Università degli studi di Padova
Dipartimento di Matematia "Tullio Levi-Civita"

Locating leading components in networks through the optimization of nonlinear modularity functions

Laurea Magistrale in Matematica

Supervisor
Prof. Francesco Rinaldi
Assistant Supervisor
Prof. Francesco Tudisco

Candidate
Fabio Brotto
Identification Number
1110875

Academic year 2016/2017
September 22, 2017

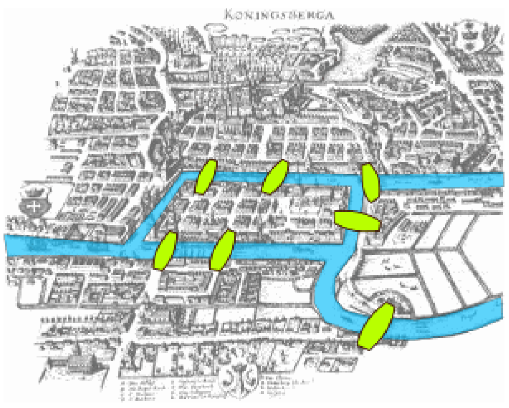
Contents

Chapter 1

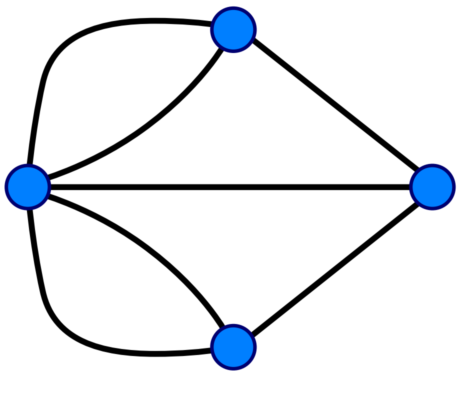
Introduction

1.1 Graph theory: beginnings and main applications

The first time for graph theory in the history of mathematics was in 1736, thanks to Euler's solution of the puzzle of Königsberg's bridges. The city of Königsberg in Prussia, currently Kaliningrad in Russia, was set on both sides of the Pregel river and included two large islands which were connected to each other, or to the two mainland portions of the city, by seven bridges. The problem was to devise a walk through the city that would cross each of those bridges once and only once. Euler reformulated the problem in abstract terms, laying the foundations of graph theory. An undirected graph is a pair $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ is a finite set of vertices, also called nodes, and E is the edge set, a set of undirected pairs of vertices. In the Euler's problem the vertices are the different parts of the city and the edge set is made of the seven bridges.



(a) *Königsberg city.*



(b) *Graph representation.*

Figure 1.1: Seven bridges problem.

So by eliminating all features except the list of land masses and the bridges connecting them, he proved that the problem has no solution. An urban legend says that, around

1750, on Sundays the wealthy citizens of Königsberg walked around the city trying in vain to find a solution to the problem, for further information we refer the interested reader to [1]. Since then a lot has been done on graphs and their mathematical properties. In the 20th century they have also become extremely useful as representation of a wide variety of systems in different areas. Biological, social, technological and information networks can be studied as graphs, and graph analysis has become crucial to understand the features of those systems. For instance, social network analysis started in the 1930's and has become one of the most important topics in sociology. The main focus of social network analysis is the study of relationships among social entities and the patterns and implications of these relationships. Many researchers have realized that the network view brings new ideas for answering standard social and behavioral science research questions by giving precise formal definition to aspects of the political, economic, or social structural evironment. In the social network analysis, the social environment can be expressed as a graph by setting the individuals as vertices and the relationships among interacting units as edges, for the interested reader we refer to [2].

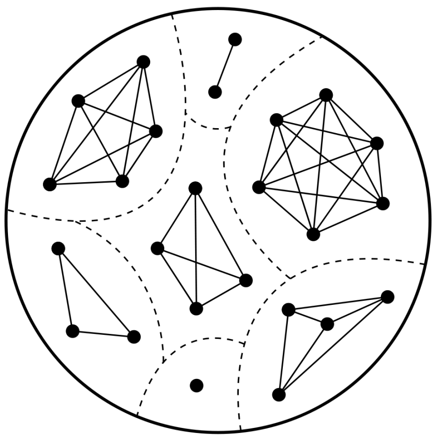


Figure 1.2: Graph with seven communities.

In recent times, the computer revolution has provided researchers with a huge amount of data and computational resources to process and analyze those data. One can potentially handle real networks composed of milions or even billions of vertices. So discovering efficient algorithms able to control huge networks is one of the most important topic of graph theory for the computational point of view. Another relevant topic of graphs representing real systems is community detection, or clustering, i. e. the organization of vertices in clusters. Communities, also called clusters or modules, are groups of vertices which probably share common properties and/or play similar roles within the graph. A simple idea of community is a subset of the vertex set that is highly connected inside and poorly connected with the rest of the graph, for an example see Figure 1.2.

Communities occur in many network systems coming from biology, computer science, engineering, economics, politics and so on. For instance in [3], the authors show how an algorithm that partitions weighted graph into communities is a useful tool in studying the modularity organization of biological networks: genes in the same functional module confer similar phenotype deletions, known protein complexes are largely contained in the functional modules in their entirety and module identifiction could be very useful for gene annotation. In the graph of the World Wide Web the communities may correspond to groups of pages dealing with the same or related topics. The web graph is made of web pages as nodes and hyperlinks as arcs, for further information we refer to [4]. Clustering Web clients detects groups that have similar interests and are geografically near to each other. This may improve the performance of services provided on the World Wide Web in a way that each cluster of clients could be served by a dedicated mirror server, we refer to [6]. Moreover, identifying clusters of customers with similar interests in the network of buy relationships between customers and products of online retailers (like, e. g., www.amazon.com) can set up efficient recommendation systems that better guide customers through the list of items of the retailer and improve the business opportunities, for further details we refer to [7]. Another topic touched by community detection is food web. Compartments have been difficult to detect in empirical food webs because of incompatible approaches or insufficient methodological rigour. In [5] it is shown that the social network science is a helpful tool to detect empirical food web comparments, see Figure 1.3. The method identifies compartmental boundaries in which interactions are concentrated, so it is compatible with the definition of compartments. As we will see in the second chapter of this work, the method is rigorous because it maximizes an explicit function, identifies the number of non-overlapping compartments, assigns membership to compartments.

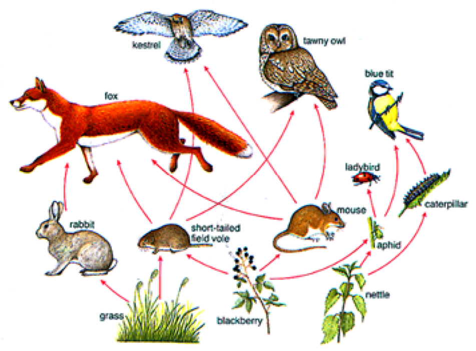
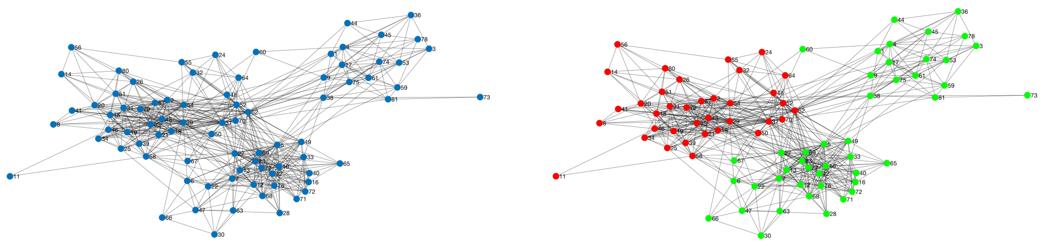


Figure 1.3: A food web.

Even if it will not be part of in this work, another important aspect related to community structure is the hierarchical organization displayed by many network systems in the real world. Real networks are usually composed by communities including smaller communities, which in turn include smaller communities, etc. The human body offers a paradigmatic example of hierarchical organization: it is composed by organs, organs are composed by tissues, tissues by cells, and so on. To conclude graph communities is a popular topic in computer science too. For instance in parallel computing is crucial to know what is the best way to allocate tasks to processors in order to minimize the communications between them and enable a rapid performance of the calculation. This can be accomplished by splitting the computer cluster into groups with roughly the same number of processors, such that the number of physical connections between processors of different groups is minimal, we refer to Section IV.A of [8].

1.2 About this thesis

Here we give the details related to the main topics of this thesis. The main goal is the community detection of given graphs through maximization of a nonlinear function. What we are going to maximize is the modularity function of a graph as it has been studied in [8], [10] or [11]. The definition of the modularity function is based on a random graph, called null model. In a random graph, structure introduced by [9], the probability of having an edge between a pair of vertices is equal for all possible pairs. For istance, the distribution of the degree of the nodes could be binomial so most vertices have equal or similar degree. The formal introduction of graph theory is displayed at the beginning of Chapter 2. Given a graph $G = (V, E)$, we can create a particular null model with the same nodes of G. Even if no definition is universally accepted, we can say that modularity function computes the difference between the connection of G and the connection of a null model, where connection means how many vertices there are between nodes. Positive value of modularity indicates higher connection respect to the null model, so these nodes assemble a community. For sure, modularity depends on the null model that we choose, we're going to explain formally this matter in the next chapter. Since real systems can be represented by graphs, the final task will be finding communities in real networks, see Figure 1.4.



(a) *UKFaculty network.* (b) *UKFaculty network bi-partitioned.*

Figure 1.4: Example of bi-partitioning of UKFaculty real network using basin hopping global algorithm with non monotone spectral projected gradient method as local search.

In the next chapter we introduce the mathematical theory of community detection, in particular we choose a precise definition of modularity measure and show how this is connected with a particular matrix M, called modularity matrix, we refer to [10]. The leading eigenvectors of this matrix give us an upper bound of the best bi-partitioning of a given graph. The next step will be describing how to find $\mathcal{M}$, a nonlinear extension of M, and show the strict relation between leading eigenvector of $\mathcal{M}$ and the best bi-partitioning of the graph, for further information we refer to [11]. Actually this strict relation is a maximization problem, the main goal of this thesis will be to give an equivalent constrained formulation of the problem in order to use strong optimization algorithms over a convex feasible set. We study four different iterative feasible local optimization algorithms: the two classical Frank-Wolfe and projected gradient method, with Armijo line search, and their two variations with a non monotone Armijo line search. For all of them we give a convergence theorem and we apply them over a toy-model graph in order to select the best parameters inside the codes. A toy model graph is a graph where we can control the communities inside it in order to check the goodness of our results, for further information we refer to Chapter 4 or [31]. For different cases of graph we will study which are the best parameter setting to give to the different algorithms. After that we introduce a probabilistic global optimization method, the Basin Hopping algorithm, that embeds as local searches the algorithms listed before. With the basin hopping we try to compute a global maximum, that will be the best bi-partitioning of the given graph, and we can study the differences between the local algorithms respect to CPU time or maximum value. In the last chapter we show the results. We start by analyzing the best parameter setting for the local algorithms as a function of computational time, by using the performance metric studied in [12]. Then we apply in real network problems and we compare our results with the results in [11]. Summarizing, this work is divided into four different chapters:

- **Community detection:** community detection theory is displayed with a particular focus on modularity measures and its nonlinear expansion.

- **Optimization algorithms:** after rewriting the main bi-partitioning problem using a constrained non convex model, we introduce four local optimization algorithms and how to use them inside basin hopping, a global probabilistic algorithm. Convergence results are studied for each of them.

- **Computational results:** here we show the computational results of the thesis. First, we study how to choose the best parameters of local algorithms using the performance profile technique analyzed in [12]. To do that we apply the codes on toy-model graphs. After that, we use the parameter setting to run the global algorithm over real networks and we discuss the similarities and the differences between the different approaches.

- **Conclusions:** in the last chapter we briefly summarize the whole work, explaining the strength of our approach. Finally, we try to give some further developments for this subject.

Chapter 2

Community detection

This section has the aim to explain the mathematical theory built to analyze community detection in a given network. In particular we are interested in detecting a partition of the network into two sets, namely called communities. A simple definition of community is a set of nodes highly liked with each other and poorly connected with the rest of the nodes in the graph. A first problem is to give a precise mathematical definition of what a community is. No definition is universally accepted but a very popular idea is based on the concept of modularity introduced by Newman and Girvan, [10]. First of all we need to define a null model that is a graph used as a term of comparison, to verify whether the graph under analysis shows some community structure or not. Even the null model could have different definitions, the one proposed by Newman and Girvan consists of a randomized version of the original graph, where edges are chosen randomly, under the constraint that the expected degree of each vertex matches the degree of the vertex in the original graph. This null model brings us to the definition of modularity, a function that calculates the goodness of partition of a graph into cluster. In this formulation of modularity, a subgraph is a community if the number of edges significantly exceeds the number of internal edges that the same subgraph would have in the null model. The N-G modularity function is a particular type of what is called quality functions. The latter are functions that assign a number to each partition of a graph. In this way one can rank a partition based on their score given by the quality function. Partitions with high score are "good" and the one with the largest score is, by definition, the best. So by assumption, high value of modularity indicate good partitions, the partition corresponding to its maximum value on a given graph should be the best, or at least a very good one. This is the main motivation for the topic of modularity maximization.

Notation. In order to describe this maximization problem we use the following notation. Let $G = (V, E)$ be an undirected graph, where V indicates the vertex set and E the edge set. The vertices are enumerated from 1 to n. $\mu : V \to \mathbb{R}_{>0}$ and $\omega : E \to \mathbb{R}_{>0}$ are the positive measures respectively of V and E. We denote the weighted scalar product by $\langle x, y \rangle_\mu = \sum_{i=1}^n \mu_i x_i y_i$. Similarly, for $p \geq 1$, we write $||x||_{p,\mu}^p = \sum_i \mu_i |x_i|^p$ for the weighted l^p norm on V. Given two subsets $A, B \subseteq V$, the set of edges betweeen nodes in A and B is denoted by $E(A,B)$. When A and B coincide we use the short notation $E(A)$. The overall weight of a set is the sum of weights in the set, thus for $A, B \subseteq V$, we write

$$\mu(A) = \sum_{i \in A} \mu_i, \quad \text{and} \quad \omega(E(A,B)) = \sum_{i,j \in E(A,B)} \omega(ij).$$

In particular $\omega(E(i),V) = d_i$ stands for the degree of node i, and $\omega(E(A,V)) = \mathrm{vol}(A) = \sum_{i \in A} d_i$ stands for the volume of the set A. For a subset $A \subseteq V$ we write $\bar{A}$ to denote the complement $V \setminus A$ and $\mathbb{1}_A \in \mathbb{R}^n$ represents the characteristic vector

$$(\mathbb{1}_A)_i = \begin{cases} 1, & \text{if } i \in A, \\ 0, & \text{if } i \notin A. \end{cases}$$

2.1 Modularity measure

As told before, the modularity measure of a set of nodes $A \subseteq V$, quantifies the difference between the actual weights of edges in A with respect to the expected weight of edges of a null model. A subgraph $G(A)$ is then identified as a community if the modularity measure of A is "large enough". Let $\mathcal{G}_0 = (V_0, E_0)$ be the expected graph of the random ensemble $\mathcal{G}_0$, with weight measure $w_0 : E_0 \to \mathbb{R}^+$ (the null model). The definition of modularity $Q(A)$ of $A \subseteq V$ is as follows

$$Q(A) = w(E(A)) - w_0(E_0(A)), \tag{2.1}$$

so that $Q(A) > 0$ if the actual weight of edges in $G(A)$ exceeds the expected one in $\mathcal{G}_0(A)$. A set of nodes A is a cluster (or community) if it has positive modularity, and the associated subgraph $G(A)$ is called a module. Thus the Newman-Girvan modularity is based on the assumption that $A \subseteq V$ is a community of nodes if the induced subgraph $G(A) = (A, E(A))$ contains more edges than expected, if edges were placed at random according to a null model $\mathcal{G}_0$. The Newman-Girvan null model is based on the definition of weighted Chung-Lu model that we recall here below, for further information we refer to [13].

Definition 1. Let $\delta = (\delta_1, \ldots, \delta_n)^T > 0$, and let $X(p)$ be a nonnegative random variable parametrized by the scalar parameter $p \in [0,1]$, whose expectation is $\mathbb{E}[X(p)] = p$. We say that a graph $G = (V,E)$ with weight function ω follows the X-weighted Chung-Lu random graph model $\mathcal{G}(\delta, X)$ if, for all $i,j \in V$, $\omega(ij)$ are independent random variables distribuited as $X(p_{ij})$ where $p_{ij} = \frac{\delta_i \delta_j}{\sum_{i=1}^n \delta_i}$.

According to Chung-Lu model, if $\tilde{G}$ is a random graph drawn from $\mathcal{G}(\delta, X)$ then the expected degree of node i is $\mathbb{E}[d_i] = \delta_i$. In our problem we assume that the null model $\mathcal{G}_0$ agrees with the Chung-Lu random graph model $\mathcal{G}(\delta, X)$ with the vector δ equal to the degree sequence $d = (d_1, \ldots, d_n)^T > 0$ of the actual network $G = (V,E)$. Under this assumption, the modularity measure (2.1) becomes

$$Q(A) = \omega(E(A)) - \frac{\mathrm{vol}(A)^2}{\mathrm{vol}(V)},$$

and we have that $Q(A) = Q(\bar{A})$, for any $A \subseteq V$. We recall that we want to find the bi-partitioning $\{A, \bar{A}\}$ of the vertex set having maximal modularity. It's easy to extend the modularity of a subset to a modularity measure of a partition of G, by calculating the sum of the modularities: given a partition $\{A_1, \dots A_k\}$ of V, its modularity value is

$$q(A_1, \dots A_k) = \frac{1}{\mu(V)} \sum_{i=1}^{k} Q(A_i) \tag{2.2}$$

An important alternative is a normalized version of modularity, since small groups are difficult to be identified using the standard version. We denote this normalized modularity with

$$Q_\mu(A) = \frac{Q(A)}{\mu(A)},$$

and the normalized modularity of a partition $\{A_1, \dots A_k\}$ of V as

$$q_\mu(A_1, \dots A_k) = \frac{1}{\mu(V)} \sum_{i=1}^{k} Q_\mu(A_i) = \frac{1}{\mu(V)} \sum_{i=1}^{k} \frac{Q(A_i)}{\mu(A_i)}. \tag{2.3}$$

In both equations (2.2) and (2.3) when the partition consists of only two sets $\{A, \bar{A}\}$ we use the shorter notation $q(A)$ and $q_\mu(A)$ for $q(A, \bar{A})$ and $q_\mu(A, \bar{A})$ respectively. So our problem consists of finding the maximal modularity of

$$q(A) = \frac{2}{\mu(V)} Q(A), \quad \text{and} \quad q_\mu(A) = \mu(V) \frac{Q(A)}{\mu(A)\mu(\bar{A})},$$

that is:

$$q(G) = \max_{A \subseteq V} q(A), \quad \text{and} \quad q_\mu(G) = \max_{A \subseteq V} q_\mu(A). \tag{2.4}$$

Finding a global solution of Q is impossible, due to the huge number of ways in which it is possible to partition a graph. Indeed it is proved in [14] that modularity optimzation is an NP-complete problem. Nevertheless there are many algorithms that brings us to a good approximations of the modularity maximum in reasonable time.

2.1.1 Modularity matrix

One of the most used technique is spectral optimization. This approach consists the leading eigenvalue and eingevector of a special matrix, the modularity matrix, to approximate the maximum value of q. According with the Newman-Girvan null model, if $d \in \mathbb{R}^n$ is the degrees vector of the graph G, the normalized modularity matrix of G, with vertex measure μ and edge measure ω, is

$$(M)_{ij} = \frac{1}{\mu_i} \Big(\omega(ij) - \frac{d_i d_j}{\mathrm{vol}(V)} \Big), \qquad \text{for}\, i, j = 1, \dots, n. \tag{2.5}$$

Note that if we consider the special edge measure $\omega(ij) = 1$ if $(ij) \in E$ and $\omega(ij) = 0$ if $(ij) \notin E$, the first term $\omega(ij)$ is the usual adjacency matrix of G. Moreover the second

term, $\frac{d_i d_j}{\text{vol}(V)}$, is the adjacency matrix of the expected graph described in Definition 1. We can also observe that M is a symmetric matrix so all its eigenvalues are real and $M\mathbb{1} = 0$, where $\mathbb{1} = (1, \dots, 1)^T \in \mathbb{R}^n$.
Suppose the eigenvalues are enumerated in descending order $\lambda_1(M) \geq \dots \geq \lambda_n(M)$. Then its largest eigenvalue $\lambda_1(M)$ is strictly positive or it is 0, since we have just observed that $\mathbb{1}$ is an eigenvector with corresponding eigenvalue 0. If $\lambda_1(M) = 0$, the graph is said to be algebraically indivisible, i.e. it has no community structure (for more details see [15]).

Spectral method

To link $\lambda_1(M)$ with the maximal value of the modularity function we observe that every partition of a graph with n vertices in two groups (A and $\bar{A}$) can be represented by an index vector $v_A \in \mathbb{R}^n$ with components $\{+1, -1\}$: $(v_A)_i = 1$ if vertex $i \in A$ and $(v_A)_i = -1$ if $i \in \bar{A}$. By using the identities $\mathbb{1} = \mathbb{1}_A + \mathbb{1}_{\bar{A}}$ and $M\mathbb{1} = 0$, we have

$$\langle v_A, Mv_A \rangle_\mu = \sum_{i=1}^n \mu_i (\mathbb{1}_A - \mathbb{1}_{\bar{A}})_i \big(M(\mathbb{1}_A - \mathbb{1}_{\bar{A}})\big)_i =$$

$$= \sum_{i=1}^n \mu_i (2\mathbb{1}_A - \mathbb{1})_i \big(M(2\mathbb{1}_A - \mathbb{1})\big)_i = 4\sum_{i=1}^n \mu_i (\mathbb{1}_A - \frac{1}{2}\mathbb{1})_i \big(M(\mathbb{1}_A)\big)_i =$$

$$= 4\left(\sum_{i=1}^n \mu_i (\mathbb{1}_A)_i \frac{1}{\mu_i}\Big(\sum_{j\in A}\omega(ij) - \frac{d_i \text{vol}(A)}{\text{vol}(V)}\Big) - \frac{1}{2}\sum_{i=1}^n \mu_i \frac{1}{\mu_i}\Big(\sum_{j\in A}\omega(ij) - \frac{d_i \text{vol}(A)}{\text{vol}(V)}\Big)\right) =$$

$$= 4\left(\sum_{i=1}^n (\mathbb{1}_A)_i \sum_{j\in A}\omega(ij) - \sum_{i=1}^n (\mathbb{1}_A)_i \frac{d_i \text{vol}(A)}{\text{vol}(V)} - \frac{1}{2}\sum_{i=1}^n \sum_{j\in A}\omega(ij) + \frac{1}{2}\sum_{i=1}^n \frac{d_i \text{vol}(A)}{\text{vol}(V)}\right) =$$

$$= 4\left(\omega(E(A)) - \frac{\text{vol}^2(A)}{\text{vol}(V)} - \frac{1}{2}\text{vol}(A) + \frac{1}{2}\frac{\text{vol}(V)\text{vol}(A)}{\text{vol}(V)}\right) = 4Q(A).$$

The vector v_A can be rewritten as $v_A = \sum_{i=1}^n a_i w_i$, where w_i, $i = 1, \dots, n$, are the eigenvectors of M. If v_A is properly normalized, then

$$Q(A) = \frac{1}{4}\langle v_A, Mv_A \rangle_\mu = \frac{1}{4}\sum_{i=1}^n \mu_i a_i^2 \lambda_i, \tag{2.6}$$

where λ_i is the eigenvalue of M corrisponding to eigenvector w_i. It is worth remarking that the sum contains at most $n-1$ terms, as M has at least the zero eigenvalue. We suppose, as before, that the eigenvalue are ordered in a decreasing way and the largest eigenvalue λ_1 is strictly positive.
For an upper bound approximation we can use the spectral method. Consider the the Rayleigh quotient of M that is the real valued function

$$r_M(x) = \frac{\langle x, Mx \rangle_\mu}{||x||^2_{2,\mu}} \tag{2.7}$$

As the matrix M is symmetric with respect to the weighted scalar product $\langle \cdot, \cdot \rangle_\mu$, its eigenvalues can be characterized as variational values of r_M. In particular, it holds:

$$\lambda_1 = \max_{x \in \mathbb{R}^n} r_M(x).$$

Notice that $||v_A||^2_{2,\mu} = \mu(V)$. Thus the quantity $q(G)$ can be rewritten in terms of r_M, so in terms of M:

$$q(G) = \max_{A \subseteq V} q(A) = \max_{A \subseteq V} \frac{2Q(A)}{\mu(V)} = \frac{1}{2} \max_{A \subseteq V} r_M(v_A) = \frac{1}{2} \max_{x \in \{-1,1\}^n} r_M(x). \tag{2.8}$$

But computing $\max r_M(x)$ with the constraint $x \in \{-1,1\}^n$ is a NP-hard problem. We can easily find a relaxation of $q(G)$ by dropping the binary constraint, hence, we obtain what we look for, an upper bound for the optimal bi-partition of G. The step we need to take in order to pass from $q(G)$ to $\lambda_1(M) = \max_{x \in \mathbb{R}^n} r_M(x)$ is what we call linear modularity relaxation.

Untill now we have discussed the unnormalized version $q(A)$. Here we will show that even if the solution of $q(G)$ and $q_\mu(G)$ are in general far from being the same, the linear modularity relaxation is valid also for $q_\mu(G)$. During the proof of the Proposition 2.1.1 we are going to rewrite $q_\mu(G)$ in term of M as well.

Proposition 2.1.1. If the largest eigenvalue of M is positive, then it coincides with the linear modularity relaxation of both $q(G)$ and $q_\mu(G)$.

Proof. We have already observed that λ_1 is the linear relaxation of $q(G)$. Similarly we do with $q_\mu(G)$. Consider $u_A = \mathbb{1}_A - \frac{\mu(A)}{\mu(V)} \mathbb{1}$. Notice that $\mu(\bar{A}) = \mu(V) - \mu(A)$.
$||u_A||^2_{2,\mu} = \frac{\mu(A)\mu(\bar{A}}{\mu(V)}$, since

$$||u_A||^2_{2,\mu} = \sum_{i=1}^n \mu_i (\mathbb{1}_A)^2 + \sum_{i=1}^n \mu_i \frac{\mu(A)^2}{\mu(V)^2} - 2 \sum_{i=1}^n \mu_i \frac{\mu(A)}{\mu(V)} (\mathbb{1}_A) =$$

$$= \frac{\mu(A)\mu(V) + \mu(A)^2 - 2\mu(A)^2}{\mu(V)} = \frac{\mu(A)(\mu(V) + \mu(A))}{\mu(V)} = \frac{\mu(\bar{A})\mu(A)}{\mu(V)}.$$

Moreover

$$r_M(u_A) = \frac{\langle u_A, M u_A \rangle_\mu}{||u_A||^2_{2,\mu}} = \frac{\mu(V)}{\mu(\bar{A})\mu(A)} \sum_{i=1}^n \mu_i \big(\mathbb{1}_A - \frac{\mu(A)}{\mu(V)} \mathbb{1}\big)_i \big(M(\mathbb{1}_A - \frac{\mu(A)}{\mu(V)} \mathbb{1})\big)_i =$$

$$= \frac{\mu(V)}{\mu(\bar{A})\mu(A)} \sum_{\substack{j \in A \\ 1 \le i \le n}} \left((\mathbb{1}_A)_i \omega(ij) - (\mathbb{1}_A)_i \frac{d_i \mathrm{vol}(A)}{\mathrm{vol}(V)} - \frac{\mu(A)}{\mu(V)} \omega(ij) + \frac{\mu(A)}{\mu(V)} \frac{d_i \mathrm{vol}(A)}{\mathrm{vol}(V)} \right) =$$

$$= \frac{\mu(V)}{\mu(\bar{A})\mu(A)} \left(\sum_{\substack{j \in A \\ 1 \le i \le n}} \omega(ij) - \sum_{i \in A} \frac{d_i \mathrm{vol}(A)}{\mathrm{vol}(V)} - \frac{\mu(A)}{\mu(V)} \mathrm{vol}(A) + \frac{\mu(A)}{\mu(V)} \frac{\mathrm{vol}(A)\mathrm{vol}(V)}{\mathrm{vol}(V)} \right) =$$

$$= \frac{\mu(V)}{\mu(\bar{A})\mu(A)}\left(\omega(E(A)) - \frac{\text{vol}(A)^2}{\text{vol}(V)}\right) = \frac{\mu(V)}{\mu(\bar{A})\mu(A)}Q(A) = q_\mu(A,\bar{A}).$$

It is easy to observe that $\langle u_A, \mathbb{1}\rangle_\mu = 0$, so

$$q_\mu(G) = \max_{A\subseteq V} \frac{\mu(V)Q(A)}{\mu(A)\mu(\bar{A})} = \max_{A\subseteq V} r_M(u_A) = \max_{\substack{x\in\{-a,b\}^n \\ \langle x,\mathbb{1}\rangle_\mu=0}} r_M(x), \tag{2.9}$$

where $a, b \geq 0$.
As M has a positive eigenvalue, dropping the binary constraint $x \in \{-a, b\}^n$ and recalling that $\mathbb{1} \in \ker(M)$, we get

$$\max_{\substack{x\in\mathbb{R}^n \\ \langle x,\mathbb{1}\rangle_\mu=0}} r_M(x) = \lambda_1.$$

□

On the other hand, a lower bound fo $q(G)$ can be defined as follow. Notice that maximizing $Q(A)$ is equal to maximize the right-hand side of (2.6), the main idea is to choose v_A parallel to the corrisponding eigenvector w_1: this would increase the sum. But the index vector cannot be perfectly parallel to w_1 by construction, because of the binary constraint. The best choice is to match the sign of components. So, one can set

$$(v)_i = \begin{cases} 1, & \text{if } (w_1)_i > 0, \\ -1, & \text{if } (w_1)_i < 0. \end{cases}$$

So, using the identities (2.8) and that $||x||^2_{2,\mu} = \mu(V)$ for $x \in \{-1, 1\}^n$, we have

$$\frac{1}{2\mu(V)} \langle v, Mv\rangle_\mu \leq \frac{1}{2\mu(V)} \max_{x\in\{-1,1\}^n} \langle x, Mx\rangle_\mu = q(G).$$

2.2 Nonlinear modularity operator

In this work we use the spectral method just as a starting point. By following the studies of [11], now we explain how to obtain a nonlinear modularity operator $\mathcal{M}$ through a generalization of M that gives us a tight relaxation of $q(G)$ and $q_\mu(G)$. We discover that certain eigenvalues of $\mathcal{M}$ coincide with the graph modularities. Later we will show four algorithm to compute an approximation of those eigenvalues. For the nonlinear modularity operator we need to introduce the Clarke subdifferential Φ of the modulus function $t \to |t|$. The absolute value is not differentiable at zero, so Φ is set valued and defined by

$$\Phi(t) = \begin{cases} 1, & \text{if } t > 0, \\ [-1, 1], & \text{if } t = 0, \\ -1, & \text{if } t < 0. \end{cases}$$

When $x \in \mathbb{R}^n$ the notation $\Phi(x)$ stands for the operator that applies Φ entrywise to x. For more information about the Clarke subdifferential we refer to [16]. In order to define the nonlinear modularity function, we need to start with the following rewriting of M, that exploits the fact that $\mathbb{1} \in \ker(M)$. For any $i = 1, \dots, n$ we have,

$$\big(Mx\big)_i = \sum_{j=1}^{n} M_{ij}x_j - x_i \sum_{j=1}^{n} M_{ij} = \sum_{j=1}^{n}(-M_{ij})(x_i - x_j).$$

Further, using the symmetry of the matrix M, notice that

$$\sum_{i,j=1}^{n} (-M)_{ij}|x_i - x_j|^2 = \sum_{i,j=1}^{n} (-M)_{ij}(x_i^2 - 2x_ix_j + x_j^2) =$$

$$= \sum_{i,j=1}^{n} (-M)_{ij}(x_i^2 - 2x_ix_j) + \sum_{i,j=1}^{n} (-M)_{ij}x_j^2 = \sum_{i,j=1}^{n} (-M)_{ij}(x_i^2 - 2x_ix_j) + \sum_{i,j=1}^{n} (-M)_{ij}x_i^2 =$$

$$= \sum_{i,j=1}^{n} (-M)_{ij}(2x_i^2 - 2x_ix_j) = 2\sum_{i,j=1}^{n} (-M)_{ij}x_i(x_i - x_j).$$

Combining the last equalities gives us the following identity:

$$\langle x, Mx\rangle_\mu = \sum_{i,j=1}^{n} \mu_i(-M)_{ij}x_i(x_i - x_j) = \sum_{i,j=1}^{n} \mu_i(-M)_{ij}|x_i - x_j|^2, \tag{2.10}$$

for any $x \in \mathbb{R}^n$.
We define the nonlinear modularity operator as follows:

$$\mathcal{M}(x)_i = \sum_{j=1}^{n}(-M)_{ij}\Phi(x_i - x_j), \qquad i = 1, \dots, n, \tag{2.11}$$

as a consequence $\mathcal{M}$ identifies the set vectors of $\mathbb{R}^n$ such that

$$\langle x, y\rangle_\mu = \frac{1}{2}\sum_{i,j=1}^{n} \mu_i(-M)_{ij}|x_i - x_j|, \qquad \forall y \in \mathcal{M}(x). \tag{2.12}$$

From now we write $\langle x, \mathcal{M}(x)\rangle_\mu$ to denote the quantity above. Then we introduce the following two Rayleigh quotients associated with $\mathcal{M}(x)$,

$$r_{\mathcal{M}}(x) = \frac{\langle x, \mathcal{M}(x)\rangle_\mu}{||x||_{1,\mu}}, \qquad r^*_{\mathcal{M}}(x) = \frac{\langle x, \mathcal{M}(x)\rangle_\mu}{||x||_\infty}, \tag{2.13}$$

where $||x||_{1,\mu} = \sum_{i=1}^{n} \mu_i|x_i|$ and $||x||_\infty = \max_i |x_i|$. First of all, we show briefly why $r_{\mathcal{M}}$ and $r^*_{\mathcal{M}}$ are the nonlinear generalization of r_M. For simplicity, we fix $\mu_i = 1$, for $i = 1, \dots, n$. Therefore the modularity matrix becomes $(-M)_{ij} = \frac{d_id_j}{\mathrm{vol}(V)} - \omega(ij)$ for all $i, j = 1, \dots, n$. Given the graph $G = (V, E)$, consider the linear difference operator

$B : \mathbb{R}^n \to \mathbb{R}^{|E|}$ entrywise defined by $(Bx)_{ij} = x_i - x_j$, $ij \in E$. And let $\omega_M : E \to \mathbb{R}$ be the real valued function $\omega_M(ij) = \frac{(-M)_{ij}}{2}$. So we can write

$$\langle x, Mx\rangle_\mu = \langle Bx, Bx\rangle_{\omega_M} = ||Bx||^2_{2,\omega_M} = \sum_{ij\in E} \omega_M(ij)(Bx)^2_{ij},$$

where we use the compact notation $|| \cdot ||_{2,\omega_M}$, even tought that quantity is not a norm on $\mathbb{R}^{|E|}$, as ω_M attains positive and negative values. We have as a consequence

$$r_M(x) = \left(\frac{||Bx||_{2,\omega_M}}{||x||_{2,\mu}}\right)^2.$$

A natural generalization of such quantity is therefore given by

$$r_p(x) = \left(\frac{||Bx||_{p,\omega_M}}{||x||_{p,\mu}}\right)^p,$$

where, for $p \geq 1$ and $z \in \mathbb{R}^{|E|}$, we are using the notation $||z||^p_{p,\omega_M} = \sum_{ij} \omega_M(ij)|z_{ij}|^p$. Clearly r_M is retrivied from r_p for $p = 2$. Now, let p^* be the Hölder conjugate of p, that is the solution of the equation $\frac{1}{p} + \frac{1}{p^*} = 1$. Notice that $2^* = 2$, so $r_M(x) = r_2(x) = r_{2^*}(x)$: primal and dual quantities are the same. The Rayleigh quantities in (2.13) are obtained with $p = 1$ for $r_\mathcal{M}$ and $p^* = \infty$ for $r^*_\mathcal{M}$. Another interesting point about these definitions is that the optimality conditions for $r_\mathcal{M}$ and $r^*_\mathcal{M}$ are related to a notion of eigenvalues for the nonlinear modularity operator $\mathcal{M}$. Let Ψ be the set-valued map $\Psi(x) = \{\sigma_1 \mathbb{1}_{m_1}, \dots, \sigma_k \mathbb{1}_{m_k}\}$, where, for $i = 1, \dots, k$, $1 \leq k \leq n$, $m_i = \arg\max_j |x_j|$ and $\sigma_1 = \mathrm{sgn}(x_{m_i})$. Notice that Ψ is the Clarke differential of $||x||_\infty$. We have the following result.

Proposition 2.2.1. Let x be a critical point of $r_\mathcal{M}$, then x is such that $0 \in \mathcal{M}(x) - \lambda\Phi(x)$ with $\lambda = r_\mathcal{M}(x)$. Similarly, if x is a critical point of $r^*_\mathcal{M}$, then $0 \in \mathcal{M}(x) - \lambda\Psi(x)$ with $\lambda = r^*_\mathcal{M}(x)$.

Proof. Let ∂ be the Clarke generalized derivative, thus note that $\partial||x||_{1,\mu} = D_\mu\Phi(x)$. Using the chain rule for ∂ we have

$$\partial r_\mathcal{M}(x) \subseteq \frac{1}{||x||^2_{1,\mu}}\{||x||_{1,\mu}\partial\langle x, \mathcal{M}(x)\rangle_\mu - \langle x, \mathcal{M}(x)\rangle_\mu \partial||x||_{1,\mu}\} =$$

$$\frac{1}{||x||_{1,\mu}}\{D_\mu\mathcal{M}(x) - r_\mathcal{M}(x)D_\mu\Phi(x)\}.$$

□

For further detail we refer to [16]. Thus critical points and critical values of $r_\mathcal{M}$ and $r^*_\mathcal{M}$ satisfy generalized eigenvalue equations for $\mathcal{M}$. The number of eigenvalues of $\mathcal{M}$ defined by means of the Rayleigh quotients in (2.13) is much larger than just the set of varational ones. However, recall that in the linear case the eigenvalues of M coincide with the varational values of r_M.

2.2.1 Exact relaxation

Consider the dominant eigenvalues of $\mathcal{M}$, corresponding to suitable variational values of $r_{\mathcal{M}}$ and $r^*_{\mathcal{M}}$. Here we want to discuss how to use these eigenvalues to locate a leading module in the network by means of a nonlinear spectral method. The task of multiple community detection can also be addressed by successive bipartitions. As already observed the relaxation coming from (2.8) and (2.9) shows that the leading eigenvalue of the modularity matrix is an upper bound for the quantities we want to compute. Instead, in what follows we prove that the quantities

$$\lambda_1(r^*_{\mathcal{M}}) = \max_{x\in\mathbb{R}^n} r^*_{\mathcal{M}}(x), \qquad \lambda_1^{\perp}(r_{\mathcal{M}}) = \max_{\substack{x\in\mathbb{R}^n\\ \langle x,\mathbb{1}\rangle_\mu=0}} r_{\mathcal{M}}(x) \tag{2.14}$$

coincide with the cut-modularities of the graph, $q(G)$ and $q_\mu(G)$ respecively. So, we can localize the best bi-partitioning of G into communities according to the sign of the entries in the vector achieving the maxima in (2.14). For the case of $q(G)$ we use the Lovász extension of a set valued function. We recall its definition

Definition 2. Given the set of vertices V, let $\mathcal{P}(V)$ be the power set of V, and consider a function $F:\mathcal{P}(V)\to\mathbb{R}$. For a given vector $x\in\mathbb{R}^n$ let σ be the permutation such that $x_{\sigma(1)}\le x_{\sigma(2)}\le\cdots\le x_{\sigma(n)}$ and let $C_i(x)\subseteq V$ be the set

$$C_i(x)=\{k\in V : x_{\sigma(k)}\ge x_{\sigma(i)}\}$$

The Lovász extension $f_F:\mathbb{R}^n\to\mathbb{R}$ is defined by

$$f_F(x)=\sum_{i=1}^{n-1}F(C_{i+1}(x))(x_{\sigma(i+1)}-x_{\sigma(i)})+F(V)x_{\sigma(1)}.$$

An important property of the Lovász extension of $F:\mathcal{P}(V)\to\mathbb{R}$, is that for any $A\subseteq V$ it holds $F(A)=f_F(\mathbb{1}_A)$. We have:

Lemma 2.2.2. Let $F,H:\mathcal{P}(V)\to\mathbb{R}$ be the set valued functions such that $0<H(A)\le 1$ for all $A\subseteq V$ such that $A\notin\{\emptyset,V\}$. If $F(V)=0$, then

$$\max_{A\subseteq V}\frac{F(A)}{H(A)}\ge\frac{1}{2}\max_{||x||_\infty\le 1} f_F(x).$$

Proof. Suppose w.l.o.g. that the entries of $x\in\mathbb{R}^n$ are labeled in ascending order. So

$$f_F(x)=\sum_{i=1}^{n-1}F(C_{i+1}(x))(x_{i+1}-x_i)=\sum_{i=1}^{n-1}\frac{F(C_{i+1}(x))}{H(C_{i+1}(x))}H(C_{i+1}(x))(x_{i+1}-x_i).$$

Since $0<H(C_{i+1}(x))\le 1$ and $(x_{i+1}-x_i)\ge 0$ we get

$$f_F(x)\le\max_{i=2,\dots,n}\frac{F(C_i(x))}{H(C_i(x))}(x_n-x_i)\le\left(\max_{i=1,\dots,n}\frac{F(C_i(x))}{H(C_i(x))}\right)2||x||_\infty.$$

Since $\{C_i(x) : ||x||_\infty \leq 1\} = \mathcal{P}(V)$, we can conclude

$$\max_{||x||_\infty \leq 1} f_F(x) \leq 2 \max_{||x||_\infty \leq 1} \max_{i=1,\dots,n} \frac{F(C_i(x))}{H(C_i(x))} = 2 \max_{A \subseteq V} \frac{F(A)}{H(A)}$$

□

Thanks to Lemma (2.2.2) we can show that the nonlinear relaxation of $q(G)$ is optimal.

Theorem 2.2.3. *Let $r^*_{\mathcal{M}}$ be the dual Rayleigh quotient defined in (2.13) and let $\lambda_1(r^*_{\mathcal{M}}) = \max_{x \in \mathbb{R}^n} r^*_{\mathcal{M}}(x)$. Then*

$$q(G) = \max_{A \subseteq V} q(A) = \frac{\lambda_1(r^*_{\mathcal{M}})}{\mu(V)}$$

Proof. Consider a subset $A \subseteq V$ and the vector $v_A = \mathbb{1}_A - \mathbb{1}_{\bar{A}}$. Remember that

$$\langle v_A, M(v_A) \rangle_\mu = \frac{1}{2} \sum_{i,j=1}^n \mu_i (-M)_{ij} |(v_A)_i - (v_A)_j|^2 = 4Q(A)$$

and notice that $||v_A||_\infty = 1$ and

$$|(v_A)_i - (v_A)_j| = \begin{cases} 0, & \text{if } i,j \in A \text{ or } i,j \notin A \\ 2, & \text{otherwise.} \end{cases}$$

We get that

$$\langle v_A, M(v_A) \rangle_\mu = \frac{1}{2} \sum_{i,j=1}^n \mu_i (-M)_{ij} |(v_A)_i - (v_A)_j| = 2Q(A).$$

Therefore $r_{\mathcal{M}}(v_A) = 2Q(A)$ and

$$\mu(V) q(G) = \max_{A \subseteq V} r_{\mathcal{M}}(v_A) \leq \max_{x \in \mathbb{R}^n} r_{\mathcal{M}}(x).$$

To show the reverse inequality we use Lemma (2.2.2). Given a graph $G = (V, E)$ let W_G denote its weight matrix, and let $\mathrm{cut}_G(A) = \omega(E(A, \bar{A}))$. Then the modularity $Q(A)$ coincides with $Q(A) = (\mathrm{cut}_{K_0}(A) - \mathrm{cut}_G(A))/2$, where $K_0 = (V, V \times V)$ is the complete graph with edge matrix $(W_{K_0})_{ij} = \frac{d_i d_j}{\mathrm{vol}(V)}$. The Lovász extension of cut_G is $f_{\mathrm{cut}_G}(x) = \sum_{i,j=1}^n (W_G)_{ij} |x_i - x_j|$, thus the Lovász extension of $Q(A)$ is

$$f_Q(x) = f_{\frac{1}{2}(\mathrm{cut}_{K_0} - \mathrm{cut}_G)}(x) = \frac{1}{2}(f_{\mathrm{cut}_{K_0}}(x) - \mathrm{cut}_G(x)) =$$

$$= \frac{1}{2}\Big(\sum_{i,j=1}^n (W_{K_0})_{ij} |x_i - x_j| - \sum_{i,j=1}^n (W_G)_{ij} |x_i - x_j| \Big) = \langle x, \mathcal{M}(x) \rangle_\mu ,$$

due to the linearity of the extension. For more information and the calculi [17]. Let $H : \mathcal{P}(V) \to \mathbb{R}$ be the constant function $H(A) = 1$. As $Q(V) = 0$, we can use such H into Lemma (2.2.2) with $F = H$ to get

$$\max_{A\subseteq V} Q(A) \geq \frac{1}{2} \max_{||x||_\infty \leq 1} \langle x, \mathcal{M}(x)\rangle_\mu .$$

Notice that f_Q is positively one-homogeneous, that is $f_Q(\alpha x) = \alpha f_Q(x)$, for any $\alpha \geq 0$, so we get

$$\frac{1}{2} \max_{||x||_\infty \leq 1} \langle x, \mathcal{M}(x)\rangle_\mu \geq \frac{1}{2} \max_{x\in\mathbb{R}^n} \langle \frac{x}{||x||_\infty}, \mathcal{M}(\frac{x}{||x||_\infty})\rangle_\mu = \frac{1}{2} \max_{x\in\mathbb{R}^n} r^*_{\mathcal{M}}(x).$$

So we have both inequalities and using the identity $q(A) = \frac{2Q(A)}{\mu(V)}$, we conclude. $\square$

Now it is the turn of $q_\mu(G)$ and $r_{\mathcal{M}}$. Again for the main result we need a preparing Lemma. This time we omit the proof that can be found in [18].

Lemma 2.2.4. A function $f : \mathbb{R}^n \to \mathbb{R}$ is positively one-homogeneous, even, convex and $f(x + y) = f(x)$ for any $y \in \text{span}(\mathbb{1})$ if and only if exists $\mu : V \to \mathbb{R}_{\geq 0}$ such that $f(x) = \sup_{y\in Y} \langle x, y\rangle_\mu$ where Y is a closed symmetric convex set such that $\langle y, \mathbb{1}\rangle_\mu = 0$ for any $y \in Y$.

For the next theorem we introduce the function $\nu : \mathcal{P}(V) \to \mathbb{R}$ such that $\nu(A) = \frac{\mu(A)\mu(\bar{A})}{\mu(V)}$. Then $q_\mu(A) = \frac{Q(A)}{\nu(A)}$. Moreover, let $w_A = \mathbb{1}_A - \frac{\mu(A)}{\mu(V)}\mathbb{1}$. As in proposition (2.1.1), we have $||w_A||_{1,\mu} = 2\nu(A)$ and as in the proof of the theorem (2.2.3), $\langle w_A, \mathcal{M}(w_A)\rangle_\mu = \langle \mathbb{1}_A, \mathcal{M}(\mathbb{1}_A)\rangle_\mu = Q(A)$. Thus $r_{\mathcal{M}}(w_A) = \frac{q_\mu(A)}{2}$ and $q_\mu(G) = 2 \max_{\substack{x\in\{-a,b\}^n \\ \langle x,1\rangle_\mu=0}} r_{\mathcal{M}}(x)$.

Theorem 2.2.5. *Let $r_{\mathcal{M}}$ be the dual Rayleigh quotient defined in (2.13) and let $\lambda_1^\perp(r_{\mathcal{M}}) = \max_{\substack{x\in\mathbb{R}^n \\ \langle x,1\rangle_\mu=0}} r_{\mathcal{M}}(x)$. Then*

$$q_\mu(G) = \max_{A\subseteq V} q_\mu(A) = 2\lambda_1^\perp(r_{\mathcal{M}}).$$

Proof. For $x \in \mathbb{R}^n$ and $t > 0$ consider the level set $A_x^t = \{i \in V : x_i > t\}$. Let $x_{\min} = \min_i x_i$ and $x_{\max} = \max_i x_i$. We have

$$\langle x, \mathcal{M}(x)\rangle_\mu = \sum_{x_i>x_j} \mu_i(-M)_{ij} \int_{x_j}^{x_i} dt = \int_{x_{\min}}^{x_{\max}} \sum_{x_i>t\geq x_j} \mu_i(-M)_{ij}\, dt = \int_{x_{\min}}^{x_{\max}} Q(A_x^t)\, dt \leq$$

$$\leq \left\{ \max_t \frac{Q(A_x^t)}{2\nu(A_x^t)} \right\} \int_{x_{\min}}^{x_{\max}} 2\nu(A_x^t)\, dt = \left\{ \max_t \frac{Q(A_x^t)}{2\nu(A_x^t)} \right\} \underbrace{\int_{x_{\min}}^{x_{\max}} ||w_{A_x^t}||_{1,\mu}\, dt}_{(*)}$$

Now we study the $(*)$ term using Lemma (2.2.4). Let $P : \mathbb{R}^n \to \mathbb{R}^n$ be the orthgonal projection onto $\{x : \langle x, \mathbb{1}\rangle_\mu = 0\}$. Then $P(x) = x - \frac{\langle x,\mathbb{1}\rangle_\mu}{\mu(V)}\mathbb{1}$. Consider the function

$f(x) = ||P(x)||_{1,\mu}$. Note that f satisfies all the hypothesis of Lemma (2.2.4). Moreover note that $f(\mathbb{1}_{\mathbb{A}}) = ||w_A||_{1,\mu}$ for any $A \subseteq V$. Thus there exists $Y \subseteq \text{range}(P)$ such that

$$\int_{x_{\min}}^{x_{\max}} f(\mathbb{1}_{A_x^t})\, dt = \int_{x_{\min}}^{x_{\max}} ||w_{A_x^t}||_{1,\mu}\, dt = \sup_{y \in Y} \int_{x_{\min}}^{x_{\max}} \langle \mathbb{1}_{A_x^t}, y \rangle_\mu \, dt.$$

Assume w.l.o.g. that x is ordered so that $x_1 \leq \dots \leq x_n$. The function $\phi(t) = \langle \mathbb{1}_{A_x^t}, y \rangle_\mu$ is constant on the intervals $[x_i, x_{i+1}]$. Letting $A_i = A_x^{x_i}$ we have

$$\int_{x_{\min}}^{x_{\max}} \phi(t)\, dt = \sum_{i=1}^{n-1} (x_{i+1} - x_i) \langle \mathbb{1}_{A_i}, y \rangle_\mu = \sum_{i=1}^{n} x_i \langle \mathbb{1}_{A_{i-1}} - \mathbb{1}_{A_i}, y \rangle_\mu = \langle x, y \rangle_\mu \,,$$

thus $||P(x)||_{1,\mu} = f(x) = \int_{x_{\min}}^{x_{\max}} f(\mathbb{1}_{A_x^t})\, dt$, our $(*)$ term. Denote by A_x^* the set that attains the maximum $\max_t \frac{Q(A_x^t)}{\nu(A_x^t)}$. As $\langle P(x), \mathcal{M}(P(x)) \rangle_\mu = \langle x, \mathcal{M}(x) \rangle_\mu$, all together we have

$$\lambda_1^\perp(r_\mathcal{M}) = \max_{\substack{x \in \mathbb{R}^n \\ \langle x, 1 \rangle_\mu = 0}} \frac{\langle x, \mathcal{M}(x) \rangle_\mu}{||x||_{1,\mu}} = \max_{x \in \mathbb{R}^n} \frac{\langle P(x), \mathcal{M}(P(x)) \rangle_\mu}{||P(x)||_{1,\mu}} = \max_{x \in \mathbb{R}^n} \frac{\langle x, \mathcal{M}(x) \rangle_\mu}{||P(x)||_{1,\mu}} \leq$$

$$\leq \max_{x \in \mathbb{R}^n} \frac{Q(A_x^*)}{\nu(A_x^*)} \underbrace{||P(x)||_{1,\mu}}_{(*)} \frac{1}{||P(x)||_{1,\mu}} = \max_{x \in \mathbb{R}^n} \frac{Q(A_x^*)}{\nu(A_x^*)} \leq \frac{q_\mu(G)}{2}.$$

On the other hand,

$$q_\mu(G) = 2 \max_{\substack{x \in \{-a, b\}^n \\ \langle x, 1 \rangle_\mu = 0}} r_\mathcal{M}(x) \leq 2 \max_{\substack{x \in \mathbb{R}^n \\ \langle x, 1 \rangle_\mu = 0}} r_\mathcal{M}(x) = 2\lambda_1^\perp(r_\mathcal{M}).$$

The statement is proved. $\square$

Chapter 3

Optimization algorithms

Summarising, given a graph G we want to find the best bi-partitioning $q(G)$, (2.4). To obtain it we use the modularity measure that can be defined in terms of the modularity matrix M, (2.6). Now the problem is that the best bi-partitioning is a constrained maximization problem, very difficult to solve, cf. the problem gives in (2.8). One can easily have a relaxation of $q(A)$ calculating the largest eigenvalue of M but it is just an upper bound, Proposition 2.1.1. Moving from the linear operator to the nonlinear modularity, (2.13), we have obtained an exact relaxation of $q(G)$, Theorem 2.2.3. We have seen that the normalized version $q_\mu(G)$ has similar conclusions. In what follows we study the classic modularity $q(G)$, it can be easily change with $q_\mu(G)$ with few differences. This section want to show four feasible algorithms that allows us to find a maximum of $q(G)$. The algorithms are the classical ones Frank-Wolfe algorithm and projected gradient algorithm and their non monotone versions. All of them are iterative and feasible algorithms, this means that every point generated from the iterations of the algorithm has to belong to the feasible set. Moreover they use information of the first order, so we can optimize a function $f : \mathbb{R}^n \to \mathbb{R}$ on the compact & convex feasible set $S \subset \mathbb{R}^n$ if $f \in \mathcal{C}^1(S)$. To apply these algorithms to our problem (Theorem 2.2.3) we have to modify the function so that it becomes continuously differentiable and choose a suitable feasible set, over we are looking for the maximum, to get a compact & convex set. For the second task we have the following results.

Proposition 3.0.1. Let $q(G)$ be the best bi-partitioning defined in (2.4) and let $\mathcal{M}(x)$ the nonlinear modularity operator defined in (2.11). Then

$$q(G) = \max_{A \subseteq V} q(A) = \frac{1}{\mu(V)} \max_{\substack{x \in \mathbb{R}^n \\ ||x||_\infty \leq 1}} \langle x, \mathcal{M}(x) \rangle_\mu . \tag{3.1}$$

Proof. We have proved in Theorem 2.2.3 that

$$q(G) = \max_{A \subseteq V} q(A) = \frac{\lambda_1(r^*_{\mathcal{M}})}{\mu(V)},$$

where $r^*_{\mathcal{M}}$ is the dual Rayleigh quotient defined in (2.13) and $\lambda_1(r^*_{\mathcal{M}}) = \max_{x \in \mathbb{R}^n} r^*_{\mathcal{M}}(x)$. Moreover we have noticed that $f_Q(x) = \langle x, \mathcal{M}(x) \rangle_\mu$ is one-homogeneous, that is $f_Q(\alpha x) =$

$\alpha f_Q(x)$, for any $\alpha \geq 0$. Thus

$$\max_{x\in\mathbb{R}^n} r^*_{\mathcal{M}}(x) = \overbrace{\max_{x\in\mathbb{R}^n} \frac{\langle x, \mathcal{M}(x)\rangle_\mu}{||x||_\infty}}^{C} = \max_{x\in\mathbb{R}^n} \frac{f_Q(x)}{||x||_\infty} =$$

$$= \max_{x\in\mathbb{R}^n} f_Q\big(\frac{x}{||x||_\infty}\big) = \max_{||x||_\infty=1} f_Q(x) = \underbrace{\max_{||x||_\infty=1} \langle x, \mathcal{M}(x)\rangle_\mu}_{B}$$

Now, obviuosly we have

$$\underbrace{\max_{||x||_\infty=1} \langle x, \mathcal{M}(x)\rangle_\mu}_{B} \leq \underbrace{\max_{||x||_\infty\leq 1} \langle x, \mathcal{M}(x)\rangle_\mu}_{A}$$

and

$$\underbrace{\max_{x\in\mathbb{R}^n} \frac{\langle x, \mathcal{M}(x)\rangle_\mu}{||x||_\infty}}_{C} \geq \underbrace{\max_{||x||_\infty\leq 1} \frac{\langle x, \mathcal{M}(x)\rangle_\mu}{||x||_\infty}}_{D} \geq \underbrace{\max_{||x||_\infty\leq 1} \langle x, \mathcal{M}(x)\rangle_\mu}_{A}.$$

Combining all together we have

$$A \geq B = C \geq D \geq A,$$

so they are all identities and we have the statement. □

A similar proof allows us to write the next Proposition for the normalized version $q_\mu(G)$.

Proposition 3.0.2. Let $q_\mu(G)$ be the best bi-partitioning defined in (2.4) and let $\mathcal{M}(x)$ the nonlinear modularity operator defined in (2.11). Then

$$q_\mu(G) = \max_{A\subseteq V} q_\mu(A) = 2 \max_{\substack{||x||_{1,\mu}\leq 1 \\ \langle x, \mathbb{1}\rangle_\mu = 0}} \langle x, \mathcal{M}(x)\rangle_\mu . \tag{3.2}$$

Proof. As in the Proposition (3.0.1), remember the result of the Theorem (2.2.5). □

3.1 Smooth approximation of non-smooth function

The smooth approximation of non-smooth function is a big branch of the nonlinear programming, the main ideas and results one can find in [19] and [20] and references therein. We are not going to handle deeply this topic, it is not the goal of this thesis, we want just to show our reasonable idea. The function that we need to make smooth is (3.1) or (3.2). Remembering its definition (2.12), for any $x \in \mathbb{R}^n$, we have:

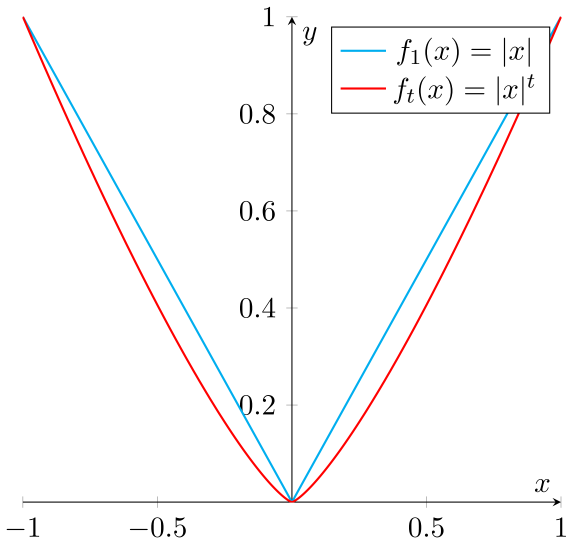
$$f_Q(x) = \langle x, \mathcal{M}(x)\rangle_\mu = \frac{1}{2}\sum_{i,j=1}^{n} \mu_i(-M)_{ij}|x_i - x_j| = f_1(x).$$

We introduce f_1 just to make the notation clearer. Our smooth approximation idea is adding an exponent $t \in \mathbb{R}$, $t > 1$, at each absolute values. What we obtain is

$$f_t(x) = \frac{1}{2} \sum_{i,j=1}^{n} \mu_i (-M)_{ij} |x_i - x_j|^t, \tag{3.3}$$

and obviously now f_t belongs to the family of $\mathcal{C}^1(\mathbb{R}^n)$ functions. For t close to 1 the effect is a rounded absolute value as we can easily see in one dimension, Figure 3.1.

Figure 3.1: Function f_1 and its approximation f_t, with $t = 1.3$, in $\mathbb{R}$.



We can expect that $f_t(x)$ is a continuos function of t, that is for any $x \in \mathbb{R}^n$ and for any $\epsilon > 0$ there exists $\delta > 0$ such that, if $t - 1 < \delta$, then $f_t(x) - f_1(x) < \epsilon$. We do not prove this result formally.

3.2 Optimization theory

Now we are going to introduce the fundamental ideas of the optimization theory in order to study our bi-partitioning problem in a mathematical way. An optimization problem is defined as the minimization or maximization of a real function over a given set. In general we have the following form:

$$\min_{x \in \mathcal{S}} f(x) \tag{3.4}$$

where $f : \mathcal{S} \to \mathbb{R}$ is the objective function and $\mathcal{S}$ is the feasible set. Here we give some definitions.

Definition 3. The problem (3.4) is said to be infeasible if $\mathcal{S} = \{\emptyset\}$, i.e. there are no feasible solutions.

Definition 4. The problem (3.4) is said to be unlimited (from below) if for any $M > 0$ there exists $x_M \in \mathcal{S}$ such that $f(x_M) < -M$.

Definition 5. It is said that the problem (3.4) admits optimal (finite) solution if there exist $x^* \in \mathcal{S}$ such that $f(x^*) \leq f(x)$ for any $x \in \mathcal{S}$. $f(x^*)$ is said optimal value.

Notice that we can always handle minimization problems, since a maximum point of the problem

$$\max_{x\in\mathcal{S}} f(x),$$

it is a point $x^* \in \mathcal{S}$ where, by definition, we have

$$f(x^*) \geq f(x), \quad \forall\, x \in \mathcal{S}.$$

This is equivalent to:

$$-f(x^*) \leq -f(x), \quad \forall\, x \in \mathcal{S},$$

so x^* is a minimum point of

$$\min_{x\in\mathcal{S}} -f(x).$$

We get

$$\max_{x\in\mathcal{S}} f(x) = -\min_{x\in\mathcal{S}}(-f(x)). \tag{3.5}$$

So w.l.o.g. we are giving definitions and algorithms for a minimization problem. Moreover we consider $\mathcal{S} \subseteq \mathbb{R}^n$. We give the definition of minimum points.

Definition 6. A point $x^* \in \mathcal{S}$ is said to be local minimum of f on $\mathcal{S}$ if there exists a neighborhood $B(x^*;\rho) \subset \mathbb{R}^n$, with $\rho > 0$ such that:

$$f(x^*) \leq f(x), \quad \forall\, x \in \mathcal{S} \cap B(x^*;\rho).$$

Definition 7. A point $x^* \in \mathcal{S}$ is said to be strict local minimum of f on $\mathcal{S}$ if there exists a neighborhood $B(x^*;\rho)$, with $\rho > 0$ such that:

$$f(x^*) < f(x), \quad \forall\, x \in \mathcal{S} \cap B(x^*;\rho).$$

Definition 8. A point $x^* \in \mathcal{S}$ is said to be global minimum of f on $\mathcal{S}$ if

$$f(x^*) \leq f(x), \quad \forall\, x \in \mathcal{S},\, x \neq x^*.$$

Definition 9. A point $x^* \in \mathcal{S}$ is said to be strict global minimum of f on $\mathcal{S}$ if

$$f(x^*) < f(x), \quad \forall\, x \in \mathcal{S},\, x \neq x^*.$$

Since we have proved propositions 3.0.1 and 3.0.2, we are interested in optimizing a function on a convex set. We can easily check that both $\{x \in \mathbb{R}^n : ||x||_\infty \leq 1\}$ and $\{x \in \mathbb{R}^n : ||x||_{1,\mu} \leq 1, \langle x, \mathbb{1}\rangle_\mu = 0\}$ are convex sets. Before of introducing the first algorithm, we remember the definition of feasible direction set:

Definition 10. Let $\mathcal{S} \subseteq \mathbb{R}^n$ be a nonempty set. It is defined the set of feasible directions of $\mathcal{S}$ in $\bar{x} \in \mathcal{S}$ the following set $S(\bar{x})$:

$$S(\bar{x}) = \{d \in \mathbb{R}^n, d \neq \underline{0} : \exists\, \delta > 0 \text{ such that } \bar{x} + \alpha d \in \mathcal{S}, \forall\, \alpha \in (0, \delta)\},$$

where $\underline{0}$ is the null vector of $\mathbb{R}^n$.

Moreover we get for the particular case of convex feasible set,

Proposition 3.2.1. *Let $\mathcal{S} \subseteq \mathbb{R}^n$ be convex set and $\bar{x} \in \mathcal{S}$. If $\mathcal{S} \neq \{\bar{x}\}$, for all $x \in \mathcal{S}$ with $x \neq \bar{x}$, the direction*

$$d = x - \bar{x}$$

is feasible for $\mathcal{S}$ in $\bar{x}$.

If $\mathcal{S} \subseteq \mathbb{R}^n$ is convex, we can define the feasible directions set of $\bar{x} \in \mathcal{S}$ in the new following form:

$$S(\bar{x}) = \{d \in \mathbb{R}^n : d = x - \bar{x},\, x \in \mathcal{S},\, x \neq \bar{x}\} \tag{3.6}$$

3.3 Monotone local algorithms

Here we propose two classical monotone algorithms for the minimization of a non linear function. In the next section we are going to discuss their non monotone version.

3.3.1 Frank-Wolfe method

The Frank-Wolfe method is based on the following proposition.

Proposition 3.3.1. *Let $x^* \in \mathcal{S}$ be a minimum local point of the problem*

$$\min_{x \in \mathcal{S}} f(x)$$

with $\mathcal{S} \subseteq \mathbb{R}^n$ convex and $f \in \mathcal{C}^1(\mathbb{R}^n)$. Then

- *$\nabla f(x^*)^T (x - x^*) \geq 0$, for all $x \in \mathcal{S}$. In this case we call x^* a critical point.*
- *Moreover if $f \in \mathcal{C}^2(\mathbb{R}^n)$, for any $x \in \mathcal{S}$ such that $\nabla f(x^*)^T (x - x^*) = 0$, we have:*

$$(x - x^*)^T \nabla^2 f(x^*)(x - x^*) \geq 0.$$

We have noticed that our objective functions, defined in (2.13), are both not differentiable. But we are going to use the approximation that we have showed in Section 3.1, with the equation (3.3). Furthermore the two Rayleigh quotients that we want to maximize are determined by the nonlinear modularity operator (2.11), so instead of $\max r_{\mathcal{M}}(x)$ and $\max r^*_{\mathcal{M}}(x)$ we are going to $\min -r_{\mathcal{M}}(x)$ and $\min -r^*_{\mathcal{M}}(x)$ using the equivalence (3.5).

Now we are ready to describe the Frank-Wolfe algorithm, at every iteration this algorithm computes a feasible direction of decrease in x_k by solving the following problem:

$$\min_{x \in \mathcal{S}} \nabla f(x_k)^T (x - x_k). \tag{3.7}$$

If $\mathcal{S}$ is compact, the problem has always a solution $\hat{x}_k \in \mathcal{S}$. If $\nabla f(x_k)^T(\hat{x}_k - x_k) = 0$, then we have

$$0 = \nabla f(x_k)^T(\hat{x}_k - x_k) \leq \nabla f(x_k)^T(x - x_k),$$

for any $x \in \mathcal{S}$ and x_k is a critical point. If $\nabla f(x_k)^T(\hat{x}_k - x_k) < 0$ we can define a new feasible direction of decrease in x_k:

$$d_k = \hat{x}_k - x_k,$$

and determinate a new point

$$x_{k+1} = x_k + \alpha_k d_k,$$

with $\alpha_k \in (0, 1]$ that is a step size chooses by a search line. If the objective function is quite structured we can compute the step size by solving the exact problem

$$\min_{\alpha \in (0,1]} f(x_k + \alpha d_k).$$

But in our case that problem needs too much time to be solved. So we use an approximated line search method, so called Armijo line search that we are seeing below. Notice that can not be $\nabla f(x_k)^T(\hat{x}_k - x_k) > 0$ since (3.7) is a problem of minimum and for $x_k \in \mathcal{S}$ we obtain $\nabla f(x_k)^T(x_k - x_k) = 0$. Here we propose the basic scheme of Frank-Wolfe algorithm:

Algorithm 1 : Frank-Wolfe method.

```
1: Fix an initial point $x_0 \in \mathcal{S}$;
2: for $k = 0, 1, \ldots$ do
3:     Compute $\hat{x}_k \in \mathcal{S}$ solution of
           $\min_{x \in \mathcal{S}} \nabla f(x_k)^T (x - x_k)$;
4:     if $\nabla f(x_k)^T(\hat{x}_k - x_k) = 0$ then return $x_k$;
5:     Compute $\alpha_k > 0$ along $d_k = \hat{x}_k - x_k$ with line search algorithm;
6:     $x_{k+1} \leftarrow x_k + \alpha_k d_k$;
```

Before of studying the Armijo line search, we show the main result about the convergence of Frank-Wolfe algorithm:

Proposition 3.3.2. Consider

$$\min_{x \in \mathcal{S}} f(x)$$

with $f \in \mathcal{C}^1(\mathbb{R}^n)$ and $\mathcal{S} \subseteq \mathbb{R}^n$ compact and convex set. Let $\{x_k\}_{k \in \mathbb{N}}$ be the sequence produced by Frank-Wolfe algorithm with a line search that respects the following conditions:

C1. $x_{k+1} \in \mathcal{S}$;

C2. $f(x_{k+1}) < f(x_k)$;

C3. $\lim_{k\to\infty} \nabla f(x_k)^T d_k = 0$.

Then:

- either there exists $T \geq 0$ such that x_T is a critical point;
- or $\{x_k\}_{k\in\mathbb{N}}$ is a infinite sequence and each limit point is a critical point.

Proof. If the algorithm produces a infinite sequence $\{x_k\}_{k\in\mathbb{N}}$, since $\mathcal{S}$ is compact there exists an limit point $\bar{x} \in \mathcal{S}$. From C3. we have

$$\lim_{k\to\infty} \nabla f(x_k)^T d_k = 0.$$

Moreover,

$$||d_k|| = ||\hat{x}_k - x_k|| \leq ||\hat{x}_k|| + ||x_k||.$$

So d_k is finite for any $k \in \mathbb{N}$. We can define two sub-sequences $\{x_k\}_{k\in K}$ and $\{d_k\}_{k\in K}$, with $K \subset \mathbb{N}$, convergent to $\bar{x}$ and $\bar{d}$ respectively. Thus we have

$$\nabla f(\bar{x})^T \bar{d} = 0.$$

From the definition of d_k,

$$\nabla f(x_k)^T d_k \leq \nabla f(x_k)^T (x - x_k), \quad \forall\, x \in \mathcal{S}.$$

Fix $x \in \mathcal{S}$, considering the limit we get

$$0 = \nabla f(\bar{x})^T \bar{d} \leq \nabla f(\bar{x})^T (x - \bar{x}), \quad \forall\, x \in \mathcal{S},$$

so $\bar{x}$ is a critical point. □

3.3.2 Armijo line search

When the exact line search is too expensive, for instance too many objective function evaluations are required, different rules can be used for the stepsize calculation. The line search algorithm that we use is the one proposed by Armijo in 1966, ref. [21]. We fix the scalars $\delta \in (0, 1)$ and $\gamma \in (0, \frac{1}{2})$ and we set $\alpha = \delta^m$, where m is a non negative integer and we try these steps sequentially untill

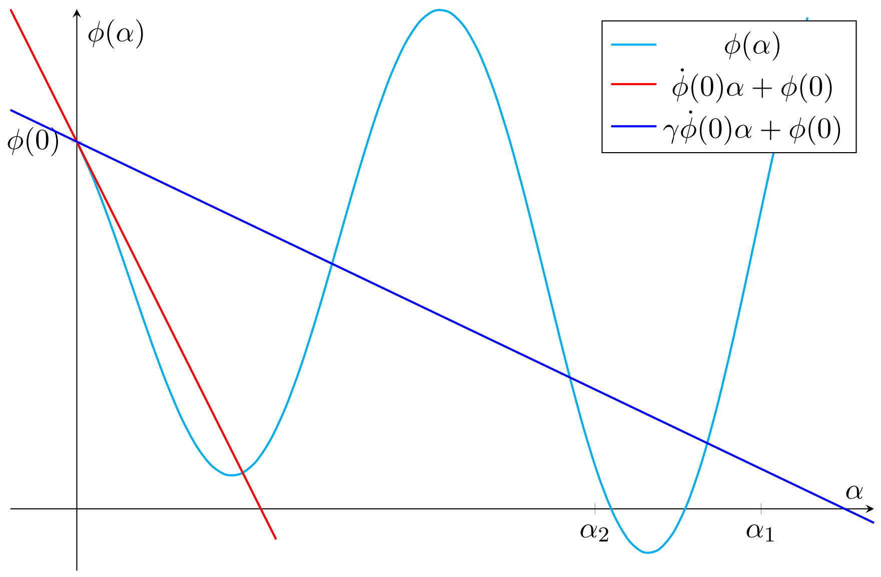
$$f(x_k + \alpha d_k) \leq f(x_k) + \gamma \nabla f(x_k)^T d_k \alpha. \tag{3.8}$$

The inequality is satisfied for $m = m_k$ and so we define $\alpha_k = \delta^{m_k}$. Geometrically it is easy to observe that the condition (3.8) is equivalent to ask that the value of

$$\phi(\alpha_k) = f(x_k + \alpha_k d_k)$$

will be below of the line passes through the point $(0, \phi(0))$ with slope $\gamma\dot{\phi}(0)$.

Figure 3.2: Example of line search with Armijo algorithm.



Actually, $f(x_k) + \nabla f(x_k)^T(x_k + \alpha d_k - x_k)$ is an approximation of the first order with the Taylor expansion of $\phi(\alpha) = f(x_k + \alpha d_k)$ centered in x_k. Moreover we know that $\dot{\phi}(0) < 0$, since d_k is a decrease direction by construction. So multiplying $\nabla f(x_k)^T d_k \alpha$ by $\gamma \in (0, \frac{1}{2})$ we just lift the tangent line in $(0, \phi(0))$. It can be proved that the sequence prodeces by Armijo method satisfies the condition C1, C2, C3 of the proposition 3.3.2.

Algorithm 2 : Monotone Armijo line search.

```
1: Fix $\delta \in (0,1)$ and $\gamma \in (0, 0.5)$;
2: Set $\alpha = 1$ and $j = 0$;
3: while $f(x_k + \alpha d_k) > f(x_k) + \gamma \nabla f(x_k)^T d_k \alpha$ do
4:     $\alpha \leftarrow \delta\alpha$;
5:     $j \leftarrow j + 1$;
6: Set $\alpha_k = \alpha$; return $\alpha_k$;
```

3.3.3 Projected gradient method

We introduce formally the idea of projection on a convex set.

Definition 11. Let $\mathcal{S} \subset \mathbb{R}^n$ be a closed convex set and $\bar{x} \in \mathbb{R}^n$ a given point. We define the projection of $\bar{x}$ on $\mathcal{S}$ the solution $p(\bar{x})$ of the following problem:

$$\min_{x \in \mathcal{S}} \frac{1}{2} ||x - \bar{x}||^2,$$

where $|| \cdot ||$ indicates the euclidean norm.

Here we show some properties about the projection:

Proposition 3.3.3. With the same notation of the Definition (11), we have that

- $y^* \in \mathcal{S}$ is the projection of $\bar{x}$ on $\mathcal{S}$, i.e. $p(\bar{x}) = y^*$, if and only if
$$(\bar{x} - y^*)^T(y - y^*) \leq 0, \quad \forall\, y \in \mathcal{S}.$$
- The projection $p : \mathbb{R}^n \to \mathcal{S}$ is a continuous and no-expansive function, that is:
$$||p(x) - p(y)|| \leq ||x - y||, \quad \forall\, x, y \in \mathbb{R}^n.$$

Proof. We skip the proof, it is enough write the necessary conditions of minimum for the feasible convex set. □

Proposition 3.3.4. Let $x^* \in \mathcal{S}$ be a local minimum point of the problem
$$\min_{x \in \mathcal{S}} f(x),$$
with $\mathcal{S} \subseteq \mathbb{R}^n$ convex and $f \in \mathcal{C}^1(\mathbb{R}^n)$. Then
$$x^* = p(x^* - s\nabla f(x^*)),$$
with $s \geq 0$.

Proof. x^* is a local minimum by hypothesis, so $\nabla f(x^*)^T(y - x^*) \geq 0$, $\forall\, y \in \mathcal{S}$. Multiplying by $-s$, $s \geq 0$ we get
$$-s\nabla f(x^*)^T(y - x^*) \leq 0,$$
and adding and removing $x^{*T}(y - x^*)$ we obtain
$$(\underbrace{x^* - s\nabla f(x^*)}_{\bar{x} \text{ of Prop. 3.3.3}} - x^*)(y - x^*) \leq 0, \quad \forall\, y \in \mathcal{S}.$$
So $x^* = p(x^* - s\nabla f(x^*))$. □

Now we can describe the projected gradient algorithm that, at every step, computes a descent direction in x_k using
$$\hat{x}_k = p(x_k - s\nabla f(x_k)),$$
so we define
$$d_k = \hat{x}_k - x_k$$
and obtain a new point
$$x_{k+1} = x_k + \alpha_k d_k,$$
with $\alpha_k \in (0, 1]$ found by a line search as Armijo, Algorithm 2, and $s \geq 0$. In the following the projected gradient pseudocode:

Also for the projected gradient method we show the result about its convergence.

Algorithm 3 : Projected gradient method.

```
1: Choose an initial point $x_0 \in \mathcal{S}$ and $s \geq 0$;
2: for $k = 0, 1, \ldots$ do
3:     Compute $\hat{x}_k = p(x_k - s\nabla f(x_k))$;
4:     if $\hat{x}_k = x_k$ then return $x_k$;
5:     Compute $\alpha_k > 0$ along $d_k = \hat{x}_k - x_k$ with line search algorithm;
6:     $x_{k+1} \leftarrow x_k + \alpha_k d_k$;
```

Proposition 3.3.5. Consider

$$\min_{x \in \mathcal{S}} f(x)$$

with $f \in \mathcal{C}^1(\mathbb{R}^n)$ and $\mathcal{S} \subseteq \mathbb{R}^n$ compact and convex set. Let $\{x_k\}_{k\in\mathbb{N}}$ be the sequence produced by projected gradient algorithm with a line search that respects the following conditions:

C1. $x_{k+1} \in \mathcal{S}$;

C2. $f(x_{k+1}) < f(x_k)$;

C3. $\lim_{k\to\infty} \nabla f(x_k)^T d_k = 0$.

Then:

- either there exists $T \geq 0$ such that x_T is a critical point;
- or $\{x_k\}_{k\in\mathbb{N}}$ is a infinite sequence and each limit point is a critical point.

Proof. From the properties of the projection we have:

$$(x_k - s\nabla f(x_k) - \hat{x}_k)^T (x - \hat{x}_k) \leq 0, \quad \forall x \in \mathcal{S}.$$

Setting $x = x_k$,

$$(x_k - s\nabla f(x_k) - \hat{x}_k)^T (x_k - \hat{x}_k) \leq 0,$$

that we can rewrite as

$$\nabla f(x_k)^T (\hat{x}_k - x_k) \leq -\frac{1}{s} ||x_k - \hat{x}_k||^2. \tag{3.9}$$

Thus d_k is a decrease if $||x_k - \hat{x}_k|| \neq 0$. If the algorithm generates an infinite succession $\{x_k\}_{k\in\mathbb{N}}$, from the compactness of $\mathcal{S}$ there exists an limit point $\bar{x} \in \mathcal{S}$. We can define a subsuccession $\{x_k\}_{k\in K}$, with $K \subset \mathbb{N}$, convergents to $\bar{x}$. From the condition C3. and (3.9) we have

$$\lim_{\substack{k\to\infty \\ k\in K}} ||p(x_k - s\nabla f(x_k)) - x_k|| = 0.$$

Due to the continuity of the projection we get:

$$p(\bar{x} - s\nabla f(\bar{x})) = \bar{x}.$$

So $\bar{x}$ is a critical point of f. □

3.4 Non monotone local algorithm

We have seen two classical algorithms for the optimization of a nonlinear function on a closed set. In this section we are going to describe the non monotone versions of them, that is we change the monotone line search with a non monotone one in order to obtain better results in less time.

3.4.1 Non monotone Frank-Wolfe method

The idea of non monotone Frank-Wolfe method (NMFW) is the same as in Algorithm 1, what changes is the line search. The non monotone line search consists of accepting a stepsize as soon as it yields a point which allows a sufficient decrease with respect to a given reference value. A classical choice for the reference value is the maximum among the last M objective function values computed, where M is a positive integer constant. Here is the non monotone Armijo line search pseudocode:

Algorithm 4 : Non monotone Armijo line search.

```
1: Fix the parameters $\delta \in (0,1)$, $\gamma_1 \in (0, \frac{1}{2})$, $\gamma_2 \geq 0$, $M > 0$;
2: Update
       $\bar{f}_k = \max_{0 \leq i \leq \min\{M,k\}} f(x_{k-i})$;
3: Choose initial stepsize $\alpha \in (0, \alpha_{max}]$;
4: while $f(x_k + \alpha d_k) > \bar{f}_k + \gamma_1 \alpha \nabla f(x_k)^T d_k - \gamma_2 \alpha^2 ||d_k||^2$ do
5:     Set $\alpha = \delta\alpha$;
6: Set $\alpha_k = \alpha$; return $\alpha_k$;
```

Even in this case we have a result about the convergence of the method.

Proposition 3.4.1. Consider

$$\min_{x \in \mathcal{S}} f(x),$$

with $f \in \mathcal{C}^1(\mathbb{R}^n)$ and $\mathcal{S} \subseteq \mathbb{R}^n$ compact and convex. Let $\{x_k\}_{k\in\mathbb{N}}$ be te sequence of points produced by Frank-Wolfe algorithm with non monotone line search (NMFW). Then, either an integer $T \geq 0$ exists such that x_T is a critical point, or the sequence $\{x_k\}_{k\in\mathbb{N}}$ is infinite and every limit point $\bar{x}$ is a critical point.

Proof. For the proof, that is similar to the monotone case, we refer to [22]. □

We are showing the results in the next chapter of the thesis.

3.4.2 Non monotone spectral projected gradient method

The projected gradient method is considered to be very slow, even if projecting is inexpensive, as in box-constrained case. In [24] and [25] a non monotone spectral line search is used to boost the projected gradient method. Applying those innovations, on one hand we extend the typical globalization strategies to the non monotone line search schemes as in NMFW; on the other hand we propose to associate the spectral steplenght. The algorithm starts with an feasible initial point and uses an integer $M \geq 1$ as a "memory" parameter, a small parameter $\alpha_{min} > 0$, a large parameter $\alpha_{max} > \alpha_{min}$, a sufficient decrease parameter $\gamma \in (0,1)$, and safeguarding parameters $0 < \sigma_1 < \sigma_2 < 1$. Initially, $\alpha_0 \in [\alpha_{min}, \alpha_{max}]$ is arbitrary. Given $x_k \in \mathcal{S}$ and $\alpha_k \in [\alpha_{min}, \alpha_{max}]$. The Algorithm 5 decribes how to obtain x_{k+1} and α_{k+1} and when terminate the process. The one dimen-

Algorithm 5 : Non monotone spectral projected algorithm.

```
1:  Choose an initial point $x_0 \in \mathcal{S}$;
2:  for $k = 0, 1, \ldots$ do
3:      if $||p(x_k - \nabla f(x_k)) - x_k|| = 0$ then return $x_k$;
4:      $\lambda \leftarrow \alpha_k$;
5:      Step 1.
6:      Set $x_+ = p(x_k - \lambda \nabla f(x_k))$;
7:      if
            $$f(x_+) \leq \max_{0 \leq j \leq \min\{k, M-1\}} f(x_{k-j}) + \gamma \langle x_+ - x_k, \nabla f(x_k) \rangle ; \quad (3.10)$$
        then
8:          Define $\lambda_k = \lambda$, $x_{k+1} = x_+$, $s_k = x_{k+1} - x_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$;
9:          goto → Step 2;
10:     else Define
            $$\lambda_{new} \in [\sigma_1 \lambda, \sigma_2 \lambda];$$
11:         Set $\lambda \leftarrow \lambda_{new}$ and goto → Step 1;
12:     Step 2.
13:     Compute $b_k = \langle s_k, y_k \rangle$;
14:     if $b_k \leq 0$ then
15:         Set $\alpha_{k+1} = \alpha_{max}$;
16:     else Compute $\alpha_k = \langle s_k, s_k \rangle$;
17:         Set
            $$\alpha_{k+1} = \min\{\alpha_{max}, \max\{\alpha_{min}, \frac{\alpha_k}{b_k}\}\};$$
```

sional search procedure of Algorithm 5, called NMSPG from now on, takes into account points of the form $p(x_k - \lambda \nabla f(x_k))$ for $\lambda \in (0, \alpha_k]$, which, in general, form a curvilinear path (piecewise linear if $\mathcal{S}$ is a polyhedral set). For this reason, the scalar product $\langle x_+ - x_k, \nabla f(x_k) \rangle$ in the non monotone Armijo condition (3.10) must be computed for each trial point x_+. It can prove that also Algorithm 5 is a well defined algorithm and has the property that every limit point $\bar{x}$ is a constrained critical point. We omit the proof that anyone can find in [23].

3.5 Global optimization algorithms

We are not only interested in finding a local minimum of the objective function, but rather look for a global minimum since our main goal it is to discover the best bi-partitioning of a given graph and so we want to solve problems (3.1) and (3.2). How it is easy to understand, this task it is harder than just compute a critical point of the first order. There are many global optimization algorithm, some of them use global information like convexity, concavity or Lipschitz constant of the objective function or of the constraints and/or number of global minimums. Other algorithms use local information like values of the objective function and of the constraints on trial points and/or values of the derivatives. In both cases we can get convergence properties only through a suitable sampling. Mainly there are two kind of global algorithms:

- **Deterministic algorithms**: where the points are sampled on the basis of information obtained during the iterations of the algorithm.
- **Probabilistic algorithms**: where the points are sampled randomly, basing on the information given by the algorithm.

From now on, we assume that the feasible set $\mathcal{S} \subseteq \mathbb{R}^n$ is a no-empty and compact set such that:

$$\mathcal{S} = Cl(\mathring{\mathcal{S}}) \tag{3.11}$$

where $Cl(\cdot)$ indicates the closure of a set and $\mathring{\mathcal{S}}$ is the interior of $\mathcal{S}$.

Definition 12. Let $\mathcal{S}$ be a subset of a topological space F. $\mathcal{S}$ is said to be dense in F if $Cl(\mathcal{S}) = F$.

Notice that in a metric space F, for any point $x \in F$ and for any $\epsilon > 0$ there exists a point $y \in B(x, \epsilon)$ such that $y \in S$. For the probabilistic algorithms we have the following convergence theorem:

Theorem 3.5.1. *We assume the following hypotheses hold;*

- $\mathbb{S} \subseteq \mathbb{R}^n$ *satisfies the assumption (3.11);*
- $\mathcal{C}$ *is the set of the continuos functions on* $\mathcal{S}$*;*
- A_P *is a probabilistic global algorithm that uses local information.*

Then for any function $f \in \mathcal{C}$, with x^* global minimum of f on $\mathcal{S}$, A_P generates a sequence $\{x_k\}_{k\in\mathbb{N}}$ that converges at x^* with probability bigger than $p \in (0,1)$ if and only if any point of $\mathcal{S}$ belongs to the closure of the points produced by A_P, as $k \to \infty$, with probability bigger than p.

Proof. For the proof we refer to [26] or [27]. □

There is a similar result for determistic algorithms, we refer again to [26] or [27]. Anyway, convergence is a minimum requirement and does not ensure efficiency. A classical way to work is to explore the feasible set by giving priority to promising region. We have to balance the computational work between exploration of the feasible region and the approximation of global optimum. The algorithms can be devided into four families:

- **Incomplete algorithms**: they use heuristic method for the search, but they can finish in a local minimum.
- **Asymptotically complete algorithms**: they guarantee to gain a global minimum (almost with probability 1) if they are performed for infinite time, but there is no way to know when we obtain a global minimum.
- **Complete algorithms**: they guarantee to find a global minimum if they are performed for infinite time and in a finite time we can obtain an approximation of global minimum given a tollerance.
- **Strict algorithms**: they guarantee to identify an approximation of the global minimum, given a tollerance, even if there are errors.

In leterature one of the first algorithm that one can find is multistart algorithm. This algorithm chooses a random feasible initial point, then it computes a local minimum by a local optimization method and it does the same with a new random feasible point. So, iteration by iteration, it keeps the best value that it has found so far. It's quite obvious that this algorithm satisfies the hypotheses of Theorem (3.5.1), nevertheless it does not take advantage of the information of the previous iterations and the number of local minimization could be very big, becoming inefficient. An improvement of random search brings us to the monotonic basin hopping: Notice how a method of local opti-

Algorithm 6 : Monotonic basin hopping.

```
1: Choose randomly $x_0 \in \mathcal{S}$;
2: Set $x_0^* = \text{LocalSearch}(x_0)$ and $k = 1$;
3: Step 1.
4: Set $x_k = \text{Perturb}(x_{k-1}^*)$;
5: Set $y_k = \text{LocalSearch}(x_k)$;
6: if $f(y_k) < f(x_{k-1}^*)$ then
7:     Set $x_k^* = y_k$;
8: else $x_k^* = x_{k-1}^*$
9: $k = k + 1$ and goto $\rightarrow$ Step 1;
```

mization, LocalSearch$(\cdot)$, and a process for the perturbation of the solution, Perturb$(\cdot)$, characterize this new algorithm. In particular if Perturb$(\cdot)$ is just choose a random feasible point, we obtain the multistart algorithm. Usually, for the perturbation of x_{k-1}^*, a random point is choosen in a ball $B(x_{k-1}^*, \delta)$, where $\delta > 0$. Actually we have to take care about the dimension of this ball: if δ is too small, we will get stuck in a local minimum; on the other side if δ is too big, the algorithm will be equivalent to multistart. In fact LocalSearch and Perturb depend on the problem. In our case we are going to use the previous four methods (FW, GP, NMFW, NMSGP) as LocalSearch and a swap technique as Perturb. We are explaining the reason of these choices in the next chapter. Other global probabilistic algorithms could be simulated annealing or genetic algorithms but they are not considered in this thesis, we refer the interested reader to [28] and [29]. From now on we write BH for monotonic basin hopping algorithm and the abbreviations of local methods before BH to indicate the global basin hopping algorithm with the particular LocalSearch, for example FW-BH or NMSGP-BH.

Chapter 4

Numerical results

All the results have been computed with a MacBook Air with a processor of 1.4 GHz Intel Core i5. In order to be as clear as possible, we make a list of the topics studied in this chapter.

- Description of the problems.
- Description of the stochastic block model: the algorithm that we use to produce the model graph.
- Introduction of the performance profile idea and selection of the best parameters to be used inside the local optimization algorithms.
- Analysis of results of the global algorithm applied over the toy model graph.
- Study of real networks and comparison with the results in [11].
- Difference between non linear and linear method to detect the best bi-partioning of a real graph.

4.1 Our riformulated problem

Given a graph $G = (V, E)$, the equation (3.1) is presented again:

$$q(G) = \max_{A \subseteq V} q(A) = \frac{1}{\mu(V)} \max_{\substack{x \in \mathbb{R}^n \\ ||x||_\infty \leq 1}} \langle x, \mathcal{M}(x) \rangle_\mu .$$

This is the problem we consider here, the weighted case can be seen as a natural continuity but it is not studied in this thesis. In particular the function that we want to maximize is

$$f_1(x) = \langle x, \mathcal{M}(x) \rangle_\mu .$$

Just to fix the ideas, we consider the unit measure of V: $\mu(i) = \mu_i = 1$, for all $i \in V$. Therefore the weighted scalar product $\langle x, y\rangle_\mu = \sum_{i=1}^n x_i y_i = \langle x, y\rangle$ is the usual scalar product of $\mathbb{R}^n$. Remebering (2.12), for any $x \in \mathbb{R}^n$, we obtain

$$f_1(x) = \langle x, \mathcal{M}(x)\rangle = \frac{1}{2}\sum_{i,j=1}^{n} (-M)_{ij}|x_i - x_j|. \tag{4.1}$$

As seen in Section 3.1, we are going to consider an approximation of f_1, adding an exponent $t \in \mathbb{R}$, $t > 1$,

$$f_t(x) = \frac{1}{2}\sum_{i,j=1}^{n} (-M)_{ij}|x_i - x_j|^t.$$

Therefore we gain $f_t \in \mathcal{C}^1(\mathbb{R}^n)$, moreover it is easy observe that the constrain set

$$\mathcal{S} = \{x \in \mathbb{R}^n : ||x||_\infty \leq 1\},$$

is a convex compact set so we can utilize each algorithms displayed in the Chapter 3. From now on, we present all the results as maxima, in Section 3.2 we have discussed how to move from a minimization to a maximization problem. Furthermore we opt for the particular edge measure ω such that

$$\omega(ij) = \begin{cases} 1, & \text{if } ij \in E, \\ 0, & \text{if } ij \notin E. \end{cases}$$

So by adopting the Newman - Girvan null model, the modulaity matrix M is

$$M = A - B, \tag{4.2}$$

where:

- A is the adjacency matrix of G,
- B is the matrix resulting from the null model, we have for any $i, j \in V$, $(B)_{ij} = \frac{d_i d_j}{\mathrm{vol}(V)}$.

In most cases the adjacency matrix of a graph is a sparse matrix, a fact that has to be taken into account to make the computational aspect efficient.
To conclude this section we remember what is the projection in a box set as $\mathcal{S}$. It is easy to prove that the projection of $y \in \mathbb{R}^n$ over $\mathcal{S}$ is the function $\pi : \mathbb{R}^n \to \mathcal{S}$ such that, for any $i = 1, \dots, n$,

$$(\pi(y))_i = \begin{cases} 1, & \text{if } y_i \geq 1, \\ -1, & \text{if } y_i \leq -1, \\ y_i, & \text{otherwise.} \end{cases} \tag{4.3}$$

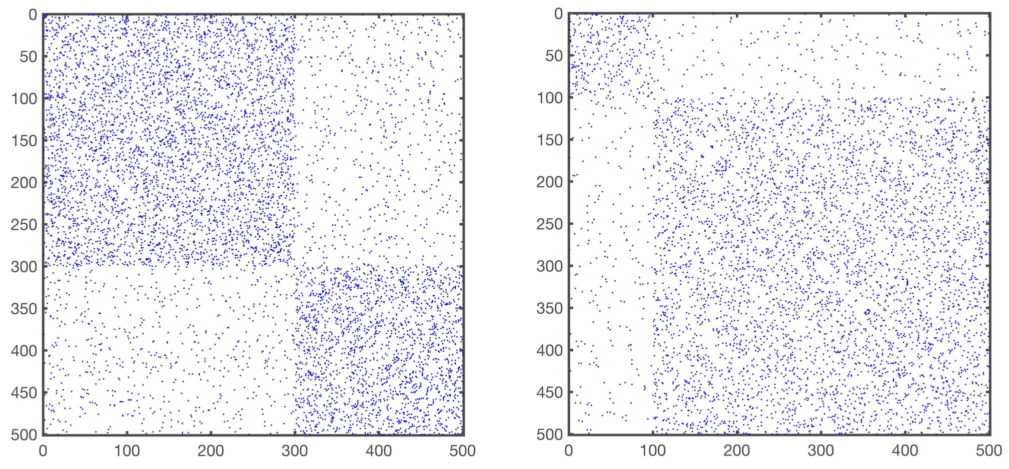
Notice that this projection it is very easy to program and fast to compute.

4.2 Stochastic block model (SBM)

Probabilistic network models can be used to model real networks as it is studied in [30], the stochastic block model (SBM) is one of the most popular network models exhibiting community structures. The model was first proposed in the 80's and received significant attention in mathematics and computer science literature, for further information we refer to [31]. The SBM puts a distribution on n-vertices graphs with a hidden (or planted) partition of the nodes into k communities. We define the vector $size \in \mathbb{R}^k$ so that its components are the dimension of k-th community and we assume that a pair of nodes in communities i and j connects independently with probability Q_{ij}. The inputs for the SBM code are the vector $size$ and Q a $k \times k$ symmetric matrix with entries in $[0, 1]$. In this work we use the SBM function showed in Algorithm 7, the outputs are A the sparse adjency matrix of the graph and C the ground truth vector. Actually the stochastic block model it is used to set benchmarks for clustering algorithms with well defined ground truth that is typically no avaible in real networks. We work with $k = 2$ so the graph is made of just two communities. The probabilities matrix is

$$Q = \begin{pmatrix} p & q \\ q & p \end{pmatrix},$$

where p is the probabilitiy of connection inside the two communities and q the probability between them. According to [31] we are setting low probabilities in order to create a sparse graph and make our detection more difficult and so more interesting, moreover it is showed that the model graph depends on the difference $p-q$ and not on the particular p and q choosen, so we are proposing results depending on the difference $p-q$. Figure 4.1 represents the non null entries of the adjency matrix of two different graph obteined with SBM.



(a) *size=[300,200], p=0.05, q=0.01.* (b) *size=[100,400], p=0.03, q=0.005.*

Figure 4.1: Example of graph from SBM.

Algorithm 7 : Stochastic block model function.

```
function [A, C] = SBM(Q, size)
    I=[ ];
    J=[ ];
    k = length(B);
    n = sum(size);
    for i1=1:k do
        for i2 = 1:size(i1) do
            i = sum(size(1:i1-1)) + i2;
            lrow = size(i1) - i2;
            e = rand(1,lrow);
            Jnew = find(e <= Q(i1,i1));
            enew = length(Jnew);
            if enew >0 then
                I = [I, repmat(i,1,enew)]; J = [J i+Jnew];
            for j1 = i1+1:k do
                j = sum(size(1:j1-1));
                e = rand(1,size(j1));
                Jnew = find(e ¡= Q(i1,j1));
                enew = length(Jnew);
                if enew > 0 then
                    I = [I, repmat(i,1,enew)]; J = [J j+Jnew];
    A = sparse([I J],[J I],1,n,n);
    C = zeros(n,k);
    for i = 1:k do
        imin = sum(size(1:i-1)) + 1;
        imax = sum(size(1:i-1)) + size(i);
        C(imin:imax,i) = 1;
```

4.3 Local results

Here we are going to present the results computed with the local algorithms showed in Chapter 3. First of all we introduce performance profile as a tool for evaluating and comparing the performance of optimization software. After that we present the results and successively choose the best paramenters. The parameters studied are the approximation exponent t and the parameter $s \geq 0$ that we find inside the code of the projected gradient method, Algorithm 3. Instead the "memory" parameters M in NMFW and NMSGP or the initial step size inside the Armijo line search and other parameters are choosen by the indication of leterature since we do not recognize important variations in the results. At the end we give some indications/suggestions to reduce the computational time and to detect a smart Perturb function for the global optimization algorithm.

4.3.1 Performance profile

The benchmarking of optimization software has improved a lot in the last years, one of the most important contribution can be found in [32]. Most benchmarking efforts involve tables displaying the performance of some solver applies to problems using as metrics CPU time or number of function evaluations. In this section we introduce the notion of performance profile as a mean to evaluate and compare the performance of a set of solvers S on a test set U. Let n_s, n_u be the number of solvers and problems respectively. What we use as performance measure is the *computing time*. For each problem u and solver s, we define

$$T_{u,s} = \{\text{computing time required to solve problem } u \text{ by solver } s\}.$$

We compare the performance on problem u by solver s with the best performance by any solver on this problem. We define a performance ratio:

$$r_{u,s} = \frac{T_{u,s}}{\min\{T_{u,s} : s \in S\}} \tag{4.4}$$

We assume that a parameter $r_N \geq r_{u,s}$ for all u, s is chosen, and $r_{u,s} = r_N$ if and only if the solver s does not solve the problem u. In [12] it is showed that the choice of r_N does not affect the performance evaluation and we refer to that paper for further information. We define

$$\rho_s(\tau) = \frac{1}{n_u}|\{u \in U : r_{u,s} \leq \tau\}|, \tag{4.5}$$

where $|\cdot|$ indicates the cardinality of the set, $\rho_s(\tau)$ is the probability for solver $s \in$ S that a performance ratio $r_{u,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio. The function ρ_s is the (cumulative) distribution funciton for the performance ratio. We use the term performance profile for the distribution function of a performance metric, a plot of performance profile reveals all the major performance features. In particular, if the set of problems U is suitably large and representative of problems that are likely to occur in applications, then solvers with large probability $\rho_s(\tau)$ are to be preferred.

4.3.2 Parameters detection

Now we are going to present the first results for the parameters detection of the local algorithms using performance profiles. In order to be coherent with the notations above, for any algorithm, the set of solvers S depends only on the exponent t for FW, NMFW and NMSGP algorithms

$$S_t = \{1.15; 1.2; 1.3; 1.4\},$$

while for the case of GP method we study also the inner parameter $s \geq 0$ and we have $S_{(t,s)} = \{(1.15, 0.15); (1.2, 0.2); (1.3, 0.25); (1.4, 0.3)\}$. The set U is characterized by 4 probabilities differences $\{0.01; 0.02; 0.03; 0.04\}$ and for each of them we create 10 model graphs with SBM with 500 nodes, so the test set U is composed of 40 different problems. For all the algorithms, the initial point $x_0 \in \mathbb{R}^n$, where n is the dimension of the graph, is chosen randomly inside the feasible set $\mathcal{S}$. From the definition, the performance profile $\rho_s : \mathbb{R} \to [0, 1]$ for a solver is a nondecreasing, piecewise constant function, continuous from the right at each breakpoint and the value of $\rho_s(1)$ is the probability that the solver will win over the rest of the solvers. We fix $r_N = 1000$, a value big enough for our case. What we obtain are graphics like Figure 4.2 and 4.3 for GP and FW algorithms respectively.

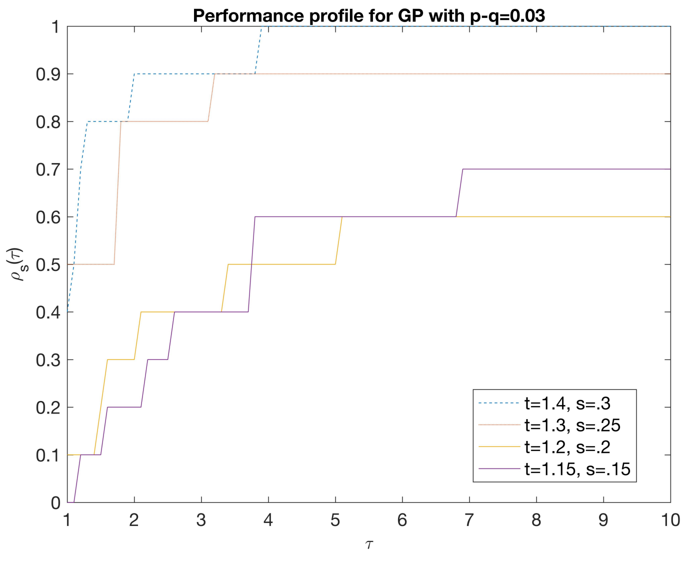


Figure 4.2: Performance profile for GP algorithm with $p - q = 0.03$.

The plots represent the probability for the different solvers that a performance ratio (4.4) is within a factor $1 \leq \tau \leq 10$ of the best possible ratio. We notice that in both figure the best choice of exponent is for $t = 1.4$, since the blu dotted line is the best and reaches the probability 1 in almost 4 times of the best possible ratio for GP and in less than 2 times for FW. Moreover, by looking at Figure 4.3 we can notice that for the parameter $t = 1.15$ only the 60% of the problems are solved within $\tau = 10$.

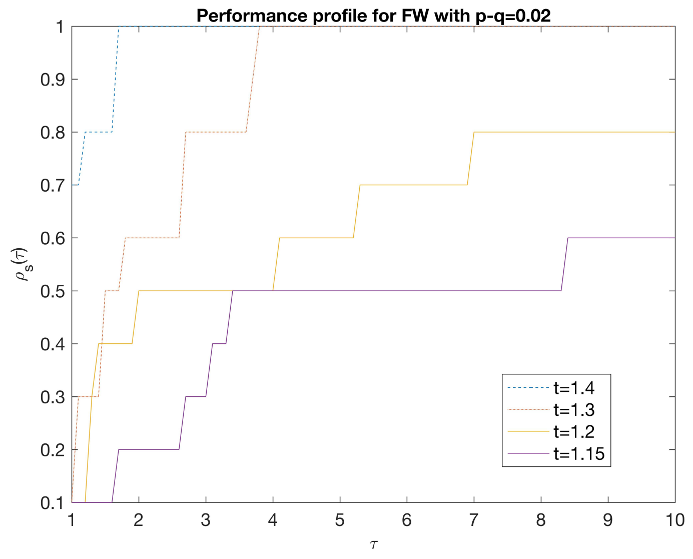


Figure 4.3: Performance profile for FW algorithm with $p - q = 0.02$.

We study all the cases and we choose the best parameters by analizing the plots, we resume the result in the Table 4.1 for the GP algorithm and Table 4.2 for the others.

$p - q$	0.01	0.02	0.03	0.04
Exponent t	1.4	1.3	1.4	1.4
Parameter s	0.3	0.25	0.3	0.3

Table 4.1: Parameters detection for GP with performance profile.

As we can see, the most common result for the approximation exponent is the value 1.4, this can be due to the fact that the function for this exponent is more likely a quadratic function and therefore it is faster to detect a maximum point. We decide to not choose an exponent bigger than 1.4 otherwise we would find the same results of the linear spectral method (2.10), for further information we refer to [11].

$p - q$	0.01	0.02	0.03	0.04
FW	1.3	1.4	1.4	1.4
NMFW	1.4	1.4	1.4	1.4
NMSGP	1.2	1.4	1.4	1.4

Table 4.2: Best approximation exponent t computed by performance profile.

4.3.3 Selection of Perturb function and codes improvement

From these results we find out an important property that brings us to determinate the Perturb function inside the basin hopping global algorithm and to improve the efficiency of the evaluation of the objective function. The property concerns the solution of the problem (3.1), in all the cases this is approximately a vertex of the feasible set $\mathcal{S}$, this means that most of the solution's components are 1 or -1. It seems that the iterative algorithms, after few steps, try to reach a maximum move from a vertex to another. This feature and the combinatoric nature of the problem gives us the idea for the Perturb function of the basin hopping, Algorithm 6. We base it on a swap idea. In particular, we define a function that changes a fixed percentage, from 10% to 25%, of components of the local optimum found at every iteration of the algorithm. In this way we change vertex of $\mathcal{S}$ and so, using (2.8), the particular bi-partitioning of the graph. Moreover in order to not stop in a attracted local basin we multiply the vector for a random positive constant $\eta < 1$.
Another suggestion comes from the results concerning the function evaluation. Each method showed is iterative algorithm and it computes the objective function many times. The fact that the iteratives seem to move from a vertex to another and the particular nature of the function bring us to reformulate the code for the evaluation function in such a way to keep the information of the previous step and update the value just computing the function over the components that change from step to step. In order to have clear notation, choosen any local algorithm, let x, $y \in \mathbb{R}^n$ be the solutions at step k and $k+1$. We have

$$f_t(x) = \sum_{i,j=1}^{n} (-M)_{ij}|x_i - x_j|^t,$$

and

$$f_t(y) = \sum_{i,j=1}^{n} (-M)_{ij}|y_i - y_j|^t.$$

We define $I \subseteq \{1, \dots, n\}$ such that $I = \{i : x_i = y_i\}$, the set of index where x and y are equal. Remembering that the modularity matrix M is symmetric, we obtain

$$f_t(y) = \sum_{\substack{i\in I\\ j\in I}} (-M)_{ij}|y_i - y_j|^t + 2\sum_{\substack{i\in I\\ j\in I^C}} (-M)_{ij}|y_i - y_j|^t + \sum_{\substack{i\in I^C\\ j\in I^C}} (-M)_{ij}|y_i - y_j|^t =$$

$$= \sum_{\substack{i\in I\\ j\in I}} (-M)_{ij}|x_i - x_j|^t + 2\sum_{\substack{i\in I\\ j\in I^C}} (-M)_{ij}|y_i - y_j|^t + \sum_{\substack{i\in I^C\\ j\in I^C}} (-M)_{ij}|y_i - y_j|^t.$$

And so if we define the function $g_t : \mathbb{R}^n \to \mathbb{R}$ as

$$g_t(y) = 2\sum_{\substack{i\in I\\ j\in I^C}} (-M)_{ij}|y_i - y_j|^t + \sum_{\substack{i\in I^C\\ j\in I^C}} (-M)_{ij}|y_i - y_j|^t,$$

we have

$$f_t(y) = f_t(x) - g_t(x) + g_t(y).$$

The result is that savig the solution x and the value $f_t(x)$ at step k, we can compute the function value in y at step $k+1$, $f_t(y)$, with just $|I| \cdot |I^C| + |I^C|^2 = n|I^C|$ operations, instead of n^2 operations.

4.4 Global results on toy model

After the detection of the best parameters, we are ready to test the global algorithm on the model graph given by SBM. Since basin hopping, global Algorithm 6, is characterized by the specific local search we are going to show the differences between the four local methods displayed in this thesis. We present the box plots of the computational time that allows us to select which is the best method to use. We need to be precise about how we have obtained the results: first of all we chose a probability difference $p - q \in \{0.01, 0.02, 0.03, 0.04\}$ and we create 10 model graphs of $n = 500$ nodes with SBM for each differences. We have decided to choose a random feasible starting point and run 10 local iterations of basin hopping. We have focused our attention on the speed of the algorithms, trying to find if there exists a method that works better with respect to the others when the connectivity of the graph changes. A model graph obtained with $p - q = 0.01$ will be more random respect to a model with $p - q = 0.04$ where it will be easier to detect a bi-partitioning. So, after the detection of the best method to use in function of the randomness of the graph we can say what are the best method to use in a real network that presents similar features. From both Figure 4.4 and Figure 4.5 we can observe that BH with NMSGP is really better than any other alternative.

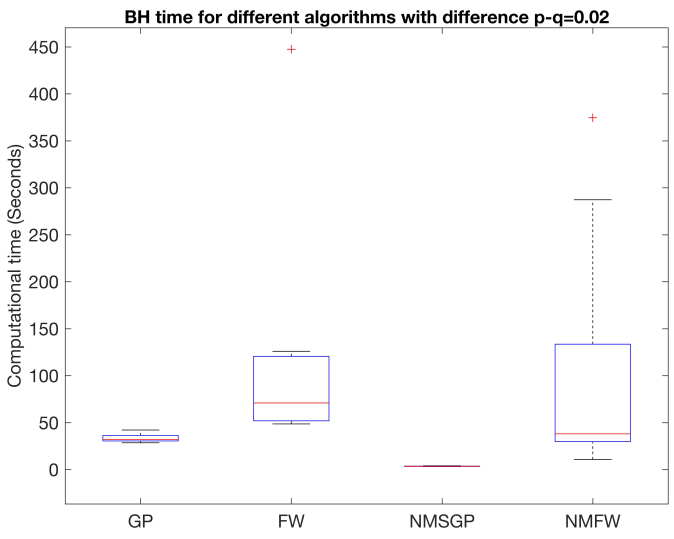


Figure 4.4: Box plots of BH computational time with $p - q = 0.02$.

Moreover, as literature said, FW is lower than GP and the non monotone versions are better then monotone ones. We do not display here the results of the maximum since every time we call SBM function the model graph changes and therefore the maximum of (3.1), but we present a median of the maxima in Table 4.3.

$p-q$		GP	FW	NMSGP	NMFW
0.01	Median	0.11193	0.11300	0.11336	0.11485
	Count	0	9	1	6
0.02	Median	0.11660	0.12023	0.11678	0.12137
	Count	0	5	0	7
0.03	Median	0.12225	0.12192	0.11963	0.12550
	Count	0	1	0	9
0.04	Median	0.12368	0.12629	0.12649	0.13051
	Count	0	3	0	3

Table 4.3: Median and count value for BH.

By looking at Table 4.3, we have to decide to keep the FW version of BH since the median computed with FW and NMFW is a little better than what we have computed with GP and NMSGP. In Table 4.3, we can observe also the value *Count* that indicates how many times the local search method does not find a critical point in 1000 iterations. We notice that GP and NSGP have a better behaviour also for this point of view.

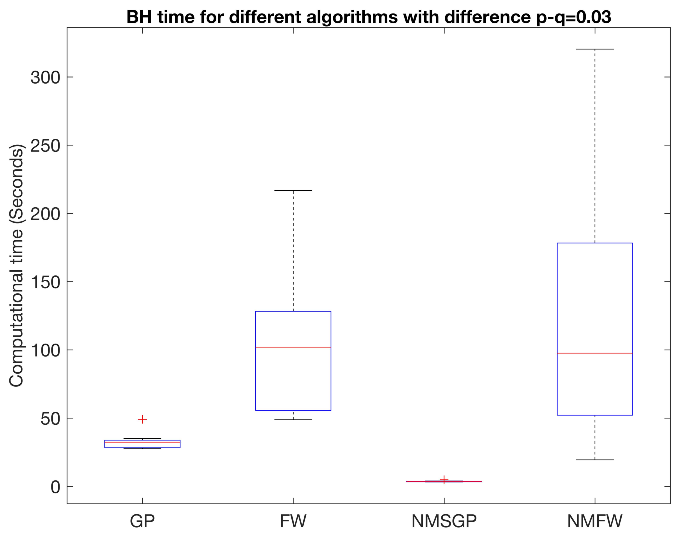


Figure 4.5: Box plots of BH computational time with $p-q=0.03$.

4.5 Real networks

The real world networks are given by Prof. Francesco Tudisco, Marie Skłodowska Curie Fellow, Department of Mathematics and Statistics, University of Strathclyde. The real graphs represent data taken from different fields, from ecological networks (Benguela, Skipwith, StMarks, Ythan2), social and economic networks (SawMill, UKFaculty, Corporate, Geom, Erdös), protein-protein interaction networks (Malaria, Drugs, Hpylori, Ecoli, PINHuman), technological and informational networks (Electronic2, USAir97, Internet97, Internet98) and transcription networks (YeastS). The results are compared with the results in [11], where the authors use a Generalized ratioDCA algorithm to find out the solutions and for the values showed in Table 4.4 and Table 4.5 they consider 100 random starting points, including w_1 the leading eigenvector of M. Instead we only consider w_1 as starting point and we compute 25 iterations of basin hopping algorithm. In the first Table 4.4, we have inserted the best value found by BH between the four different local search. For real graphs with few nodes we have spoted the same maximum.

Network	n	$q(G)$ in [11]	$q(G)$ with BH
Benguela	29	0.09	0.0939
Skipwith	35	0.07	0.0659
StMarks	20	0.2	0.1971
Ythan2	92	0.22	0.2209
SawMill	12	0.39	0.3870
UKFaculty	81	0.37	0.3711
Malaria	229	0.35	0.3439
Drugs	616	0.48	0.4842
Hpylori	710	0.35	0.3484
Electronic2	252	0.47	0.4749
USAir97	332	0.3	0.2983

Table 4.4: Experiments on real networks.

A very interesting and important aspect in what we have computed is the computational time needed to BH for different local searches. In Figure 4.6, we propose a histogram representing the running time needed to find a solution with BH on three real networks with more than 200 nodes, for each network there are four columns that show the total time spent by BH with the particular algorithm choices of local search. We can notice that the results are similar to what we have computed over the toy model with Figure 4.4 and Figure 4.5, where NMSGP is the faster algorithm and FW, NMFW the slower ones.

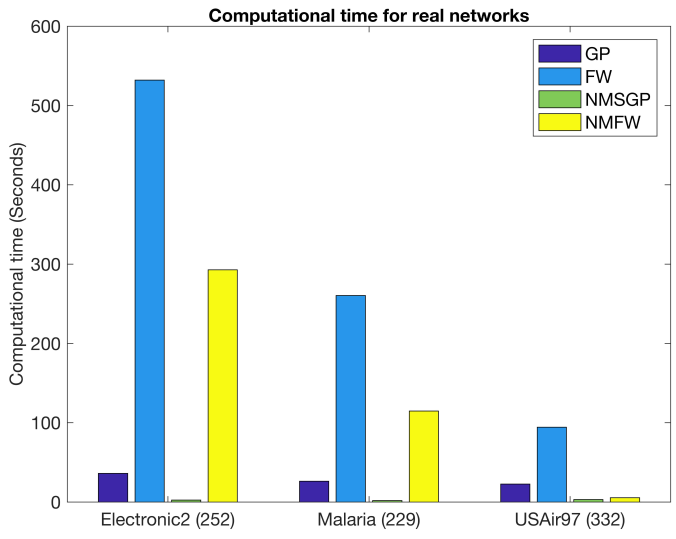


Figure 4.6: Histogram of computational time of different versions of BH apply on real networks.

But in the real cases the difference of computational time is huge, this brings us to choose only NMSGP for computing the best bi-partitioning of real graphs with more than 1000 nodes. Therefore the results in Table 4.5 are obtained with the NMSGP version of BH. By examining the two Tables of real networks we can notice that we obtain always the same results give in [11], except for *YeastS* and *Malaria* networks. Actually for *Geom* network we get a better solution.

Network	n	$q(G)$ in [11]	$q(G)$ NMSGP-BH
Corporate	1586	0.33	0.3281
Geom	3621	0.42	0.4257
Erdös	6927	0.42	0.4162
Ecoli	1251	0.28	0.2770
PINHuman	2783	0.4	0.4028
Internet97	3015	0.4	0.4020
Internet98	3522	0.4	0.4084
YeastS	2224	0.39	0.3795

Table 4.5: Experiments on real networks. BH with NMSGP as local algorithm.

If we look at Figure 4.7, we can observe in a particular case what we have said about the computational time and the real efficiency of the NMSGP version of BH. The histogram show how much the versions of BH with FW and NMFW are slow with respect to the GP and NMSGP ones. Precisely, the computational times are: 6527.9 seconds for FW-BH, 1185.9 seconds for GP-BH, 4761.8 seconds for NMFW-BH and 17.9 seconds for NMSGP-BH .

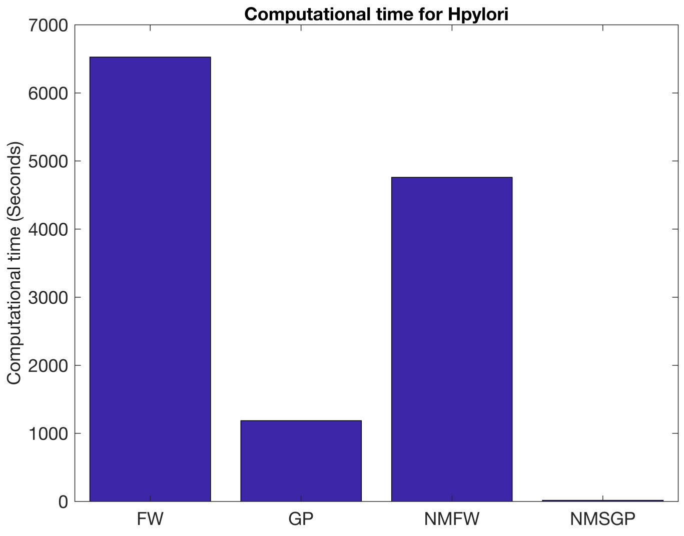


Figure 4.7: Histogram of computational time of BH applies on Hpylori real netwrok.

We have to say that the best value of $q(G)$ on the *Hpylori* real network is computed by FW version of BH with $q(G) = 0.34260$, and the worst by NMSGP with $q(G) = 0.33994$. But 27 ten-thousandths of difference is not worth the huge difference of time. The goodness of our result derives from two aspects: on one hand we have choosen one initial points and only 25 iterations of basin hopping algorithm, on the other hand the computational time for NMSGP-BH version that is quite fast. Table 4.6 reports the computational time of NMSGP-BH applies on all the real networks. As we expected, bigger is the network more time is necessary to find a maximum. To complete the analysis of the results we decide to study where the algorithms spent more time during the computation. To do that we use the profiler of Matlab, for further information we refer to the MathWorks web site. By focusing on the monomote algorithms, we can say that both FW-BH and GP-BH spend the most of time on the Armijo line search. The problem is not that they compute a step size too much small but the number of function and derivative evaluations needed that is very big. By remembering the equation (4.2) we can be more precise. Indeed, both algorithms spent lots of time to evaluate the matrix B since it is a dense matrix. Also for NMFW-BH and NMSGP-BH the most of computational time is spent on the function and derivative evaluations but they need less evaluations thank to the non monotone line search and what makes NMSGP-BH the best algorithm is the combination of non monotone line search with a spectral method.

Network	n	Computational time of NMSGP-BH
Benguela	29	1.2”
Skipwith	35	0.5”
StMarks	48	0.5”
Ythan2	92	0.9”
SawMill	36	0.3”
UKFaculty	81	0.4”
Malaria	229	2.5”
Drugs	616	8.2”
Hpylori	710	17.9”
Electronic2	252	2.4”
USAir97	332	3.1”
Corporate	1586	53.5”
Geom	3621	337.3”
Erdös	6927	2842.7”
Ecoli	1251	56.7”
PINHuman	2783	525.5”
Internet97	3015	535.0”
Internet98	3522	888.0”
YeastS	2224	152.3”

Table 4.6: Computational time on real networks for BH with NMSGP as local algorithm.

4.6 Comparison between linear and non linear method

To conclude this chapter we are going to show the difference between the linear and non linear method to compute the best bi-partitioning of the graph. In Table 4.7 it represents the maxima of the modularity computed by linear and non linear method. For the graphs with more than 1000 the results of non linear method are computed by NMSGP version of BH. We notice that for all the real world networks the maximum computed by the non linear extension of M is really better than the maximum found by the leading eigenvector of the modularity matrix M. We propose the linear results present in [11].

Network	n	$q(G)$	
		Linear	Non linear
Benguela	29	0.07	0.0939
Skipwith	35	0.04	0.0659
StMarks	20	0.16	0.1971
Ythan2	92	0.2	0.2209
SawMill	12	0.33	0.3870
UKFaculty	81	0.33	0.3711
Malaria	229	0.25	0.3439
Drugs	616	0.43	0.4842
Hpylori	710	0.28	0.3484
Electronic2	252	0.36	0.4749
USAir97	332	0.28	0.2983
Corporate	1586	0.27	0.3328
Geom	3621	0.25	0.4257
Erdös	6927	0.28	0.4162
Ecoli	1251	0.25	0.2770
PINHuman	2783	0.32	0.4028
Internet97	3015	0.27	0.4020
Internet98	3522	0.25	0.4084
YeastS	2224	0.25	0.3795

Table 4.7: Comparison between modularity values computed by linear method and non linear method.

The histogram in Figure 4.8 shows the gain of the modularity values obtained by non linear method with respect to linear one. The gain is $q_{NL}(G)/q_L(G)$, where $q_L(G)$ is the modularity linear value and $q_{NL}(G)$ the modularity non linear value.

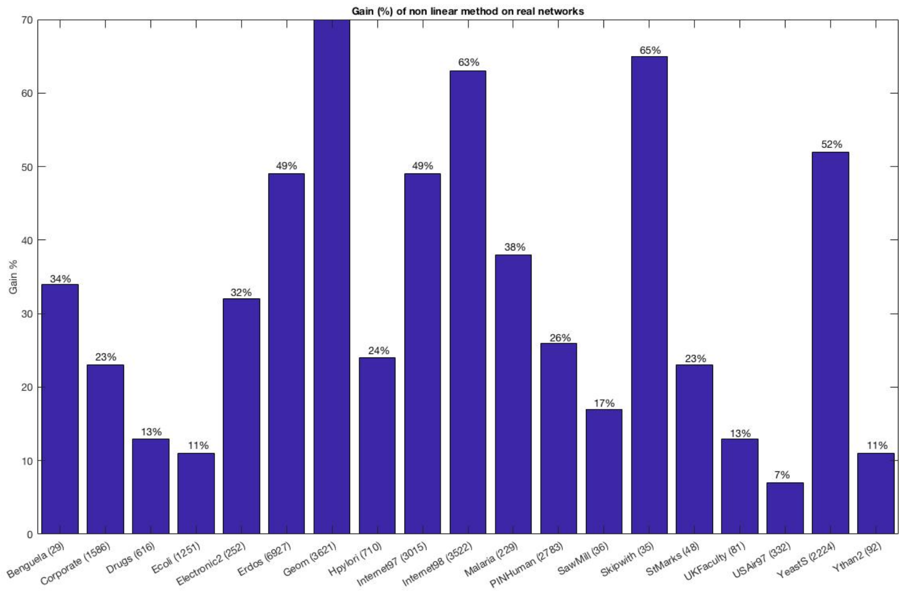


Figure 4.8: Histogram of the gain of non linear method respect to linear method.

Chapter 5

Conclusions

In this thesis we analyzed the problem of bi-partitioning a given graph. We started from a natural linear view of the problem and then we moved to a non linear extension that gives us a strict relation between the best bi-partitiong and the maximum of a continuous functions. Moreover we proved a new constrained formulation of the maximization problem with a convex and compact region as feasible set. From there we decided to approximate the objective function with a $\mathcal{C}^1$ function in order to use first-order local optimization algorithms. We showed these algorithms, giving for each of them convergence theorems and related schemes. Finally we described a probabilistic global method that we applied first on a toy model graph, given by a stochastic block model, and then on real world graphs. We compared results with some state of the art methods, trying to highlight what is the strength of our approach and which is the best algorithm to use. Some possible continuations of this work can be to maximize the weighted version (3.2) of modularity function or to change the null model at the base of the modularity measure's definition, in that way it would change the B term of modularity matrix (4.2). It would also interesting to study some further theoretical properties of modularity functions, like e.g. hidden concavity. This might help to develop tailored algorithms for the problem.

Bibliography

[1] Shields, Rob. *Cultural Topology: The Seven Bridges of Königsburg 1736.* Theory Culture and Society. 29 (4-5): 43–57. doi:10.1177/0263276412451161, [2012].

[2] Wasserman, S., and K. Faust. *Social network analysis.* Cambridge University Press, Cambridge, UK, [1994].

[3] Chen, J., and B. Yuan. *Bioinformatics 22(18), 2283*, [2006].

[4] Dourisboure, Y., F. Geraci, and M. Pellegrini. *WWW '07: Proceedings of the 16th international conference on the World Wide Web* (ACM, New York, NY, USA), pp. 461– 470, [2007].

[5] Krause, A. E., K. A. Frank, D. M. Mason, R. E. Ulanowicz,and W. W. Taylor. *Nature 426, 282*, [2003].

[6] Krishnamurthy, B., and J. Wang. *SIGCOMM Comput. Commun. Rev. 30(4), 97*, [2000].

[7] Reddy, K. P., M. Kitsuregawa, P. Sreekanth, and S. S. Rao. *DNIS '02: Proceedings of the Second International Workshop on Databases in Networked Information Systems*, (Springer-Verlag, London, UK), pp. 188–200, [2002].

[8] S. Fortunato. *Community detection in graphs*, arXiv:0906.0612v2, [physics.soc-ph], [2010].

[9] Erdös, P., and A. Rényi. *Publ. Math. Debrecen 6, 290*, [1959].

[10] M. E. J. Newman and M. Girvan. *Finding and evaluating community structure in networks.* Physical review E, 69:026113, [2004].

[11] F. Tudisco, P. Mercado, M. Hein. *Community detection in networks via nonlinear modularity eigenvectors*, arXiv:1708.05569 [cs.SI], [2017].

[12] E. D. Dolan, J. J. Moré. *Benchemarking optimization software with performance profiles*, Ser. A 91:201-213 (2002), Springer-Verlag, [2001].

[13] N. Arcolano, K. Ni, B. A. Miller, N. T. Bliss, and P. J. Wolfe. *Moments of parameter estimates for Chung-Lu random graph models.* In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3961–3964, [2012].

[14] Brandes, U., D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikolski, and D. Wagner. URL http://digbib.ubka.uni-karlsruhe.de/volltexte/documents/3255, [2006].

[15] D. Fasino and F. Tudisco. *An algebraic analysis of the graph modularity.* SIAM Journal on Matrix Analysis and Applications, 35:997–1018, [2014].

[16] F. H. Clarke. *Optimization and nonsmooth analysis, volume 5.* SIAM, [1990].

[17] F. Bach. *Learning with submodular functions: A convex optimization perspective.* arXiv preprint arXiv:1111.6453, [2011].

[18] M. Hein and S. Setzer. *Beyond spectral clustering-tight relaxations of balanced graph cuts. In Advances in neural information processing systems*, pages 2366–2374, [2011].

[19] D. Bertsekas, Massachussetts Institute of Technology. *Nonlinear Programming*, Athena Scientific, Belmont, Massachussetts, [1995].

[20] Yu, Nesterov. *Smooth minimization of non-smooth functions*, Mathematical Programming, [2005].

[21] L. Armijo. *Minimization of functions having Lipschitz continuous first partial derivaties.* Pacific J. Math 16 (1): 1-3, [1966].

[22] C. Buchheim, M. De Santis, F. Rinaldi, L. Trieu. *A Frank-Wolfe based Branch-and-Bound algorithm for mean-risk optimization.* arXiv:1507.05914v4. [2015].

[23] E. G. Birgin, J. M. Martínez, M. Raydan. *Nonmonotone spectral projected gradient methods on convex sets.* SIAM J. OPTIM. Vol. 10, No. 4, pp. 1196-1211, [1999].

[24] J. Barzilai and J. M. Borwein. *Two point step size gradient methods*, IMA J. Numer. Anal., 8, pp. 141–148, [1988].

[25] M. Raydan. *On the Barzilai and Borwein choice of steplength for the gradient method*, IMA J. Numer. Anal., 13, pp. 321–326, [1993].

[26] Stephens, CP and Baritompa. *Global optimization requires global information.* Journal of Optimization Theory and Applications, volume 96, number 3, pages 575-588, Springer, [1998].

[27] Stefano Lucidi. *Notes of 'Global optimization'.* Sapienza University of Rome, [2013/2014].

[28] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*, John Wiley and Sons, [1990].

[29] Whitley, D. Stat Comput 4: 65. https://doi.org/10.1007/BF00175354, [1994].

[30] M. Newman, *Networks: an introduction*, Oxford University Press, Oxford, [2010].

[31] Abbe, Emmanuel and Sandon, Colin, *Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery*, Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on, pp. 670-688, [2015].

[32] H. Mittelmann, *Benchmarks for optimization software*, http://plato.la.asu.edu/bench.html