is $k = \frac{1}{\ln 2}\,W\left(\frac{d\ln 2}{12}\right)$.

We still need to see for which of the two regions $\mathcal{L}(F) > \mathcal{L}(F^-)$ holds. We determine it via the limit of $\mathcal{L}(F) - \mathcal{L}(F^-)$ when d tends to infinity.

$$\lim_{d\to\infty}\left(\mathcal{L}(F) - \mathcal{L}(F^-)\right) = \sqrt{4^k - 3\,2^k + 2} - \sqrt{4^k - 2^k} + 1 < 0 \tag{5.8}$$

This serves to prove our claim. □

In particular, merging together the results of these two last propositions, we see that F is a sink if $k = \Omega(\log(d))$.

In conclusion, even if we did not see that all faces are sinks, we proved that apart from the faces of smaller dimension, faces are sinks. To illustrate it, we present Figure 5.1, where we plot the function $\frac{1}{\ln 2}\,W\left(\frac{d\ln 2}{12}\right)$ that delimites which faces are sinks. Observe that for $d = 1000$, **all** faces with dimension greater than 4 are sinks.

Now that we have already seen which faces are sinks, we can still conjecture that all reachable sinks are faces. However, this not true.

[2]The Lambert W-Function is the inverse function of $f(x) = xe^x$. It is defined for $x \geq \frac{1}{e}$, and double-valued on $(1/e, 0)$. Its Taylor series is $W(x) = \sum_{n=1}^{\infty} \frac{(-n)^{n-1}}{n!}\,x^n$, and can be also expressed as $W(x) = \ln x - \ln\ln x + \sum_{k=0}^{\infty}\sum_{m=0}^{\infty} c_{km}(\ln\ln x)^{m+1}\ln x^{-k-m-1}$ for certain c_{km}.

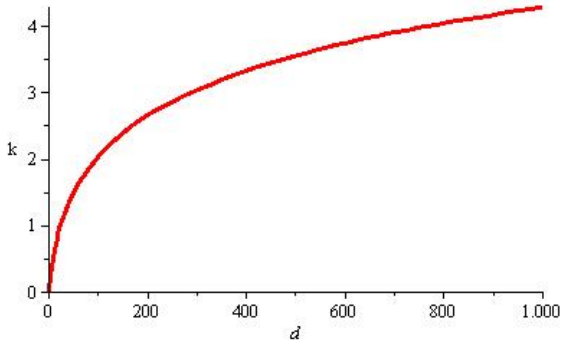


Figure 5.1: $\frac{1}{\ln 2}\, W\left(\frac{d \ln 2}{12}\right)$

EXAMPLE 5.1. *As a counterexample we study two disjoint subcubes of different size Q_k and Q_l at distance one. In this case, depending on the size of both, the union can be a sink.*

Suppose that $k \geq l$, and let Q be the union of both. For appropriate values of l depending on k, Q will be a sink. For example, the union of Q_4 with a neighbouring Q_2 is a sink of Q_{10}.

2.2. Conductance case. Now we undertake the study of the conductance on the same example. As $\mathrm{E}_{\mathrm{out}}(F) = (d-k)2^k$, the conductance of F is

$$\phi(F) = \frac{(d-k)2^k}{k2^{k-1} + (d-k)2^k} = \frac{2(d-k)}{2d-k}$$

We now prove see that k-faces are sinks of $\Gamma(\phi)$ for all k.

PROPOSITION 5.6. *All faces F of a d-cube Q_d are sink of $\Gamma(\phi)$.*

PROOF. First of all, we recall the E_{in}'s and state the $\mathrm{E}_{\mathrm{out}}$'s

$$\begin{aligned}
\mathrm{E}_{\mathrm{out}}(F^+) &= (d-k)2^k + d - 2,\\
\mathrm{E}_{\mathrm{in}}(F^+) &= k2^{k-1} + 1,\\
\mathrm{E}_{\mathrm{out}}(F^-) &= (d-k)2^k - d + 2k,\\
\mathrm{E}_{\mathrm{in}}(F^-) &= k2^{k-1} - k.
\end{aligned}$$

Now,

$$\phi(F) - \phi(F^+) = \frac{-d\,(k-2)}{(2\,d-k)\left(2^k\,(d-\frac{k}{2}) + d - 1\right)}, \tag{5.9}$$

which is negative as long as $2 \leq k \leq d$. Therefore, $\phi(F) < \phi(F^+)$. Moreover,

$$\phi(F) - \phi(F^-) = -\frac{dk}{(2\,d-k)\left(2^k\,(d-\frac{k}{2}-\frac{d}{2^k}) + k)\right)} < 0; \tag{5.10}$$

hence, having in mind that we minimize conductance, F is a sink. □

3. A random graph

We are all aware of the importance of finding communities in a well communitized graph. Nevertheless, it is also of great importance not to find them when they do not exist.

This is why we present this study case. We try out our community definition in a Erdős-Rényi random graph $G(n,p)$ [**17**]. Remember that $G(n,p)$ is a graph with n nodes, having an edge between two nodes independently with probability p. We should imagine it as a prefixed set of n nodes, where for each possible edge a coin is tossed[3]to determine the existence of an edge.

It is intuitively clear that in general, a random graph should not be communitized. We want to see how this theoretical lack of communities gets translated in our objective functions. In order to study the distribution of sinks in the graph, we will make use of the following random variables.

$$E = |\mathrm{E}_{\mathrm{in}}(S)| \text{ for a subset } S \tag{5.11}$$

$$N_u = |\mathrm{nb}_S(u)| \text{ for a subset } S \text{ and a node } u \notin S \tag{5.12}$$

for a fixed subset of nodes $S \subset V$, and a fixed external node u.

The key of our success resides in three facts. First, these random variables are independent and follow binomial distributions

$$\begin{aligned} E &\sim B\left(\frac{k(k-1)}{2}, p\right) \\ N_u &\sim B\left(k, p\right) \end{aligned}$$

where $k = |S|$. The second key is the well known approximation of a binomial by a normal distribution:

$$\begin{aligned} E &\sim N\left(\frac{k(k-1)}{2}\,p, \frac{k(k-1)}{2}\,p\,(1-p)\right) \\ N_u &\sim N\left(k\,p, k\,p(1-p)\right) \end{aligned}$$

And last but not least, the third key will be the linearity of the normal distribution. If $X \sim N(\mu_X, \sigma_X^2)$, $Y \sim N(\mu_Y, \sigma_Y^2)$ are two independent normal variables, and $a, b, c \in \mathbb{R}$, then

$$aX + bY + c \sim N(a\mu_X + b\mu_Y + c, (a\sigma_X)^2 + (b\sigma_Y)^2) \tag{5.13}$$

We will extensively use this expression later on.

We must remark that here we are assuming $0 < p < 1$, as otherwise even the approximation by normals does not work.

3.1. Laplacian.

Our main result is concentrated in the following theorem. It states that for a fixed subset S, the probability that it is a sink tends to 0 as the size of the random graph grows to infinity.

THEOREM 5.1. *Let $G \in G(n,p)$, and $S \subset V(G)$ a fixed subset of its nodes. Then $\lim_{n\to\infty} \mathbb{P}(S \text{ is a sink}) = 0$.*

PROOF. We are interested in the probability that a fixed subset $S \subset V\left(G(n,p)\right)$ is a sink. We remark that a set is a sink if and only if neither adding a node nor removing one increases the value of the directed Laplacian.

[3]A special coin that gives heads p times for each $1-p$ tails

The proof will be split into three sections, depending on the size of S. We will define two functions, $\alpha(n,p)$ and $\beta(n,p)$, and divide the proof between sets with size larger than α, sets smaller than β and sets of size between α and β. The definition of these functions will be made precise when the proof motivates it.

We will focus on the addition of a node. Observe that the removal of a node can also be seen from the point of view of the smaller set and thus can be studied as an addition.

First, we study larger sets, which means that $k \geq \alpha(n,p)$ for a certain function α that we will analyse later on. We do this by defining a new random variable

$$X_u = \mathcal{L}(S \cup u) - \mathcal{L}(S) \text{ for each node } u \notin S. \tag{5.14}$$

This variable models the addition of a single node to the fixed set. When there is any u such that $X_u \geq 0$ then S is not a sink. Therefore we can bound the probability of S being a sink by the probability that no addition of a node increases the objective value.

$$\mathbb{P}(S \text{ is a sink}) \leq \mathbb{P}\left(\forall u \notin S : X_u \leq 0\right)$$

So just proceed to study the variables X_u. If we translate (4.14) to our new universe of random variables, we obtain

$$X_u = A(k,c)\,E + B(k,c)\,N_u + C(k), \tag{5.15}$$

where

$$A(k,c) = 2c\left(\frac{k-2}{\sqrt{k(k-1)}} - \frac{k-1}{\sqrt{k(k+1)}}\right) \tag{5.16}$$

$$B(k,c) = 2c\left(1 - \frac{k-1}{\sqrt{k(k+1)}}\right) \tag{5.17}$$

$$C(k) = 1 - \sqrt{k}\big(\sqrt{k+1} - \sqrt{k-1}\,\big) \tag{5.18}$$

So we can finally use (5.13) to state that X follows a normal distribution, precisely

$$X_u \sim N\left(\mu(k,c,p), \sigma^2(k,c,p)\right) \tag{5.19}$$

where

$$\mu(k,c,p) = A(k,c)\,\frac{k(k-1)}{2}\,p + B(k,c)\,k\,p + C(k,c) \tag{5.20}$$

$$\sigma^2(k,c,p) = A(k,c)^2\frac{k(k-1)}{2}\,p\,(1-p) + B(k,c)^2\frac{k(k-1)}{2}\,p\,(1-p) \tag{5.21}$$

Now, $Z = \frac{X_u - \mu}{\sigma}$ wher $Z \sim N(0,1)$; and $X_u \leq 0$ if and only if $Z \leq \frac{-\mu}{\sigma}$.

The cumulative distribution function of the standardized normal distribution is

$$\mathbb{P}(Z \leq t) = 1/2 + 1/2\,\mathrm{erf}\left(\frac{t}{\sqrt{2}}\right),$$

where erf is the *Error function*: $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,dt$. Therefore,

$$\mathbb{P}(X_u \leq 0) = \mathbb{P}(Z \leq \frac{-\mu}{\sigma}) = 1/2 + 1/2\,\mathrm{erf}\left(\frac{-\mu}{\sqrt{2}\,\sigma}\right). \tag{5.22}$$

We will be able to study this asymptotically. But first we need to make a crucial observation. So far, we have not commented at all on the role that c is playing here. But c is also a random variable which depends on n and p. Moreover, c is not independent with respect to E nor N_u. We have to take this into account in order to analyse the probabilities correctly.

We concentrate on the study of $\frac{-\mu(k,c,p)}{\sigma(k,c,p)}$. If we set $\mu'(k,c,p) = \mu(k,c,p) - C(k) = A(k,c)\,\frac{k(k-1)}{2}\,p + B(k,c)\,k\,p$, we can observe that c is a constant factor in both μ' and σ. Therefore, the quotient $\frac{\mu'}{\sigma}$ does not depend on c! We are thus left with

$$\frac{-\mu(k,c,p)}{\sigma(k,c,p)} = \frac{-\mu'(k,1,p)}{\sigma(k,1,p)} + \frac{-C(k)}{\sigma(k,c,p)}$$

and asymptotically

$$\frac{-\mu'(k,c,p)}{\sigma(k,c,p)} \sim -\frac{p\sqrt{k}}{2\,\sqrt{(1-p)\,p}} \xrightarrow{k\to\infty} -\infty \tag{5.23}$$

and moreover, $C(k)$ and $\sigma(k,c,p)$ are essentially $C(k) \sim -\frac{1}{8}\,k^{-2}$ and $\sigma(k,c,p) \sim 3\,\frac{c\sqrt{(1-p)p}}{\sqrt{k}}$.

And now, we will need to make use of the results obtained in Section 3.1. In particular, we need Theorem 3.2 to limit the effect of c. Namely, as a consequence of this theorem we get $\lambda_n \geq -\frac{n}{2}$, and thus we can bound c by

$$c \geq \frac{2}{n}.$$

It is now the time to establish the function

$$\alpha(n) = n^{2/3+\varepsilon}, \tag{5.24}$$

because now we can use that $k \geq \alpha(n)$ to get

$$\begin{aligned}
\frac{-C(k)}{\sigma(k,c,p)} &\sim \frac{-\sqrt{k}}{24\,k^2\,c\sqrt{(1-p)\,p}} \\
&= \frac{-k^{-3/2}}{24\sqrt{(1-p)\,p}\,c} \\
&\leq \frac{-n\,k^{-3/2}}{48\sqrt{(1-p)\,p}} \\
&\leq \frac{-n\,\alpha(n)^{-3/2}}{48\sqrt{(1-p)\,p}} \\
&= \frac{-n^{-\varepsilon}}{48\sqrt{(1-p)\,p}} \xrightarrow{n\to\infty} 0,
\end{aligned}$$

and therefore we can dismiss the effect of $C(k)$. As a conclusion, asymptotically

$$\frac{-\mu(k,c,p)}{\sigma(k,c,p)} \sim \frac{-\mu'(k,1,p)}{\sigma(k,1,p)} \xrightarrow{k\to\infty} -\infty$$

Moreover, the Taylor expansion of erf at minus infinity is

$$\operatorname{erf}(t) = 1 + \left(-\frac{1}{\sqrt{\pi}t} + \frac{1}{2\sqrt{\pi}t^3} - \frac{3}{4\sqrt{\pi}t^5} + O\left(t^{-7}\right)\right)\left(e^{t^2}\right)^{-1}$$

and

$$1/2 - 1/2\,\operatorname{erf}\left(1/2\,t\sqrt{2}\right) \sim \frac{1}{t\sqrt{2\,\pi e^{t^2}}} \tag{5.25}$$

Hence, combining (5.22), (5.23) and (5.25) and using from (5.24) that $\alpha(n) \xrightarrow{n\to\infty} \infty$, we can state

$$\mathbb{P}(X_u \le 0) \sim \sqrt{2\frac{1-p}{\pi p\,k}}\,e^{-\frac{pk}{8(1-p)}} \le \sqrt{2\frac{1-p}{\pi p\,\alpha(n)}}\,e^{-\frac{p\alpha(n)}{8(1-p)}} \xrightarrow{n\to\infty} 0$$

So finally, in this case where $k \ge \alpha(n)$, we take any node not belonging to the set $u_0 \notin S$ and

$$\mathbb{P}(S \text{ is a sink}) \le \mathbb{P}\left(\forall u \notin S \quad X_u \le 0\right) \le \mathbb{P}\left(X_{u_0} \le 0\right) \xrightarrow{n\to\infty} 0$$

Which proves our claim when $k \ge \alpha(n,p)$.

Once adjusted the larger sets, we have to face the case of small sets. These will also be split into two portions. We will first concentrate on the sets in the middle, those with $\alpha(n,p) \ge k \ge \beta(n,p)$, where $\beta(n,p)$ is a function that tends to infinity but that will also be made precise later.

We must still deal with the problem that c is a random variable that we do not control, and that it is not even independent with respect to the other variables in play. To overcome this problem, we first completely forget about c, and concentrate only on the distribution of edges, that is, on the variables E and N_u, which are indeed independent, and later on we will deal with c.

From our set S, we pick a node u. Our strategy consists in finding a node $w \notin S$ such that $\mathcal{L}(S \cup w) - \mathcal{L}(S) \ge \mathcal{L}(S) - \mathcal{L}(S \setminus u)$. This will give us a win-win solution: either $\mathcal{L}(S \cup w) - \mathcal{L}(S) \ge 0$ and S is not a sink, or $\mathcal{L}(S) - \mathcal{L}(S \setminus u) \le \mathcal{L}(S \cup w) - \mathcal{L}(S) < 0$ and S is not a sink either! Remark that we might be able to prove this without having to worry about the value of c, because c will determine in which of the two situations above we are, but we finish the proof in both situations because both imply that S is not a sink.

Let's see it. Now, the variables E and N_u will refer to $S \setminus u$, and N_w to S. Moreover, instead of concentrating on the changes of $\mathcal{L}$, we will concentrate on the changes of the c-homogeneous part[4], and later on we will see that this is enough. That is, we will concentrate in the cases when

$$A(k+1,1)\,(E + N_u) + B(k+1,1)\,N_w \ge A(k,1)\,E + B(k,1)\,N_u,$$

which translates to

$$B(k+1,1)\,N_w \ge (A(k,1) - A(k+1,1))\,E + (B(k,1) - A(k+1,1))\,N_u =: Y.$$

[4]Observe that $X_u = c\,(A(k-1,1)\,E + B(k-1,1)\,N_u) + C(k)$

But now, it follows again from (5.13) that Y follows a normal distribution. Specifically, using the notation $A(k) = A(k,1)$, etc. the distribution of Y is

$$Y \sim N(\mu, \sigma^2)$$

with

$$\mu = (A(k) - A(k+1))\,\frac{p(k-1)(k-2)}{2} + (B(k) - A(k+1))\,p(k-1),$$
$$\sigma^2 = p(1-p)(k-1)\left((A(k) - A(k+1))^2\frac{k-2}{2} + (B(k) - A(k+1))^2\right).$$

However, now we have an advantage, because $\lim_{k\to\infty}\sigma^2 = 0$.[5] Now, even if we do not specify β yet, we impose that $\beta(n,p) \xrightarrow{n\to\infty} \infty$. Thus, together with $k \geq \beta(n,p)$, this implies that the value of Y concentrates around its mean! In particular, $\mathbb{P}(Y \in \mu \pm \varepsilon) \to 1$. From now on we will work with the probability conditioned to this event.

Now if we name Z_w the event

$$\begin{aligned} B(k+1,1)\,N_w \geq &(A(k) - A(k+1))\,\frac{p(k-1)(k-2)}{2} \\ &+ (B(k) - A(k+1))\,p(k-1) + \varepsilon \end{aligned}$$

we are interested in the probability $\mathbb{P}(Z_w)$.

But we have already what we need, because for large enough k's the expression $\frac{(A(k)-A(k+1))\,\frac{p(k-1)(k-2)}{2} + (B(k)-A(k+1))\,p(k-1))}{B(k+1,1)}$ is smaller than the mean of N_w, which is $(k+1)p$. To see this, it is enough to use the Taylor expansions

$$\begin{aligned} (A(k) - A(k+1))\,\frac{p(k-1)(k-2)}{2} + (B(k) - A(k+1))\,p(k-1)) = \\ p\left(3 - \frac{19}{4k} + \frac{63}{8k^2} - \frac{1067}{64k^3} + \mathcal{O}(\frac{1}{k^4})\right) \end{aligned}$$

$$B(k+1,1) = \frac{1}{k}\left(3 - \frac{19}{4k} + \frac{63}{8k^2} - \frac{867}{64k^3} + \mathcal{O}(\frac{1}{k^4})\right)$$

Just setting ε small enough we have that

$$P(Z_w) \geq \frac{1}{2}$$

for each $w \notin S$.

[5] This is because $A(k) - A(k+1) = \mathcal{O}(\frac{1}{k^3})$ and $B(k) - A(k+1) = \mathcal{O}(\frac{1}{k})$

So, using the crucial property that all events Z_w are pairwise independent and recalling that $|S| \leq \alpha(n,p)$, we obtain

$$\begin{aligned}
\mathbb{P}(\exists w \notin S \text{ such that } Z_w \text{ is true}) &= 1 - \mathbb{P}(\forall w \notin S \ Z_w \text{ is false}) \\
&= 1 - \mathbb{P}(w \notin S \ Z_w \text{ is false})^{|V \setminus S|} \\
&\geq 1 - \mathbb{P}(\forall w \notin S \ Z_w \text{ is false})^{n-\alpha(n,p)} \\
&\geq 1 - (1 - \mathbb{P}(Z_w))^{n-\alpha(n,p)} \\
&\geq 1 - \left(\frac{1}{2}\right)^{n-\alpha(n,p)} \xrightarrow{n\to\infty} 1
\end{aligned}$$

And now the proof goes by itself, because just observing that $C(k+1) - C(k) \geq 0$ along with these previous results shows that $\mathcal{L}(S \cup w) - \mathcal{L}(S) \geq \mathcal{L}(S) - \mathcal{L}(S \setminus u)$. The only knowledge about c we used is that $c \geq 0$. And this is what we wanted to prove.

Finally, we have to deal with really small sets, that is $|S| \leq \beta(n,p)$.

For now just assume there is a function $\gamma(n,p)$ such that for all graphs H with at most $2\gamma(n,p)$ nodes, we have with probability 1 that H is an induced subgraph of G.

In particular, we can impose that the complete bipartite graph $H = K_{\gamma,\gamma}$ be a subgraph of G, and we will almost surely be right (with probability 1). Then again from Theorem 3.2 and Observation 3.10, we have that $\lambda_n(G) \leq -\gamma(n)$ and

$$c \leq \frac{1}{\gamma(n,p)}. \tag{5.26}$$

We will prove that when k is small enough, then there is not any node the addition of which can increase the Laplacian, and therefore the only sink with less than k elements is the empty set.

To see this, we focus on (5.15). We observe that it is monotone in E and N, as it is a linear function in each of these variables. An easy calculation from (5.16) and (5.17) allows us to see that $A(k,c) < 0$ and $B(k,c) > 0$ for $k \geq 2$. Thus, X_u is decreasing in E and increasing in N and it attains its maximum when E is minimal and N is maximal, meaning $E = 0$ and $N = k$. We then get

$$X_u \leq 2\,ck\left(1 - \frac{k-1}{\sqrt{(k+1)\,k}}\right) + \sqrt{k\,(k-1)} + 1 - \sqrt{(k+1)\,k} = c\,D(k) + E(k).$$

This is a linear equation in c, which evaluates to zero precisely when $c = -\frac{E(k)}{D(k)}$, and is smaller than zero when $c < -\frac{E(k)}{D(k)}$. And now we have what we need, because asymptotically

$$c = -\frac{E(k)}{D(k)} \sim \frac{1}{24\,k^2}.$$

Now we set $\beta(n,p) = \sqrt{\frac{\gamma(n,p)}{24}}$ and according to (5.26), if $k < \sqrt{\frac{\gamma(n,p)}{24}} = \beta(n,p)$ this condition holds and our claim is true. Notice that as $\gamma(n,p) \xrightarrow{n\to\infty} \infty$, so does β. And this is what we had previously required.

Before advancing to the next step, we must still give an observation: in this ending of proof we have *not* used the Normal approximation of the Binomial, which does not hold for very small parameters.

We still need to prove the existence of such a γ. What we will proceed to do is to bound the probability that there is no copy of $K_{\gamma,\gamma}$ in G, and prove that if γ is small enough, then this probability tends to 0 when n tends to infinity. With that intention, we first partition G in pieces of size 2γ, and try to fit $K_{\gamma,\gamma}$ in each of these pieces fixing where each half of the bipartite graph shall go. The probability that none fits in the way we described (event that we will call M) is the probability that it does not fit in each piece. If we name N the event that it fits in a piece, we have

$$\mathbb{P}(M) = (1 - \mathbb{P}(N))^{\frac{n}{2\gamma}}$$

But now,

$$\mathbb{P}(N) = p^{\gamma^2} (1-p)^{\gamma^2-\gamma} \geq q^{2\gamma^2-\gamma}$$

where $q = \min\{p, (1-p)\}$. And then

$$\mathbb{P}(M) \geq (1 - q^{2\gamma^2-\gamma})^{\frac{n}{2\gamma}} \sim e^{-q^{2\gamma^2-\gamma}\frac{n}{2\gamma}}$$

To have $\mathbb{P}(M) \longrightarrow 0$, we need $q^{2\gamma^2-\gamma}\frac{n}{2\gamma} \longrightarrow \infty$, but we can manage to get this, because if $r = \frac{1}{q} \geq 0$, then setting $\gamma = \sqrt{\frac{\log_r(n^{1-\varepsilon})}{2}}$, we have

$$q^{2\gamma^2-\gamma}\frac{n}{2\gamma} = \frac{n}{2\gamma\, r^{2\gamma^2-\gamma}} \sim \frac{n}{2\sqrt{\frac{\log_r(n^{1-\varepsilon})}{2}}\, n^{1-\varepsilon}} \xrightarrow{n\to\infty} \infty$$

just as we wanted to prove. And observe that in the step of bounding p and $1-p$ with q, this proof holds for any graph H not necessarily the complete bipartite graph.

□

And yet one observation on the consequences of this theorem. Here we proved that the probability of a set being a sink tends to 0. However, this does not mean at all that there are no sinks. The speed of decreasing towards zero is important. What we have seen so far is that the expected number of sinks in a random graph is smaller than exponential in order of magnitude, which is far from being 0. To have an expected number of sinks closer to 0, we would need to prove that the probability decreases faster than exponential. Even if this was true, it is difficultly proved without further knowledge on the smallest eigenvalue.

4. Qualitative differences between $\mathcal{L}$ and ϕ

By observing the differences between the two definitions we can deduce some properties of the communities selected by the different objective functions. The Laplacian $\mathcal{L}$ ignores the connection of the community with the rest of the graph, and only looks at its internal strength and size. Thus, it regards a subset of nodes with many internal connections as a community even if it has relations with the rest of the graph. In contrast, the conductance ϕ involves $\mathrm{E}_{\mathrm{out}}$, leading to a different concept of community that also takes into account the relation with the rest of G. However, this conflicts with the goal of overlappedness: A well-connected node that *should* belong

to a lot of communities will not appear in any of them, because each one will have many edges towards the exterior, and therefore will not be considered a community.

On the other hand, as have seen in Observation 5.1, the "hairs" or less-connected nodes are well classified by ϕ, but are not taken in any community with $\mathcal{L}$. Thus, we do not expect an OCA_g scan with the Laplacian to cover all nodes of G, but just those that are relevant to the community structure (in contrast to the conductance that typically *will* cover all nodes). However, this is not a handicap, as it agrees with the philosophy that not every node necessarily belongs to some community. In Chapter 6 we propose a postprocessing scheme in case we do need to cover the whole graph.

CHAPTER 6

Community detection: the algorithm

Once we have a definite fitness function (indeed, we have two), we can start the process of finding the communities of a graph. In this chapter we will present an algorithmic framework that handles the community detection in a graph.

Our community detection algorithm is based in a very simple method: it consists in constructing communities around some seed subsets by looking for sinks of the fitness function. This procedure is repeated over and over for different seeds until a halting criterion is reached. The communities encountered are collected and returned as the community structure of the graph.

Before developing this basic idea into an algorithm, we will reformulate our definition of community. From their introduction at Chapter 3 until now we have presented a relation of equivalence between the definitions of sink and community. Stating that a community was a sink of certain objective function. However, when we approach the real algorithm this will not be completely accurate. Concretely, we will impose that communities are sinks, but not all sinks will be declared communities.

In the examples in Chapter 5, we observed graphs where with both the conductance and the directed Laplacian there were many communties, too many communities. We can not afford to search and store so many communities, and neither we need to.

Not all communities will be of the same importance. In fact, we will be have to return a small but yet significant part of the possible sinks. This yields to our operational definition of a community: a community will be a set returned by our algorithm.

Even if this might seem inaccurate, it is indeed a reasonable method. The cause is the following. Given a sink, it has an influence area of those nodes whose emerging walks lead to the sink. We expect those sinks with larger influence area to be more relevant communities, so if the algorithm allows for it, the larger this area is ther is greater probability of finding the sink. Therefore, we will expect to return the more relevant communities.

Once clarified this point, we are now in condition of presenting the *Overlapping Community Algorithm* OCA.

It is designed to be used in two kind of queries. It can be used globally, in order to obtain all the communities in a graph (OCA_g), and locally, to explore the communities of a specific node (OCA_l). In both cases, the underlying method used to explode and determine a community from a seed is the same and it is called LOCA ("Local OCA"). In general, we will focus on the global version, OCA_g, since OCA_l is based on the same ideas.

OCA_g searches for the community structure of the whole graph. It repeatedly chooses some *seed communities* randomly distributed along the whole graph, and grows to a locally optimum community using LOCA. The collection of communities found will become the community structure of the graph.

On the other hand, OCA_l is a variation of the same concept where all the seeds are chosen in the neighborhoods of a specific node.

Notice that even if the procedure is essentially the same, the underlying query is not. While with OCA_g our final objective is to obtain a picture of the topology of the whole graph, with OCA_l we seek much preciser results to obtain the exact local structure.

1. LOCA

The core of OCA is the local search subroutine LOCA, which eventually is the one that will determine subsets that are communities (guided by the objective function ϕ).

It maps each initial seed set S_0 to a community in the neighborhood[1]. This community will correspond to a sink in $\Gamma(\phi)$ i.e. a local maximum of ϕ. A sketch of the algorithm is presented in Algorithm 2.

Algorithm 2 LOCA: local search (sketch)

```
1: procedure LOCA(S_0, φ, P)
2:     S_0: the seed community
3:     φ: the objective function
4:     P: the pivot rule that selects an improving node
5:     Set S := S_0
6:     Set L := ∅
7:     while some neighbor u of S is not yet visited do
8:         if φ(S ∪ u) > φ(S), put u into L
9:     end while
10:    while some element u of S \ S_0 is not yet visited do
11:        if φ(S \ u) > φ(S), put u into L
12:    end while
13:    if L = ∅ then return S; end if
14:    Use P to select u_0 ∈ L; set S := S ∪ {u_0}; goto 6
15: end procedure
```

LOCA starts from an initial seed subset $S := S_0$ and gradually modifies it until it becomes a local sink. For that, it looks at all the nodes at S and tries to remove them, as well as looks at all the nodes connected to S, which it tries to add to S. This provides a set of values of the fitness function $\big\{\phi(S \setminus v) : v \in S\big\} \cup \big\{\phi(S \cup v) : v \text{ connected to } S\big\}$. If none of these improves $\phi(S)$, LOCA stops and declares S to be a *community* in V. Otherwise, it uses a pivot rule P to choose the next subset S. Details and further discussion on the pivot rule are given later on in Section 2.

[1]The definition of neighborhod is somehow fuzzy. It becames in fact an operational definition. Thus, by neighborhood is meant the sets reachable travelling in the direction of the edges in $\Gamma(\phi)$

2. OCA: Global community search

OCA_g uses repeatedly LOCA to explore the entire graph G and return its community structure. Its behaviour is presented at Algorithm 3.

Algorithm 3 OCA_g: Global exploration

```
1: procedure OCA(G, φ, P, N, H, I)
2:     G = (V, E): an undirected graph
3:     φ: the objective function
4:     P: the pivot rule that selects an improving node
5:     N: the rule that selects a neighborhood of S_0
6:     H: The halting criterion
7:     I: The rule that selects initial subsets of G
8:     Initialize C = ∅
9:     while H is not fulfilled do
10:        Use I to select an initial community S_0 ⊂ V
11:        Use N to select a neighborhood S of S_0
12:        Add LOCA(S, φ, P) to cC
13:    end while
14:    return C
15: end procedure
```

We will do a brief presentation of the algorithm.

First, it uses some rules I and N to select a initial subset S. I, the *Initial subset rule*, determines a seed node that must be expanded to a full community. N the *Neighborhood selection* is used to give an initial push to the community by selecting some nodes from the neighborhood and adding them to the initial seed (without even consulting the fitness function). This step not only saves some computation time (it skips having to run LOCA to add these neighbors, which have great chances to be added anyway), but will be a crucial point in order to obtain good results with OCA_l.

Once selected the initial set S, it applies LOCA on S to obtain a community. This procedure is repeated until a halting criterion H is reached. Then the collection of all communities that LOCA has found are returned.

Thus, the criteria that parametrize LOCA and OCA_g determine the search strategy used in the quest for local maxima. In the following, we describe the most important aspects of the algorithm in more detail:

Pivot Rule P. Given a subset $S \subseteq V$ and a list L of nodes whose addition or removal will improve the objective value $\phi(S)$, LOCA greedily takes the node that maximizes increase, using a steepest increment strategy.

This "hill climbing" approach appears to be the most appropriate one in both terms of efficiency and quality of the resulting communities.

Experimentally, this strategy has yielded the best results. Although this could seem counter-intuitive, it is explained by the fact that we are not exclusively interested in global optima, but local sinks, therefore a greedy approach fits perfectly in this case. Other more conservative algorithms such as iterative-improving or simulated annealing[2] skip these local maxima with

[2]Details of other optimization algorithms can be found at [**43**]

higher probability (looking for global ones), and spend more time walking to find a sink, which worsens the overall execution time.

Initial subset rule I. In the selection of the initial subset there lies the main difference between OCA_g and OCA_l.

In OCA_g, we pick a random node that has not yet been added to any community, and then expand it using N. Thus, we discard not only those nodes that have been chosen as initial seeds, but all those that have ever been included in a community.

On the other hand, with OCA_l the selected node is always the one whose neighborhood we need to explore.

Neighborhood selection N. The purpose of LOCA is to find a community containing a given (small) seed subset S_0. For this, we select some larger subset $S \supset S_0$ and start the local search from there.

We explored two different selection rules. One of them takes the set of all nodes at graph distance at most k from S_0, while the other constructs a random subset of those. The value of k is usually chosen around 2. We did not implement more sophisticated criteria based on random walks as in [**46**], as our simple-to-implement criteria turned out to be entirely sufficient.

In OCA_g, both methods produced equivalent results. While choosing random neighbours yields better results when performing OCA_l. In fact, this random selection of the neighborhood is key in OCA_l.

According to I, the initial seed is always the same, and thus if there is no randomness added, the result would always be the same! In addition without this criterion, there may exist small communities that could be missed in favour of those larger and stronger. But this way we leave open the possibility that in some iteration, the selected nodes belong mainly to this small community, and thus the community is detected.

With OCA_g, the problem is not so highlighted because the different initial seeds already allow for detecting most of the communities.

REMARK 6.1. *With the directed Laplacian, an extra constraint is added to this neighborhood selection. Observing the ascending condition at Lemma 4.5, we obtain that edges at Γ between sets smaller than a critical size (that depends on the value of c) all point downwards, and therefore we would not be able to recover any community different from singletons. Hence, we must impose an initial minimum size.*

To determine it, we focus in the clique ascension, that is, when a subclique grows to a clique. This happens as long as $c \geq \frac{1}{12\,|S|^2}$, and thus we impose that with the neighborhood rule we select communities of size $|S| \geq \frac{1}{\sqrt{12\,c}}$.

Halting criterion H: Recall from Remark 5.1 that we do not expect to cover all the nodes in a community. Therefore we need to establish a different criterion to stop searching for communities.

The evolution of the number of nodes covered is similar to a logarithmic function: at the beginning it increases very fast, but soon the velocity slows down to finally converge to the covering threshold. This encourages us to stop when the decreasing slope is smaller than a tolerance τ. The election

of τ marks a compromise between the degree of covering and efficiency, and even though variations of τ might have a slight effect on the quality of the solution, "structurally" speaking the most important communities will be those found during the first few iterations of the algorithm. Therefore, the importance of τ is diminished.

3. Implementation details

In this section, we present some further details of the implementation and the study on the complexity of the algorithm.

3.1. The smallest eigenvalue. When working with the directed Laplacian, one of the crucial elements is the constant c that determines the scalar product between adjacent vertices. Then, to use the objective function, first of all we need to find it!

Since calculating all eigenvalues of a large graph is out of the question, we must approximate or bound the smallest eigenvalue $\lambda_{\min} < 0$. The well-known *power method* can be used to efficiently calculate $\lambda_{\max}$, the largest absolute value of any eigenvalue. However, this is usually not the answer we are after, since the theorem of Perron–Frobenius [**20**] tells us that $|\lambda_{\min}| \leq \lambda_{\max}$, with equality iff G is bipartite. Nevertheless, we can adapt the power method to find the smallest eigenvalue.

THEOREM 6.1. *The most negative eigenvalue λ_{min} of any symmetric matrix A can be calculated from any upper bound $\lambda_{\max} \leq \kappa$ for the largest eigenvalue.*

PROOF. Set $B = A - \kappa I_n$. Its eigenvalues are $\mu_i = \lambda_i - \kappa$, where λ_i are the eigenvalues of A. Since $\mu_i \leq 0$ for all i, the eigenvalue of B of largest absolute value is $\mu_{\min} < 0$, which can be calculated using the power method. To finish the proof, note that $|\lambda_{\min}| = |\mu_{\min}| - \kappa$. □

As a consequence of Theorem 6.1, we just need an appropriate upper bound of $\lambda_{\max}$ to be able to efficiently retrieve the most negative eigenvalue with the power method. We can soften (3.12) and use the upper bound $\lambda_{\max} \leq \Delta$, where Δ is the maximum degree of a node in G.

3.2. Preprocessing. In some situations, performing a preprocess of the graph before running the OCA avoids performing redundant operations. In particular, Remark 5.1 tells us that when we use $\mathcal{L}$ the nodes of degree one will probably not be part of any community. In a scale-free network, which represents most of the large graphs available, most of the nodes have very small degree, which means that with high probability, G has many nodes of degree one. To avoid exploring these irrelevant nodes that will not be taken by any community, we perform a previous scan on the graph removing all "hairs" and isolated nodes.

3.3. Implementing the local search. The local search algorithm LOCA repeatedly examines a current community S and its neighbors in G to find the node whose inclusion or exclusion maximizes the increase in quality. To set up notation, let $\mathrm{nb}(S) = \left(\bigcup_{u \in S} \mathrm{nb}_V(u)\right) \setminus S$ be the set of neighbors of S.

Algorithm 2 is of course only a sketch. In the real implementation, LOCA maintains a priority queue $\mathcal{Q}$ with all *priorities* $\Pi_+(u) = \phi(S \cup u) - \phi(S)$ for $u \in \mathrm{nb}(S)$, respectively $\Pi_-(u) = \phi(S \setminus u) - \phi(S)$ for $u \in S$. In each iteration, the node yielding the greatest improvement is found using $\mathcal{Q}$, and the data structure is updated.

THEOREM 6.2. *In any graph with n nodes and maximum degree Δ, the local search* LOCA *with the squared-length Laplacian $\mathcal{L}$ can be implemented in $O\big(\Delta s(\log s + \log n)\big)$ steps, where s is the size of the community found. For the conductance ϕ, this run-time rises to $O\big(\Delta s(s + \log n)\big)$.*

PROOF. We examine the updates required to add a node u to S; removing one is quite similar. Formulas (4.14) and (4.16) say that for both choices of objective function, LOCA needs to update the values of *all* $u' \in \Sigma := S \cup \mathrm{nb}(S)$, because all priorities depend on data such as $\mathrm{E_{in}}$ and s that change with the properties and size of S. This yields an algorithm whose running time is quadratic in the size of the output set. By Equation (4.16), this is the best possible for the conductance ϕ.

In the case of the Laplacian, we improve the efficiency of LOCA as follows. Write $f_+\big(\mathrm{nb}_S(u), s, \mathrm{E_{in}}(S)\big)$ for the right-hand side of $\Pi_+(u) = \phi(S \cup u) - \phi(S)$ in Formula (4.14), and distinguish the possible cases for $u' \in \Sigma \cup \mathrm{nb}_V(u)$:

(a) If $u' \in \mathrm{nb}_\Sigma(u)$, we replace $\Pi(u')$ by

$$f_+\big(\mathrm{nb}_S(u') + 1, s + 1, \mathrm{E_{in}}(S) + \mathrm{nb}_S(u)\big). \tag{6.1}$$

This happens at most $\Delta = \mathrm{maxdeg}(G)$ times.

(b) If $u' \in \Sigma \setminus \mathrm{nb}_\Sigma(u)$, we replace $\Pi(u')$ by

$$f_+\big(\mathrm{nb}_S(u'), s + 1, \mathrm{E_{in}}(S) + \mathrm{nb}_S(u)\big). \tag{6.2}$$

We cannot bound the number of times this happens, because we have no control over the size of Σ.

(c) If $u' \in \mathrm{nb}_V(u) \setminus \mathrm{nb}_\Sigma(u)$, LOCA needs to query $\mathrm{nb}_V(u')$.

By expanding into powers of $\frac{1}{s}$, the difference δ_2 between the value $\Pi_+(u') = f_+\big(\mathrm{nb}_S(u'), s, \mathrm{E_{in}}(S)\big)$ stored in $\mathcal{Q}$ and the updated value (6.2) turns out to be an order of magnitude smaller than the analogous difference δ_1 with (6.1). In particular, $\frac{\delta_2}{\delta_1} < \frac{1}{100}$ for $|S| > 150$. This has the very practical consequence that after an initial stage of constant duration C, we may ignore the updates in case (b) entirely. (Again, for lack of space we omit the details of this calculation, and of the analogous one for $\phi(S \setminus u) - \phi(S)$. Moreover, we assume that the total number of such "removal steps" is small compared to the number of "addition" steps.)

After this initial stage, step (a) for a community of size j requires at most Δ updates of $\mathcal{Q}$, of cost $O(\log j)$ each.

Step (c), finally, is executed at most Δs times during the entire algorithm. The cost of execution of the query for $\mathrm{nb}_V(u')$ will depend on the data structures we can use. If the graph does not fit into the core memory we can handle the graph with the structures provided by DEX [**32**] in order to manage our large data graph, making it possible to execute the query for $\mathrm{nb}_V(u')$ in $O(\log n)$ time. If the graph fits into memory, then we can use

larger data structures that allow a faster execution of this query. In this case, we could perform this execution in constant time.

If we stick to the general case, the total cost of the algorithm is thus

$$C' + \Delta s\, O(\log n) + \sum_{j=C+1}^{s} \Delta O(\log j) \;=\; O\big(\Delta s(\log n + \log s)\big).$$

This concludes the proof. □

3.4. Implementing the global search. We have presented I, N and H, the criteria that guide the search of OCA_g. We will proceed now to detail how do they materialize in the final implementation.

To select the initial node, we keep a list Λ with all the nodes shuffled, and at each iteration we *pop* its first element. However the final implementation does not need to materialize Λ. We have all the nodes indexed by consecutive natural numbers and use a large prime number p. Then, starting at an intial randomly chosen seed node a_0, the k-th node we *pop* corresponds to $a_k = (a_0 + p\,k) \mod (n)$, where n is the number of nodes.

We declare a node *visited* if it has been added to any community during a previous iteration (even if it has been eventually dropped). If the node we pop from Λ has been visited, we discard it and look for the next one.

To select the neighborhood of the initial node a_i, first we need to compute the minimal size M as stated in Remark 6.1, then for each neighbor of the initial node, we decide to add it to the set with a probability equal to $\min\left(1, \frac{2M}{\deg(a_i)}\right)$. Hence, the initial communities are either of size around $2M+1$ or contain all the neighbors of the seed. We do not add nodes at distance greater than one.

Finally we halt the execution whenever we reach any of these situations:

(i) The whole graph is covered.
(ii) $a_i = a_0$
(iii) During the last $\bar{\tau}$ iterations no new node was visited.

Item (iii) is the way how we materialized the halting criterium, and where in the original notation of Section 2 we should state $\bar{\tau} \sim \frac{1}{\tau}$. So far, setting $\bar{\tau} = 50$ has shown to be more than enough to recover the full community structure of all the graphs we have tested.

3.5. Postprocessing. Two different postprocessing issues arise from the results:

Noise fusion: At times, our community search is "too precise", in the sense that it returns communities differing in very few nodes. Even though there may exist some structural difference between them, we will often not be interested in such details but in a more global picture of the graph.

We define the *similarity* between communities C and D as

$$\rho(C, D) \;=\; 1 - \frac{|C \setminus D| + |D \setminus C|}{|C \cup D|}. \tag{6.3}$$

This number lies between 0 and 1, is maximal for equal communities, and minimal when they are disjoint.

If OCA returns two communities with a similarity over a certain threshold, we merge them in a sole new community. This way we avoid having pairs of communities that differ only in some small noise.

Community reassignation: Sometimes we want to fully cover G by assigning each node to at least one community. For this, we assign each "orphan" (uncovered node) to the community to which most of its neighbors belong.

CHAPTER 7

The results

In this chapter, we present the results of several experiments run to test two different aspects of our proposal:

(i) quality of results using OCA_g and
(ii) local use of OCA_l.

The structure of the chapter is as follows: first we present the datasets we tested OCA on. And then, we will give the results classified by each of the properties we wanted to analyse.

1. The gallery of graphs tested

We present here the various datasets we have used to test the performance of OCA. Table 1 summarizes them. We have tested OCA on four datasets.

Name	# nodes	# edges
Erdős-Rényi	10^5	$\sim 10^5$–10^7
Word Association	7 211	31 798
LFR	10^4–10^6	$\sim 10^5$–10^7
Daisy	10^5	$\sim 4 \cdot 10^5$

Table 1: Data sets analyzed by OCA

Erdős-Rényi random graphs: Finding communities when they appear is as important as certifying their non-existence. A good example of a family of graphs in which communities probably do not exist are random graphs. We already made a theoretical analysis in Chapter 5, and now we will extend this analysis with the computational results. We use the Erdős-Rényi model [**17**] to generate a random graph $G(n, p)$ with n nodes and an edge between two with probability p.

Word Association Network: This data set based on [**33**] has been used for community detection in [**39, 44**]. It is a directed weighted graph that links two words if they are meaningfully related or strongly associated. The weight of each edge reflects the degree of relationship between the linked words. We use this data set to study specific cases, since it permits us to intuitively determine the quality of the communities found by OCA. For our experiments, we turn it into an undirected graph on the original set of vertices by connecting two of these if the sum of the weights of the two

directed edges exceeds a given threshold ω^*. Using the same ω^* as in [**39**] and ignoring isolated nodes yields a network of 7 211 nodes and 31 798 edges. Removing all nodes of degree 1 yields a graph with 5 353 nodes and 29 940 edges.

LFR (Communitized benchmark graphs): Lacichinetti, Fortunato and Radicchi propose in [**29**] a benchmark for community detection (from now on LFR). To our knowledge, it is the only benchmark to date that allows to create graphs of different sizes containing non-overlapping communities that satisfy certain constraints given by parameters. Since we know a priory in which community each node should be contained, this dataset yields a quantitative way to determine the quality of the communities found by OCA. The following are the values of the parameters we consider by default: size equals to 10^5, average degree equals to 20, maximum degree equals to 90, communities size is in $(25, 150)$, exponent of the power law for degree distribution equals to $\gamma = 2.5$, exponent of the power law for community size distribution equals to $\beta = 1.5$, *mixing parameter* $\mu = 0.2$. This parameter μ gives the proportion of edges that a node shares with other nodes that do not belong to its community.

Daisy (Overlapped Test Graph): To our knowledge, a benchmark for testing overlapped community search algorithms has not been proposed previously. In order to test the ability of our different fitness functions to detect intersecting communities, we propose a very simple family of synthetically generated graphs with consecutively labelled nodes. We create an edge with probability p_1 between two nodes u and v if $f(u)$ and $f(v)$ hold, where $f(x)$ is the condition $(x \equiv 0 \mod p) \vee (x \equiv 0 \mod q)$, for coprime p, q. We also create an edge with probability p_2 between any two nodes u and v such that $u \equiv v \mod p$ and $u, v \not\equiv 0 \mod q$. The resulting graph consists of p communities (a central community, composed of multiples of p or q, and $p - 1$ communities overlapped with this one). The parameter q affects the degree of overlap between each community and the central one. Because of its shape, we will call it a *daisy graph*. Our overlapped graph is a *daisy tree*, which consists in a collection of daisy graphs where daisies are joined by selecting one petal from each, and creating edges between nodes in these petals them with probability p_2.

The experiments were performed using a computer with a single processor at 2.4 GHz and 2 GB of RAM. The operating system is Linux Debian etch 4.0 (kernel version 2.6.18).

The algorithms of the results presented here were writen in C++ with special data structures created ad hoc for the problem.

2. Quality Analysis

In this first part of the experiments, we analyze the quality of the results yielded by OCA. We focus on three different aspects, namely, the coverage of our algorithm, the degree of overlapping detected in each graph, and the

accuracy of the community search algorithm. Finally, we show that the algorithm does not find communities in a randomly generated graph, as expected.

Coverage. For each data set, Figure 7.1 plots the percentage of the graph that is covered (until the halting criterion is reached) against the number of executions of LOCA during a full run of OCA_g using $\mathcal{L}$ for the Words dataset. We found 1 988 communities, covering 82% of the graph. Modifying τ will not have a significant impact on the quality of the results in terms of coverage, since around 70% of the graph is covered during the first 1000 iterations.

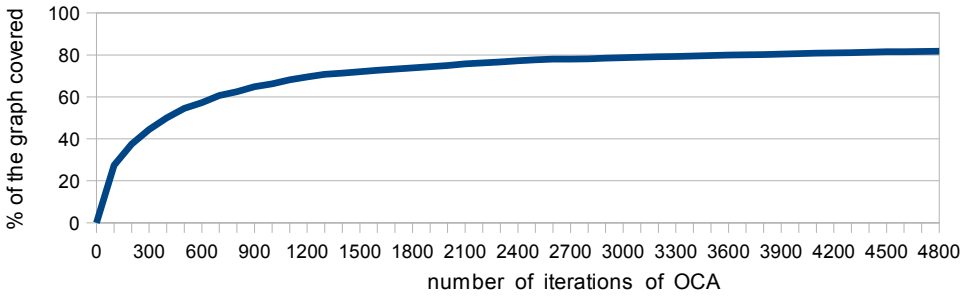


Figure 7.1: Evolution of the percentage of the graph covered during an execution of OCA_g.

At this point, we would like to reiterate that it is part of the philosophy of our approach, when using the Laplacian fitness function, to assume that some nodes might not be part of any community. Therefore, depending on the graph, a low graph coverage does not imply low quality.

Figure 7.2 shows an example of the degree distribution of the word graph and the degree distribution of the nodes not included in any community, using $\mathcal{L}$ as the fitness function, after up to 5000 iterations. The experiments show that the nodes not included are those with the lowest degrees. This makes sense, since these are the nodes with lower connectivity and, therefore, the less expected to be in the core of any community. We must also stress that, while it is true that the leftover nodes have small degree, it is not only the degree that determines if a node is covered or not, but it depends on how important the node is for the community structure. Also, we can see that after the first 1000 iterations, more than half of the nodes are included in a community. Experiments using other data sets show the same trends.

Using ϕ instead of $\mathcal{L}$, the graph is covered completely.

As we have said, the coverage of OCA_g using $\mathcal{L}$ depends on the degree distribution, and for LFR benchmarks therefore on γ, the exponent of the power law. Figure 7.3 shows the coverage for $\gamma \in (2.0, 3.0)$ on a graph containing 10^5 nodes.

Low values of γ result in a more uniform distribution and OCA_g only classifies the nodes with high degree. On the other hand, high values give a better coverage, because low-degree nodes take importance in the graph.

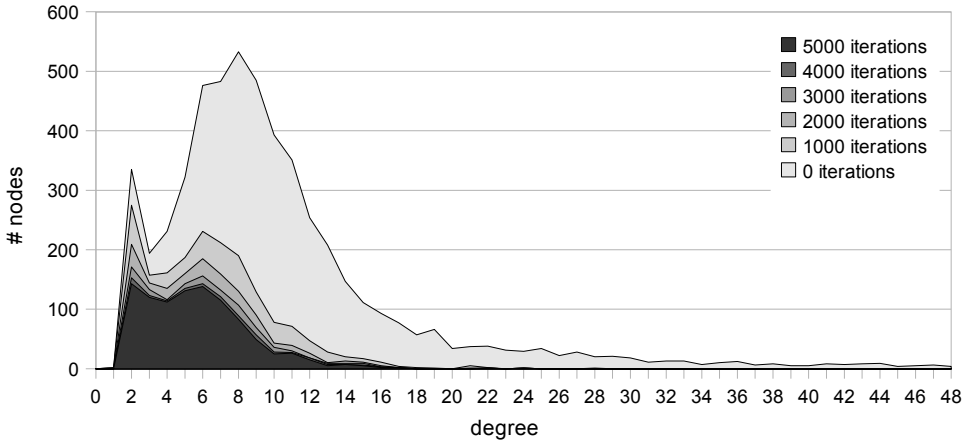


Figure 7.2: Degree distribution of all nodes and nodes not covered

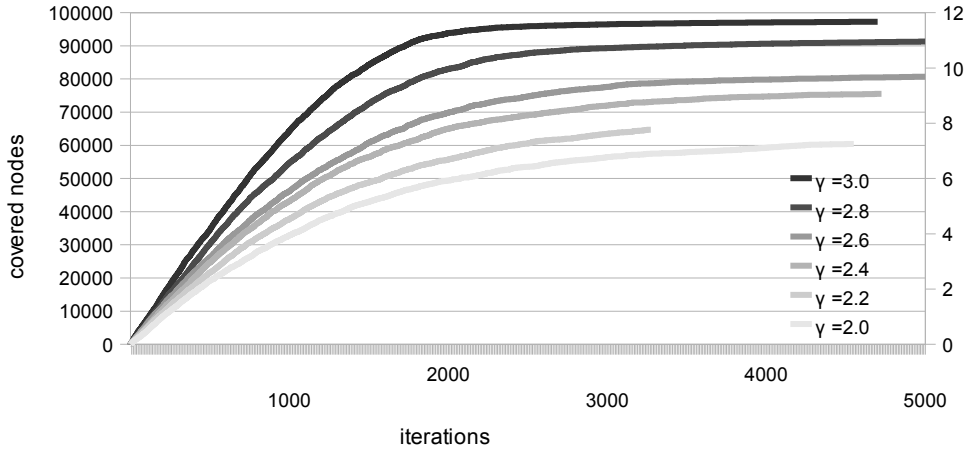


Figure 7.3: Coverage of LFR graphs for various exponents. The graphs and γ-values are both ordered top-to-bottom.

Degree of overlap. Figure 7.4 shows to how many different communities the nodes of the Word graph belong. We observe that while before the post-processing a node is contained in 7.49 communities on average, after the fusion this number drops to 3.43, which is almost the same as the 3.38 achieved by the conductance. However, we can see that the behaviour is not the same, as in the conductance almost all communities are contained in 3, 4 or 5 communities, while with the Laplacian the distribution is much more disperse. To be more precise, the standard deviation with $\mathcal{L}$ is 11.95, and after the fusion is 4.11, while with ϕ we obtain a standard deviation of only 1.92.

We do not run this experiment using the LFR data set, since communities are not overlapped.

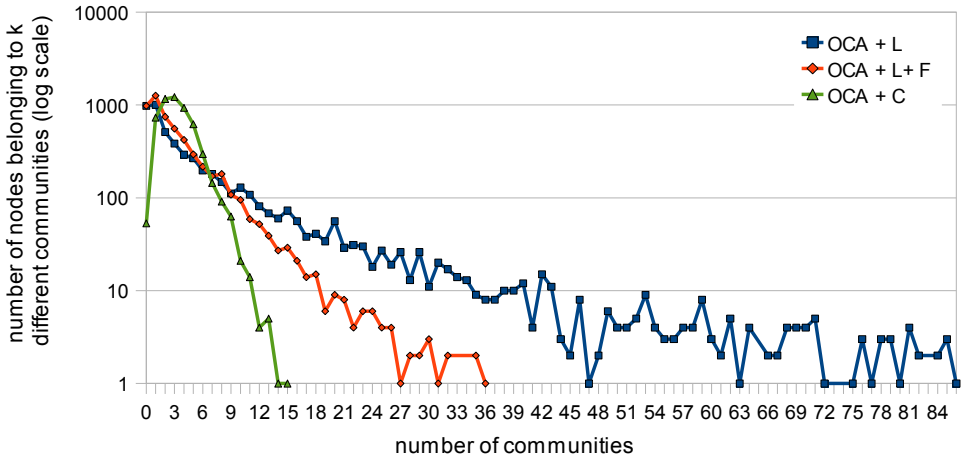


Figure 7.4: Number of nodes contained in k communities

Community Search Accuracy for Graph Partitioning. Next, we test OCA accuracy in finding non-overlapping communities for the two proposed fitness functions. The LFR benchmark tells us the "true" communities a priory.

Recall the similarity function ρ defined in Equation (6.3). Let $F = F_1, \ldots, F_\ell$ be the communities defined by LFR, and $O = O_1, \ldots, O_m$ those found by OCA_g. Denote by $V_i = \{O_j \mid \operatorname{argmax}_k \rho(F_k, O_j) = i\}$ the subset of communities that fit better with F_i than with the other F_k. We will use

$$\Theta(F, O) = \frac{1}{\ell} \sum_{i=1}^{\ell} \frac{1}{|V_i|} \sum_{O_j \in V_i} \rho(F_i, O_j). \tag{7.1}$$

as a quantitative similarity measure between both configurations. Θ also takes values between 0 and 1, where the maximum implies exactly the same community structure, and 0 indicates a totally different one. Note that this function definition also holds for overlapping structures.

In the following example we analyse the behaviour of Θ against the mixing parameter μ with different variants of OCA_g. LFR set the value of μ between 0 and 0.5 if a community structure is wanted. Above 0.5, there are no clear communities, and $\mu \geq 1$ yields a random graph.

Figure 7.5 shows that for the LFR benchmark, ϕ returns better communities, because that benchmark is geared toward an interpretation of community that implies isolation, cf. Chapter 5. Because postprocessing adds "hairs" to communities, it improves the quality of the results of $\mathcal{L}$.

This quality increment after the postprocessing reinforces our posture of not covering the whole graph: the untaken nodes are those less significant to the community structure. This gets reflected in the fact that the naive method of reassignation classifies them correctly.

To test better the Laplacian we compare the size of the "real" LFR-communities to those obtained with OCA_g. With the same parameters as

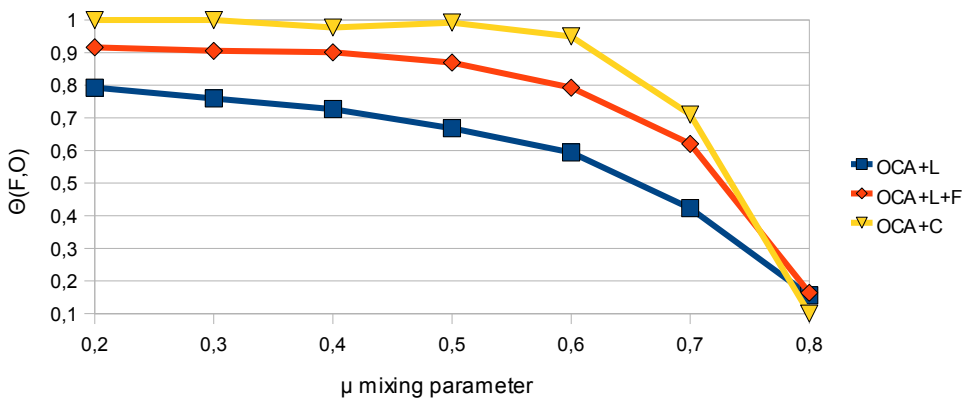


Figure 7.5: Evolution of Θ against μ.

before, Figure 7.6 shows no substantial difference between the "real" distribution of communities sizes and the ones found by OCA. Note that, in effect, the distribution follows a power law.

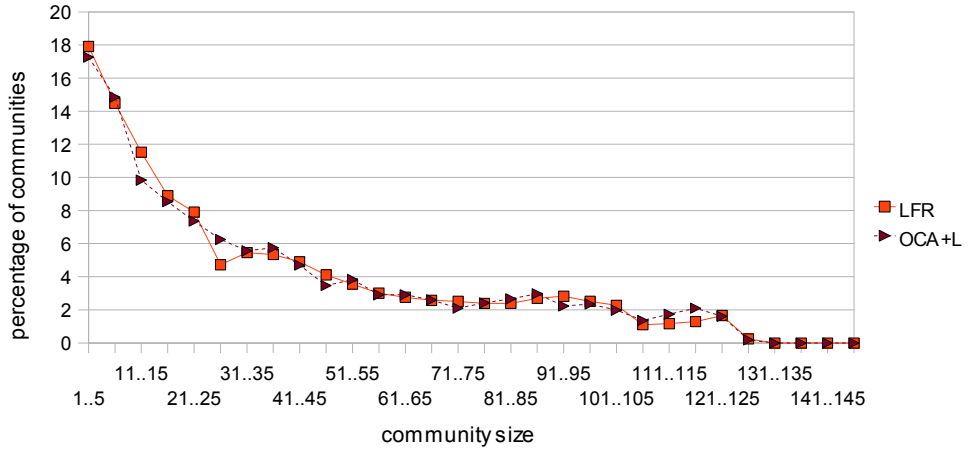


Figure 7.6: Size of real communities and those found with OCA_g

Community Size Analysis. We have theoretically studied the complexity of OCAshowing that it depends on the degree of the nodes in the graph. It is well-known that the node degrees in a social network-like graph follow a power-law distribution. In practice, this makes our algorithm suitable for large graphs. In Figure 7.7, we can observe the distribution of community sizes using the Laplacian fitness function before (OCA+L) and after (OCA+L+F) the fusion postprocessing and with conductance (OCA+C), normalized over the number of communities, for the Words dataset. We get a mean community size of 21.64 with OCA_g, and a mean of 18.05 after the fusion. With the conductance we get a mean of 29.13. Thus, the communities for ϕ are larger than those for $\mathcal{L}$, but in both cases their size is negligible compared to the number of nodes in the graph.

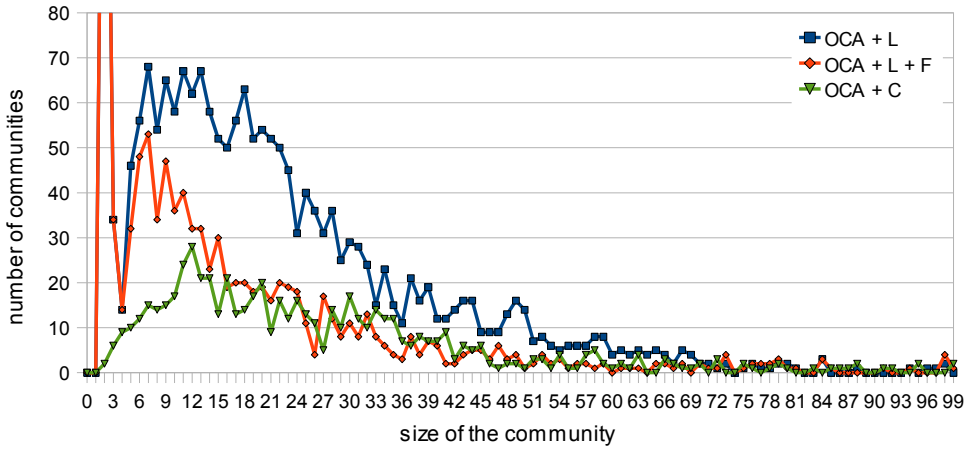


Figure 7.7: Distribution of community sizes

Scale-Independent Quality. The graphs generated with LFR also allows us to test the quality of OCA_g with sizes of different magnitude. Table 2, shows Θ function for graphs with different sizes.

Table 2: Θ of community structure

Size	10^3	10^4	10^5	10^6
OCA+L	0.860	0.847	0.848	0.835
OCA+L+F	1.000	0.990	1.000	0.987
OCA+C	1.000	1.000	0.992	1.000

Note that the quality of the structures found by OCA_g does not significantly depend on the size of the graph.

Execution Time Analysis. Now, we present the execution time evolution for several graphs generated in ascending order using the LFR benchmark. We created LFR with the default parameters varying the number of nodes. That is, a graph with degrees following a power law of parameter $\gamma = 2.5$ and average degree 20. The graph sizes we tested where from 10^3 to 10^6. Figure 7.8 shows the time in seconds to find all the communities using OCA_g. It is interesting to note that the execution time grows linearly with the number of nodes in the graph[1]. This results are based on the fact that the size of communities is similar regardless the size of the entire graph. We assume this to be realistic since it is well-known that small-world networks usually contain a large number of small communities.

Refering of the settings of the parameters of OCA_g used to obtain this execution: The initial seed was taken randomly between the not visited nodes, while the neighborhood was taken randomly such that its expected size was twice the minimum size fixed by Remark 6.1. The stopping

[1]Notice that the average degree remains constant and thus the number of edges is proportional to the number of nodes and does not affect the complexity analysis

threshold was set to 50 iterations without visiting new nodes, which resulted enough, as all the preset communities by the model LFR where found.

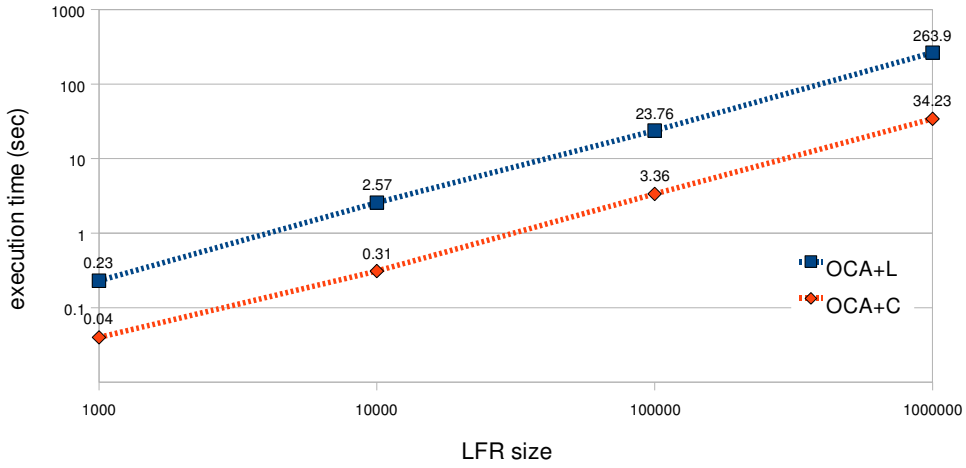


Figure 7.8: Execution Time Evolution

We have also run the experiments using the conductance fitness function. Although the time of executing an iteration is larger than in the case where the laplacian fitness function is used, in some experiments (as this one) we have observed that using conductance the number of iterations is lower and thus the execution time is also lower. However, as we see later in this section, the laplacian will be the fitness function of choice for its ability to detect overlapped communities. Moreover in these graphs the community sizes were all of size smaller than 150, so we could not use Theorem 6.2.

Accuracy for Overlapped Communities. Next, we analyze OCA_g's ability to find overlapped communities with our new Laplacian function $\mathcal{L}$, and compare the results to those obtained using the classical conductance ϕ. We use the Daisy dataset (with $p_1 = 0.9$, $p_2 = 0.5$) for these experiments, since it contains overlapping communities that we know a priory. In Figure 7.9 we can contrast both approaches on a little daisy with $n = 70$, $p = 6$ and $q = 5$.

In this example, nodes that are multiples of 5 belong to two communities. Note that, when conductance is used (Figure 7.9, right), the algorithm assigns these nodes to the central community, and excludes them from their weakest communities, thus omitting the overlap. This behaviour agrees with the qualitative explanation in Section 4. On the other side, the Laplacian (Figure 7.9, left) detects all the overlapping nodes.

In order to test this for larger graphs, we construct daisy trees with different sizes. The daises ($p = 10$, $q = 11$) are always the same size, 100 nodes, and we only change the number of them. Table 3 lists the values of $\Theta(D, O)$, cf. (7.1), where D is the daisy community structure.

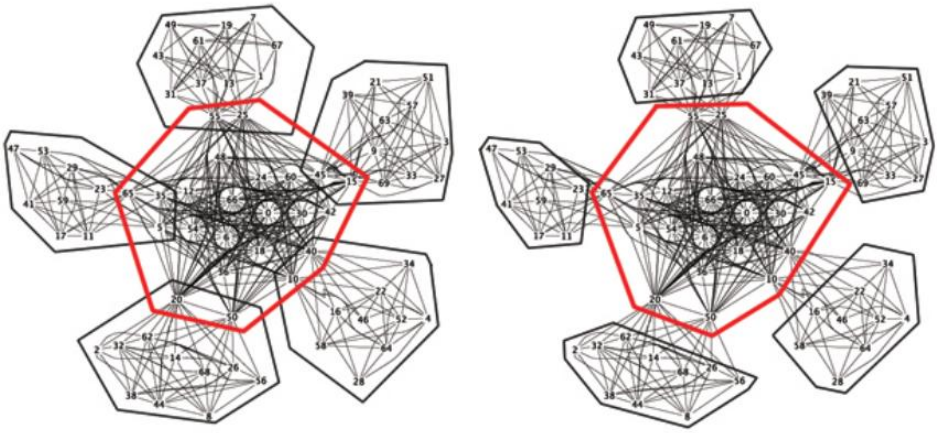


Figure 7.9: Communities found in the daisy graph. *Left:* Laplacian $\mathcal{L}$, *Right:* Conductance ϕ

Table 3: Θ of daisy community structure

Size	10^3	10^4	10^5
OCA+L	0.850461	0.840459	0.825850
OCA+L+F	0.873666	0.855303	0.849742
OCA+C	0.734057	0.730351	0.734778

From the behaviour observed for each fitness function in Figure 7.9, we can see that the Laplacian estimates better overlapping community structure. On the other hand, conductance obtains a low value of Θ, since it does not detect the overlaps.

Random Graphs Quality Test. Finally, we execute our algorithm on radom graphs from the Erdős-Rényi model [**17**]. In particular, we generate graphs $G(n, p)$ with $n = 10^5$ and different values of p. In particular, we set $p \in \{0.0002, 0.0004, 0.0006, 0.0008, 0.001\}$, that yields graphs with average degree from 20 to 100. But first in Table 4 we show the values we obtain for the constant c in each of these graphs.

Table 4: Constant c for different random graphs

p	0.0002	0.0004	0.0006	0.0008	0.001
c	0.111058	0.0784636	0.0644881	0.056078	0.0504645

In Figure 7.10 we show the size of the communities we found with the directed Laplacian in a log-log scale. We can see in this plot the tendences used in Theorem 5.1. There, we divided the possible communities in three regions: small, medium and large.

The great quantity of communities of small size in fact reflects the behaviour we showed in the proof: when c is small enough, then small communities cannot add new nodes and instead they decrease in size to the minimal communities (of size two). Moreover, even if it does not get reflected in this chart, there are no communities of size 2 for $p = 0.0002$. This is coherent with Remark 4.3, as the constant c of the graph $G(p, 0.0002)$ is the only one that belongs to the case where edges grow to paths of lenght three (a very probable situation), and thus we do not recover any community of size 2. In contrast, for smaller values of c edges can only grow to triangles, which is not as probable, giving as a result a lot of small communities.

On the other hand, when p becomes larger, we can appreciate some difference in the behaviour of middle-sized communities: there appears a small bell of communities of medium size. The nature of our algorithm does not reach to explore the larger communities.

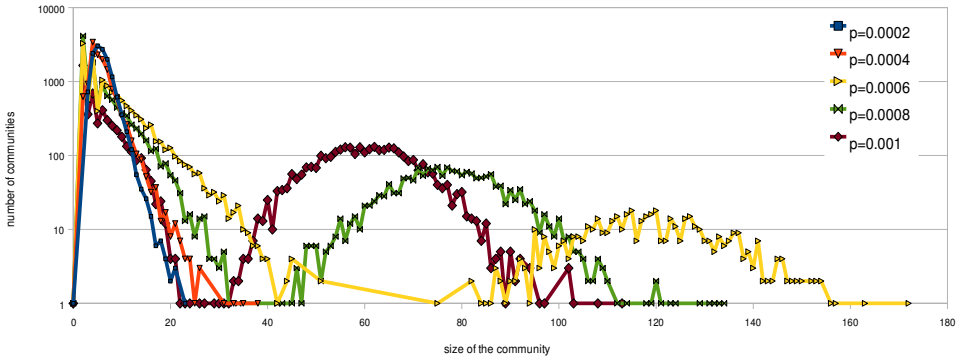


Figure 7.10: Distribution of community sizes of the application of OCA_g with the directed Laplacian on the randoms graphs $G(10^5, p)$ in a log-log scale

In Figure 7.11 we plot the sizes of the communities recovered with the Conductance. We grouped the values of the communities sizes in intervals of 5 in order to make the plot clearer. In particular, we wanted to stress the apparition of communities of size ~ 50000 for the graphs with probability greater than 0.0006.

The different behaviour of the two fitness functions we have contempled gets reflected in the average size of the communities found. The results get encoded in Table 5.

In Table 6 we show the execution times for these graphs. However, the different behaviour of the laplacian and the conductance against random graphs makes this times not comparable, as, for example, the conductance obtains communities of very large size (that take a lot of time to recover), while the communities with the Laplacian are significatively smaller.

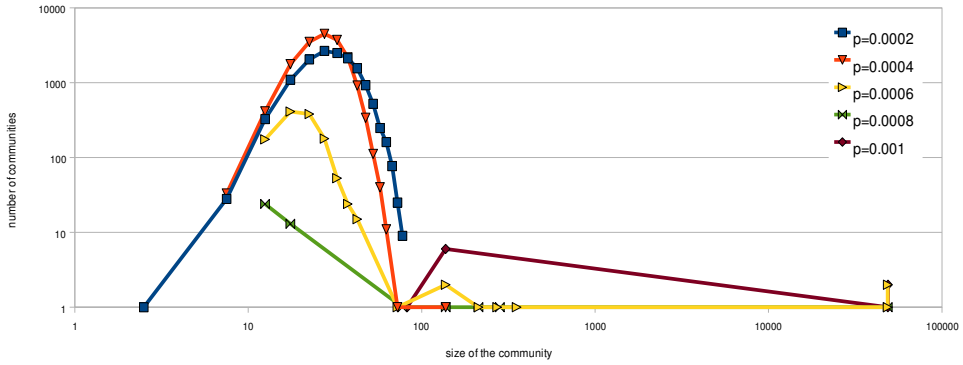


Figure 7.11: Distribution of community sizes of the application of OCA_g with the conductance on the randoms graphs $G(10^5, p)$ in a log-log scale

Table 5: Average community size

p	0.0002	0.0004	0.0006	0.0008	0.001
$\mathcal{L}$	6.049	5.825	11.524	15.122	27.745
ϕ	32.608	28.357	392.402	6770.046	16595.462

Table 6: Execution times (in seconds) of OCA_g with the Laplacian and the Conductance

p	0.0002	0.0004	0.0006	0.0008	0.001
$\mathcal{L}$	37	65	66	98	84
ϕ	30	44	1138	1716	2356

Cartesian product of cliques and the d-cube. We just add this small section to complete the theoretical analysis given at Chapter 5.

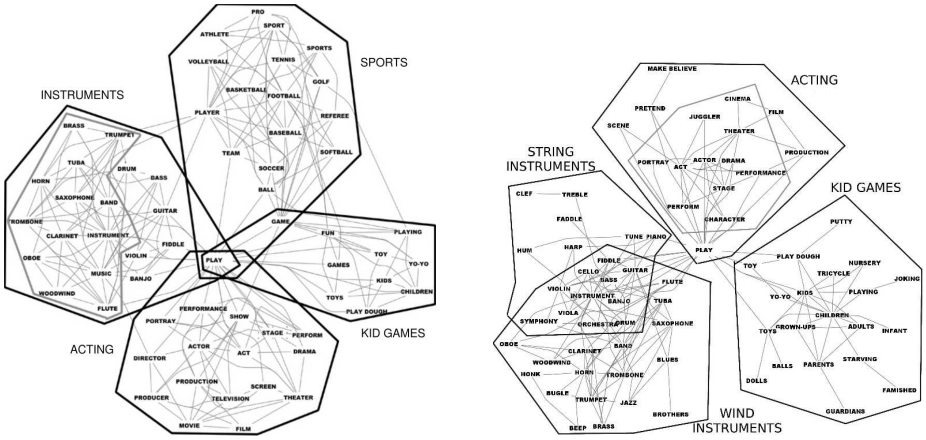
We analyzed three different products of cliques: $K_{100}\square K_{10}$, $K_{100}\square K_{100}$ and $K_{100}\square K_{1000}$. The results where the same with both objective functions, and coincided with our expectations. For $K_{100}\square K_{10}$ we obtained the ten 100-cliques, for $K_{100}\square K_{1000}$ we obtained the hundred of 1000-cliques and for $K_{100}\square K_{100}$ the two hundred copies of the 100 cliques. Observe that in the last case each node belongs to two communities.

For the graphs of cubes, we tested Q_5, Q_{10} and Q_{20}. With both objective functions, the resulting communities consisted in faces and unions of faces of low dimension.

3. Local Search Analysis

Now, we show the use of OCA for local exploration of communities in large graphs (OCA_l). We show that, given a node $v \in V$, we can efficiently calculate the overlapped communities containing v. We also show an example where a hierarchical communities arise naturally. Finally, we compare

the Laplacian fitness function and the conductance with respect to the quality of the results obtained.



(a) OCA_l with $\mathcal{L}$ (b) OCA_l with ϕ

Figure 7.12: Communities containing PLAY found by OCA_l with $\mathcal{L}$ and ϕ. Notice how the conductance excludes the word PLAY from all communities but one, but includes nodes of degree one and two. Also notice the nested communities found by both objective functions.

First, we search for the communities containing a given word using OCA_l with $\mathcal{L}$. Figure 7.12 highlights the overlapping and hierarchical structure of the communities resulting from a local exploration of the word **PLAY**. In the case of the Laplacian, we can see that this many-facetted word belongs to Instruments, Sports, Kid Games and Acting. Their hierarchical structure is reflected, for example, in the nesting of Wind Instruments inside Instruments. However, when we perform the same query using ϕ, results show that the fitness function is not able to detect the fact that the word **PLAY** belongs to different communities, and it excludes it from all of them but one.

All experiments for the Words dataset are run in less than a second. Note that for local community search, we assume that eigenvalues are calculated only once and, thus, time for calculating them is not considered here. In any case, thanks to the approximate calculus presented in Theorem 6.1, this value can be calculated very quickly: less than a second for the Words dataset.

CHAPTER 8

Conclusions and Future Work

We have presented a novel way of finding communities that it is not only efficient for handling large graphs, but also takes into account the fact that nodes in a graph might belong to several communities at the same time. We have seen that it is possible to adapt the community search method to the properties of the graph, without having to rely on complex and non-intuitive parameters set by the user. Our results show that the Laplacian fitness function proposed in this Master Thesis provides a natural way to detect overlapped communities, and is able to find them more accurately than previous proposals adapted to non-intersecting communities. Also the complexity of our approach is computationally lower and it is easily updatable.

From here, some interesting future research topics have already been pointed out with observations. Moreover, in the mathematical side, vector representations have shown to be useful to derive results on graphs, and can surely be pushed further to obtain stronger properties.

A deeper study of the directed Laplacian operator must be done, applying it in general cases and different functions.

From the computational point of view, it would be interesting to design a system to keep and update the community information once it has been calculated, in order to respond to successive queries more efficiently, and to explore the *hierarchy* of the community structure in the graph. Another interesting research line for the next future would be studying strategies for the efficient parallelization of our algorithm, making it possible to handle even larger graphs.

Finally, this work offers the possibility to new explore neighboring areas, such as graph summarization for graphs containing overlapped communities.

Bibliography

1. Noga Alon and Benny Sudakov, *Bipartite subgraphs and the smallest eigenvalue.*, Comb. Probab. Comput. **9** (2000), no. 1, 1–12 (English).
2. Luis A. Nunes Amaral, Antonio Scala, Marc Barthelemy, and H. Eugene Stanley, *Classes of behavior of small-world networks*, (2000).
3. Kenneth Appel and Wolfgang Haken, *Every planar map is four colorable*, Contemporary Mathematics, vol. 98, American Mathematical Society, Providence, RI, 1989, With the collaboration of J. Koch. MR MR1025335 (91m:05079)
4. Albert-Laszlo Barabasi, Reka Albert, and Hawoong Jeong, *Scale-free characteristics of random networks: the topology of the world-wide web*, Physica A: Statistical Mechanics and its Applications **281** (2000), no. 1-4, 69–77.
5. Norman Biggs, *Algebraic graph theory*, Cambridge University Press, February 1994.
6. B. Borchers, *CSDP, a C library for semidefinite programming*, **11** (1999), 613–623.
7. Ulrik Brandes, Daniel Delling, Martin Höfer, Marco Gaertler, Robert Görke, Zoran Nikoloski, and Dorothea Wagner, *On finding graph clusterings with maximum modularity*, Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'07), Lecture Notes in Computer Science, Springer, 2007.
8. A. Capocci, V. D. P. Servedio, G. Caldarelli, and F. Colaiori, *Detecting communities in large networks*, Physica A: Statistical and Theoretical Physics **352** (2005), no. 2-4, 669–676.
9. Fan Chung, *Spectral graph theory*, American Mathematical Society, February 1997.
10. ———, *Laplacians and the cheeger inequality for directed graphs*, Annals of Combinatorics **9** (2005), 1–19.
11. Sebastian M. Cioabă, *The spectral radius and the maximum degree of irregular graphs*, Electr. J. Comb. **14** (2007), no. 1.
12. Sebastian M. Cioabă, David A. Gregory, and Vladimir Nikiforov, *Note: Extreme eigenvalues of nonregular graphs*, J. Comb. Theory Ser. B **97** (2007), no. 3, 483–486.
13. Aaron Clauset, M. E. J. Newman, and Cristopher Moore, *Finding community structure in very large networks*, August 2004.
14. Inderjit Dhillon, Yuqiang Guan, and Brian Kulis, *A fast kernel-based multilevel algorithm for graph clustering*, KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (New York, NY, USA), ACM, 2005, pp. 629–634.
15. Luca Donetti and Miguel A. Munoz, *Detecting network communities: a new systematic and efficient algorithm*, October 2004.
16. Jordi Duch and Alex Arenas, *Community detection in complex networks using extremal optimization*, Physical Review E **72** (2005), no. 2, 027104+.
17. P. Erdös and A. Rényi, *On random graphs, i*, Publicationes Mathematicae (Debrecen) **6** (1959), 290–297.
18. Santo Fortunato and Claudio Castellano, *Community structure in graphs*, 2007.
19. Michelle Girvan and M. E. J. Newman, *Community structure in social and biological networks*, Proc. Natl. Acad. Sci. USA **99** (2002), 7821.
20. Chris Godsil and Gordon Royle, *Algebraic graph theory*, Springer, 2001.
21. M Grotschel, L Lovasz, and A Schrijver, *Geometric algorithms and combinatorial optimization*, 1988.
22. Roger Guimerà, Marta Sales-Pardo, and Luís A. Nunes Amaral, *Modularity from fluctuations in random graphs and complex networks*, Phys. Rev. E **70** (2004), no. 2, 025101.

23. J. William Helton and Victor Vinnikov, *Linear matrix inequality representation of sets*, 2003.
24. A J Hoffman, *On eigenvalues and colorings of graphs*, In Graph theory and its Applications (Proc. Advanced Sem., Math. Research, Academic Press, 1969, pp. 79–91.
25. Ravi Kannan, Santosh Vempala, and Adrian Vetta, *On clusterings: Good, bad and spectral*, Journal of the ACM **51** (2004), no. 3, 497–515.
26. Jon M. Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew S. Tomkins, *The web as a graph: Measurements, models, and methods*, 1999, pp. 1+.
27. Ioannis Koutis and Gary L. Miller, *Graph partitioning into isolated, high conductance clusters: theory, computation and applications to preconditioning*, SPAA '08: Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures (New York, NY, USA), ACM, 2008, pp. 137–145.
28. Andrea Lancichinetti, Santo Fortunato, and Janos Kertesz, *Detecting the overlapping and hierarchical community structure of complex networks*, 2008.
29. Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi, *Benchmark graphs for testing community detection algorithms*, Physical Review E (Statistical, Nonlinear, and Soft Matter Physics) **78** (2008), no. 4.
30. Adrian S. Lewis, Pablo A. Parrilo, and Motakuri V. Ramana, *The lax conjecture is true*, PROCEEDINGS OF THE AMS **133** (2005), 2495.
31. László Lóvasz, *On the Shannon capacity of a graph*, IEEE Trans. Inform. Theory **25** (1979), 1–7.
32. N. Martinez-Bazán, J. Nin, S. Gómez-Villamayor, M. A. Sánchez-Martínez, V. Muntés-Mulero, and J. L. Larriba-Pey, *DEX: High-performance exploration on large graphs for information retrieval*, CIKM '07: Proceedings of the sixteenth ACM conference on information and knowledge management (New York, NY, USA), ACM, 2007, pp. 573–582.
33. D. L. Nelson, C. L. McEvoy, and T. A. Schreiber, *The University of South Florida word association, rhyme, and word fragment norms*, 1998.
34. M. E. Newman and M. Girvan, *Finding and evaluating community structure in networks.*, Phys Rev E Stat Nonlin Soft Matter Phys **69** (2004), no. 2 Pt 2.
35. M. E. J. Newman, *Fast algorithm for detecting community structure in networks*, September 2003.
36. ———, *Finding community structure in networks using the eigenvectors of matrices*, May 2006.
37. M. E. J. Newman and E. A. Leicht, *Mixture models and exploratory analysis in networks.*, Proc Natl Acad Sci USA **104** (2007), no. 23, 9564–9569.
38. Arnau Padrol, Guillem Perarnau, Julian Pfeifle, and Victor Muntés, *Overlapping community search for large graph mining*, Unpublished, 2009.
39. Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek, *Uncovering the overlapping community structure of complex networks in nature and society*, Nature **435** (2005), 814.
40. Pascal Pons and Matthieu Latapy, *Computing communities in large networks using random walks (long version)*, 2005.
41. Alex Pothen, *Graph partitioning algorithms with applications to scientific computing*, Parallel Numerical Algorithms, Kluwer Academic Press, 1997, pp. 323–368.
42. Matthew J. Rattigan, Marc Maier, and David Jensen, *Graph clustering with network structure indices*, ICML '07: Proceedings of the 24th International Conference on Machine learning (New York, NY, USA), ACM, 2007, pp. 783–790.
43. Stuart Russell and Peter Norvig, *Artificial intelligence: A modern approach*, 2nd edition ed., Prentice-Hall, Englewood Cliffs, NJ, 2003.
44. Huawei Shen, Xueqi Cheng, Kai Cai, and Mao-Bin Hu, *Detect overlapping and hierarchical community structure in networks*, Physica A: Statistical Mechanics and its Applications **388** (2009), no. 8, 1706 – 1712.
45. Bill Smith, *The pagerank citation ranking: Bringing order to the web - page, brin, motwani, winograd (researchindex)*, Canadian journal of botany **32** (1999), no. 12, 1–99.

46. Daniel A. Spielman and Shang-Hua Teng, *A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning*, Preprint http://arxiv.org/abs/0809.3232, 2008.
47. R P Stanley, *A bound on the spectral radius of graphs with e edges*, Linear Algebra Appl (1987), no. 87, 267–269.
48. Dragan Stevanović, *The largest eigenvalue of nonregular graphs*, J. Comb. Theory Ser. B **91** (2004), no. 1, 143–146.
49. J J Sylvester, *Chemistry and algebra*, Nature, no. 17, 28–4.
50. Luca Trevisan, *Max cut and the smallest eigenvalue*, CoRR **abs/0806.1978** (2008).
51. L. Vandenberghe and S. Boyd, *Semidefinite programming*, **38** (1996), 49–95.
52. Walter Willinger, David Alderson, and John C. Doyle, *Mathematics and the internet: A source of enormous confusion and great potential*, Notices of the AMS **56** (2009), no. 5.
53. Xiao-Dong Zhang, *Eigenvectors and eigenvalues of non-regular graphs*, Linear Algebra and its Applications **v409** (2005).