Lukas Klug, BSc

The Naming Game on Empirical and Synthetic Networks

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Denis Helic, Assoc. Prof. Dipl.-Ing. Dr.techn.

Knowledge Technologies Institute

Graz, August 2015

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis dissertation.

Date

Signature

Abstract

In this thesis, it is investigated how the computational intensive Naming Game can be precalculated. This minimalistic model emerging from linguistics proved itself appropriate for agreement dynamics and opinion formation in social networks. In this approach, opinions are pinned to cohesive groups of individuals reflecting the community structure of such networks. This work focuses on how the community detection technique spectral analysis correlates with the outcome of Naming Game simulations. It is found that for the opinion clusters a distinct pattern could always be observed, which becomes clearer with the number of clusters. Moreover the analysis yields a remarkably accurate one-dimensional ordering of the nodes. These results can be obtained from synthetic and verified on two empirical networks which comprise strong community structure.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Similarity breeds connection. This fact advancing from the field of social sciences refers to the homophily principle (McPherson et al., 2001). Individuals exhibit different qualities like ethnicity, gender, status or religion. The more similar they are to each other, the more likely an interaction between them is. By agreement on their opinions they get strengthened in what they are thinking and further get localized in the social space. Through the isomorphic principle of homophily, structures in various networks describing friendship, work, opinion or co-membership are obtained. It limits individuals in the way they receive information, how they form attitudes or shape opinions.

In social networks homophily is reflected in community structure (Girvan and Newman, 2002). Individuals display a lot of social ties to others within their group but only a few to individuals outside of the it. Community structure is one feature beside network transitivity, the small-world property (Watts and Strogatz, 1998) or scale-free degree distributions (Barabási and Albert, 1999). It is nevertheless the main force which hinders individuals reaching consent on a global level. One example of a large empirical network is the well-known political blogosphere which was captured during the U.S. election in 2004 (Adamic and Glance, 2005). Two camps, one for the liberals and one for the republicans, were built by the bloggers. They mainly linked blogs belonging to their camps while cross-links remained rare.

Various agent-based models try to describe the process of reaching consent in social networks (Epstein, 1996), (Anghel et al., 2004), (Lambiotte and Ausloos, 2007), (Candia and Mazzitello, 2008). They advance from various sciences. In such simulations each individual holds an opinion. Through *microscopic* interactions between them, they shape their opinion over time. On the long run, certain *macroscopic* features can be observed. Such a behavior is referred to as self-organized as it emerges without global coordination. In this thesis, the Naming Game is used as a minimalistic agent-based model which has already been analyzed intensely and it represents a powerful tool for modeling opinion dynamics.

The Naming Game was investigated considering agreement processes in a high and middle school network in which students are connected if they share activities (Lu et al., 2009). At the beginning of the simulations four communities emerged where the students were divided by race and school grade. On the long run, the students from middle school coalesced into one big cluster. This study shows very accurately that students divide themselves by race with age. It further points out that the Naming Game is an authentic model for opinion formation. The model however exhibits the problem that it is computational intensive, whether it is time complexity or memory load. Moreover several simulations need to be carried out and compared due to its randomness.

This thesis sheds new light on how the outcome of the Naming Game can be calculated through the well-known community detection technique spectral analysis. In this analysis, the eigenvector corresponding to the second smallest eigenvector, which is called the *Fiedler vector*, of the Laplacian matrix is utilized. The nodes of the network are reordered according to their value in the vector. Based on the gaps in the

sorted vector, they are separated in communities. It is then analyzed how these communities correlate to the opinion clusters from the outcomes of the Naming Game simulations. For a better visualization of the network structure, a *k*-core decomposition is employed. This method can be useful on networks which exhibit communities which are only poorly connected.

The *Fiedler vector* is first analyzed on synthetic networks. The networks are generated according to a Stochastic Block Model with two, three and four communities. In this thesis one probability is stated for links between nodes belonging to the same community and another probability for links between nodes belonging to different communities. The quotient of the probabilities depicts the *inter-connectivity* ν. Networks generated with a certain *inter-connectivity* are then categorized in three distinct structures based on the outcome of the Naming Game simulations. An asymmetric structure, where the number of communities in an outcome equals the number of communities defined in the Stochastic Block Model in all of the simulation outcomes. A symmetric structure in which consensus is reached, i.e. the simulation outcome depicts only one opinion. Finally a metastable structure where the number of opinions differ in the simulation outcomes.

The results from the synthetic networks were then compared with the results of simulations on two empirical networks. First a co-authoring network from arXiv[1] was investigated. Scientists are connected if they jointly wrote papers. This network exhibits interesting properties, as it grants insight in the collaboration of a scientific community. The results are further investigated on a subset of the Facebook network[2] which consists of ten Ego-networks. It accurately reflects real-world friendships and is thus highly valuable for research on consensus engineering.

The results showed that the communities in the *Fiedler vector* always follow a characteristic curve. They first increase very sharply, then change into a linear slope and increase again very sharply at the end. In the synthetic networks an interesting behavior emerges in networks generated with an asymmetric structure. It can be observed that with the increasing number of pre-defined communities in the Stochastic Block Model, the shape of the curve more precisely approximates a plateau, thus they can be separated more clearly. The analysis however exhibits problems with the one-dimensional reordering of the nodes. If there are more than two communities involved, they cannot always be separated by the gaps in the vector. It is nevertheless remarkable, that the ordering of the nodes on the vector is very accurately. The results from the empirical networks confirm the observations from the synthetic networks.

[1] http://arxiv.org/
[2] https://www.facebook.com

Chapter 2

History and Related Work

Agent based modeling has a long history in computer science. Furthermore it has proven itself as an interdisciplinary field of research. The beginnings date back to the 1940s where Von Neumann and Ulam introduced the concept of the cellular automaton (Neumann, 1966), (Ulam and Marcin, 1960). Von Neumann's idea was to create a machine which is able to replicate itself, which is referred to as Self-Reproducing Automaton. Konrad Zuse stated in his book "Rechnender Raum" that the whole universe can be simulated using cellular automata (Zuse, 1969).

Such automata consist of a grid of cells in typically two dimensions, although the number of dimensions can vary. In the beginning each cell takes on one of the predefined discrete states, for example 1 or 0. Further, a neighborhood is defined, famous examples are the Von Neumann neighborhood where the cells on the top, bottom, left and right are considered or the Moore neighborhood in which also the diagonal cells are included. Based on the neighborhood and the state of the cell itself, an update rule is applied which determines the new state. This is done simultaneously for the whole grid.

A very popular version of cellular automata is Conway's Game of Life. The game consists of an infinite two-dimensional checkerboard, where only the update rules and the initial states are defined. From this very simple apparatus various static and repeating patterns can be observed which are named Beehive, Beacon, Glider and so forth.

Further work on cellular automata has been carried out in the field of Mathematics for describing complex models by the automaton rules and observing self-organizing behavior (Wolfram, 1984). Such models can be employed for describing the structure of natural patterns like snowflakes or mollusc shells.

Cellular automation models have also been utilized for medical simulations. In (Düchting and Vogelsaenger, 1984) three-dimensional lattices were used to investigate tumor growth concerning several therapies, namely surgery, radiation therapy and chemotherapy. Such modeling of tumor growth is still applied in ongoing research for example in the field of Theoretical Biology (Kansal et al., 2000). These approaches have even been extended with evolutionary game theory (Mansury et al., 2005). Through this technique, the linkage between genotype and phenotype concerning tumor morphology is examined. An evolutionary hybrid cellular automaton model for cancer research was stressed, where each cell on a two-dimensional grid is either occupied by cancer or not (P. Gerlee, 2007). Every element on the automaton is endowed with a feed-forward neural network as input, which comprises the environment of the cell.

In the field of Distributed Artificial Intelligence multiple autonomous agents work together. The agents are loosely-coupled and can therefore utilize massive computational power. Applications of such systems include signal processing, air traffic control, connectionist cognitive science, natural language (Davis, 1982) or robot planning (Konolige and Nilsson, 1980).

In "The society of mind" (Minsky, 1988) an interesting philosophy was introduced to model intelligence. The aim is to describe the human mind or other natural cognitive systems fostering agents. Each agent is

able to solve only simple tasks. Intelligence is established through connecting them to a society, sometimes referred to as agency. In this way, the connection between humans and machines is attempted to be made. Research on agent-based system is continued by the artificial intelligence community, such as combining autonomous agents with biologic phenomena (Maes, 1990), designing robotic multi-agents or simulating competition for food between ant colonies (Cliff, 1994).

The first attempts on modeling agent-based systems on social science were made by Thomas Schelling (Schelling, 1969) (Schelling, 1971). The investigated issue was segregation along societies in spatial neighborhoods on a two-dimensional grid. This segregation is based on affiliation, like a professor who wants to live among other researchers or the "ultimate concern" segregation by color. The computational resources were quite limited at that time, therefore the scope of the simulations remained small. Though he came to the conclusion that even a very subtle intolerance can lead to collective separation.

The first simulations on a large scale were carried out by (Epstein, 1996), in which artificial societies were grown from the bottom up. The model carries the name *sugarscape* and is composited of three constituents. First the environment, a two-dimensional lattice that represent resource-bearing sites. On that environment agents are moving. They hold states that can be fixed, like sex, the metabolic rate or vision or they can be dynamic like culture, wealth or opinion. For the relation of these two entities the third compound rules is introduced. An *agent-environment* rule could be to look around and go to the richest site for food. Resource growth would be an example for an *environment-environment* rule and similar rules can be defined for *agent-agent* interactions, e.g. mating, trade. Through this framework, a vast amount of simulations were carried out to gain insight into macroscopic social structures and collective group behavior like demographics, transmission of culture, trade or the emergence of groups.

Concerning social dynamics, a lot of different models are present and if all the variations are taken into account, the diversity is overwhelming (Castellano et al., 2009). A classical approach coming from physics is the *Ising-paradigm* (Landau and Binder, 2009). Monte-Carlo programs are used to originally describe ferromagnetic transition from a disordered to a ordered state. Each agent is represented by a spin which can take on the values ± 1. The total energy in the system is comprised of the sum of the energy of all nearest-neighbors in the system. A spin-flip occurs following a probability given by an exponential function. The function is parameterized by the energy change and a temperature for thermal noise.

The *voter model* (Liggett, 1999) follows a very simple definition. Each agent holds a binary state, i.e. ± 1. At each time step a random agent takes over the state of one of its neighbors.

In the *majority rule model* (Galam, 2000) also two types of opinion exist. The update rule selects a group with a fixed size of agents and all of them take over the opinion of the majority. If there is a tie, i.e. half of the agents are of opinion $+1$ and half of the agents are of the opinion -1, one introduces a bias and the whole group accepts it.

In the original version of the *Sznajd model* (Sznajd-Weron, 2005) each agent occupies a site on a linear chain bearing again opinion ± 1. At each timestep a pair of adjacent agents is selected, say A and B. If their opinions match both neighbors adopt it. Otherwise the neighbor of A takes over the opinion of B and the neighbor of B takes over the opinion of A accordingly. An variant of the model omits the second rule (antiferromagnetic case) because it was unrealistic to represent community behavior.

One important common mutuality of the models introduced so far is their restriction to only two states. An example of an approach which incorporates multiple states would be the well known *Axelrod model* (Axelrod, 1997). It investigates the dissemination of culture with respect to two assumptions: Social influence, which means that people influence themselves more when they interact. The second is homophily, the preference of individuals to associate with similar people. Each agent is endowed with a vector of features or dimensions of a culture, each can take on discrete values which are called traits. In every step an agent and one of its neighbors is selected. A probability for their communication is calculated based on the traits they share. If an interaction takes place a random feature is selected where they disagree and the traits are set equal. There exist various other models, such as for bounded confidence, crowd behavior, hierarchy formation or human dynamics.

In language dynamics a distinction between *sociobiological* and *sociocultural* models can be roughly made (Steels, 2005). *Sociobiological* approaches incorporate Chomsky's nativist proposals (Chomsky,

1988), which are grounded on an evolutionary foundation. Successful communicators carry a selective advantage over others, which leads to a better biological fitness and hence leave more offspring. Several strategies can be acquired for calculating the fitness. In a subsequent study the innate language acquisition device is utilized, which represents the ability of each newborn child to learn languages (Hurford, 1989). The counterpart *sociocultural* models are not based on an evolutionary process, but rather see a language as a dynamic system where cultural choice and successful communication serve as basic principles. The Naming Game is one example for such a model.

First studies on the Naming Game have been conducted in (Baronchelli et al., 2006) which are based on an experiment from the artificial life community (Steels, 1995) known as Talking Heads. In the simulations all N agents are connected - i.e. the graph is complete - which corresponds to the mean-field case in statistical mechanics. It was found that the system exhibits a structure in which three stages take place corresponding to a S-shaped curve, which can be observed in the spreading of new language conventions. These stages turn out to be common in Naming Game simulations. First the number of total words grows following $N_w(t) = 2t$ and the number of different words $N_d(t) = t$ as a function of the time t. Here the games can be seen as uncorrelated, hence the agents get populated with random words. After a time of $O(N)$ and a memory need of $O(N^{3/2})$ - i.e. every agent communicated once - the second stage is reached. Neighboring nodes now have words in common and collective behavior emerges. The success rate $S(t)$ assigns 1 for a successful communication and a 0 for an error. These outcomes are averaged over all realizations. The rate evolves in that time span as $S(t) = 3t/N^2$. The third stage is reached when $N_w(t)$ reaches its maximum. This transition is referred to as *symmetry breaking*. An interesting property of this phase is that the evolution towards global consensus takes on very steep and scales even steeper with a growing system size. A shared dictionary establishes after a time of $O(N^{3/2})$.

On low-dimensional lattices the observed behavior is similar (Baronchelli et al., 2006).The system first evolves local clusters which then slowly coalesce to global consensus. The average cluster size grows following $\sqrt{t/N}$ until convergence is reached by $O(N^{1+2/d})$ for $d \leq 4$ and a total memory need of $O(N)$.

For small-world networks the Watts-Strogatz algorithm (Watts and Strogatz, 1998) is utilized (Dall'Asta et al., 2006). Starting from a one-dimensional lattice connected at both ends (i.e. a ring lattice) links are rewired according to the probability p. If p is chosen as $1/N \ll p \ll 1$ a small-world structure is obtained. For this topology the memory used can be drastically reduced from $O(N^3)$ for a one-dimensional lattice to $O(N^{1.4})$, while the convergence times have shown to be much faster depending on the parameter p.

Small-world and scale-free networks have been further analyzed in (Dall'Asta et al., 2006). It is shown that parameters like clustering coefficient or exponents for the degree distribution don't have a lot of impact on the convergence time. This is generally given by $O(N^{1.4\pm0.1})$. It is concluded that networks following small world characteristics constitutes an trade-off between mean-field "temporal efficiency" and regular lattice "storage optimization".

Another subject under investigation were networks consisting of fully-connected cliques which are interconnected by one link. The simulations comprised one word for each cluster. Exactly this result was picked up and further analyzed (Lu et al., 2009). The friendship network utilized in this study exhibits a structure of four clusters. These clusters are composed of approximately 1,000 students which are separated by school grade, i.e. high and middle school and by race, i.e. blacks and whites. The Naming Game simulations resulted most likely in four clusters in the short-term and in three on the long run. This can be explained by homophily which intensifies over time, i.e. the races mix up in younger years and separate with their age. A further point of interest was consensus engineering on that network. The starting point for this simulations were a typical outcome of the Naming Game with three co-existing clusters. The target was to "indoctrinate" the agents with an preferred opinion. The opinion was induced through committed agents, which do not accept any new words and always transmit their own. The main goal was how the committed agents are placed in the network and how many of them are needed to induce global consensus. The strategies for the selection were:

- *Degree ranking*. Agents with the highest degrees
- *Hop-distance proximity to the core cluster*. Agents outside but not farther away than two hops from the core cluster

- *Betweenness*. Agents with the shortest paths to all others

For comparison, a fourth strategy in which the agents were placed randomly on the network was used. Three features could be observed. First, a small fraction of committed agents is needed to induce global consensus. Second, the number of surviving runs (runs with more than one opinion) decays exponentially over time. Third, the rate at which global consensus is reached is a function of the committed agents, but it quickly saturates. The selection method turned out not to make a notable difference, but they are all clearly more effective than placing the agents randomly on the network. Another approach for influencing the agents was fostered. Started again with the meta-stable state with three co-existing opinions, the original Naming Game was played, but with a certain probability p one exogenous word (opinion) was used. The intention was to simulate global external influence, which can be interpreted as mass media influence. The findings yielded that even for extremely small values of p the fraction of surviving runs decays exponentially.

The term *inter-connectivity* was introduced in another study (Lambiotte and Ausloos, 2007). The microscopic interactions in the agent-based approach were defined by a variant of the majority rule model. In this model a random node is selected and with the probability q one of the two predefined opinions is randomly assigned to it. With the probability $1-q$ two neighbors are selected, which form the *majority triplet* along with the node selected in the first place. The whole triplet is then assigned the opinion of the *local majority*. The underlying topology exposes a coupled random network, which corresponds to a Stochastic Block Model with two communities in this thesis. The model was solved analytically and confirmed by simulations.

A distinction between three stages was made: The *disordered regime* where no collective behavior emerges; the *asymmetric regime* where two stable clusters co-exist and the *symmetric regime* where all nodes share the same opinion. The results showed that an outcome with a disordered regime solely depends on the random flips governed by the parameter q and takes place in the interval $]3/5,1]$. The transition between the asymmetric and the symmetric states occurs at $\nu \approx 0.1547$ with $q = 0$.

The same parameter ν was investigated employing a variation of the Axelrods model (Candia and Mazzitello, 2008). In this study it was termed *intercommunity connectedness*. The underlying topology was as well synthesized utilizing coupled random networks. In this model one node is selected. If it belongs to the first community an interaction with a constant vector takes place with probability M. In this way mass media is simulated. If the node is not influenced by the mass media in this timestep a random neighbor is chosen. With probability r a *cultural drift* occurs and one cultural feature is mutated at random. Otherwise the original version of the model is conducted.

For the results with $M = 0$, a distinction between three phases is made: The *monocultural phase* in which only one opinion exists; the *bicultural phase* with two opinion clusters and the *multicultural phase* where several opinions prevail. The transition limits are drawn at the point where the probabilities for two phases are equal. The transition towards the multicultural phase merely depends on the random noise r. Outcomes where two opinions prevail (bicultural phase) are analogously mainly governed by the *intercommunity connectedness* ν. The transition between the monocultural and the bicultural phase for small values of r is at $\nu \approx 10^{-3}$ which was also roughly shown analytically.

Chapter 3

Methodology

This chapter is arranged in four sections. First, the Naming Game is described in detail in section 3.1. Afterwards, in section 3.2 the theoretical background for the generation process of synthetic networks is given. The spectral partitioning, on which the community detection relies, is explained in depth in section 3.3. Finally, in section 3.4 the k-core decomposition is described which was utilized for a better visualization of the network structures.

In the Naming Game each node holds an inventory of words. In this thesis, each word is viewed as an opinion, whereas the inventory describes a mindset. The nodes start with an empty mindset. In each atomic timestep one node is selected as speaker and one of its neighbors as listener. If the speaker has no opinion - i.e. his mindset is empty - he chooses a random one. He then transmits the opinion to the listener. If the listener already knows about the opinion - i.e. he holds the opinion in his mindset - both, the speaker and the listener reconcile their mindsets to this single opinion. Otherwise the listener learns it and adds it to his mindset. These interactions take place until no further significant changes occur on the network. Depending on the modular topology of the network, either global consent is reached or multiple opinions prevail.

The synthetic networks are generated according to a Stochastic Block Model. In this model, the number of communities is stated in the first place. In this thesis, the model is defined by two probabilities. One for the probability of links between nodes belonging to the same community and one for links between different communities. The quotient of them depicts the *inter-connectivity* ν. On networks built from different values for ν, the Naming Game is played until a final stable state is reached. For each valuation of ν multiple simulations are carried out. Based on the number of opinions in the stable state networks generated with a certain *inter-connectivity* are classified in one out of three categories: An *asymmetric structure*, where the number of opinions matches the number of communities in the Stochastic Block Model in the vast majority of the simulations. A *metastable structure* where the number of opinions differ in the simulation outcomes. Finally a *symmetric structure* where one single opinion prevails in the vast majority of the simulations, i.e. global consent is reached.

Spectral partitioning is based on the LaPlacian matrix of the graph. It depicts the negative adjacency matrix with the node degrees on its diagonal. The eigenvector corresponding to the second smallest eigenvalue is called the *Fiedler vector*. The position of the nodes are sorted according to their value in the vector. The nodes are then separated in communities at the point where gaps in the vector are located. The analysis is conducted on the synthetic and empirical networks. The point of investigation is how good the opinion clusters from the Naming Game can be predicted through spectral partitioning.

In the k-core decomposition, nodes with a degree lower than k are removed. The degree states the number of links incident to the node, i.e. the number of its neighbors. Starting with a value of 3 for k it is increased up to the maximum degree occuring in the network.

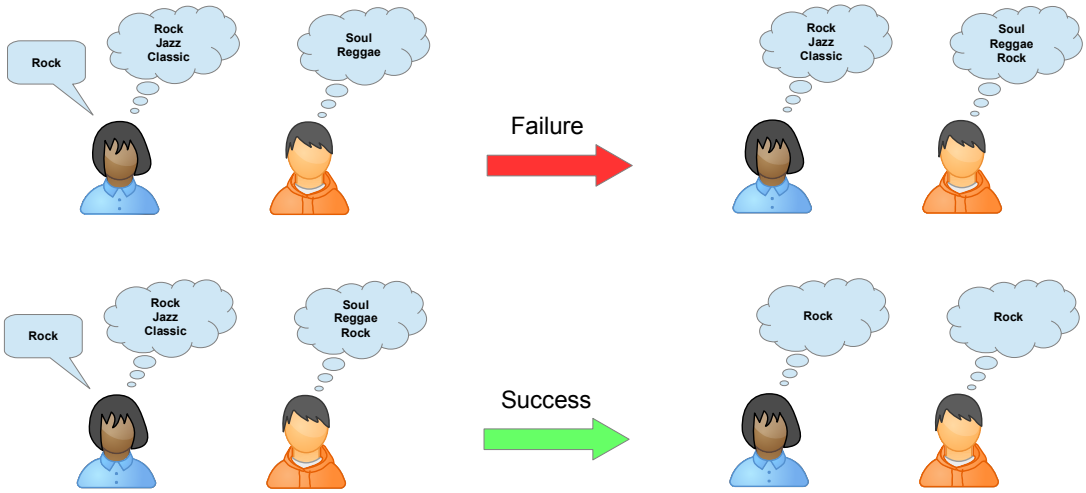


Figure 3.1.: *Elementary communication in the naming game.* The speaker transmits a random word. If the listener doesn't know the word he adds it. Otherwise both collapse their inventory to the transmitted word.

3.1. The Naming Game

The Naming Game is originally a model describing a bootstrap process for establishing a common vocabulary for a *single* entity among a set of agents through *pairwise* communications. One could think of constituting a novel term for an entity like a geometric object, a living organism, a natural phenomenon or tags for a website.

In the Naming Game every agent is endowed with an inventory of words which represents the contemporary words for the entity. At each timestep t ($t = 1, 2, 3, ...$) a random node is selected and labeled as *speaker*. Afterwards one of its neighbors is chosen again randomly and referred to as *listener*. The speaker then chooses a random word from his inventory and transmits it to the listener. If the inventory is empty a new word is created. The communication is now termed *success* if the listener holds the word in his inventory. In this case both peers agree on that single word and collapse their inventories to it. If the listener doesn't know the word he learns it and thus adds it to his inventory. Such a communication is termed *failure*. The process of an elementary communication is sketched in Figure 3.1. One simulation lasts until a stable configuration is reached and doesn't change on the long run. Due to a lot of randomness in the model multiple simulations need to be conducted and compared. The number of iterations for the networks were chosen empirically.

A very important property of the model is that no global control is involved, i.e. it is self-organized. As it is typical for agent based modeling a set of *microsopic* interactions is defined which leads to *macroscopic* phenomena that can be observed. The Naming Game originates from the research area of semiotic dynamics and is in this study utilized to represent opinion spreading. Although it does certainly not take all elements from sociology into account, it can serve as a minimal bound which was already observed on empirical networks (Lu et al., 2009). It can be interpreted as follows: One has several opinions about an issue. As he talks to other people known to him, he learns about new views. If both agree, they both then share the common opinion and forget about the others. On the long run several opinion clusters emerge. How much clusters evolve clearly depends on the topology of the network. If someone knows only a few people, his opinion is only shaped by them. Global external influence like mass media, the internet or advertisement is not considered in this study. Another important property of the model is the selection of the speaker and listener. Three different strategies exist (Dall'Asta et al., 2006):

- *The direct Naming Game.* The strategy used in this study - the speaker is selected first - which is the most intuitive one. High-degree nodes will preferentially act as hearers, since they have a high

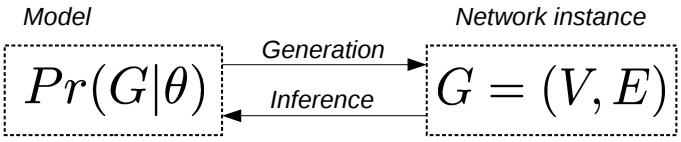


Figure 3.2.: *Generation and inference.* For the *generation* of an instance graph G the parameters θ are applied. Through *inference* these parameters are estimated.

probability to be chosen as a neighbor of a speaker.

- *The reverse Naming Game.* The inverse strategy, where the hearer is selected first. Hubs are more likely to be selected as speaker.
- *The neutral Naming Game.* An edge is selected first, speaker and hearer are selected randomly afterwards.

In this study the total number of different words N_d is limited to 100. This results in a lower memory need and faster convergence times, though it can lead to biased outcomes. These outcomes can nevertheless be identified because there are always several simulations for one network.

The simulations of the Naming Game were realized on synthetic and on empirical networks and the results are then compared.

3.2. Stochastic Block Model

Stochastic block models were used in this study for the generation of synthetic networks. They were first investigated by mathematical sociologist in (Holland et al., 1983) and further analyzed by (Wang and Wong, 1987). The model consists of the following parameters:

- Number of nodes N.
- Non-overlapping blocks (partitions) $B_1, B_2, ..., B_k$.
- $N \times 1$ vector of node type indices s
- $k \times k$ stochastic block matrix M where M_{uv} states the probability for a link (dyad) between a node of type u and a node of type v

The model incorporates two basic assumptions:

- The probability for link between two groups is *independent and identically distributed*. Mathematically the probability for two links (i, j) and (k, l) p_{uv} for $i, k \in B_u$ and $j, l \in B_v$ is $i.i.d$ for $u, v = 1, 2, ...k$.
- The link probabilities between different groups are *mutually independent*.

Both assumptions together form *structural equivalence*.

This model can be utilized for generation and the reverse process inference of a network which is illustrated in Figure 3.2. To draw a network instance G a coin is flipped for each of the $\binom{N}{2}$ possible links. Hence the existence of a link is determined by an outcome of a Bernoulli experiment.

Depending on the probabilities defined in the blockmatrix M several classes can be observed:

- *Random graphs*. The probabilities are all equal, i.e. $M_{uv} = p$. In this case the model is equivalent to a Erdös-Rényi random graph (Erdös and Rényi, 1959). Please note that on links between one group this is always the case.

- *Assortative communities*. Individuals on the network tend to connect to other individuals in their community, i.e. $M_{uu} > M_{uv}$ for $u \neq v$.
- *Disassortative communities*. Individuals tend to connect to others in different groups, $M_{uu} < M_{uv}$ for $u \neq v$.

In this thesis the groups are always of equal size and only two probabilities are defined. The first is M_{uu} which is termed p_{in} and the second is M_{uv} which is termed p_{cross}.

On average the nodes belonging to the same group are connected with a degree of $k_{in} = p_{in} * (N_{nodes} - 1)$ where N_{nodes} is the number of nodes per group. Nodes belonging to different groups on the other hand are on average connected to $k_{cross} = p_{cross} * (N_{nodes} * (N_{groups} - 1))$ nodes where N_{groups} is the number of groups.

3.3. Spectral Partitioning

In this work the community detection heavily relies on spectral partitioning. Utilizing the eigenvectors of the adjacency matrix of graphs was first considered in (Donath and Hoffman, 1972). This approach was further investigated by *Miroslav Fiedler* who introduced the term *algebraic connectivity* for the second eigenvalue of the Laplacian matrix (Fiedler, 1973) and partitioning based on the corresponding eigenvector (Fiedler, 1975) which carries the name *Fiedler vector*. Since then several approaches for spectral graph partitioning evolved (Spielmat and Teng, 1996).

A graph $G = (V, E)$ consists of a vertex V and an edge set E. Let us first introduce the well-known adjacency matrix $A = A(G)$ which is of the size $N \times N$ and defined by

$$A_{ij}(G) = \begin{cases} 1, & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Throughout this study only simple graphs are used, i.e. graphs are undirected and exhibit no loops. Therefore the matrix is symmetric and every element on the diagonal is 0.

The degree matrix $D = D(G)$ is again of size $N \times N$ and defined by

$$D_{ij}(G) = \begin{cases} deg(v_i), & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

where $deg(v_i)$ returns number of edges incident to the vertex v_i. This is commonly referred to as degree of a vertex.

Based on those two matrices the graph Laplacian can be calculated as

$$L = D - A$$

An eigenvalue λ and it's corresponding eigenvector v of an arbitrary $N \times N$ matrix M is given by

$$Mx = \lambda x$$

The spectrum of the matrix M is the multiset of its eigenvalues. We introduce the convention $\lambda_1 \leq \lambda_2 \leq ...\lambda_n$. For this study the spectrum of the graph Laplacian is of interest. Following properties can be observed:

- Since L is a real symmetric matrix all if it's N eigenvalues are real.

- The number of eigenvalues equal to 0 corresponds to the number of connected components in the graph, thus $\lambda_1 = 0$ and the corresponding eigenvector is the all ones vector $\mathbf{1}$
- Only connected graphs are analyzed, thus $\lambda_2 > 0$
- Each eigenvector $v = (v_1, v_2, ..., v_N)$ satisfies

$$\sum_{i=1}^{N} v_i = 0$$

since $\mathbf{1}$ is always an eigenvector of the graph Laplacian and the eigenvectors of a symmetric matrix are orthogonal

λ_2 is referred to as the *algebraic connectivity* or *Fiedler value* and its corresponding eigenvector *Fiedler vector*. These values are utilized for community detection in this study. The algebraic connectivity depicts a metric for how connected a graph is. Considering the diameter (number of edges on the longest shortest path) D of the graph, the *Fiedler value* is bounded by (Gross and Yellen, 2003), (Mohar and Alavi, 1991)

$$\frac{1}{ND} \leq \lambda_2 \leq \frac{4}{ND}$$

Concerning spectral partitioning several methods can be used (Spielmat and Teng, 1996). The idea is to split the nodes in the *Fiedler vector* $v = (v_1, v_2, ..., v_N)$ in two groups B_1 and B_2 which is called *Fiedler cut*. Therefore a splitting value s is defined, such that $v_i \in B_1$ if $v_i \leq s$ and $v_i \in B_2$ if $v_i > s$. Such an approach for finding s can be:

- *Bisection.* s is the median of v
- *Sign cut.* Partitioning is based on the sign, i.e. $s = 0$
- *Gap cut.* v is sorted and s is chosen that it separates the groups based on the largest gap

3.4. *k*-core decomposition

For the structural analysis of networks k-core analysis has been proposed in (Seidman, 1983). Such an analysis examines the cohesiveness or "knittedness" of networks. A k-core of a graph G is a subgraph such that each node has at least the degree k. The network can be seen as successively enclosed k-cores, which can be compared to a russian nesting doll. The foundation for this approach builds the definition of cliques (Luce and Perry, 1949). In a clique every node has to know all the other ones, therefore a k-core can be seen as a looser definition. Clique analysis on empirical networks exposed two sincere difficulties. Such networks rarely contain cliques and they furthermore frequently overlap.

The algorithm for gathering k-cores is very simple and fast.

1. Set k to 2
2. Remove every vertex v_i with $deg(v_i) < k$
3. If there are nodes left increment k by 1 and goto 2

With respect to k the resulting graph represents the robust core of the network.

Chapter 4

Framework

The framework was throughout written in Python 2.7. It is compounded of altogether four independent projects:

- *The Naming Game*. The simulations for the Naming Game (see 3.1). Every atomic step is saved in textfiles for further analysis.
- *Analyzer*. Takes the output files from the Naming Game as input and iterates its files. Although it produces reasonable output it is merely a preprocessor for the analysis. The analyzer can also be used for a quick overview of a simulation.
- *Analysis*. Contains several scripts for the in-depth analysis of a simulation. Only the output from the analyzer is used anymore. This results in improved performance and extensibility.
- *Stochastic Block Model*. Implementation of the stochastic block model (see 3.2). It is utilized for the generation of the synthetic networks. Furthermore it contains scripts for running the Naming Game on the generated network and subsequent analysis.

Each of the projects is described in detail in the oncoming sections.

4.1. The Naming Game

The project uses the object-oriented features of Python which allow an easy reuse. One important design goal of the code was exchangeability of the different components. Therefore the strategy design pattern was stressed. The class diagram can be found in the appendix (see A.1). For graph operations the networkx package was incorporated.[1]

The project is divided in following modules:

- *game*. Contains the core of the naming game. The classes are:
  - `NamingGame`
  - `RoundObserver`
- *gameio*. The code for IO operations. The classes are:
  - `IOUtil`
  - `DifferentialGameStringPresenter`

[1] https://networkx.github.io/

  - `GameLogger`
  - `VocabularyIO`
  - `InputArguments`
  - `GameCreator`
- *meeting*. Holds the classes which are used how a meeting between nodes is established. The classes are:
  - `MeetingStrategy`
  - `RandomNeighborMeeting`
- *communication*. The module for the communication between the different nodes in a meeting. The classes are:
  - `GraphManipulationAction`
  - `VocabularyChangeType`
  - `VocabularyChangeAction`
  - `CommunicationStrategy`
  - `NamingGameCommunication`
- *selectors*. The selectors for choosing the committed agents. The classes are:
  - `BinaryOperator`
  - `NodeSelector`
  - `SimpleDictionarySelector`
  - `RandomSelector`
  - `HighestDegreeSelector`
  - `HighestBetweennessSelector`
  - `CoreClusterProximitySelector`

The main class is `NamingGame`. It needs to be parameterized and started with the `play()` method afterwards. The program flow is always receiving the attendees of a meeting through the meeting strategy, communication of those attendees according to the communication strategy and finally notifying all registered observers (see Figure 4.1). This is done `NamingGame.iterations` times.

In this work only one meeting strategy was implemented represented by the class `RandomNeighborMeeting`. It is the *direct naming game* where one node is chosen randomly and one of it's neighbors is then chosen again at random.

For communication as well only one strategy was implemented as the `NamingGameCommunication`. It follows the rules explained in section 3.1.

Committed agents were first introduced in (Lu et al., 2009). Such an agent never accepts a new word, but always uses his own single one if he is the speaker. Even though committed agents were not used in this study all strategies in (Lu et al., 2009) were implemented. For a listing of those see section 2. Therefore the *selectors* module was integrated. How those agents are treated is subject to the meeting and communication strategy.

Output in the naming game is produced by round observers. By default there is only one registered which is an object of the class `GameLogger`. Upon initialization it writes the full representation of the graph to `GameLogger.filename_complete_graph`, which is per default `complete.txt` (Paths are always joined with `GameLogger.output_directory`). The file format follows some simple rules: For each node a text block is created followed by a new line character. A block consists of:

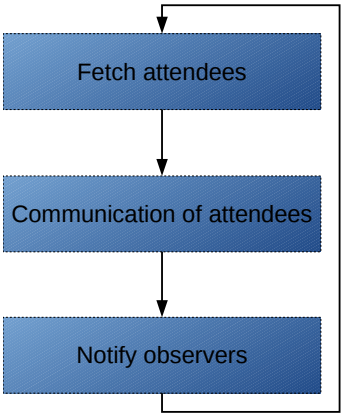


Figure 4.1.: *Program flow of the naming game.* The meeting strategy fetches the attendees. These attendees communicate according to the communication strategy. At the end of each iteration every observer is notified.

```
<nodeid>
<nodevocabulary>
{<neighbor-nodeid><EOL>}
```

where `<nodevocabulary>` represents the words from the node vocabulary separated by a comma. After the simulation has finished a file with the same format is written to `GameLogger.filename_complete_end_graph`, which is per default `complete_end.txt`.

The elementary steps at each iteration are written to the full and small differential files. The full differential files are snapshots which are taken after `GameLogger._step_size_differentials` small differential files are written. The file format follows the same as for the full graph representation but only nodes were changes occurred are written. The small differentials contain every step conducted during each round. If a round is a failed communication one or two nodes get a new word. For those nodes a line is added with the format `<nodeid>+<word>\n`. If a communication is a failure they both reconcile their vocabularies to one single word. Therefore the line `<nodeid>=<word>` is added. For a finished round a new line character is inserted.

The Naming Game simulations are started via the `main.py` script. The following parameters are mandatory:

- `--input-graph`. The path to the graph. It can be absolute or relative. The format of the file needs to correspond to the parameter `--input-graph-type`.
- `--iterations`. The number of iterations t_{max} for the Naming Game. Needs to be a positive integer

The following parameters are optional:

- `--input-graph-type`. The format of the input file. The only valid and default value is `edge-list`
- `--input-vocabularies`. The path to input vocabularies. The format is specified below.
- `--agent`. The strategy for the selection of committed agents. The possible values are `random`, `degree`, `betweenness` and `cluster-proximity`.
- `--agent-count`. The count of agents which are selected by the selection strategy.
- `--meeting`. The meeting strategy. The only valid and default value is `random`.
- `--meeting-attendees`. The number of attendees which are selected for a meeting.
- `--communicate`. The communication strategy. The only valid and default value is `naming-game`.

- `--count-full-differentials`. The approximate number of full differential files.
- `--count-small-differentials`. The approximate number of small differential files in total.
- `--output-directory`. The path to the output directory. It can be absolute or relative. The default value is `out`.
- `--overwrite`. A switch if the output directory should be overwritten in case it exists.

If an input vocabulary is passed to the program each line has the format `<nodeid>:<vocabulary>` where `<vocabulary>` is composited of the words of the vocabulary separated by a comma.

4.2. Analyzer

The analyzer consumes the output of the Naming Game and can serve as either a preprocessor for the analysis or as a printer for statistics which offer a quick overview of the outcome of a simulation. The design of the project is similar to that of the Naming Game. It is built out of two modules:

- *analyzer*. The core module containing the analyzer classes and some observers. The classes are:
  - `Analyzer`
  - `StaticAnalyzer`
  - `WalkingAnalyzer`
  - `RoundObserver`
  - `WordOccurrenceObserver`
  - `VocabulariesObserver`
  - `AnalzyerUtil`
  - `AnalyzerVocabularyChangeAction`
- *analyzerio*. The code responsible for the io operations. The classes are:
  - `IORoundObserver`
  - `VocabularyChangeObserver`
  - `WordDistributionObserver`
  - `GroupObserver`
  - `InputArguments`
  - `AnalyzerCreator`
  - `GraphReader`

The analyzer is started with the `main.py` script. Only the parameter `--directory` which is the path to the Naming Game simulation directory is mandatory. The following parameters are optional:

- `--overwrite`. Overwrite the analyzer files if they already exist.
- `--from`. Read the simulation files from a particularly full differential file.
- `--to`. Read the simulation files only to a particularly full differential file.
- `--step-size`. Write the output files with the specified step size.
- `--stats`. Call the static analyzer and write the output only to the standard output.
- `--verbose`. Tell the static analyzer to be verbose in its output.

Based on the input arguments either an object of the `StaticAnalyzer` or the `WalkingAnalyzer` class is created. Those modes of operation are explained in the following two subsections.

4.2.1. Static analyzer

The static analyzer reads solely the full differential files and prints a quick overview about a simulation to the standard output. Only the `GraphReader` is used therefore, no observers are needed. The static analyzer is built from the `AnalyzerCreator` if the switch `--stats` is set in the `main.py` script. The first block of output states the number of nodes and edges and if the simulation used input vocabularies. A sample output could be:

```
Nodes: 5242
Edges: 14496
Input vocabularies: no
```

The overhaul information is succeeded by the word distributions of the full differentials. For each file a block of the following structure is generated:

```
Differential <differential-id>
  Word count: <number-of-different-words>
  Distribution: <word-distribution>
```

where `<word-distribution>` lists each word in the game along with the number of its occurrences. A sample output is:

```
Differential 1
  Word count: 34
  Distribution: [('ck', 4), ('ca', 4), ('an', 4), ('cr', 4), ('bl', 4), ('bo',
     4), ('bz', 4), ('e', 4), ('i', 4), ('l', 4), ('aa', 2), ('cn', 2), ('aj',
     2), ('al', 2), ('ce', 2), ('aq', 2), ('ap', 2), ('av', 2), ('cp', 2), ('
    cu', 2), ('ad', 2), ('ak', 2), ('be', 2), ('bm', 2), ('bi', 2), ('bv', 2),
     ('bw', 2), ('by', 2), ('g', 2), ('k', 2), ('m', 2), ('q', 2), ('s', 2),
    ('z', 2)]

Differential 2
  Word count: 56
  Distribution: [('cr', 12), ('ap', 8), ('e', 8), ('al', 6), ('an', 6), ('bl',
     6), ('l', 6), ('ck', 4), ('ca', 4), ('aq', 4), ('bb', 4), ('bo', 4), ('bi
    ', 4), ('bw', 4), ('by', 4), ('bz', 4), ('i', 4), ('k', 4), ('q', 4), ('aa
    ', 2), ('ac', 2), ('ab', 2), ('ae', 2), ('cn', 2), ('cj', 2), ('ai', 2),
    ('ah', 2), ('aj', 2), ('ce', 2), ('au', 2), ('at', 2), ('av', 2), ('cp',
    2), ('cu', 2), ('ad', 2), ('co', 2), ('ak', 2), ('be', 2), ('bg', 2), ('bm
    ', 2), ('cf', 2), ('bv', 2), ('bp', 2), ('ao', 2), ('br', 2), ('c', 2), ('
    d', 2), ('g', 2), ('f', 2), ('j', 2), ('m', 2), ('o', 2), ('s', 2), ('r',
    2), ('y', 2), ('z', 2)]
```

Additionally the switch `--verbose` can be passed to the `main.py` script. In this case the final groups for the words are appended. For each group a line of the format `<word>: <node-ids>\n\n` is created. A sample output would be:

```
u: [8443, 17595, 12687, 15596, 5407, 24293]

y: [8261, 24960, 9408, 1695, 10180, 18600, 19380, 11609, 4822, 21860, 2334,
    1403, 14543, 21221]
```

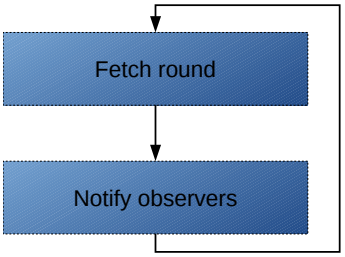


Figure 4.2.: *Program flow of the walking analyzer.* Each new round is read through the `GraphReader` by the `WalkingAnalyzer` and the changes are then submitted to the observers.

4.2.2. Walking analyzer

The walking analyzer is the common way of operation for the analyzer. It iterates over all small differential files and writes its output to the simulation directory. After each round read by the `GraphReader` all observers are notified. This is illustrated in Figure 4.2.

There exist two types of observers. The ones directly inherited from the `RoundObserver` class, which gather information from the input files and store it. Those are instances of either `VocabulariesObserver` or `WordOccurrenceObserver`. These observers are then further the information holders for the subclasses of `IORoundObserver`, thus the order in which they are registered is important.

The `VocabularyChangeObserver` writes the number of different words each `VocabularyChangeObserver.step_size` round to `VocabularyChangeObserver.filename` which is `vocabulary_changes.txt` per default. It needs an instance of the `WordOccurrenceObserver` prior to it registered in the analyzer. The file carries a header of the following structure:

```
#<information about the file as a comment>
@start round: <start-round>
@step size: <step-size>
```

for later information and processing instructions about the file.

The `WordDistributionObserver` writes the word distributions for each `WordDistributionObserver.step_size` round to `WordDistritubionObserver.filename` which is `distributions.txt` per default. It also needs an instance of the `WordOccurrenceObserver` prior to it registered in the analyzer. Each line has the format `<list-begin>{('<word>', <occurrences>), }<list-end>` where `<list-begin>` is the `[` and `<list-end>` is the `]` character. The file contains the same header as the `VocabularyChangeObserver`.

The `GroupObserver` writes all words and the corresponding nodes which carry only this word in its vocabulary after the game has finished to `GroupObserver.filename` which is `groups.txt` per default. Each line has the format `<word>: <list-begin>{<node-id>, }<list-end>`.

After the walking analyzer has processed the simulation directory it usually contains following files:

- `diff-<full-diff-id>-<small-diff-id>.txt`. The small differentials.
- `full-<full-diff-id>.txt`. The full differentials.
- `complete.txt`. The full graph representation before the simulation.
- `complete_end.txt`. The full graph representation after the simulation.
- `vocabulary_changes.txt`. The number of different words per round.
- `distributions.txt`. The word distributions per round.
- `groups.txt`. The node ids of the groups for each word after the finished simulation.

4.3. Analysis

The analysis project is compounded of several scripts which all address their own purpose and thus are to a large extent independent of each other. They are all written following the design paradigm that they can be called from the command line or just be parametrized in the header and executed. For a complete analysis the superscript `run_all.py` can be called which invokes all the other scripts. In the following subsections every script is described in detail. It should be noted that the script described in the section *Calculate cores* (see 4.3.1), *Spectral analysis* (see 4.3.2) and partially *Degree distributions* (see 4.3.3) investigate the network while the others inspect the simulations.

Throughout this section the well-known Zachary's Karate Club network (Zachary, 1977) was utilized. A plot of the whole graph can be found in Figure 4.3. An representative simulation of the Naming Game was conducted which involved 3,000 iterations. The outcome exposed two groups, one consisting of 29 and the other of 6 nodes. Each group is labeled with one word, a node is part of it if the word is held in the inventory.

4.3.1. Calculate cores

The script `calculate_cores.py` is an implementation of the k-core decomposition described in section 3.4. Starting from $k = 3$ for each k-core two plots are generated:

- `core-<k>.png`. The plot of the k-core.
- `core_with_neighbors-<k>.png`. The plot of the k-core including its neighbors. A neighbor is a node which is not included in the core but adjacent to it. The neighbors are plotted in red.

The cores are written to the file `cores.txt`. For each k a block of the following structure is generated:

```
Coresize: <k>
{<component-i><EOL>}
```

where `<component-i>` is the i-th connected component of the core. It is a list of the corresponding nodes in the format `<list-begin>{<node-id>, }<list-end>`.

The script takes following parameters as input:

- `--filename-network`. The filename of the network. It needs to be in the edge-list format of the networkx package.
- `--output-directory`. The directory where the output files are written.

The output is illustrated in Figure 4.4. Since the network has a maximum degree of 4, the plots for the 3- and 4-cores are shown.

4.3.2. Spectral analysis

The spectral analysis in the script `spectral.py` is based on the spectral partitioning described in section 3.3. It is the only script which is built according to object-oriented design principles for better reusability. Usually the methods `write_analysis()` and `write_groups_treshold()` of the only class `SpectralClustering` are invoked. The class relies heavily on the Pythons *numpy*[1] package. The whole output is written to the directory which is passed in the constructor.

[1] http://www.numpy.org/

The `write_analysis()` method first plots the whole graph to `graph.eps` (for an example see Figure 4.3).

Afterwards the *fiedler value* and the *fiedler vector* are calculated. The *fiedler vector* is then sorted. The positions of the nodes on the adjacency matrix are reordered according to their position in the *fiedler vector*. The unmodified adjacency matrix is saved to `matrix_initial.eps` and the sorted one to `matrix_sorted.eps`. The mapping for the sorted matrix is written to the file `adjacancy_matrix_mapping.txt`. Beginning from the first node each position is written to one line.

A plot of the *fiedler vector* and the sorted one is saved to `vector_unsorted.eps` and `vector_sorted.eps` respectively. Example plots are shown in Figure 4.6. The values of the sorted vector are saved to `vector_sorted.txt`. The *fiedler value* as well as the count of eigenvalues equal to 0 are saved to `spectral.txt`. The *fiedler value* for the Karate Club is 0.4685.

The `write_groups_treshold()` takes the float parameter `treshold` as input and writes the groups to `groups.txt`. Each group is separated by at least `treshold` in the sorted *fiedler vector*. Since the vector is normalized, this value varies significantly depending on the number of nodes.

4.3.3. Degree distributions

The `plot_histogram.py` script plots the degree distribution of the whole network and/or the groups from a simulation outcome depending on its input parameters. The plots are always histograms with a count of bins equal to the maximum degree.

If the `--plot-graph` switch is set a histogram of the degree distribution of whole graph is plotted to the joined path of the parameter `--graph-directory` and `degree_histogram.eps`. An example is shown in Figure 4.7. The values of the histogram are retained in the file `degree_distribution.eps.txt`. For each bin a line with the format `(<degree>, <node-count>)` is created.

If the `--plot-groups` switch is set the same plot is created but for each group in an outcome of an simulation. Therefore the file `groups.txt` (generated by the analyzer) needs to exist in the directory pointed to from the input parameter `--group-directory`. The plot is saved in the directory assigned with the parameter `--output-directory` as `group-<group-id>-degree_histogram.eps` with its data in `group-<group-id>-degree_histogram.eps.txt`.

The input parameters of the script are the following:

- `--plot-graph`. Plots the degree distribution of the graph if set. Needs the parameters `--filename-network` and `--graph-directory` set in order to work.
- `--filename-network`. The path to the network file. It needs to have the networkx edge-list format.
- `--graph-directory`. The output directory for the degree distribution of the whole graph.
- `--plot-groups`. Plots the degree distribution of each group if set. Needs the parameters `--filename-network`, `--group-directory` and `--output-directory`.
- `--group-directory`. Directory where the `groups.txt` file from the analyzer resides.
- `--output-directory`. The output directory for the degree distribution of the groups.

4.3.4. Group differences

A comparison of the groups from multiple simulations is made in the script `group_diff.py`. The output is written to the file pointed to from the input parameter `--output-filename`. All runs in the base directory `--input-directory` with the directory pattern given by `--run-directories-pattern` are examined. In each of the directories a `groups.txt` file from the analyzer needs to be found.

The header of the output file obtains the following structure:

```
Runs: <simulation-count>
Total groups: <total-group-count>
Unique groups: <unique-group-count>
Unsimilar groups: <unsimilar-group-count>
```

where `<simulation-count>` is the number of simulations and `<total-group-count>` is the total number of groups from all simulations. In the `<unique-group-count>` groups which match exactly are only counted once. Most interesting in this script are the unsimilar groups. These are found by the following algorithm:

```
unsimilar_groups = []

for run in runs:
    for group_run in run:
        is_similar = False

        for group_similar in unsimilar_groups:
            if group_similarity(group_run, group_similar) > treshold:
                is_similar = True
                break

        if is_similar:
            if len(group_run) > len(group_similar):
                unsimilar_groups.remove(group_similar)
                unsimilar_groups.append(group_run)

        if not is_similar:
            unsimilar_groups.append(group_run)
```

Listing 4.1: Detection of unsimilar groups

where the treshold is a percentage which is usually 0.8. The algorithm basically merges groups which are similar to at least 80 %. If two groups match only the larger one is retained.

After the header in the textfile the node ids of the unique and the unsimilar groups are stated.

Then the groups from the simulations are compared one by one. If the groups are identical a message of the format `Group <first-groupid> and group <second-groupid> are identical` is printed. Afterwards the same is done for the unsimilar groups.

The tail of the output file contains how many of the groups matched approximately (unsimilar groups) on average and how many new unsimilar groups were found in each simulation.

The input parameters of the script are the following:

- `--directory`. The directory where the run (simulation) directories are located and the output file is saved to.
- `--run-directories-pattern`. The pattern of the run directories relative to the input directory. The pattern needs to be in the format of the *glob*[1] module
- `--output-filename`. The path to the output file. Defaults to `groupdiff.txt`

[1] https://docs.python.org/2/library/glob.html

4.3.5. Statistical metrics

The script `plot_stats.py` is responsible for two plots:

- *Vocabulary changes*. The number of different words which coincide with the number of different groups over the time.
- *Word distributions*. A box plot of the word distributions over the time.

The vocabulary changes are plotted to the file `vocabulary_changes.eps`. The corresponding (x,y) tupels are written to the file `vocabulary_changes.eps.txt` in the format `(<x>, <y>)` where each line represents an entry in the plot.

The word distributions are presented as a box plot. The basis is the `distributions.txt` from the analyzer. The number of boxes is determined by the input parameter `--count-boxes`. For each word the occurrences are counted. These values are then sorted and grouped in four quartiles. A box is drawn with borders at the first and third quartile and the median displayed as a red line. The whiskers show the maximum and minimum value. The box plot also contains a subplot of the vocabulary changes with the same x-axis. Each of the values is again written to a line in the file `distributions.eps.txt` with the format `<list-begin>{<group-x-size>, }<list-end>`.

A sample of the two plots is displayed in Figure 4.8.

The input parameters of the script are the following:

- `--input-directory`. The directory which contains the `vocabulary_changes.txt` and `distributions.txt` files from the analyzer.
- `--output-directory`. The directory where the output is saved.
- `--count-boxes`. The number of boxes created in the box plot. Defaults to 30.

4.3.6. Group plots

All plots concerning group analysis are generated by the script `plot_groups.py`. The script depends on the files `adjacency_matrix_mapping.txt` and `vector_sorted.txt` from the spectral analysis described in subsection 4.3.2 The following plots are provided:

- *Adjacency matrix*. The adjacency matrix colored according to the groups.
- *Group eigenvector*. The portion of the sorted eigenvector where the nodes of the group are found. Nodes which lie within the range, but are not part of the group are emphasized.
- *Group subgraph*. The graph of the group including its neighbors.
- *Group subgraph including eigenvector nodes*. The same plot as the group subgraph, extended with the nodes which lie in between the group on the sorted eigenvector.
- *In-Out degrees*. The in- and out degrees of each node.

The shape of the *adjacency matrix* in the plot is always the same as in the spectral analysis (subsection 4.3.2). Every link which is not exclusive in one group - i.e. both ends don't have one equal word in their vocabulary - is referred to as an *inter-group* link and has the color red assigned. All of the other links are depicted in the color of their group. The plots follow the naming scheme `matrix_groups*.eps` depending on if they contain a legend and if their colors are asserted according to the positions of the groups in the matrix. An example of such an matrix is illustrated in Figure 4.9.

The *group eigenvector* plots display the portion of the eigenvector where the corresponding values of the group nodes reside. Values belonging to the group are plotted with blue crosses, while those not belonging to the group are depicted as green circles. For the approximation of the gaps to the remaining eigenvector red crosses represent the value of a neighboring node. The plots are written to `group-<group-id>`

-vector.eps. The data is written as (x,y)-tuples to group-<group-id>-vector.eps.txt corresponding to the format (<x>, <y>). The plots for the two groups are illustrated in Figure 4.12.

The group subgraph plots the nodes of the group in blue and its neighbors in red as a graph. The files are saved to group-<group-id>.eps. The plots are illustrated in Figure 4.11.

The *group subgraph including eigenvector nodes* plot is the same as the group subgraph plot, but it also contains the nodes where the values of the eigenvector lie within the groups range corresponding to the nodes displayed with a green circle in group eigenvector plots. The files are saved to group-<group-id>-vectorgroup.eps. The plots are illustrated in Figure 4.12.

The in-out degrees plots the in and out degrees of each node. The nodes are in ascending order of their in degree. The plots are saved to group-<group-id>-in_out.eps. The corresponding values are written to group-<group-id>-in_out.eps.txt in the format (<in-degree>, <out-degree>). The plots are illustrated in Figure 4.13.

The input parameters of the script are the following:

- --filename-network. The path to the network file. It needs to have the networkx edge-list format.
- --graph-directory. The directory where the files adjacency_matrix_mapping.txt and vector_sorted.txt from the spectral analysis reside.
- --run-directory. The directory where the file groups.txt from the analyzer resides.
- --output-directory. The directory where the output files are written to.

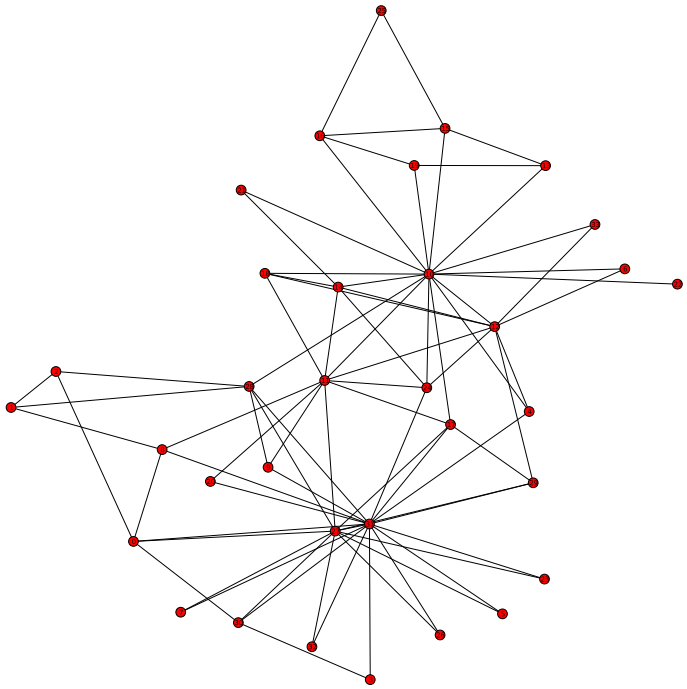


Figure 4.3.: *Plot of the Zachary's Karate Club network.*

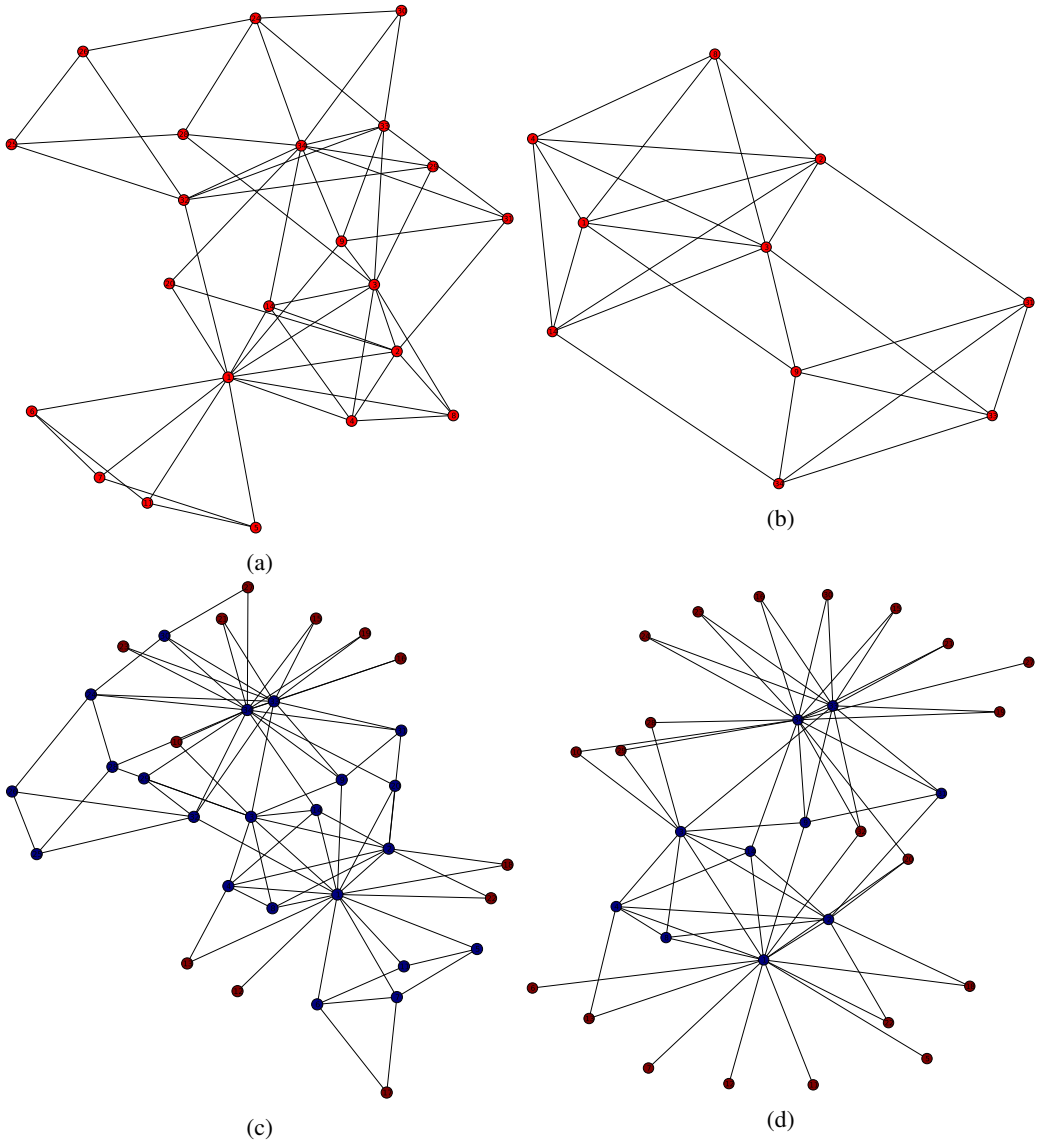


Figure 4.4.: *k-core decomposition of the Zachary's Karate Club network.* (a) 3-core network (b) 4-core network (c) 3-core network with its neighbors in red (d) 4-core network with its neighbors in red

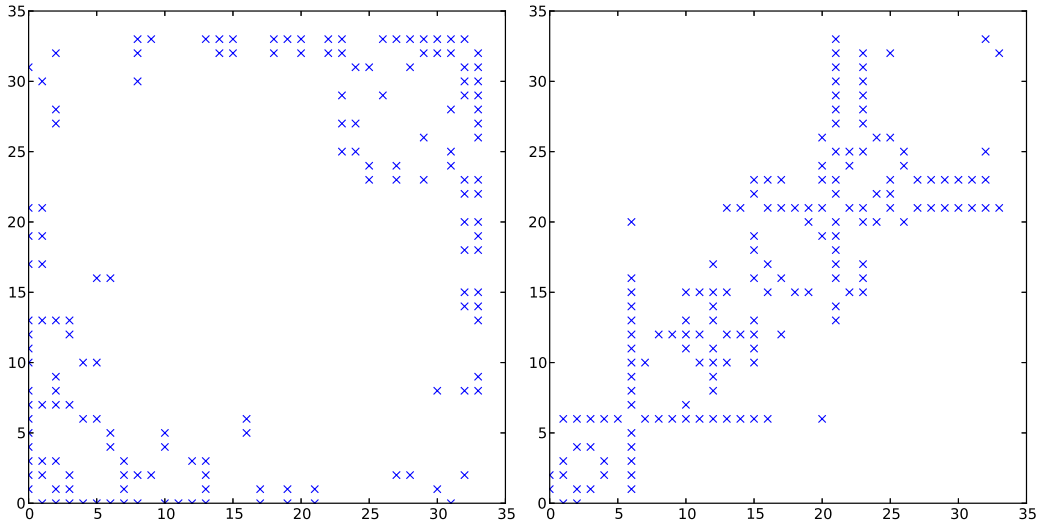


Figure 4.5.: *Unsorted and sorted adjacency matrices of the Zachary's Karate Club network.* (a) The adjacency matrix (b) The adjacency matrix with the node positions sorted according the their value in the *fiedler vector*

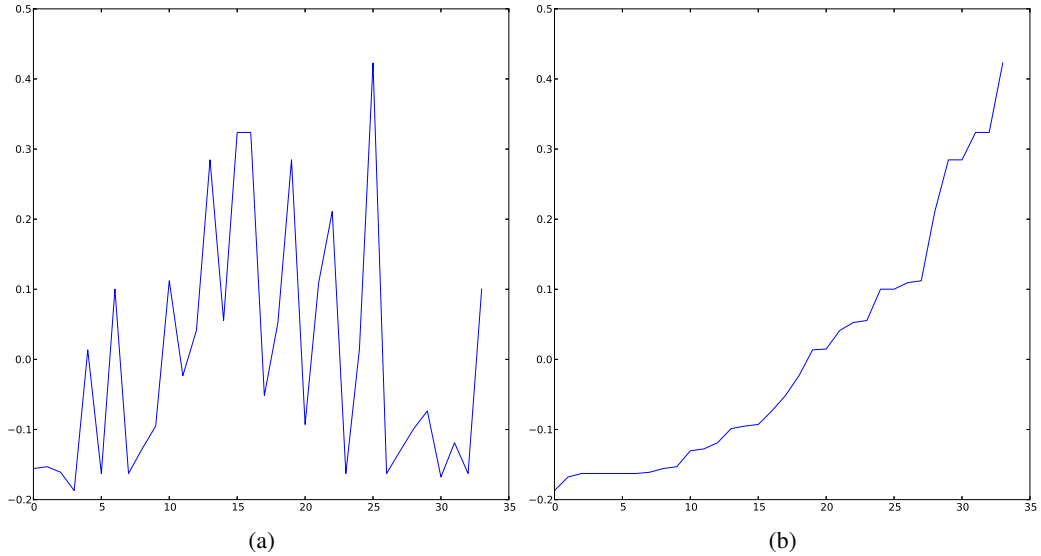


Figure 4.6.: *Unsorted and sorted fiedler vectors of the Zachary's Karate Club network.* (a) Unsorted *fiedler vector* (b) Sorted *fiedler vector*

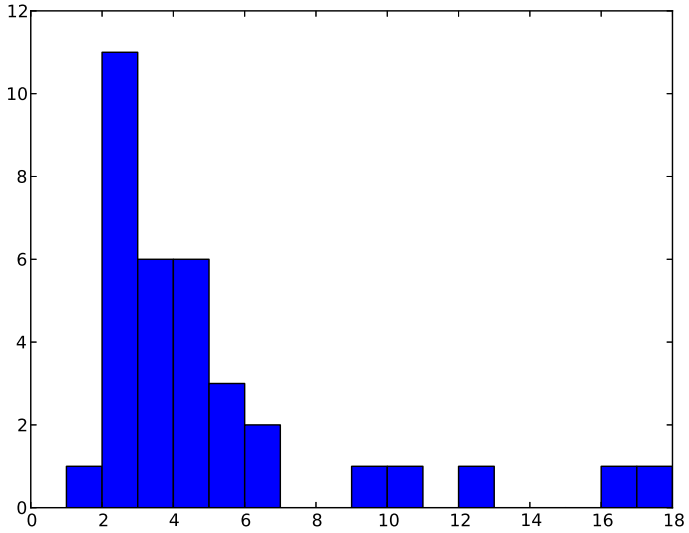


Figure 4.7.: *Histogram of the degree distribution of the Zachary's Karate Club network.*

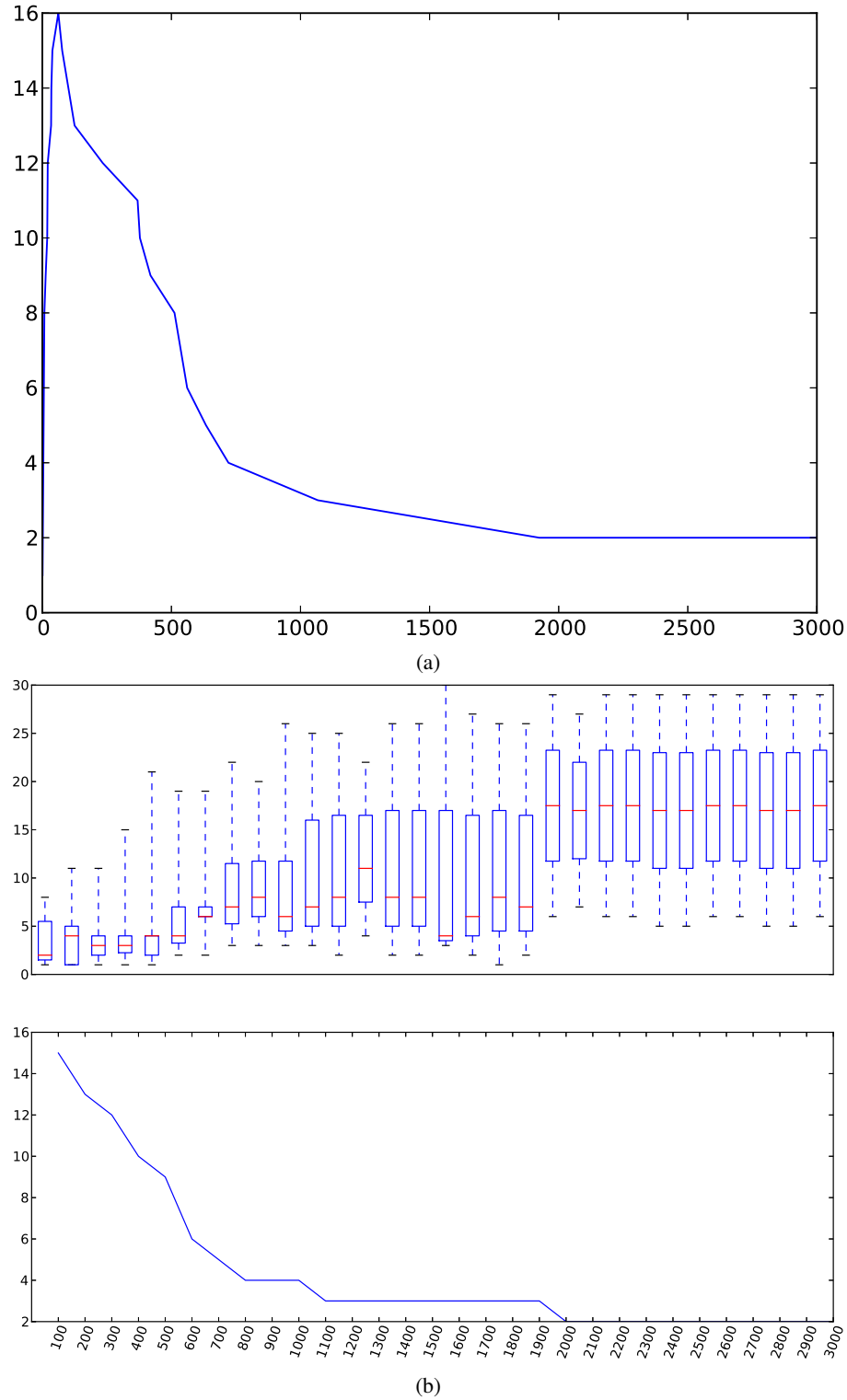


Figure 4.8.: *Vocabulary changes and word distributions of a sample simulation on the Zachary's Karate Club network.* (a) The number of different words (groups) over time (b) Box plot of the word distributions over time

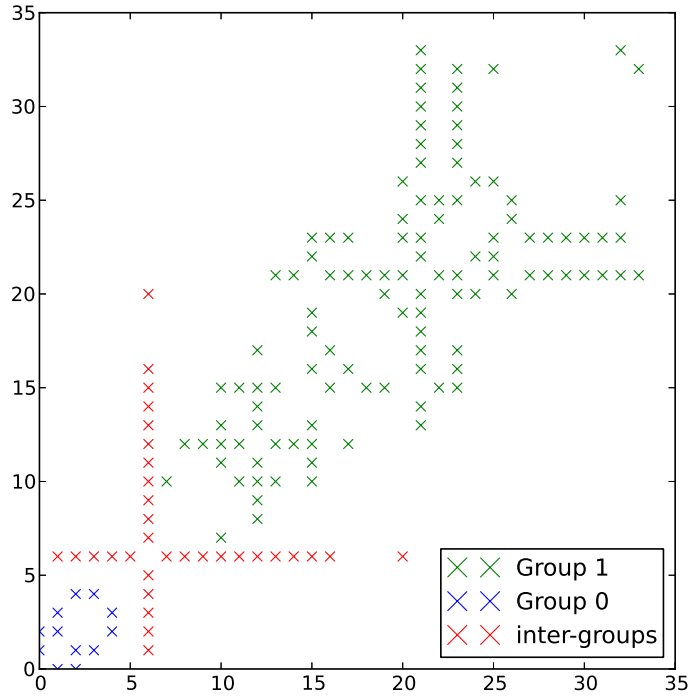


Figure 4.9.: *Adjacency matrix with links colored according to their group.*

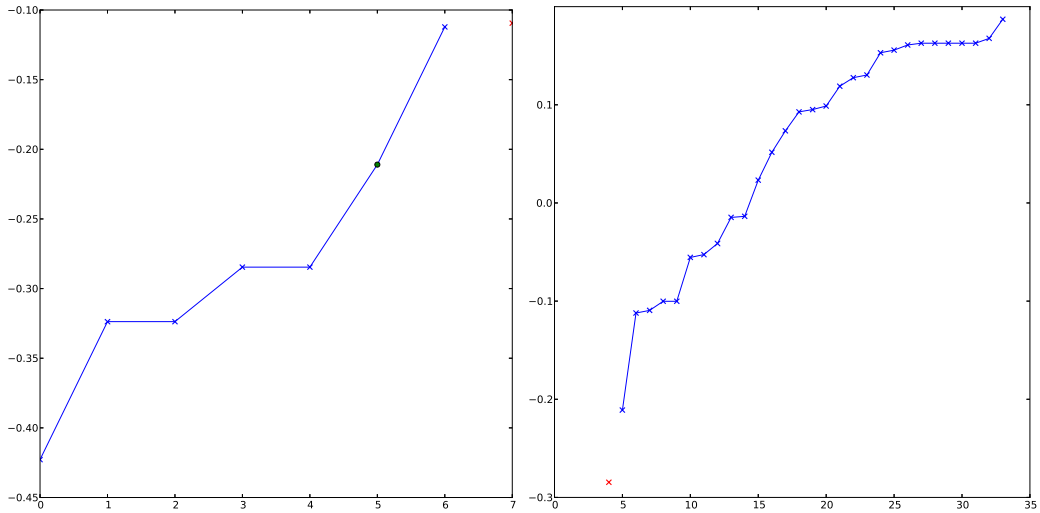


Figure 4.10.: *Plots of the group eigenvectors.* Values corresponding to the nodes within the groups are plotted as blue crosses, the others as green circles. The red crosses depict values for neighboring nodes on the eigenvector.

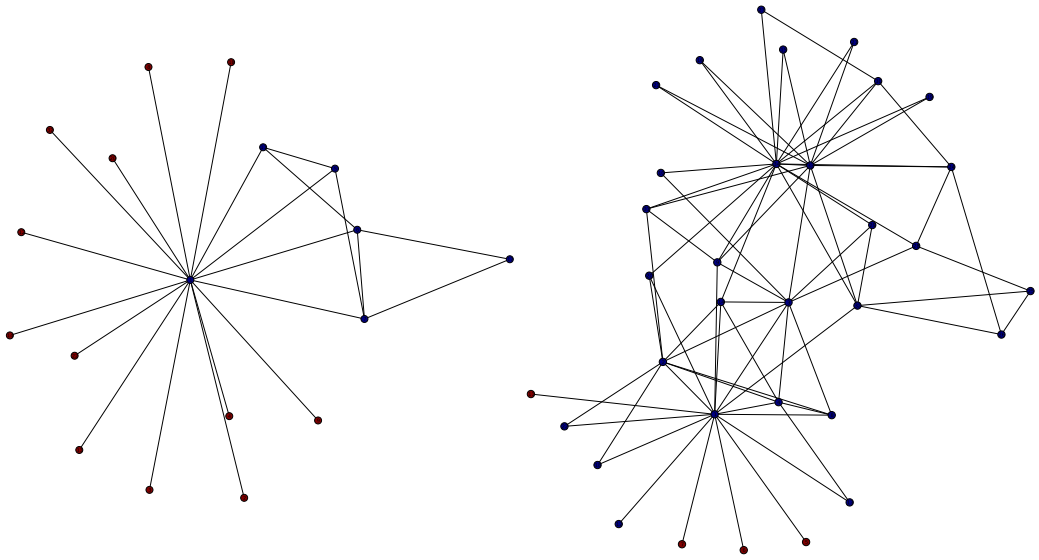


Figure 4.11.: *Plots of the groups.* The nodes belonging to the group are plotted in blue, the neighbors in red.

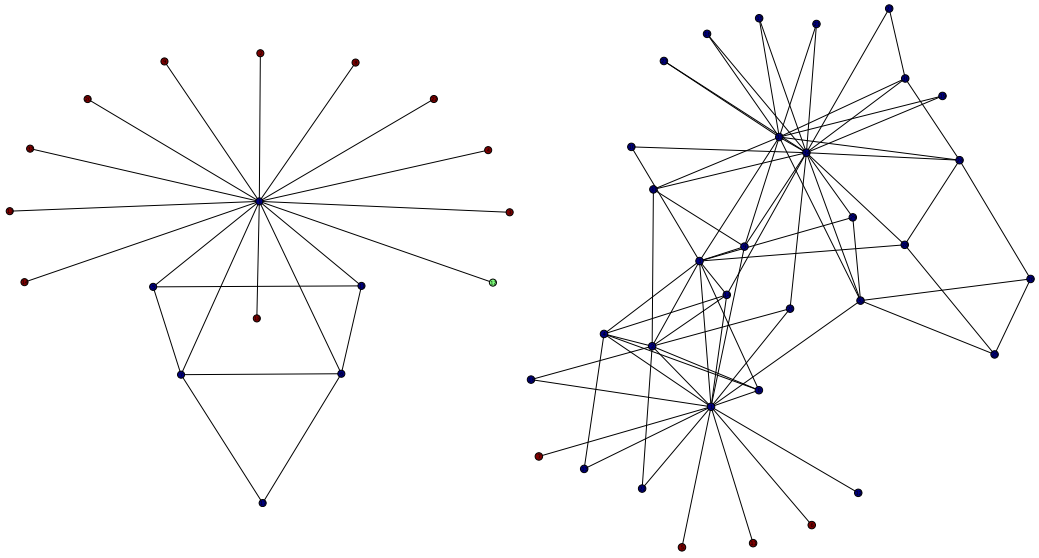


Figure 4.12.: *Plots of the groups including eigenvector nodes.* The same as the group plots but nodes whose eigenvector values lie within the groups range are plotted in green.

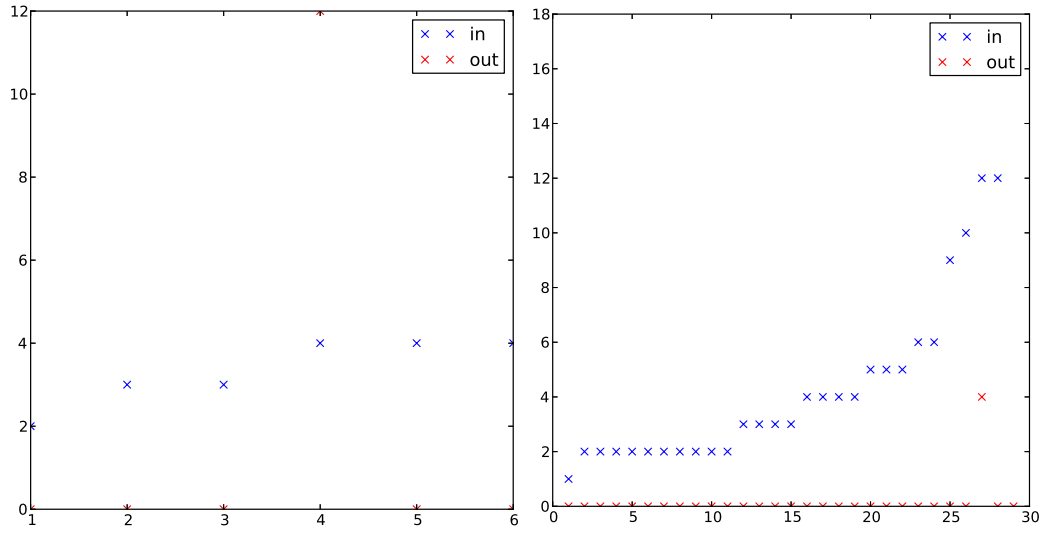


Figure 4.13.: *Plots of in-out degrees of the nodes from the groups.* The nodes are ordered ascending to their in-degree.

4.4. Stochastic Block Model

The baseline for the *Stochastic Block Model* project builds the class `StochasticBlockModel` in the `sbm` module. It is initialized with the node type indices vector `type_indices` and the symmetric block matrix `block_matrix`. Through the `get_graph()` method a graph is created from the parameters. The number of nodes is determined by the node type indices vector. For each node n_i the corresponding entry v_i in the node type indices vector states the group index, i.e. the column in the block matrix. The implemented block matrix is always square and symmetric.

The `main-1.py` script utilizes the `sbm` module and is responsible for the simulations on synthetic networks described in chapter 5. The script generates block matrices with just two probabilities. The inner-group probability p_{in} which states the probability of a link between two nodes in the same group and the cross-group probability p_{cross} for links between different groups. Each group is equal and only the number of groups and the number of nodes per group is defined. There can also be several groups. The algorithm follows the following pattern:

- Iterate over all possible p_{cross}.
- Iterate over all possible group counts and create a directory according to the pattern `out-1-<p_in>-<p_cross>` with `<p_in>` as p_{in} and `<p_cross>` as p_{cross}.
- For each group count create several graphs and create a directory with the pattern `<group-directory-prefix><group-count>-<graph-id>` where `group-count` is the number of groups and `<graph-id>` is an accumulator for the graph.
- For each graph run several simulations and the analysis scripts

The script is configured in the header and holds the following parameters:

- `group_counts`. The number of groups $N_{communities}$. The parameter is of the list type.
- `graphs_per_group_count`. The number of graphs which are created per group count.
- `group_size`. The number of nodes per group N_{nodes}.
- `probability_group`. The probability for links between nodes which belong to the same group p_{in}.
- `probability_group_to_group`. A list of probabilities for links between nodes which belong to different groups p_{cross}.
- `base_directory_pattern`. The base directory pattern for the directories which are created for a group size and a certain p_{in} and p_{cross}. The probabilities are fed to the string through the `%` operator
- `games_directory`. The directory where the games are located in the group directory
- `analysis_directory`. The directory where the files for the analysis are located in the group directory
- `group_directory_prefix`. The prefix for the group directory. For each graph a directory with the pattern `<group-directory-prefix><group-count>-<graph-id>` in the base directory is created.
- `game_directory_prefix`. The prefix for the game directory. For each simulation a folder is created in the games and analysis directory following the pattern `<game-directory-prefix><game-id>`.
- `count_games`. The number of simulations run on each graph.

An example for a directory structure from the outcome of the script is the following:

```
/
├── /out-1-03-0016
│   ├── groups3-1
│   │   ├── analysis
│   │   │   ├── cores
│   │   │   ├── game-1
│   │   │   └── game-2
│   │   └── games
│   │       ├── game-1
│   │       └── game-2
│   └── groups4-1
│       ├── analysis
│       │   ├── cores
│       │   ├── game-1
│       │   └── game-2
│       └── games
│           ├── game-1
│           └── game-2
└── /out-1-03-0017
    ├── groups3-1
    │   ├── analysis
    │   │   ├── cores
    │   │   ├── game-1
    │   │   └── game-2
    │   └── games
    │       ├── game-1
    │       └── game-2
    └── groups4-1
        ├── analysis
        │   ├── cores
        │   ├── game-1
        │   └── game-2
        └── games
            ├── game-1
            └── game-2
```

Listing 4.2: Directory structure of an outcome of the `main-1.py` script

The parameters for the directory structure in listing 4.2 were the following:

```
group_counts = [3, 4]
graphs_per_group_count = 1
group_size = 300

probability_group = 0.03
probabilities_group_to_group = [0.0016, 0.0017]

base_directory_pattern = "./out-1-%02d-%04d"
games_directory = "games"
analysis_directory = "analysis"

group_directory_prefix = "groups"
game_directory_prefix = "game-"
```

```
count_games = 4
```

Chapter 5

Simulations on Synthetic Networks

The aim of the simulations was how the outcome of a Naming Game can be precalculated through spectral analysis. Therefore the Stochastic Block Model described in section 3.2 was utilized. The simulations were carried out with two, three and four equally generated communities. It is analyzed how the number of communities affects the Naming Game and further the *Fiedler vector*. First, thresholds for networks following one out of three characteristic network structures considering the probabilities p_{in} and p_{cross} are found. The *Fiedler vector* is then analyzed on these networks. p_{in} is the probability for a link between nodes of the same community while p_{cross} depicts the probability for links between different communities. The links starting and ending in the same community are i.i.d. according to p_{in}. The links between the communities are i.i.d. according to p_{cross} respectively. For the categorization of the networks the *inter-connectivity* $\nu = p_{cross}/p_{in}$ is employed.

In this chapter a distinction between three characteristic structures is made:

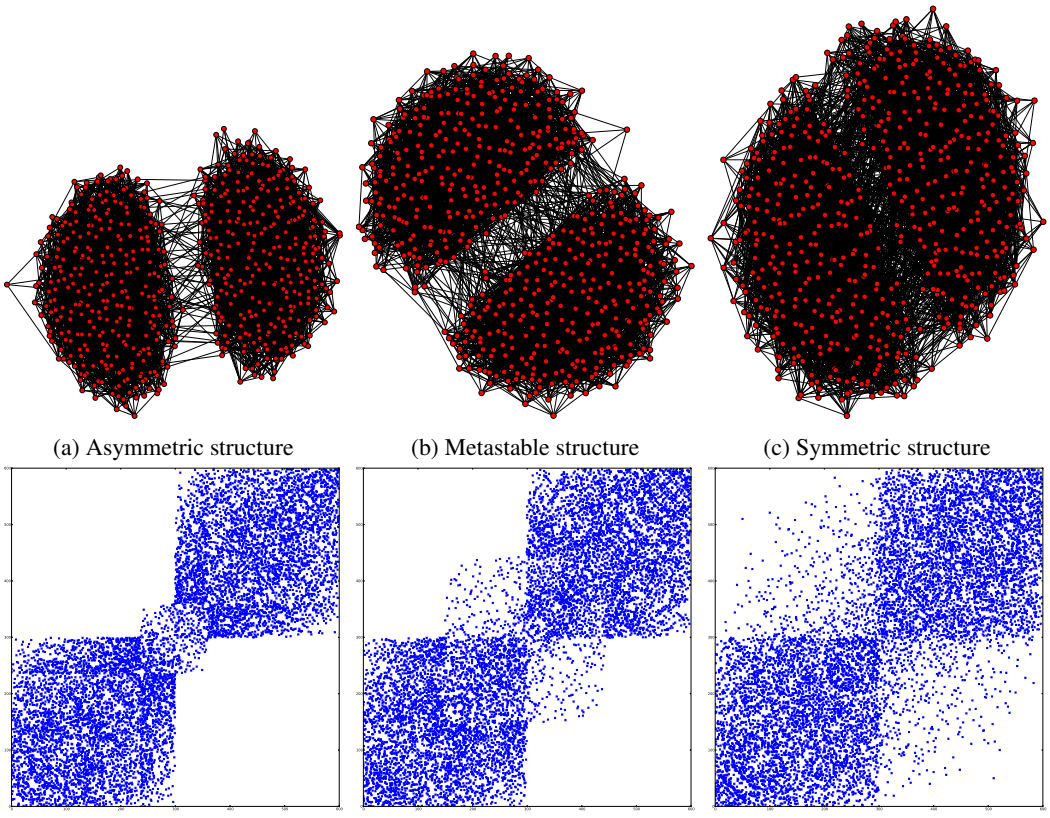
- *Symmetric structure*. Networks where all nodes reach consent, i.e. share one opinion.
- *Asymmetric structure*. Networks where the number of different opinions is exactly the number of communities defined in the Stochastic Block Model.
- *Metastable structure*. Networks where the number of opinions differ in the simulation outcomes.

The first threshold th_{asym} depicts the transition between an asymmetric and a metastable structure. Networks generated with $th_{asym} \leq \nu$ are categorized as networks following an asymmetric structure. The second threshold th_{sym} is the transition between the metastable and the symmetric structure analogously. Networks generated with $th_{sym} \geq \nu$ are categorized as networks following a symmetric structure. Networks generated with $th_{asym} < \nu < th_{sym}$ are categorized as networks following a metastable structure.

5.1. Experimental setup

In the simulations on synthetic networks in this work three different probabilities for p_{in} were investigated, which are 0.03, 0.04 and 0.05. The step size for p_{cross} is 10^{-4} for $p_{cross} > 10^{-4}$ and 10^{-5} for $p_{cross} < 10^{-4}$. For each of the (p_{in}, p_{cross}) tuples three graphs are created. On each graph four simulations are run. For the tuple where $\nu = th_{asym}$ and $\nu = th_{sym}$ four simulations for six graphs are conducted. Those simulations are always executed for two, three and four communities $N_{communities} \in \{2,3,4\}$ with a fixed number of nodes per community $N_{nodes} = 300$. The number of iterations t_{max} is always set to $100,000 * N_{communities}$.

Due to the fixed N_{nodes} the probabilities for p_{cross} need to be normalized for $N_{communities} \in \{3,4\}$ in order to be compared. They are multiplied by 2 for $N_{communities} = 3$ and multiplied by 3 for $N_{communities} = 4$.



(a) Asymmetric structure (b) Metastable structure (c) Symmetric structure

(d) Sorted adjacency matrix for asymmetric structure (e) Sorted adjacency matrix for metastable structure (f) Sorted adjacency matrix for symmetric structure

Figure 5.1.: *Example plots for a synthetic network with two communities.* The number of nodes of each community N_{nodes} is 300. (a) Network following an asymmetric structure with $\nu = 0.0008/0.05 = 0.016$ (b) Network following a metastable structure with $\nu = 0.025/0.05 = 0.05$ (c) Network following a symmetric structure with $\nu = 0.0057/0.03 = 0.114$ (d) - (f) are the sorted adjacency matrices where the nodes are reordered according to their positions in the sorted *Fiedler vector*.

5.2. Results

Example networks following the three characteristic structures are shown in Figure 5.1. Below each of the graph the sorted adjacency matrix is illustrated. The matrix gives a good overview of the spectral analysis on a link basis.

The results for the symmetric and asymmetric thresholds are shown in Table 5.1. The asymmetric thresholds th_{asym} for two communities lie in the interval $[0.0133, 0.016]$. For three communities the thresholds decrease slightly. They lie in the interval $[0.01, 0.012]$. It can already be observed that the number of communities influences the probabilities for the generation of the networks. This further intensifies drastically with four communities. The thresholds are 0.003 for all valuations of p_{in}. The communities in the networks at the thresholds are already very sparse connected. The average number of links leading to other communities is 13.5.

The symmetric thresholds th_{sym} for two communities lie in the the interval $[0.1067, 0.114]$. For three communities the boundaries of the interval increase to $[0.14, 0.156]$. It further increases for four communities to $[0.1599, 0.198]$. It can be observed that the presence of communities not only decreases the thresholds for the asymmetric structures but also increases them significantly for symmetric structures.

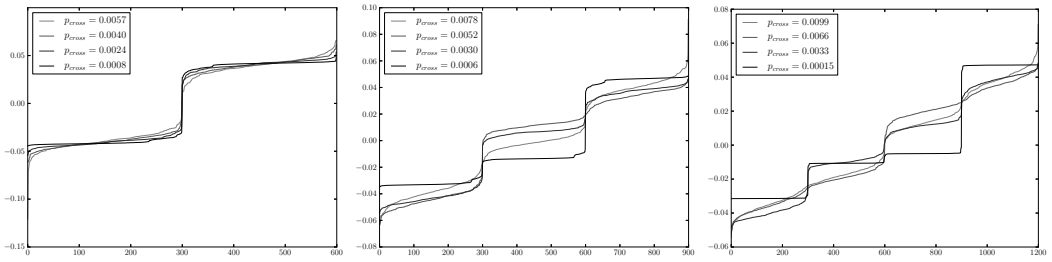
$N_{communities} = 2$		
$p_{in} = 0.03$	$p_{in} = 0.04$	$p_{in} = 0.05$
$th_{asym} = 0.0133$ with $p_{cross} = 0.0004$	$th_{asym} = 0.015$ with $p_{cross} = 0.0006$	$th_{asym} = 0.016$ with $p_{cross} = 0.0008$
$th_{sym} = 0.1067$ with $p_{cross} = 0.0032$	$th_{sym} = 0.1075$ with $p_{cross} = 0.0043$	$th_{sym} = 0.114$ with $p_{cross} = 0.0057$
$N_{communities} = 3$		
$p_{in} = 0.03$	$p_{in} = 0.04$	$p_{in} = 0.05$
$th_{asym} = 0.01$ with $p_{cross} = 0.0003$	$th_{asym} = 0.01$ with $p_{cross} = 0.0004$	$th_{asym} = 0.012$ with $p_{cross} = 0.0006$
$th_{sym} = 0.14$ with $p_{cross} = 0.0042$	$th_{sym} = 0.155$ with $p_{cross} = 0.0062$	$th_{sym} = 0.156$ with $p_{cross} = 0.0078$
$N_{communities} = 4$		
$p_{in} = 0.03$	$p_{in} = 0.04$	$p_{in} = 0.05$
$th_{asym} = 0.003$ with $p_{cross} = 0.00009$	$th_{asym} = 0.003$ with $p_{cross} = 0.00012$	$th_{asym} = 0.003$ with $p_{cross} = 0.00015$
$th_{sym} = 0.1599$ with $p_{cross} = 0.0048$	$th_{sym} = 0.18$ with $p_{cross} = 0.0072$	$th_{sym} = 0.198$ with $p_{cross} = 0.0099$

Table 5.1.: *Thresholds for synthetic networks.* An increasing number of communities $N_{communities}$ makes it harder to hold their own opinion, i.e. the thresholds for asymmetric structures decrease. On the other hand it also makes it harder to reach global consent, i.e. the thresholds for symmetric structures increase.

Subject to this thesis is how the opinion clusters correlate to the *Fiedler vector*. Example plots for each $N_{communities}$ are plotted in Figure 5.2. They show a very similar characteristic curve for each $N_{communities}$. For networks with two communities following an asymmetric structure the values in the vector increase logarithmically for the for the first ≈ 220 nodes. The values for the following ≈ 80 nodes increase exponentially until there is a big gap at 0. The values then increase again logarithmically for the first ≈ 220 nodes and exponentially for the last ≈ 80 nodes. For 70% of the nodes of each community the values of the vector differ at most 0.004. As the networks come closer to a symmetric structure the slope gets steeper. In a symmetric structure the values of the vector for 70 % of the nodes of each community differ at most 0.02.

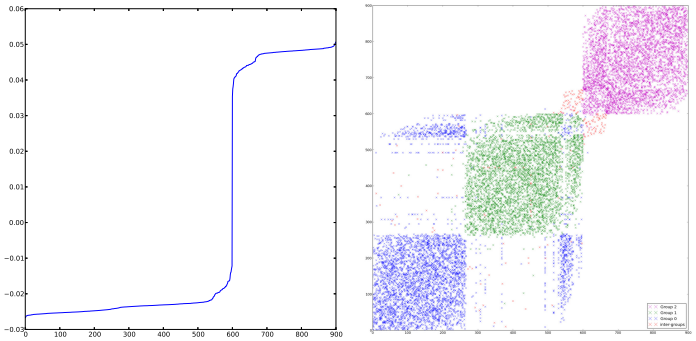
For networks with three communities the *Fiedler vector* shows a similar curve. Each community starts with a logarithmic and ends with an exponential slope. In the asymmetric structure although the plateaus from the communities are flatter. 70 % of the nodes of each community now differ at most 0.002. In the symmetric state one community can be separated best. This community comprises the biggest gap and the lowest slope in the vector. In the symmetric structure the values of the vector for 70 % of the nodes of each community differ at most 0.02. This is the same value as it was observed for two communities.

In networks with four communities the *Fiedler vector* gets even more flat in an asymmetric structure. The values for the nodes of each community now increase nearly linearly. 70 % of the values now differ only by 0.0004. In symmetric structures the behavior is again the same as in the case of two communities. In the community which can be separated most clearly, the values of 70 % of the nodes differ at most by 0.02.



(a) $N_{communities} = 2$ (b) $N_{communities} = 3$ (c) $N_{communities} = 4$

Figure 5.2.: *Example plots for Fiedler vectors with* $p_{in} = 0.05$. (a) The plateaus are flatter in the asymmetric structure and the gaps between the communities are bigger. (b) - (c) The plateaus are flatter with the increasing number of communities in the asymmetric structure. In the symmetric structure the curves for the communities stay similar. (c) The communities which can be separated best usually lie at the beginning or at the end of the vector.



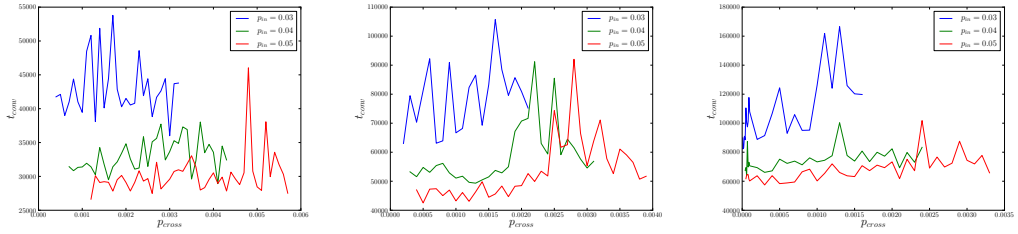
(a) Fiedler vector (b) Colored adjacency matrix

Figure 5.3.: *Incorrect detection of communities*. A few nodes of the left community are placed on the right side of the community in the middle. Therefore the gap between those communities shrinks dramatically.

The separation of the communities works precisely in most cases. It can be observed that the communities which can be separated best usually lie either at the beginning or at the end of the *Fiedler vector*. They depict the biggest gaps at their limits. Problems with the community detection arise due to the reducing to community detection to a one dimensional sorting. If nodes are linked to more than one cluster outside of their community they are also placed outside of their community in the *Fiedler vector*. This does not affect the ordering of the other nodes in their community but the gap to the neighboring community shrinks to level which makes it hard to identify. This case occurs rarely in asymmetric structures and more often as p_{cross} increases. It is illustrated in Figure 5.3. Nevertheless there is always one big gap in the vector which clearly identifies one community.

Average convergence times			
$N_{communities}$	$p_{in} = 0.03$	$p_{in} = 0.04$	$p_{in} = 0.05$
2	$\approx 43027 = N^{\approx 1.67}$	$\approx 33071 = N^{\approx 1.63}$	$\approx 30103 = N^{\approx 1.61}$
3	$\approx 77926 = N^{\approx 1.66}$	$\approx 58560 = N^{\approx 1.61}$	$\approx 52309 = N^{\approx 1.60}$
4	$\approx 107459 = N^{\approx 1.63}$	$\approx 74947 = N^{\approx 1.58}$	$\approx 67833 = N^{\approx 1.57}$

Table 5.2.: *Average convergence times* t_{conv}. The number of communities only have a small impact on the convergence times.



(a) $N_{communities} = 2$ (b) $N_{communities} = 3$ (c) $N_{communities} = 4$

Figure 5.4.: *Convergence times* t_{conv} *for* p_{cross}. The simulations converge faster if the communities are better connected.

In Table 5.2 the averages for the different number of communities $N_{communities}$ and p_{in} are shown. The convergence times mainly depend on $N_{communities}$ and p_{in}. The more connected the cluster itself is the faster the system converges. Additionally some outliers occur which need up to the double amount of time. Most of the simulations however depict a common behavior for t_{conv}. If t_{conv} is related to the number of nodes N it is on average $N^{1.62}$. It is slightly related to the number of communities, but the relation is negligible. The Naming Game on complete graphs show convergence times of $O(N^{3/2})$ (Baronchelli et al., 2006), $O(N^{1+2/d})$ for $d \leq 4$ on d-dimensional lattices (Baronchelli et al., 2006) and $O(N^{1.4})$ for small-world networks (Dall'Asta et al., 2006). The average for t_{conv} is comprehensible as it doesn't show small-world characteristics nor it is fully connected. It is remarkable that presence of groups doesn't have a lot of impact on the convergence time although it slightly increases. The convergence time is only marginally effected by its network structure. This can be observed better in Figure 5.4. In this figure t_{conv} is plotted against p_{cross} for each p_{in} and N_{groups}. Here peaks can be observed before the symmetric state is reached but t_{conv} usually stays behind the same limits.

Chapter 6

Simulations on Empirical Networks

The simulations on empirical networks were carried out on two networks obtained from the Stanford Network Analysis Project (Leskovec and Krevl, 2014). The first is a scientific collaboration (GR-QC) network where the authors are linked if they jointly wrote papers. It provides valuable insight in how close a scientific community works together. The second is a subset of the Facebook network which exhibits a social friendship structure. Although both networks come from very different areas, they both comprise small-world characteristics and are thus representative for real-world networks.

The aim of the simulations was how the results from the synthetic networks can be applied to empirical networks. Therefore six simulations with 100,000,000 and two simulations with 200,000,000 iterations where run on each of the networks. The *Fiedler vector* is then analyzed for the outcomes of the simulations and the results are compared to that of the synthetic networks.

For the GR-QC network a k-core decomposition was conducted additionally in order to visualize the network structure.

6.1. GR-QC network

The GR-QC (General Relativity and Quantum Cosmology collaboration network) network is a collaboration network from arXiv.org[1]. arXiv.org is an online archive deploying e-prints of scientific papers. For the GR-QC network only papers from the category General Relativity and Quantum Cosmology are considered. In the network each author is represented by a node. A link between two nodes is drawn if they co-authored a paper. If the paper has more than one author, they all form a clique respectively. A table of some statistical measures of the network can be found in Table 6.1.

The network consists of one giant connected component and some outliers. Those outliers are typically two or three connected nodes which are separated from the rest of the network. For the analysis in this chapter the network was trimmed to the core component only. It contains 4158 out of the 5242 total nodes. A plot of the whole network and of the core component can be found in Figure 6.1.

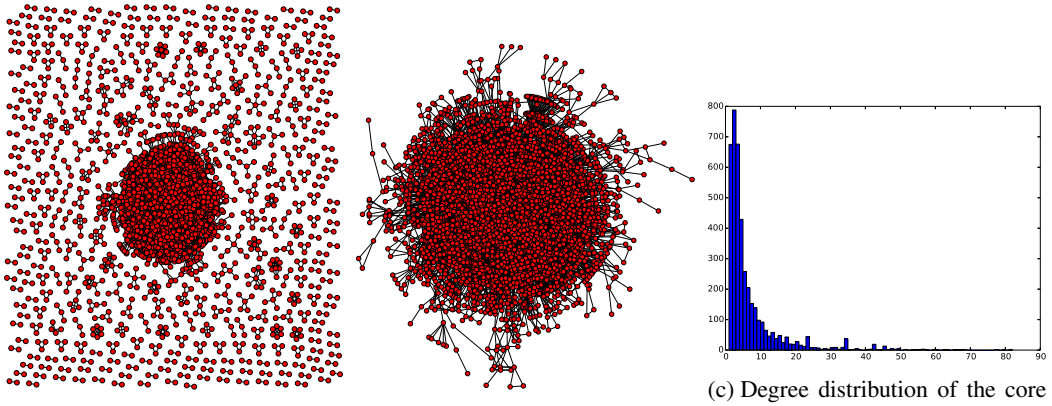
The degree distribution follows a power law which is typical for scale-free networks (Albert and Barabási, 2002). It is plotted in Figure 6.1. The high degree nodes are the hubs which are responsible for the fast diffusion of opinions. Thus the scale-free characteristic is a very important property.

The two properties of small-world networks which are a high average clustering coefficient and a short average shortest path length are basically fulfilled for the core component. The clustering coefficient C is

[1] http://arxiv.org/

Statistics of the GR-QC network	
Nodes	5242
Edges	14496
Nodes in largest SCC	4158 (0.793)
Edges in largest SCC	13428 (0.926)
Average clustering coefficient	0.5296
Number of triangles	48260
Fraction of closed triangles	0.3619
Diameter (longest shortest path)	17
90-percentile effective diameter	7.6

Table 6.1.: GR-QC network dataset statistics.



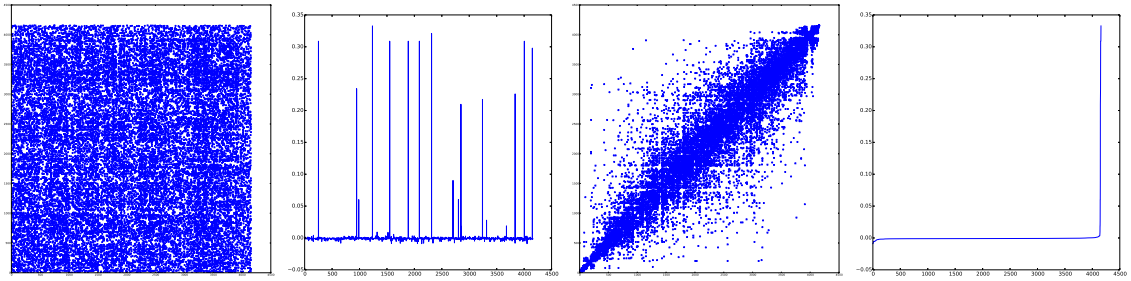
(a) GR-QC graph (b) GR-QC core component (c) Degree distribution of the core component

Figure 6.1.: *GR-QC graph and its degree distribution.* (a) The whole graph of the GR-QC network (b) Only the investigated core component of the graph (c) The core component exhibits a power-law behavior which is the property of scale-free networks.

≈ 0.5569. The average number of links per node is $k \approx 3.2294$. The clustering coefficient of a random network with the same properties according to $C_{random} \sim k/n$ is ≈ 0.0007767. Thus the core component of the network exhibits the first property of a small-world network $C \gg C_{random}$. The average shortest path length of the network L is ≈ 6.04938. The length for a random network with the same properties according to $L_{random} \sim ln(n)/ln(k)$ is ≈ 8.108. Thus the actual path length L is even below L_{random} where the second property of a small-world network is that L should not be much greater than L_{random}. (Watts and Strogatz, 1998)

The adjacency matrix and the sorted adjacency matrix according to the *Fiedler vector* is illustrated in Figure 6.2. The plots are described in subsection 4.3.6. While the unsorted adjacency matrix seems quite uncorrelated, the sorted matrix exposes some interesting charasteristics. A lot of highly connected clusters can be found, which is the result of papers with multiple authors. Nevertheless they are good connected to other nodes. Only a few clusters on the lower left and the upper right corner are only sparsely connected to the core of the network.

The corresponding unsorted and sorted *Fiedler vectors* are illustrated in Figure 6.2. They show a similar pattern like the matrices. The sorted vector is mainly flat, but it has some lower values in the beginning and a high peak in the end.



(a) Adjacency matrix (b) Unsorted *Fiedler vector* (c) Sorted adjacency matrix (d) Sorted *Fiedler vector*

Figure 6.2.: *Spectrum analysis of the GR-QC core component.* (a) The adjacency matrix of the core component (c) The sorted adjacency matrix according to the *Fiedler vector*.

Overall six simulations with $t_{max} = 100,000,000$ iterations were run. To check if more iterations can influence the result two more simulations with $t_{max} = 200,000,000$ were executed. The results are illustrated in Table 6.2. They show that the outcomes merely depict a total of three groups, which are plotted in Figure 6.3. In run 3, 4 and 6 one additional group is present in the outcome. In run 3 also a fifth group is contained which is a tendril of the network. The groups are plotted in Figure 6.4. An interesting result comes up with the outcome of run 7. Here one group contained in most runs coalesce with the third group found in all simulations. This is a result of the randomness of the simulations. The distance between the groups is small, but each of them is a well separated cluster itself. It is plotted in Figure 6.4.

The groups 2, 3 and 4 are highly connected. This can be observed from the groups plots and from the degree distributions plotted in 6.5. The groups almost build a clique. It results from the co-authoring structure where authors build cliques if they jointly wrote a paper. The opinion group 5 from run 3 is one of the tendrils of the network. The tendrils are shown later in the k-core analysis.

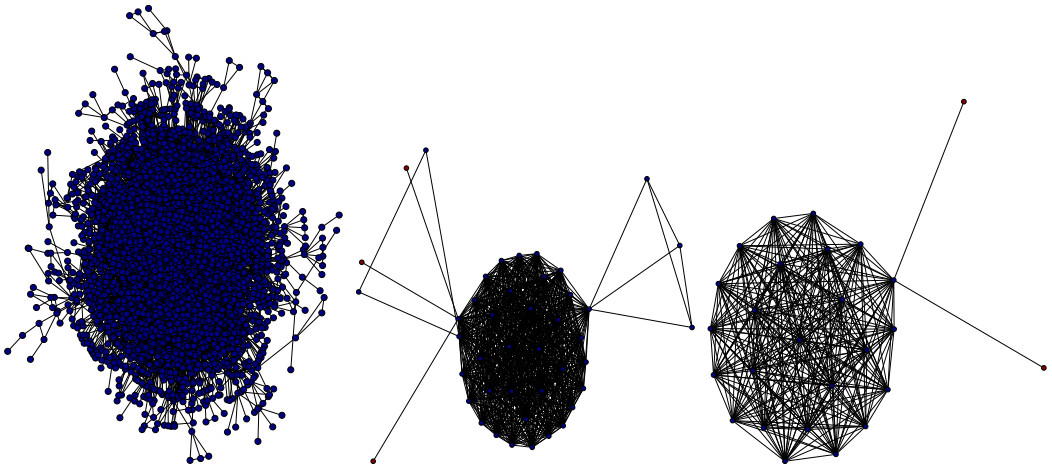
The group eigenvectors described in subsection 4.3.6 are plotted in Figure 6.6. For the first group - which contains most of the nodes - the whole network lies within its range. This inaccuracy is justifiable due to the heavily uneven sizes of the groups. The vector of the second group shows good results. The vast amount of nodes in the vector has the same values, while the outliers correspond to the nodes which are not part of the main cluster of the group. The third group shows the same pattern in the vector and has no other nodes within its range. The fourth group is also matched nearly completely in the analysis and has only one other node within its range. This is due to they nearly form cliques where all nodes are interconnected.

The vector for the fifth group yields two different groups. If looked at the group plot this would be expected. On the long run one part of the group should coalesce with the big cluster, but it sits on the end of a tendril such that the formation of a shared opinion takes much longer than for other groups. The vector has its biggest gap at the end which would suggest another group. The nodes are well separated but they would build a cluster of only five nodes which is too small for the Naming Game.

It can be concluded that the groups can be separated very precisely by the gaps in the *Fiedler vector*. Most of the groups sit on a plateau on the vector where each node has an equal value. The fifth at the beginning of the vector does not sit on a plateau but it can be separated very good by its gap. A better visualization of the results is the colored adjacency matrix. The matrices for run 1 and 3 are plotted in Figure 6.7.

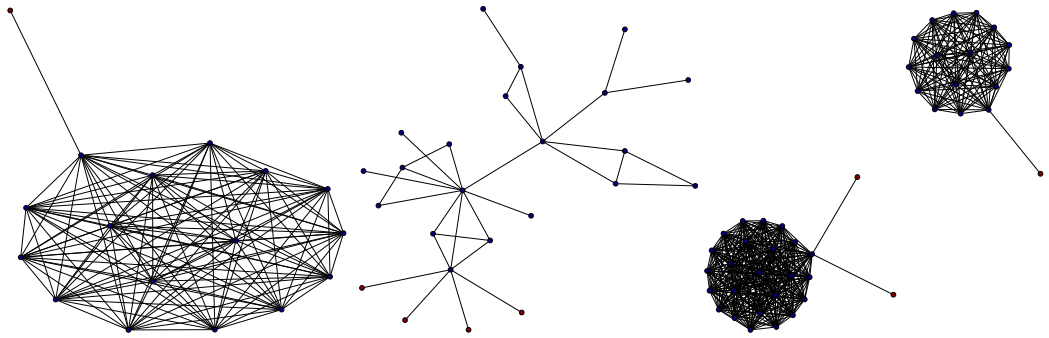
Results for the simulations	
$t_{max} = 100,000,000$	
Run	Final groups
Run 1	3
Run 2	3
Run 3	5
Run 4	4
Run 5	3
Run 6	4
$t_{max} = 200,000,000$	
Run	Final groups
Run 7	3
Run 8	3

Table 6.2.: Results for the simulations on the GR-QC core component.



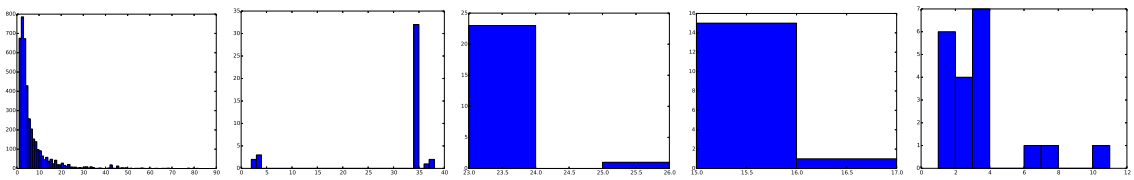
(a) First group (b) Second group (c) Third group

Figure 6.3.: *Simulation-invariant groups of the GR-QC core component including their neighbors*. Groups which are contained in all outcomes of the simulations. The members of the groups are plotted in blue, their neighbors in red.



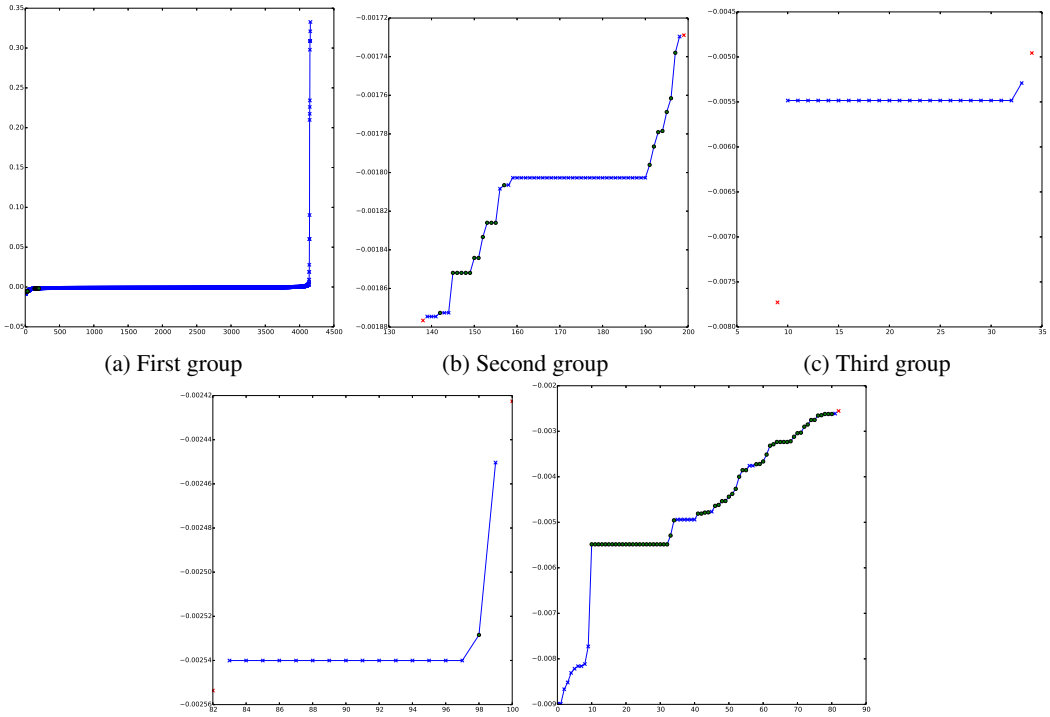
(a) Fourth group (b) Fifth group (c) Coalesced groups

Figure 6.4.: *Additional groups of the GR-QC core component including their neighbors*. Additional groups from run 3 and coalesced group from run 10. (a) The group which is also contained in run 4 and 6 (b) The group which is unique to run 3. (c) Groups which coalesced in run 10. The members of the groups are plotted in blue, their neighbors in red.



(a) First group (b) Second group (c) Third group (d) Fourth group (e) Fifth group

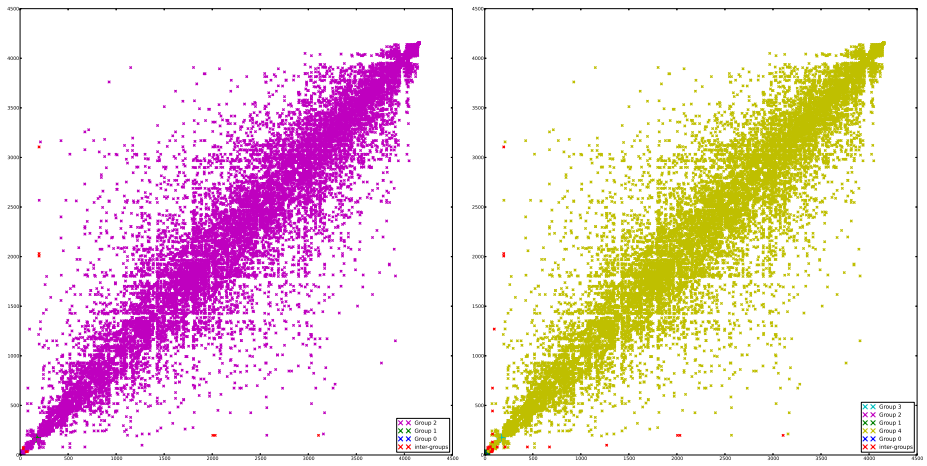
Figure 6.5.: *Degree distributions of the groups of the GR-QC core component*. (a) The first group shows an exponential degree distribution (b) - (d) The groups nearly form cliques (e) The group forms a tendril of the network.



(a) First group (b) Second group (c) Third group

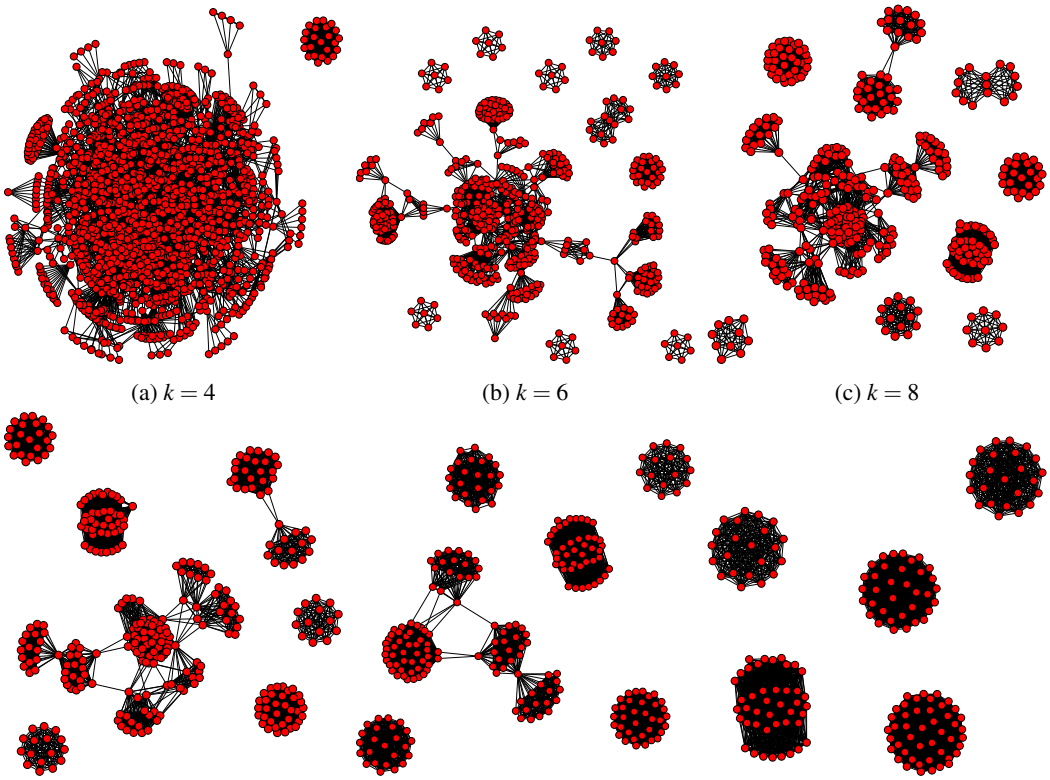
(d) Fourth group (e) Fifth group

Figure 6.6.: *Group eigenvectors for the groups of the GR-QC core component*. The range of the *Fiedler vector* where the nodes of the group are located. Nodes from the group are plotted with blue crosses, nodes which are not part of the group are plotted with green circles. Values adjacent to the group are plotted with red crosses. Each of the groups builds a plateau in the vector except of the main cluster and group five. Group five is located at the beginning of the vector and can be separated by its gap.



(a) Colored adjacency matrix run 1 (b) Colored adjacency matrix run 3

Figure 6.7.: *Colored adjacency matrices of the GR-QC core component for run 1 and 3*. The groups from the outcomes of the simulations 1 and 3 are colored in the sorted adjacency matrix. Each group has its own color. Links which do not start and end in the same group are colored in red. (a) The three groups which are contained in all simulation outcomes (b) The two additional small groups from run 3 at the left lower end of the matrix.

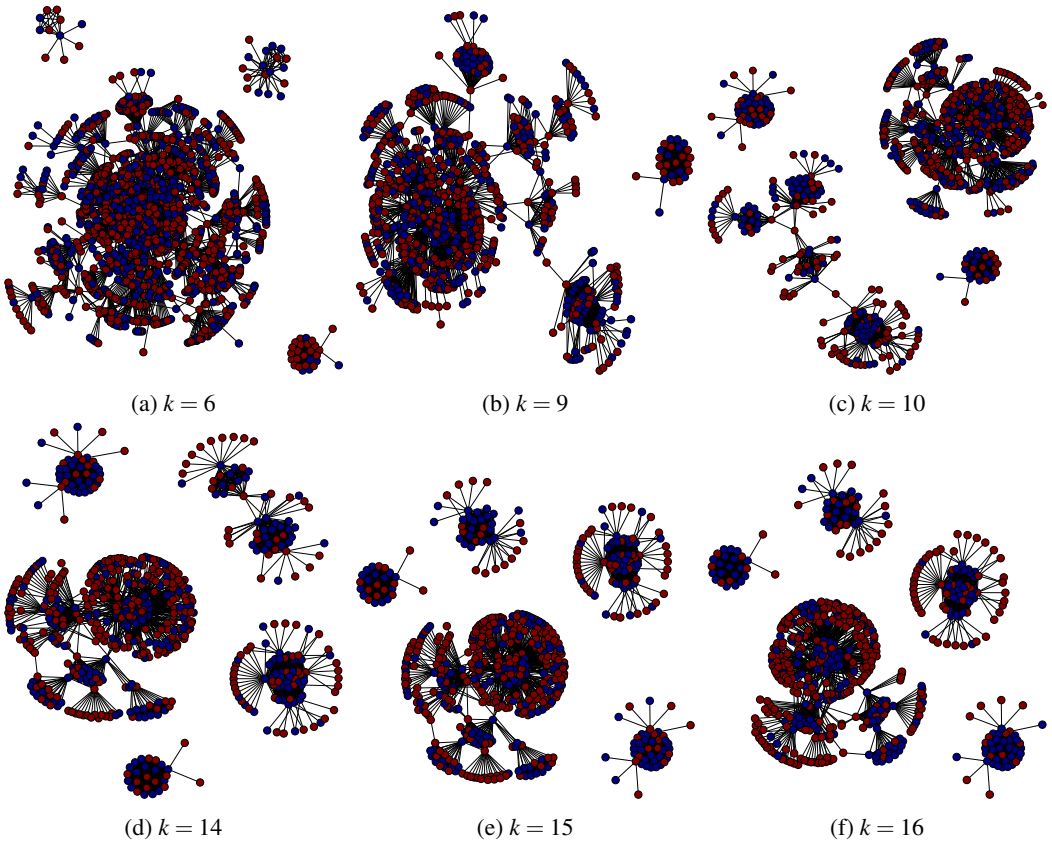


(a) $k = 4$ (b) $k = 6$ (c) $k = 8$

(d) $k = 11$ (e) $k = 15$ (f) $k = 21$

Figure 6.8.: *k-core analysis of the GR-QC core component*. Several small cluster detach early from the decomposition. The three groups which result from the Naming Game simulations are already detached from the core cluster at $k = 8$. The high connectedness of the clusters reflects the cliques which are build from the co-authoring relationships.

The k-core decomposition described in section 3.4 gives insight about the structure of the network. At $k = 4$ one cluster detaches from the network. It is one group which is included in all simulation outcomes and vanishes at $k = 24$. At $k = 6$ many small clusters detach from the main cluster. Most of them are composed of low-degree nodes and vanish with $k = 8$. Several big strongly-connected clusters then detach at $k = 8$. The five big clusters which remain at $k = 21$ can already be observed. The cluster on the upper left corner is the second group which is included in all simulation outcomes. The cluster in the middle on the bottom of the plot is the group which is included in some runs. The main cluster loosens up until it depicts one densely connected cluster at $k = 21$. The five clusters at $k = 21$ vanish from the decomposition at $k = 22, 24, 34, 35, 36$.



(a) $k = 6$ (b) $k = 9$ (c) $k = 10$

(d) $k = 14$ (e) $k = 15$ (f) $k = 16$

Figure 6.9.: *k-core analysis of the GR-QC core component including its neighbors*. The nodes of the decomposition are plotted in blue, their neighbors in red. At $k = 10$ three groups which are contained in all of the simulations can be seen. The two big connected components form one, the other clusters build the other groups.

The k-core decomposition including its neighbors which is also described in section 3.4 show a little more detailed results. The first opinion cluster which is included in all simulations also detaches from the main cluster at $k = 6$. At this stage also the tendrils of the core clusters can be observed. At $k = 10$ the second cluster which is contained in all of the simulations is detached. The two big disconnected components at $k = 10$ are too interwoven to build two opinion clusters in the Naming Game. The third opinion cluster which is included in some simulations vanishes at $k = 16$. The remaining decomposition proceeds analogously to the regular k-core decomposition.

6.2. Facebook network

The Facebook network is a subset of the original Facebook network[1]. The data was collected through the Facebook app Social Circles[2]. In this app the participants provided their friendship relationships for research purposes. They were asked to enter tags for a group of friends with which they share mutualities. The names, ids and features were collected, anonymized and provided for download. The network were all nodes are combined is a composition of ego-networks. In an ego-network the network is composed of ego-node itself and its neighborhood. The ego-node and their friends represent the nodes. To each of them a tie is drawn. If two friends of an ego-node are also friends there is also a tie drawn between them. A table of the statistical measures is illustrated in Table 6.3.

Statistics of the Facebook network	
Nodes	4039
Edges	88234
Nodes in largest SCC	4039 (1.000)
Edges in largest SCC	88234 (1.000)
Average clustering coefficient	0.6055
Number of triangles	1612010
Fraction of closed triangles	0.2647
Diameter (longest shortest path)	8
90-percentile effective diameter	4.7

Table 6.3.: Facebook network dataset statistics (Leskovec and Krevl, 2014).

The network is compounded of overall 10 ego-nodes and their neighborhoods. The type of the network is significantly different than for the GR-QC network. The groups are known a priori and can be compared to the outcome of the simulations. The GR-QC network on the other hand depicts a network where all nodes share a common feature. The whole graph for the facebook is plotted in Figure 6.10. In this graph the ego-clusters can already be observed.

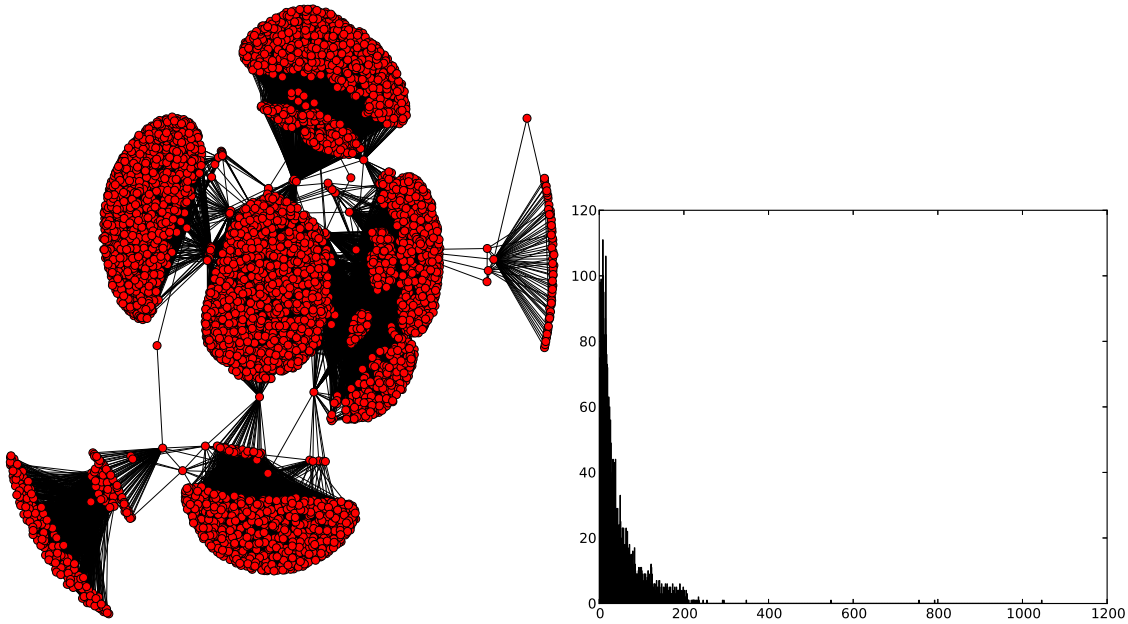
The degree distribution is plotted in Figure 6.10. It exhibits an exponential distribution like the GR-QC network which is the property for scale-free networks (Albert and Barabási, 2002).

The two properties for small-world networks are a high average cluster coefficient and a short average path length. They are both fulfilled by the network. The clustering coefficient C is ≈ 0.6055. The average number of links per node k is ≈ 21.8455. The average cluster coefficient of a random network with these parameters according to $C_{random} \sim k/n$ is 0.0054. The first property $C \gg C_{random}$ is thus fulfilled. The average shortest path length L of the network is 3.6925. According to $L_{random} \sim ln(n)/ln(k)$ the average shortest path length of a random network with the same properties is 2.6925. Thus the actual average shortest path length L is not significantly higher than for L_{random} and the second property for small-world networks is fulfilled. (Watts and Strogatz, 1998)

The unsorted and sorted *Fiedler vectors* are plotted in Figure 6.11. The unsorted vector already depicts some plateaus and reflects the structure of the unsorted matrix. The sorted vector on the other hand exposes more clusters.

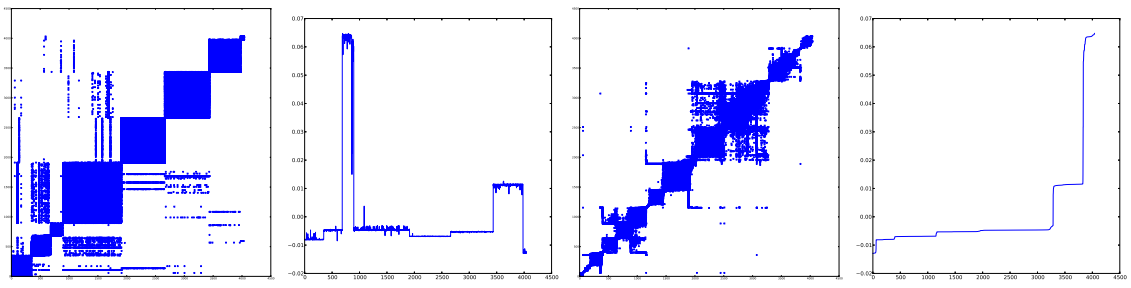
In the adjacency matrix which is illustrated in Figure 6.11 the individual Ego-node subgraphs can be seen roughly. The sorted adjacency matrix give more insight in the group structure. It reveals some more clusters in the network. Most of them can be seen very accurately, although two interwoven clusters remain.

[1] https://www.facebook.com
[2] https://www.facebook.com/apps/application.php?id=201704403232744



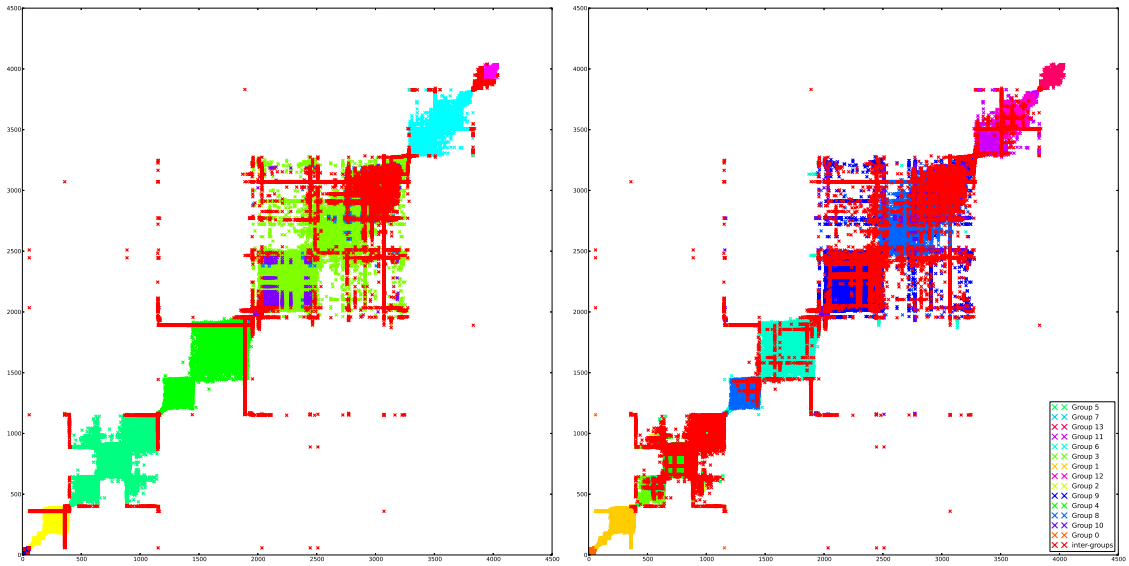
(a) Facebook graph (b) Degree distribution

Figure 6.10.: *The Facebook graph and its degree distribution.* (a) The network is compounded of 10 Ego-node networks (b) The network exhibits a power-law behaviour which is the property of scale-free networks.



(a) Adjacency matrix (b) Unsorted *Fiedler vector* (c) Sorted adjacency matrix (d) Sorted *Fiedler vector*

Figure 6.11.: *Spectrum analysis of the facebook network.* (a) The adjacency matrix of the Facebook network (c) The sorted adjacency matrix according to its *Fiedler vector*.



(a) Ego-networks (b) Final groups run 1

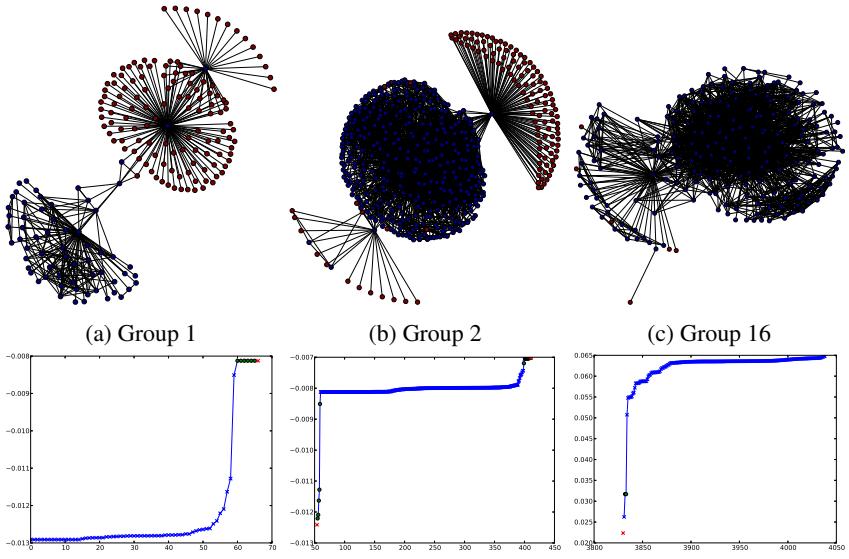
Figure 6.12.: *Colored adjacency matrices of the Facebook network.* (a) The sorted adjacency matrix where each ego-network has one color assigned. Links starting and ending in different groups are red. The majority of the networks either builds its own cluster or splits into multiple clusters. On the top right two clusters coalesce. (b) The same matrix but each group from run 1 has one color assigned. It is similar in all of the simulation outcomes.

Overall six simulations with $t_{max} = 100,000,000$ iterations were executed. To check if more iterations show different results two more runs with $t_{max} = 200,000,000$ were executed. How many groups survived at the end of each simulation is shown in Table 6.4. With $t_{max} = 100,000,000$ the usual outcome is 16. For $t_{max} = 200,000,000$ the outcomes are 13 and 14 groups. Since the last changes in N_d happen late the doubling of the number of iterations have an impact on the outcome.

Results for the simulations	
$t_{max} = 100,000,000$	
Run	Final groups
Run 1	14
Run 2	16
Run 3	16
Run 4	16
Run 5	16
Run 6	17
$t_{max} = 200,000,000$	
Run	Final groups
Run 7	14
Run 8	13

Table 6.4.: Results for the simulations on the facebook network.

The sorted adjacency matrix with each ego-network in its own color is plotted in Figure 6.12. Most communities can be seen very accurately, although three ego-networks coalesce in one cluster in the spectral analysis. In the sorted adjacency matrix they lie in the range of 2,000 - 3,300. Two further ego-networks



(a) Group 1 (b) Group 2 (c) Group 16

(d) (e) (f)

Figure 6.13.: *Group 1, 5, 16 of the Facebook network.* (a) - (c) Graph plots of the groups (d) - (e) The range of the *Fiedler vector* where the nodes of the group are located. The green dots are nodes which do not belong to the groups. The groups are contained in all simulation outcomes. They lie at the beginning or at the end of the vector and can be separated very precisely by their gap.

coalesce which is the cluster in the top right corner of the matrix. Three of the ego-networks split up in multiple communities.

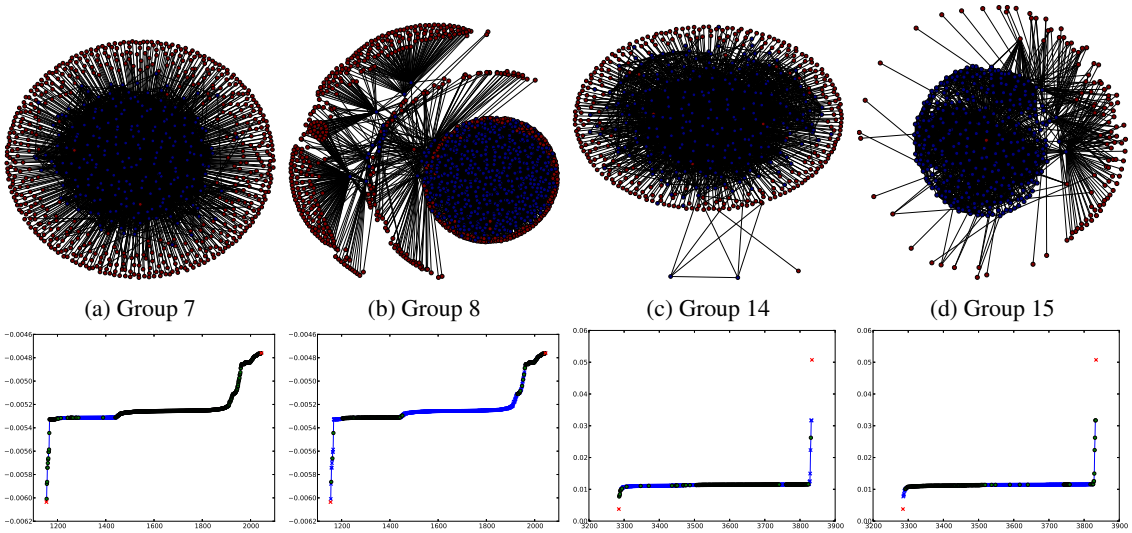
For the analysis of Facebook network a distinction between 17 different groups is made. The groups which can be separated based on the gap are 1, 2 and 16. They are plotted in 6.13. They are located at the beginning or the end of the *Fiedler vector*. If the remaining groups are separated based on their gap, the distinction would be on a too large scale such that they need to be separated further. Although the ordering of the nodes based on the *Fiedler vector* breaks the separation of the groups down to one dimension, it is remarkably good. What is also common to all of the groups is that they show the characteristic curve which could already be observed in the synthetic networks. The better they are connected within the cluster and the less they are connected to the outside the more they build a plateau. The more they are connected to another cluster the flatter the curve gets to it on the vector.

Each of the ego-networks 4 and 8 can be separated based on the gaps. Both contain two groups which ranges varies in the simulation outcomes. They are plotted in Figure 6.14.

The ego-network 3 can also be separated based on its gaps. It contains four groups in the simulation outcomes. It is plotted in Figure 6.15.

The three coalesced ego-networks which where mentioned before showed very unstable outcomes in the simulations. They can be separated as a whole based on their gaps. They usually contain four groups in the outcomes. They are plotted in Figure 6.16.

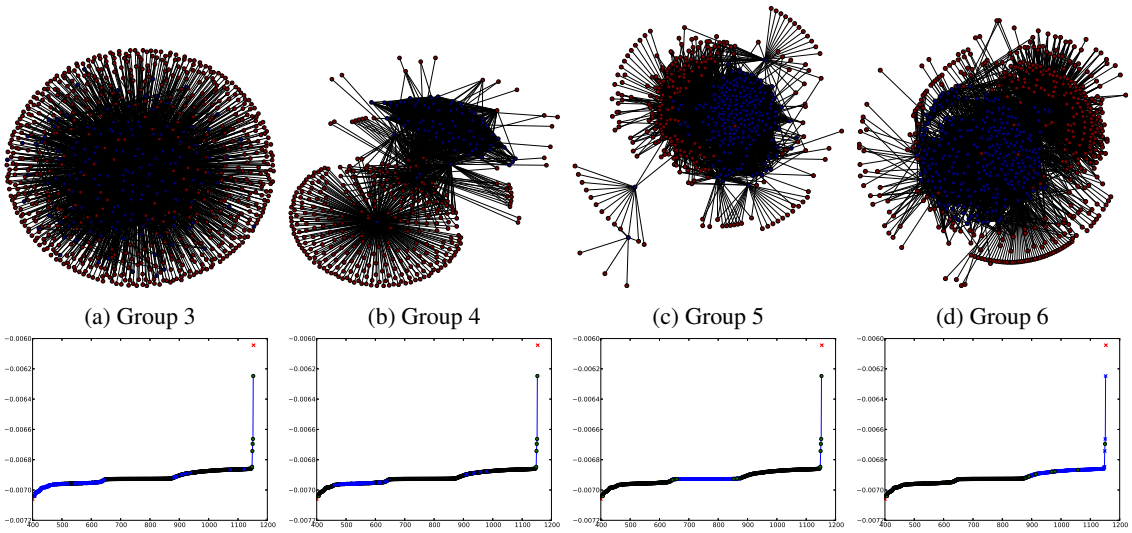
A typical pattern from the GR-QC network where nearly all nodes have the same value in the vector can be seen in the plots for the groups 9 and 17. They are highly connected and have only a few links to the outside. They are plotted in Figure 6.17.



(a) Group 7 (b) Group 8 (c) Group 14 (d) Group 15

(e) (f) (g) (h)

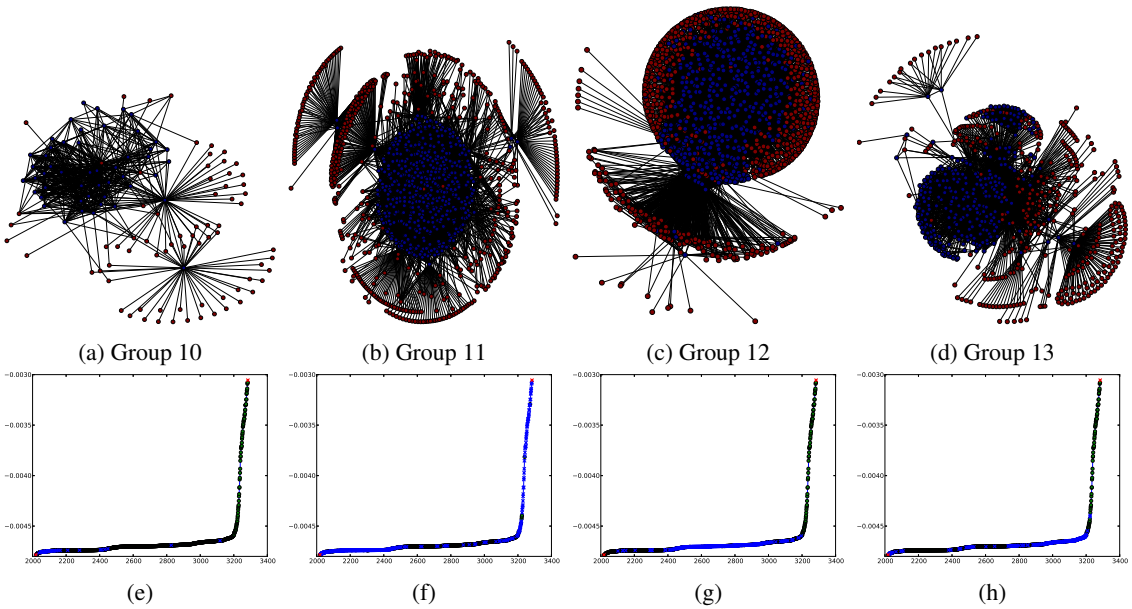
Figure 6.14.: *Ego-network 4 (group 7 and 8) and 8 (group 14 and 15) of the Facebook network.* (a) - (d) Graph plots of the groups (e) - (f) The range of the *Fiedler vector* where the nodes of the group are located. The green dots are mostly the nodes from the second group of the corresponding ego-network. Each of the ego-networks builds one community in the vector if they are separated by their gaps. The groups are contained in all of the simulation outcomes although the ranges on the vector vary within their ego-network. They show a pattern which could already be observed in the synthetic networks. Due to the one-dimensional ordering in the vector some nodes are placed in a different community and the gap shrinks such that a separation based on it is hard. The typical curve for communities although is retained.



(a) Group 3 (b) Group 4 (c) Group 5 (d) Group 6

(e) (f) (g) (h)

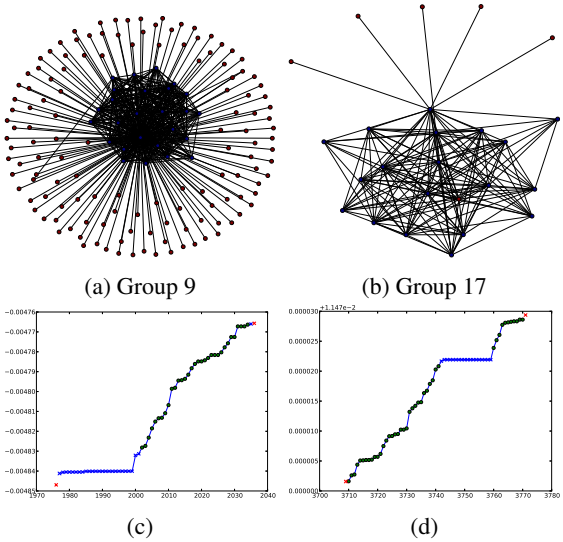
Figure 6.15.: *Ego-network 3 of the Facebook network.* (a) - (d) Graph plots of the groups (e) - (f) The range of the *Fiedler vector* where the nodes of the group are located. The green dots are other nodes from the ego-network. The ego-network itself can be separated very precisely by its gap. The groups are contained in all of the simulation outcomes. The groups 3, 5 and 6 can be seen very accurately with their characteristic curve. Group 4 shows the same curve but on a much smaller scale.



(a) Group 10 (b) Group 11 (c) Group 12 (d) Group 13

(e) (f) (g) (h)

Figure 6.16.: *Coalesced ego-networks 5, 6 and 7 of the Facebook network.* (a) - (d) Graph plots of the groups (e) - (f) The range of the *Fiedler vector* where the nodes of the group are located. The green dots are other nodes from the ego-network. The cluster containing all three ego-networks can be separated very precisely by its gap. By looking at the range of the vector two groups would be expected. The ranges from group 10 and 11 vary heavily in the simulation outcomes. Groups 12 and 13 behave the same way.



(a) Group 9 (b) Group 17

(c) (d)

Figure 6.17.: *Group 9 and 17 of the Facebook network.* (a), (b) Graph plots of the groups (c), (d) The range of the *Fiedler vector* where the nodes of the group are located. The green dots are nodes which do not belong to the groups. They are contained in some of the simulation outcomes. The vector shows a similar pattern which could be observed in the GR-QC and synthetic networks. The nodes are highly connected and have only a few links outside. Thus the values on the vector are nearly the same and build a plateau.

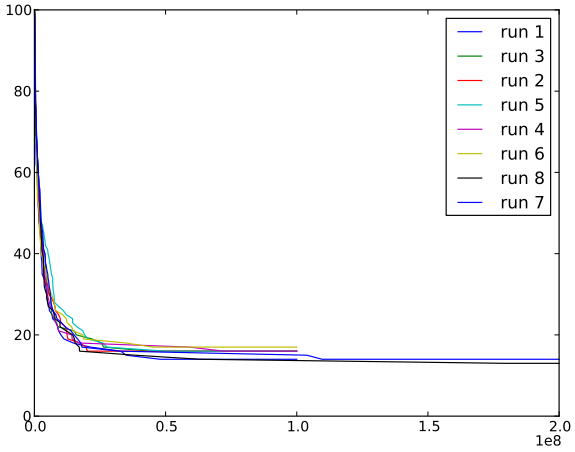


Figure 6.18.: *Number of different words N_d over time t for all runs of the Facebook network.*

The evolution of the number of different words N_d over the number of iterations t is plotted in Figure 6.18. The maximum of 100 is reached at $t \in [401,981]$ with its median at 572. N_d then decays with the typical exponential decay until it stays equal at approximately $t \in [20000000,60000000]$. One outlier is in run 4 at $t \approx 70,000,000$. The two runs with $t_{max} = 200,000,000$ show additional behavior above $t = 100,000,000$. In run 7 N_d reaches 15 at $t = 104,000,000$ and decreases further by one at $t = 110,000,000$. Run 8 reaches its final state at $t = 180,000,000$.

Chapter 7

Conclusion

This thesis investigates how the outcome of the Naming Game simulations can be precalculated through spectral analysis. Several models for describing the process of opinion formation exist. They are all based on the assumption that *microscopic* interactions between individuals form observable *macroscopic* qualities (e.g. religion, political viewpoint, cultural traits) on the long run. It was already shown that the primary cause for hindering the individuals from reaching global consent in networks is community structure.

The Naming Game is a minimalistic model which advances from the field of linguistics. Initially it was used for bootstrapping a common vocabulary for an entity, like the spreading of co-existing dialects. It was already investigated on various network topologies, among complete graphs, low-dimensional lattices, small-world or scale-free networks. In all of the listed topologies global consensus is reached, i.e. only one word survives at the end of a simulation. On social networks however, it could be observed that reaching consensus is hindered, which is a consistent behavior with the other models. The clusters - each holding a different single word - heavily correlate to the empirical datasets describing the social community structure of networks. The Naming Game is thus highly valuable for gaining insight in the process of opinion formation.

Nevertheless the Naming Game depicts the problem that it is resource intensive in both memory need and computational power. This thesis tries to reduce the finding of communities which prevail at the end of the simulations to the well-known community detection method spectral analysis. In the spectral analysis the nodes are reordered according to their value in the *Fiedler vector*. Based on the gaps in the vector they are separated in communities.

The vector was first investigated on synthetic networks. In these networks the links between communities were generated with one equal link probability and the links bridging the communities with a second probability. The results showed a characteristic curve for each community on the vector. The values increase sharply, change to a almost linear slope and then increase again sharply. The more communities are present in the network the more they build a plateau in the vector, which makes them easier to separate. This effect can be explained by the fact that the probability for links between the communities decreases with the number of communities. One problem arises due to the one-dimensional reordering of the nodes. If more than two communities are present they cannot be always separated by their gap. The characteristic curve is although retained and the ordering of the nodes is remarkably good.

The vector was then analyzed on two empirical networks which exhibit strong community structure. One was a scientific co-authoring network and the other one was a subset of the Facebook network. The results reflected the observations from the synthetic networks on a larger scale. Some communities merged if they were separated by their gaps. The characteristic curve was again retained in all of the communities and the ordering of the nodes remained accurate for communities which were included in all of the simulation outcomes. For communities which are contained in only some of the simulations, the ordering was not good either.

One improvement of the spectral analysis could be to split the network at the biggest gap in the vector and consecutively repeat this step. It is already a well-known technique for community detection. This results in additional time complexity, which should be justifiable due to better results. Although the problem would be raised when the procedure terminates. Further simulations would be needed to find appropriate thresholds.

Appendix A

Class diagrams

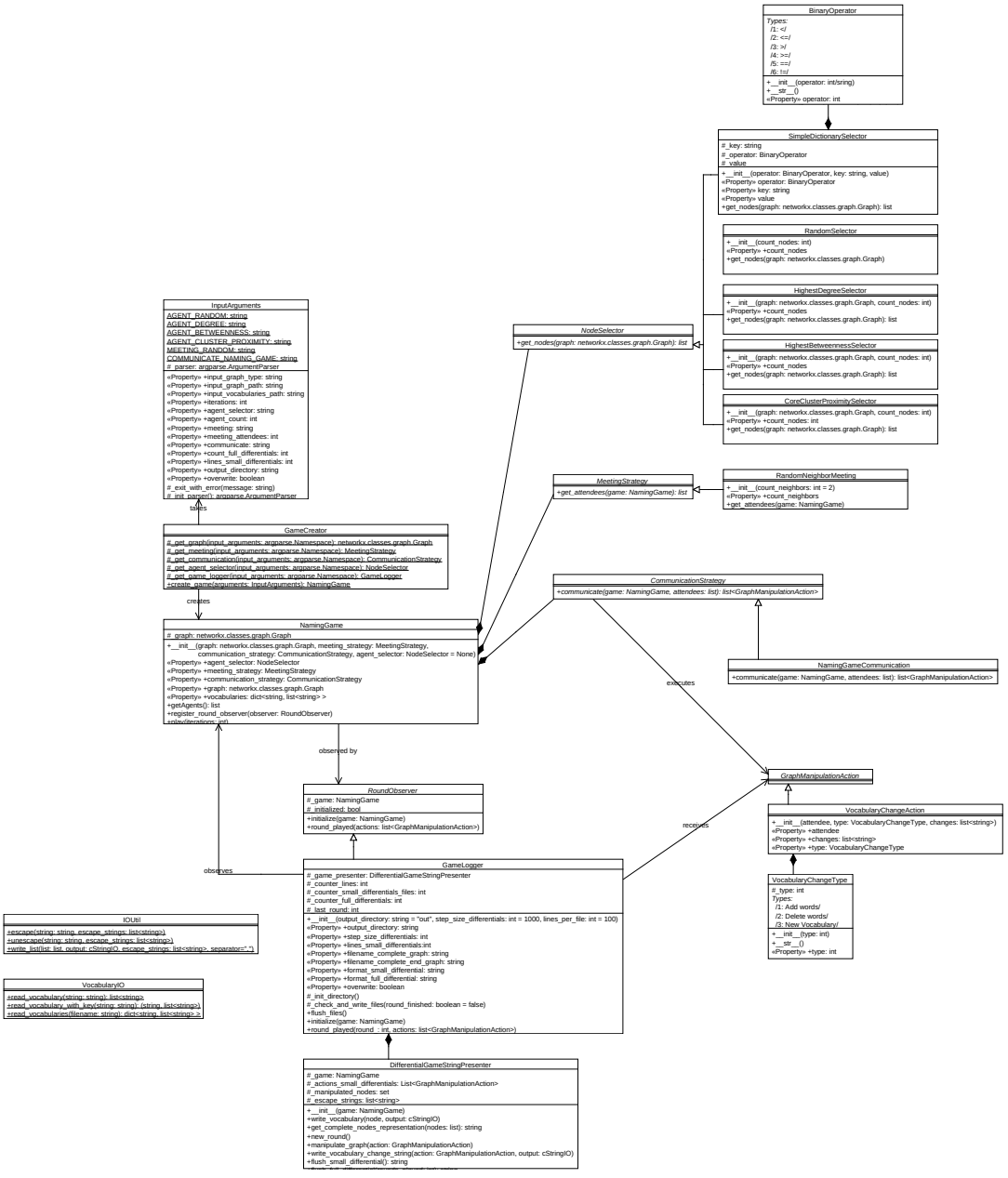


Figure A.1.: *The naming game class diagram.*

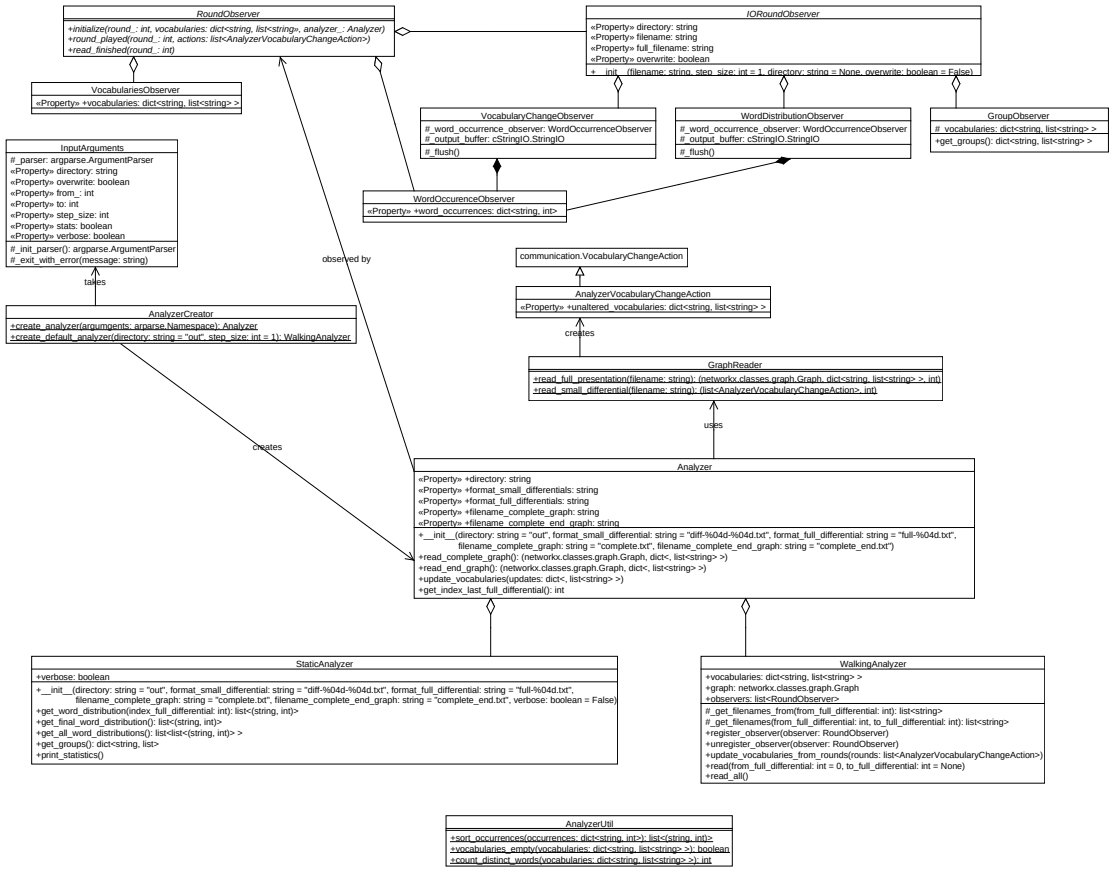


Figure A.2.: *Analyzer class diagram.*

Bibliography

ADAMIC, L. A. AND GLANCE, N. 2005. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*. ACM, 36–43. (Cited on page 1.)

ALBERT, R. AND BARABÁSI, A.-L. 2002. Statistical mechanics of complex networks. *Reviews of modern physics 74,* 1, 47. (Cited on pages 41 and 49.)

ANGHEL, M., TOROCZKAI, Z., BASSLER, K. E., AND KORNISS, G. 2004. Competition-driven network dynamics: Emergence of a scale-free leadership structure and collective efficiency. *Physical review letters 92,* 5, 058701. (Cited on page 1.)

AXELROD, R. 1997. The dissemination of culture a model with local convergence and global polarization. *Journal of conflict resolution 41,* 2, 203–226. (Cited on page 4.)

BARABÁSI, A.-L. AND ALBERT, R. 1999. Emergence of scaling in random networks. *science 286,* 5439, 509–512. (Cited on page 1.)

BARONCHELLI, A., DALL'ASTA, L., BARRAT, A., AND LORETO, V. 2006. Topology-induced coarsening in language games. *Physical Review E 73,* 1, 015102. (Cited on pages 5 and 39.)

BARONCHELLI, A., FELICI, M., LORETO, V., CAGLIOTI, E., AND STEELS, L. 2006. Sharp transition towards shared vocabularies in multi-agent systems. *Journal of Statistical Mechanics: Theory and Experiment 2006,* 06, P06014. (Cited on pages 5 and 39.)

CANDIA, J. AND MAZZITELLO, K. I. 2008. Mass media influence spreading in social networks with community structure. *Journal of Statistical Mechanics: Theory and Experiment 2008,* 07, P07007. (Cited on pages 1 and 6.)

CASTELLANO, C., FORTUNATO, S., AND LORETO, V. 2009. Statistical physics of social dynamics. *Reviews of modern physics 81,* 2, 591. (Cited on page 4.)

CHOMSKY, N. 1988. *Aspects of the Theory of Syntax*. Vol. 11. MIT press. (Cited on page 4.)

CLIFF, D. 1994. *From animals to animats 3: proceedings of the Third International Conference on Simulation of Adaptive Behavior*. Vol. 3. MIT Press. (Cited on page 4.)

DALL'ASTA, L., BARONCHELLI, A., BARRAT, A., AND LORETO, V. 2006. Agreement dynamics on small-world networks. *EPL (Europhysics Letters) 73,* 6, 969. (Cited on pages 5 and 39.)

DALL'ASTA, L., BARONCHELLI, A., BARRAT, A., AND LORETO, V. 2006. Nonequilibrium dynamics of language games on complex networks. *Physical Review E 74,* 3, 036105. (Cited on pages 5 and 8.)

DAVIS, R. 1982. Report on the second workshop on distributed ai. (Cited on page 3.)

DONATH, W. E. AND HOFFMAN, A. J. 1972. Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices. *IBM Technical Disclosure Bulletin 15,* 3, 938–944. (Cited on page 10.)

DÜCHTING, W. AND VOGELSAENGER, T. 1984. Analysis, forecasting, and control of three-dimensional tumor growth and treatment. *Journal of Medical Systems 8*, 461–475. (Cited on page 3.)

EPSTEIN, J. M. 1996. *Growing artificial societies: social science from the bottom up*. Brookings Institution Press. (Cited on pages 1 and 4.)

ERDÖS, P. AND RÉNYI, A. 1959. On random graphs i. *Publ. Math. Debrecen 6*, 290–297. (Cited on page 9.)

FIEDLER, M. 1973. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal 23,* 2, 298–305. (Cited on page 10.)

FIEDLER, M. 1975. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal 25,* 4, 619–633. (Cited on page 10.)

GALAM, S. 2000. Real space renormalization group and totalitarian paradox of majority rule voting. *Physica A: Statistical Mechanics and its Applications 285,* 1–2, 66 – 76. (Cited on page 4.)

GIRVAN, M. AND NEWMAN, M. E. 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences 99,* 12, 7821–7826. (Cited on page 1.)

GROSS, J. L. AND YELLEN, J. 2003. *Handbook of graph theory*. CRC press. (Cited on page 11.)

HOLLAND, P. W., LASKEY, K. B., AND LEINHARDT, S. 1983. Stochastic blockmodels: First steps. *Social networks 5,* 2, 109–137. (Cited on page 9.)

HURFORD, J. R. 1989. Biological evolution of the saussurean sign as a component of the language acquisition device. *Lingua 77,* 2, 187–222. (Cited on page 5.)

KANSAL, A., TORQUATO, S., IV, G. H., CHIOCCA, E., AND DEISBOECK, T. 2000. Simulated brain tumor growth dynamics using a three-dimensional cellular automaton. *Journal of Theoretical Biology 203*, 367–382. (Cited on page 3.)

KONOLIGE, K. AND NILSSON, N. J. 1980. Multiple-agent planning systems. In *AAAI*. Vol. 80. 138–142. (Cited on page 3.)

LAMBIOTTE, R. AND AUSLOOS, M. 2007. Coexistence of opposite opinions in a network with communities. *Journal of Statistical Mechanics: Theory and Experiment 2007,* 08, P08026. (Cited on pages 1 and 6.)

LANDAU, D. P. AND BINDER, K. 2009. *A guide to Monte Carlo simulations in statistical physics*. Cambridge university press. (Cited on page 4.)

LESKOVEC, J. AND KREVL, A. 2014. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`. (Cited on pages 41 and 49.)

LIGGETT, T. M. 1999. *Stochastic interacting systems: contact, voter and exclusion processes*. Vol. 324. Springer. (Cited on page 4.)

LU, Q., KORNISS, G., AND SZYMANSKI, B. K. 2009. The naming game in social networks: community formation and consensus engineering. *Journal of Economic Interaction and Coordination 4,* 2, 221–235. (Cited on pages 1, 5, 8, and 14.)

LUCE, R. D. AND PERRY, A. D. 1949. A method of matrix analysis of group structure. *Psychometrika 14,* 2, 95–116. (Cited on page 11.)

MAES, P. 1990. *Designing autonomous agents: Theory and practice from biology to engineering and back*. MIT press. (Cited on page 4.)

MANSURY, Y., DIGGORY, M., AND DEISBOECK, T. S. 2005. Evolutionary game theory in an agent-based brain tumor model: Exploring the 'genotype–phenotype' link. *Journal of Theoretical Biology 238*, 146–156. (Cited on page 3.)

MCPHERSON, M., SMITH-LOVIN, L., AND COOK, J. M. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 415–444. (Cited on page 1.)

MINSKY, M. 1988. *The Society of Mind*. Simon and Schuster. (Cited on page 3.)

MOHAR, B. AND ALAVI, Y. 1991. The laplacian spectrum of graphs. *Graph theory, combinatorics, and applications 2*, 871–898. (Cited on page 11.)

NEUMANN, J. V. 1966. *Theory of self-reproducing automata*. University of Illinois Press, Champaign, IL. (Cited on page 3.)

P. GERLEE, A. A. 2007. An evolutionary hybrid cellular automaton model of solid tumour growth. *Journal of Theoretical Biology 246*, 583–603. (Cited on page 3.)

SCHELLING, T. C. 1969. Models of segregation. *The American Economic Review*, 488–493. (Cited on page 4.)

SCHELLING, T. C. 1971. Dynamic models of segregation. *Journal of mathematical sociology 1,* 2, 143–186. (Cited on page 4.)

SEIDMAN, S. B. 1983. Network structure and minimum degree. *Social networks 5,* 3, 269–287. (Cited on page 11.)

SPIELMAT, D. A. AND TENG, S.-H. 1996. Spectral partitioning works: Planar graphs and finite element meshes. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*. IEEE, 96–105. (Cited on pages 10 and 11.)

STEELS, L. 1995. A self-organizing spatial vocabulary. *Artificial life 2,* 3, 319–332. (Cited on page 5.)

STEELS, L. 2005. The emergence and evolution of linguistic structure: from lexical to grammatical communication systems. *Connection science 17,* 3-4, 213–230. (Cited on page 4.)

SZNAJD-WERON, K. 2005. Sznajd model and its applications. *arXiv preprint physics/0503239*. (Cited on page 4.)

ULAM AND MARCIN, S. 1960. *A Collection of Mathematical Problems*. Interscience Publishers. (Cited on page 3.)

WANG, Y. J. AND WONG, G. Y. 1987. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association 82,* 397, 8–19. (Cited on page 9.)

WATTS, D. J. AND STROGATZ, S. H. 1998. Collective dynamics of 'small-world'networks. *nature 393,* 6684, 440–442. (Cited on pages 1, 5, 42, and 49.)

WOLFRAM, S. 1984. Cellular automata as models of complexity. *Nature 311*, 419–424. (Cited on page 3.)

ZACHARY, W. W. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 452–473. (Cited on page 19.)

ZUSE, K. 1969. *Rechnender Raum*. Vieweg, Wiesbaden. (Cited on page 3.)