university of groningen / faculty of science and engineering

BACHELOR'S PROJECT APPLIED MATHEMATICS

Opinion Dynamics in Spectral Clustering

Author:
H.M.G. van der Zande

First supervisor: dr. ir. B. Besselink
Second assessor: dr. A.E. Sterk

Rijksuniversiteit Groningen, The Netherlands,
Faculty of Science and Engineering

July 12, 2020

Abstract

This thesis addresses the problem of clustering networks and data sets. Networks consist of agents and links connecting the agents. Networks are used in many different fields of science such as biology and social sciences. Clustering networks can be seen as making groups of agents based on similarity. The goal of clustering networks is to improve interpretations and to facilitate a way to make the analysis of big networks easier. Even though many different methods already exist, this thesis explains a methodology based on opinion dynamics. This method relies on every agent having an opinion, which will be updated at every time step. Due to convergence of opinions, agents eventually form clusters. This methodology will be explained and substantiated with mathematical proofs. Furthermore, simulations will be performed on specific examples to understand the method. Finally, this method will be applied on an existing data set to see how this method could be used in practice.

Contents

1. Introduction

1.1 Graph theory

The famous Swiss mathematician Leonhard Euler posed a problem, which is now well-known. This problem involves seven bridges in a city and the quest to find a route to cross all seven bridges exactly once; the so-called Köningsberg Bridge Problem [13].

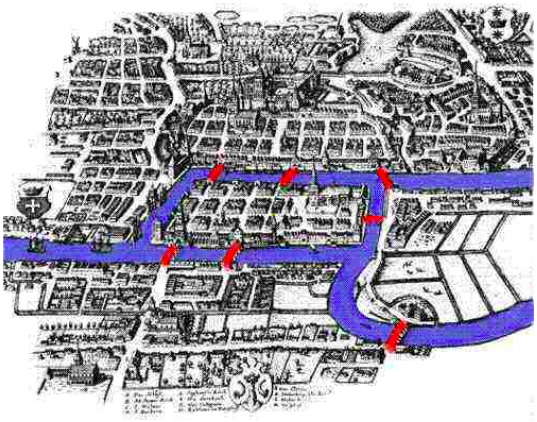


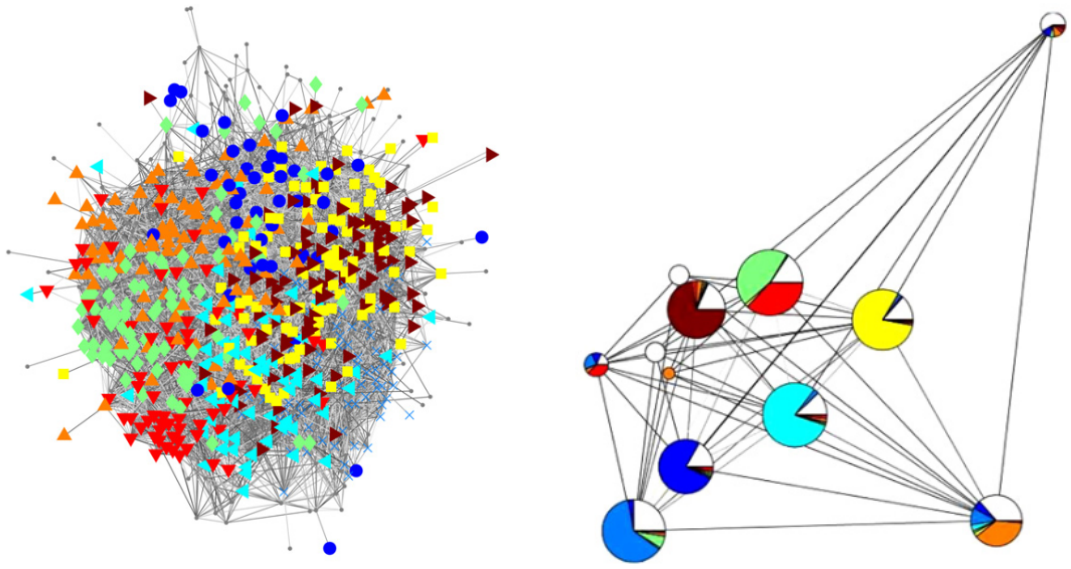
Figure 1.1: Köningsberg Bridge Problem [16]

Euler wrote an article on this problem in 1736, which is seen by many mathematicians as the beginning of graph theory. Graphs, or in some articles and books referred to as networks, can be defined as a visualization of vertices, also known as agents, and edges that join agents together [15]. Graphs can represent any type of network, from small and simple ones to big and complex ones. For example, graphs are found in the field of biology, when studying cellular systems, interactions within proteins or metabolic systems [8]. However, not only in the field of biology graphs are regularly used, but also social sciences and many other different fields. This has led to an increasing interest in these complex networks; how does one look at a network properly? What is the use of these networks? And are there ways to make interpretation of these networks more easy and efficient?

1.2 Problem description

This last question has led to the idea of finding subgroups within a graph. Imagine having a big data set, about which you need to draw a conclusion. If this big data set can be split up in subgroups based on a particular characteristic, analyzing this set becomes a lot easier. The problem addressed in this thesis is the problem of finding clusters in any graph, representing any type of network. Roughly speaking, clusters can be defined as groups of vertices for which the number of connections within this cluster is large compared to the number of connections between different clusters [8, 14]. To illustrate the idea of finding clusters in networks, consider any type of social media website such as *Facebook* or, more a professional one, *LinkedIn*. These websites allow friends, families and colleagues to communicate over big distances or to share their opinions, ideas and job vacancies with others. Anonymised data of these social websites can be used to find for example groups of friends or families. In this case people and their user pages, are considered as the agents and edges between agents can represent a friendship, following

the same celebrities or a tag in the same photo, just to name some options. By means of an algorithm, clusters of 'friendships' within five different universities in the United States have been found in [21]. Here connections represent a friendship and only friendships between students of the same universities are taken into account. In this paper, the researchers determined to what extent the demographic labels, that were included in the anonymised data, are in correspondence with the clusters found by the algorithms. In Figure 1.2 the results haven been visualized. From these figures it can be concluded that the clusters found are quite compatible with the housing of the students.



(a) Visualization of network of friendships between students at Caltech [8]. Every dot represents a user page, and hence a student, and every visible line an online friendship between two students.

(b) The different communities of the network [8]

Figure 1.2: The clusters of friendship networks of students at Caltech (with data used from September 2005). The different colors and shapes represent the different dormitories of the students.

This example shows that finding these clusters can benefit for example marketing companies, knowing that it might not be necessary to approach every individual, but only a few of every cluster. In the more advanced and technical cases, such as finding groups of related genes, the results can be used to find which genes might be relevant for a disease, by extracting associations between properties of the genes and the disease [8].

1.2.1 Existing clustering methods

Already in 1927, an article was published on finding groups of people in political bodies based on their voting behaviour [18]. Later, in 1955, an article, written by R.S. Weiss and E. Jacobsen, was published [12]. Weiss and Jacobsen studied workgroups within a government agency and made subgroups by removing agents that were working with people of various workgroups [8]. This concept of deleting connections to form groups formed a basis for the more modern algorithms and methods of finding clusters.

Over the years many different methods appeared. A popular way of finding clusters is known as the k-means clustering, which is a method to find partitions of sets [8]. This method uses distances between agents to find clusters. An extension to the k-means clustering is the fuzzy

k-means clustering, since it allows for points to belong to more than one partition. The fuzzy k-means clustering is widely used in pattern recognition [8]. The disadvantage of these two methods is that for them to work, the number of clusters needs to be determined beforehand. Since this is in many different examples not possible, these methods can often not be used.

Another category of methods to find clusters is the so-called *spectral clustering*, which is characterized by the use of eigenvalues of matrices associated to graphs. This spectral clustering was first performed by W.E. Donath and A.J. Hoffmann and published in 1973[1]. Donath and Hoffman used eigenvectors of the adjacency matrix of the graph. Later, the use of the Laplacian matrix was introduced by Miroshlav Fiedler, who used the algebraic connectivity of a graph G, which is the second smallest eigenvalue of the Laplacian matrix of G [5–7]. In some literature this is known as the Fiedler value, and its corresponding vector as the Fiedler Vector [20].

1.2.2 Opinion dynamics

In this thesis, a method for clustering described by Morărescu and Girard in [14] will be studied. Their method uses opinion dynamics to solve the problem of a partition of a graph, and hence finding clusters. They consider a graph in the original way; a set of agents and a set containing edges connecting the agents. The core of their methodology relies on agents having evolving opinions. Every agent i in V is assumed to have time dependent opinion and is denoted $x_i(t)$, which makes the problem a discrete dynamical time system. An initial opinion is assigned to every single agent and at every time-step their opinions are updated by taking a weighted average of its own and its neighbors' opinions. Finally, after sufficient number of time-steps, the opinions of every agent will converge to a limit value. This limit is the also referred to as the asymptotic opinion of an agent and will be further used in Chapter 2. If it turns out that two agents converge to the same limit, they are considered to belong to the same cluster. Finding a partition is characterized by the fact that an agent can only belong to one cluster.

1.2.3 Approach

In order to reach the goal of finding clusters, this dynamical system needs to be understood. In this thesis the opinion dynamics model of Morărescu and Girard will be explained. With the help of MATLAB, simulations of their model will be conducted. Finally, for certain cases, the behaviour will be analyzed, explained and discussed, using eigenvalues and eigenvectors. Furthermore, an application for customer services of this algorithm to a large data set, which represents items that van be bought online, will be presented as well.

[1] I was not able to find their original article as a PDF file, but it was referred to in [8].

2. Model description

The basic concept of opinion dynamics has been explained briefly in Subsection 1.2.2. Morărescu and Girard implemented opinion dynamics in a discrete dynamical system, which eventually leads to solution for the main problem of finding clusters in graphs. In this chapter the details of their model will be explained, as well as background knowledge necessary to understand their model.

2.1 Preliminaries

In order to understand their model, some concepts from matrix theory will be explained first. These concepts are used to prove important statements for this thesis.

Perron-Frobenius theorem

The Perron-Frobenius theorem, which was derived by Oscar Perron and later generalized by Ferdinand Georg Frobenius, deals with positive matrices [17]. A matrix is called a positive matrix, if it only has entries that are strictly positive. This theorem makes statements about the eigenvalues of a matrix and special eigenvectors can be defined. However, if a matrix is non-negative, with all entries non-negative, an additional property is needed for the Perron-Frobenius theorem. Therefore, there are different theorems for positive and non-negative matrices.

For any square matrix A, the set $\sigma(A)$ contains all eigenvalues of A and the spectral radius,

$$\rho(A) = \max\{|\lambda| \,|\, \lambda \in \sigma(A)\},$$

is the maximum of all eigenvalues in absolute value.

Theorem 2.1.1. *(Perron-Frobenius theorem [11, 17])*
If A is a positive matrix of size $n \times n$, then

1. *$\rho(A) > 0$, where $\rho(A)$ is the spectral radius of A.*
2. *$\rho(A)$ is a simple eigenvalue of A, hence has algebraic multiplicity equal to 1.*
3. *Any other eigenvalue λ of A is such that $|\lambda| < \rho(A)$, which makes $\rho(A)$ the eigenvalue of maximum modulus.*
4. *There exist vectors x and y, both positive, called the right- and left-Perron vectors of A satisfying $Ax = \rho(A)x$, $A^T y = \rho(A)y$ and $x^T y = 1$.*

If a matrix has entries equal to 0, which makes the matrix non-negative instead of positive, an extra restriction on this matrix is needed in order to apply the Perron-Frobenius theorem. The matrix must be irreducible as well. Matrix A is said to be irreducible if there does not exist a permutation matrix P which can bring A into block upper triangular form [11].

Theorem 2.1.2. *(Perron-Frobenius theorem for non-negative, irreducible matrices [11])*
If A is a square non-negative and irreducible matrix of size $n \times n$, then

1. *$\rho(A) > 0$.*
2. *$\rho(A)$ is a simple eigenvalue of A, with algebraic and geometrically multiplicity equal to 1.*
3. *The eigenvalue $\lambda = \rho(A)$ has a unique positive eigenvector. This unique positive eigenvector, for which holds that its components sum to 1 is called the right-Perron vector.*

Furthermore, since a square non-negative matrix A is irreducible if and only if its transpose, A^T, is irreducible, there must exist a positive left-Perron vector y as well. For this vector

$$y^T A = y^T \rho(A)$$

must hold. Since the characteristic polynomials of A and A^T are the same, their set of eigenvalues must completely overlap. This implies that their spectral radii, $\rho(A)$ and $\rho(A^T)$, are equal to one another.

Additionally, it is necessary to find a value for the spectral radius for these types of matrices. A theorem for a upper and lower bound for the spectral radii for matrices have been stated in the following theorem.

Theorem 2.1.3. *(Spectral radius of matrix A [23])*
If A is a non-negative matrix of size $n \times n$, then

$$\min_{1 \leq i \leq n} r_i(A) \leq \rho(A) \leq \max_{1 \leq i \leq n} r_i(A)$$

where $r_i(A) = \sum_{j=1}^{n} a_{ij}$ is defined as the the sum of the elements of row i, for all $1 \leq i \leq n$. Moreover, if A is irreducible, then either $\min_{1 \leq i \leq n} r_i(A) = \rho(A)$ or $\rho(A) = \max_{1 \leq i \leq n} r_i(A)$ holds if and only if $r_1(A) = \cdots = r_n(A)$.

Squeeze theorem

The squeeze theorem states that when a sequence is squeezed by two other sequences, for which the limits are the same, the squeezed sequence must converge as well.

Theorem 2.1.4. *(Squeeze theorem for convergent sequences [19])*
Let $(a_n)_{n \in \mathbb{N}}$ and $(c_n)_{n \in \mathbb{N}}$ be convergent sequences. If for all $n \geq n_0$, $a_n \leq b_n \leq c_n$ and $\lim_{n \to \infty} a_n = \lim_{n \to \infty} c_n = L$ holds, then $(b_n)_{n \in \mathbb{N}}$ must be convergent with limit $L = \lim_{n \to \infty} b_n$.

2.2 Fundamentals

In mathematical notation, a graph consisting of n agents is defined as $G = (V, E)$. Here the sets $V = \{1, 2 \ldots, n\}$ and $E \subseteq V \times V$ represent the set of agents and (initial) edges respectively [14]. In the methodology of Morărescu and Girard, the set E has some restrictions. First of all, the set must be symmetric; $(i, j) \in E$ if and only if $(j, i) \in E$ for any $i, j \in V$. Being symmetric implies that we deal only with undirected graphs, which means that that if agent i is connected to agent j, the converse must be true as well. Secondly, E must also be anti-reflexive; $(i, i) \notin E$ for any $i \in V$. The adjacency matrix of the graph G, $A(G)$, is a matrix representation of the graph. Since we are only dealing with undirected graphs, $A(G)$ is symmetric and has elements $a_{ij}(G)$ equal to 1 if agent i and j are connected, 0 otherwise.

Recall that the clustering is based on a dynamical system. At every time step every agent's opinion is updated by means of taking the weighted average

$$x_i(t+1) = \sum_{j=1}^{n} p_{ij}(t)x_j(t), \tag{2.1}$$

which can be stated in matrix-vector form

$$x(t+1) = P(t)x(t). \tag{2.2}$$

Here $x(t)$ represents a vector containing the opinion of each agent; $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T$, with $x^0 = x(0)$ representing the initial opinion vector. In the weighted average opinions of other agents are taken into account for calculating the new opinion of agent i. However, not every agent's opinion is taken into account and hence not every element of $P(t)$ is non-zero. However, agent i will always take into account its own opinion. The meaning of an interacting neighbor in this method is not the same as the meaning of a neighbor. At each time-step not only the opinions are updated, but also the set of edges $E(t)$ and hence the graph $G(t) = (V, E(t))$ are updated. The set $E(t)$ only contains $(i, j) \in E$ for which holds that the difference between the opinions of agent i and j is lower than a certain upper bound, which is called the confidence bound. This means that the elements $p_{ij}(t)$ are only non-zero if and only if j is either equal to i or an interacting neighbor of agent i. In formal notation agent j is considered to be an interacting neighbor of agent i at time t if j is an element of

$$N_i(t) = \{j \in V \mid ((i, j) \in E) \wedge (|x_i(t) - x_j(t)| \leq R\rho^t)\}. \tag{2.3}$$

with parameters $\rho \in (0, 1)$ and $R > 0$. The area of confidence of agent i at time t can now be defined as $[x_i(t) - R\rho^t, x_i(t) + R\rho^t]$. In order to simplify the equations, $E(t)$ and $G(t)$ are denoted as being solely dependent on time t, even though they both actually depend on $x(t)$. Since every agent $i \in V$ only interacts with agents that have an opinion contained in its own area of confidence, matrix $P(t)$ has to meet certain conditions. The first condition states that entries $p_{ij}(t)$ are only non-zero if and only if j is either equal to i or an element of $N_i(t)$. Secondly an assumption on the matrix $P(t)$ is made;

Assumption 1. *(Right-stochastic)*
For all $t \in \mathbb{N}$ the following two statements hold:

1. $p_{ij}(t) \in [0, 1]$ *for all* $i, j \in V$;
2. $\sum_{j=1}^{n} p_{ij}(t) = 1$ *for all* $i \in V$.

The latter states that every row of the matrix $P(t)$ sums to 1, which is also referred to as being right-stochastic. Hence, Assumption 1 will be referred to as the right-stochasticity assumption throughout this thesis. Again, $P(t)$ and $N_i(t)$ actually depend on $x(t)$, but denoted to depend solely on time for simplification. Rephrasing the restrictions on the matrix $P(t)$, using (2.1) and (2.3), one can state at every time, with $t \in \mathbb{N}$, agent i collects opinions of its neighbors in the graph $G(t) = (V, E(t))$ to determine the opinion for time $t + 1$. If the opinion of a neighbor j differs less from agent i's opinion than the given upper bound $R\rho^t$, agent i takes into account agent j's opinion and entry $p_{ij}(t)$ will be non-zero. This process is repeated at every time step and agent i will only take into account opinions of those agents whose opinion changes fast enough to their own. In this process of converging opinions it is not possible to form new links; the set of edges can only reduce in size and not increase.

As will be shown shortly, for every agent $i \in V$, the sequence $(x_i(t))_{t\in\mathbb{N}}$ converges. Using the fact that agents only interact with one another if they are within each others confidence bound, leads to an intuitive feeling that opinions come to an agreement no slower than $\mathcal{O}(\rho^t)$. This is formulated in the following proposition, for which the proof can be found in Appendix A.

Proposition 2.2.1. *The sequence of non-negative real numbers $(x_i(t))_{t\in\mathbb{N}}$ is a convergent sequence. For this limit*

$$\lim_{t\to\infty} x_i(t) = x_i^*$$

the following holds for all $t \in \mathbb{N}$

$$|x_i(t) - x_i^*| \leq \frac{R}{1-\rho}\rho^t.$$

As a consequence of this constraint on the order of convergence, links will disappear in the process. This means that, since the set of interacting neighbors change, the opinions of agents that are not interacting with each other are likely to converge to different limits. The convergence of opinions to different limits imply that the agents might not reach a global agreement, but only a local one. Agreements between agents will eventually lead to the wanted community structure within graphs.

Agents are considered to belong to the same cluster if these agents converge to the same limit. The notion of two agents converging to the same limit is called asymptotic agreement and will be used throughout this chapter.

Definition 2.2.1. *Asymptotic agreement:*
Any two agents $i, j \in V$ are said to asymptotically agree if $x_i^* = x_j^*$.

Asymptotic agreement is obviously an equivalence relation, which is used to define a cluster. Mathematically, $C \subset V$ is called a cluster of graph $G = (V, E)$ if all agents in C asymptotically agree.

2.3 Asymptotic agreement and asymptotic connectivity

The goal of this section is to prove a relation between two agents that asymptotically agree and these two agents being each other neighbors at all times. To achieve this goal, the set of edges is partitioned into two subsets:

$$E^f = \{(i,j) \in E \,|\, \exists\, T \in \mathbb{N} \,\text{such that}\, \forall\, t \geq T \,\text{it holds that}\, (i,j) \notin E(t)\}, \tag{2.4}$$

which is the set of edges, for which it holds that the corresponding agents stop interacting with each other within finite time. The set

$$E^\infty = \{(i,j) \in E \,|\, \forall\, t \in \mathbb{N},\ \exists\, s \geq t \,\text{with the property that}\, (i,j) \in E(s)\} \tag{2.5}$$

has a corresponding graph $G^\infty = (V, E^\infty)$ and exists of combinations of agents which interact with one another at any given time $t \in \mathbb{N}$. The set E is by definition a finite set, and hence so are E^f and E^∞. The number of time steps for which two agents keep interacting with each other, depends naturally on their opinion and hence E^f, E^∞ and T depend on the initial opinion vector x^0. If there exists an edge joining two agents together in the asymptotic graph G^∞, these agents are said to be asymptotically connected. If two agents i and j are not asymptotically connected, they are said to be asymptotically disconnected. Intuitively, there must be some kind of relation for two agents between asymptotic connectivity and asymptotic agreement.

2.3.1 Asymptotic connectivity implies asymptotic agreement

Morărescu and Girard formulated this intuitive relation in a proposition.

Proposition 2.3.1. *Under the stochasticity assumption, asymptotic connectivity of any two agents $i, j \in V$ implies asymptotic agreement of these agents.*

Proof. Take an arbitrary $(i, j) \in E^\infty$. By (2.5), there exists a sequence of non negative integers, strictly increasing, $(\tau_k)_{k\in\mathbb{N}}$, with the property that $(i, j) \in E(\tau_k)$ for all $k \in \mathbb{N}$. In order to show asymptotic agreement, the squeeze theorem will be used.
For any $k \in \mathbb{N}$ it holds that $|x_i(\tau_k) - x_j(\tau_k)| \leq R\rho^{\tau_k}$. Since $(\tau_k)_{k\in\mathbb{N}}$ and $\rho \in (0, 1)$, it follows that

$$\lim_{k\to\infty} \tau_k = \infty$$

and hence that

$$\lim_{k\to\infty} \rho^{\tau_k} = 0$$

which implies that $(R\rho^{\tau_k})_{k\in\mathbb{N}}$ is a convergent sequence. The squeeze theorem implies

$$\lim_{k\to\infty} (x_i(\tau_k) - x_j(\tau_k)) = \lim_{k\to\infty} x_i(\tau_k) - \lim_{k\to\infty} x_j(\tau_k) = 0.$$

Recall that x_i^* and x_j^* are the limits of $(x_i(t))_{t\in\mathbb{N}}$ and $(x_j(t))_{t\in\mathbb{N}}$ respectively, hence agent i and agent j must asymptotically agree, i.e.

$$x_i^* = \lim_{t\to\infty} x_i(t) = \lim_{k\to\infty} x_i(\tau_k) = \lim_{k\to\infty} x_j(\tau_k) = \lim_{t\to\infty} x_j(t) = x_j^*.$$

This proves the theorem. □

2.3.2 Asymptotic agreement implies asymptotic connectivity

This relation between asymptotic connectivity and asymptotic agreement can be extended. Namely, the converse of Proposition 2.3.1 can be proven as well, yet this proof is more detailed and more difficult. However, the converse statement is not true for any initial opinion vector x^0. Imagine a disconnected graph G, where every agent has initially the same opinion. Since all opinions are equal, they will all asymptotically agree. However, since a graph is disconnected if there is at least one agent for which there does not exist a path connecting it to any other agent, there must be at least one agent that is asymptotically disconnected from the rest of the agents. Hence, the converse of Proposition 2.3.1 can only be proven under additional assumptions. Since these assumptions use many definitions and notation not used before, these will be defined first.

If $I = \{v_1, \ldots, v_n\}$ is a subset of V, the set $E_I = E \cap (I \times I)$ contains the edges connecting agents in I. Furthermore, the subscript I can also be used for the opinion vector $x(t)$ and matrix $P(t)$. Namely, $x_I(t) = [x_{v_1}, \ldots, x_{v_n}]^T$ contains only the opinions of agents in the set I and has length $|I|$ and $P_I(t)$, which is generally not row stochastic, is a matrix of size $|I| \times |I|$ with elements $p_{v_i v_j}$. If for any $I \subset V$, there does not exist an agent in I that is connected to an agent in $V \setminus I$ in $G(t)$, then $x_I(t + 1) = P_I(t)x_I(t)$ holds and $P_I(t)$ is an row stochastic matrix. Moreover, if (I, E_I) is a connected graph, then $P_I(t)$ is also irreducible.

A subgraph of G can be defined by a subset I of V with the corresponding set of edges E_I. A specific example of a subgraph is $G' = (I, E')$. Here $E' \subset E_I$ is defined as the symmetric relation over $I \subseteq V$, $(i, j) = (j, i)$. Since we only take into account undirected graphs, it is always the case that if (i, j) is an element of E_I, then (j, i) must be an element of this set as well. The symmetric relation now implies that (i, j) is the same edge as (j, i) and hence the set E' has halve the number of elements compared with E_I. A subgraph G of G is called a spanning subgraph if this subgraph consists of all agents in G and the set of all spanning subgraphs of G is denoted $S(G)$.

Going steps forward in time, the opinion vector $x(t)$ is updated by taking the weighted average as in (2.2). This implies that if matrix $P(t)$ is invertible it is possible to go steps back in time

$$\begin{aligned} x(t+1) &= P(t)x(t), \\ &= P(t)P(t-1)x(t-1), \\ &\vdots \\ x(t+1) &= P(t)P(t-1)\dots P(0)x^0. \end{aligned}$$

by setting $x^0 = P(0)^{-1}P(1)^{-1}\dots P(t)^{-1}x(t+1)$. This property makes it useful for matrix $P(t)$ to be invertible, which leads Morărescu and Girard to the following assumption

Assumption 2. *The sequence of matrices $(P(t))_{t\in\mathbb{N}}$, satisfies the following two conditions:*

1. *For all $t \in \mathbb{N}$, the matrix $P(t)$ is invertible.*
2. *For any $t \in \mathbb{N}$, the matrix $P(t)$ depends on $G(t)$, $P(t) = P(G(t)) = P(G_t)$, and $P(G')$, with $G' = (I, E')$, is invertible.*

Here, the first condition can be met by taking for example a strictly diagonally dominant matrix $P(t)$, which automatically implies that the matrix is invertible [4][1]. A square matrix of size $n \times n$ is called strictly diagonally dominant if for all $i \in \{1, 2, \dots, n\}$ it holds that

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|.$$

The second condition follows from applying the first condition on the fact that $P(t) = P(G(t))$. Using the former assumption a new set of matrices can be defined. The set

$$\mathcal{Q}_t = \{P(G_0)^{-1}P(G_1)^{-1}\dots P(G_{t-1})^{-1} \,|\, G_k \in S(G), 0 \leq k \leq t-1\} \tag{2.6}$$

consist of multiplications of inverted matrices, which define steps back in time. Since it is known that $S(G)$ is a finite set, $\mathcal{Q}_t$ must be a finite set too.

Moreover, in Proposition 2.2.1 is stated that the convergence of the agent's opinions in the set V can be no slower than $\mathcal{O}(\rho)$. The assumption has been made that this convergence is even faster than $\mathcal{O}(\rho)$.

Assumption 3. *(Fast convergence)*
There exists $\hat{\rho} < \rho$, such that for all $i \in V$ and any $t \in \mathbb{N}$ it holds that: $|x_i(t) - x_i^*| \leq M\hat{\rho}^t$

This assumption can be supported and confirmed by performing numerical experiments. Now, under Assumption 1 and 2 and using Assumption 3, it can be shown that after a finite number of time steps, the interactions between agents will no longer change, which is shown in Appendix A.

Lemma 2.3.2. *Under Assumption 1 and 2, $\exists T' \in \mathbb{N}$ such that $\forall\, t \geq T'$ it holds that $G(t) = G^\infty = G_\mathcal{C}$*

Here, $\mathcal{C} = V\backslash\sim = \{[i] \,|\, i \in V\}$ is the set containing all clusters. The set $[i]$ contains all agents $j \in V$ that asymptotically agree with agent i. Using the former statements, the converse of Proposition 2.3.1 can be stated.

[1] Farid refers to an article written by H. Rohrbach in 1931 for this statement, but since this article is written in German, I chose to refer to this article.

Theorem 2.3.3. *Under Assumption 1, 2 and 3 it holds for almost all initial opinion vectors x^0 that two agents, $i, j \in V$, asymptotically agree if and only if they are asymptotically connected.*

The statement from Theorem 2.3.3 does not hold for any initial opinion vector. Take for example an initial opinion vector where every agent has the same opinion, then this theorem is not likely to be true. As another example, if the opinion of one agent differs a lot from all of the other opinions, it is likely that this agent will end up without any interacting neighbors. If this agent was initially connected to many other, this outcome is not likely to be correct and hence Theorem 2.3.3 does not seem to hold.

Proof of Theorem 2.3.3

The proof of Theorem 2.3.3 uses sets that have not yet been defined. To ensure compactness of this proof, some notation and sets needed for the proof will be clarified first.

Naturally, the set V can be partitioned into subsets that have the property that no agent of a subset is connected to an agent in another subset, which is necessary to prove Theorem 2.3.3. This convention is equivalent to being a connected component. The set $\mathcal{K}(G')$, which is defined as the set containing the connected components of the graph G', is hence a partition of V. To be more precise, a set $I \subseteq V$ is called a connected component of graph G' if $G' = (I, E')$ is a connected graph, but there is no path joining any agent in I to any other agent outside of I. The set

$$\mathcal{W}(G') = \{(I, J) | (I \subseteq V) \wedge (J \subseteq V) \wedge (I \neq J) \wedge (I \in \mathcal{K}(G')) \wedge (J \in \mathcal{K}(G'))\} \tag{2.7}$$

with $G' \in S(G)$, contains combinations of non-equal, non-empty subsets that are connected components of G'.

First, assume that (I, J) is an element of $\mathcal{W}(G')$, where I and J are defined as $\{v_1, v_2 \ldots, v_{|I|}\}$ and $\{w_1, w_2 \ldots, w_{|J|}\}$ respectively. Then by definition of $\mathcal{W}(G')$, I and J must be both connected components of G'. By definition of being connected components of G', none of the agents in I is connected to an agent in J. This leads to $P(t)$ being a block-diagonal matrix, consisting of two irreducible parts, $P_I(G')$ and $P_J(G')$. Hence, since $P(t)$ is row-stochastic, these individual blocks are row-stochastic as well. Since $P_I(G')$ and $P_J(G')$ are irreducible matrices, it follows from Theorem 2.1.2 that there must exist a right- and left-Perron vector. Let $e_I(G')$ and $e_J(G')$ be the left-Perron eigenvectors of $P_I(G')$ and $P_J(G')$ respectively, implying that the Perron-vectors satisfy the property that its elements sum to one. Furthermore, $P_I(G')$ and $P_J(G')$ have spectral radii, $\rho(P_I(G'))$ and $\rho(P_J(G'))$, equal to 1, by Theorem 2.1.3. This means

$$\begin{aligned} P_I(G')^T e_I(G') &= \rho(P_I(G'))e_I(G') = e_I(G') \\ e_I(G')^T P_I(G') &= \rho(P_I(G'))e_I(G')^T = e_I(G')^T \end{aligned} \tag{2.8}$$

Next, define a vector c_{IJ} in $\mathbb{R}^n$ with elements

$$c_{IJ,k} = \begin{cases} c_{IJ,v_k} = e_{I,k} & \text{if } v_k \in I \\ c_{IJ,w_k} = -e_{J,k} & \text{if } w_k \in J \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

and a $(n-1)$-dimensional subspace of $\mathbb{R}^n$

$$\begin{aligned} H_{IJ}(G') &= \{x \in \mathbb{R}^n | c_{IJ}(G')x = 0\} \\ &= \left\{ x \in \mathbb{R}^n \middle| \sum_{v_k \in I} e_{I,k}x_k - \sum_{w_k \in J} e_{J,k}x_k = 0 \right\}. \end{aligned} \tag{2.10}$$

In Equation 2.10, the elements $e_{I,k}$ denote the k'th element of the left-Perron eigenvector $e_I(G')$. Next, a countable union of $(n-1)$-dimensional subspaces is defined;

$$X^0 = \bigcup_{t\in\mathbb{N}} \left(\bigcup_{G'\in S(G)} \left(\bigcup_{(I,J)\in W(G')} \left(\bigcup_{Q\in\mathcal{Q}_t} QH_{IJ}(G') \right) \right) \right). \tag{2.11}$$

Here, $S(G)$ and $\mathcal{W}(G')$, for any G' in $S(G)$, are finite sets. Moreover, $\mathbb{N}$ is infinite, yet countable. Bogachev shows in [3] that there exists uncountable many sets with Lebesgue measure zero. He also states that every countable set has a Lebesgue measure equal to zero. This implies that $QH_{IJ}(G')$ is a set with Lebesgue measure zero. Hence, $X^0 \subset \mathbb{R}^n$ must be a set with Lebesgue measure zero. Having a Lebesgue measure equal to zero, implies that when taking an arbitrary vector in $\mathbb{R}^n$ the chance that this vector is an element of X^0 is close to zero.

Finally, Theorem 2.3.3 can be proven. Take an arbitrary initial vector x^0 in $\mathbb{R}^n$ and let i and j be agents in V which asymptotically agree and are asymptotically disconnected. Theorem 2.3.3 will be proven by showing that the initial vector x^0 must be an element of the set X^0, with Lebesgue measure zero.

Proof. Let I and J be connected components of G^∞, with i and j, with the properties mentioned before, being elements of I and J respectively. The asymptotic disconnectedness of i and j imply that I and J are not the same sets end hence, by (2.7), that $(I,J) \in \mathcal{W}(G^\infty)$. Now, with T' as in Lemma 2.3.2, we have that for all $t \geq T'$,

$$x_I(t+1) = P_I(G(t))x_I(t) = P_I(G^\infty)x_I(t) \tag{2.12}$$

must hold. Again the Perron-Frobenius theorem implies that there exists a left Perron eigenvector, $e_I(G^\infty)$ corresponding to the simple eigenvalue of $P_I(G^\infty)$, $\rho(P_I(G^\infty)) = 1$, and that any other eigenvalues of $P_I(G^\infty)$ have modulus smaller than one. Lemma 2.3.2 implies that when two agents are interacting neighbors at time T', they will stay interacting neighbors and hence for any $t \geq T'$ we have $x_I(t) = P_I^{t-T'}(G^\infty)x_I(T')$. Now, by the Perron-Frobenius theorems, it follows that

$$\lim_{t\to\infty} P_I(G^\infty)^t = \mathbb{1}_{|I|} e_I(G^\infty)^T$$

implying

$$\lim_{t\to\infty} x_I(t) = \lim_{t\to\infty} (P_I(G^\infty))^t x_I(T') = (e_I(G^\infty)x_I(T'))\mathbb{1}_{|I|}, \tag{2.13}$$

and, hence

$$x_i^* = e_I(G^\infty)x_I(T'). \tag{2.14}$$

Repeating the prior to $J \subseteq V$ gives

$$\lim_{t\to\infty} x_J(t) = (e_J(G^\infty)x_J(T'))\mathbb{1}_{|J|}, \tag{2.15}$$

such that

$$x_j^* = e_J(G^\infty)x_J(T'). \tag{2.16}$$

Agents i and j where chosen to asymptotically agree, which implies $e_I(G^\infty)x_I(T') = e_J(G^\infty)x_J(T')$. In (2.10) the set $H_{IJ}(G')$ has been defined, from which it can be concluded that $x(T')$ must be an element of $H_{IJ}(G^\infty)$. Moreover,

$$\begin{aligned} x(T') &= P(G_{T'-1})x(T'-1) \\ &= P(G_{T'-1})P(G_{T'-2})x(T'-2) \\ &\vdots \\ &= P(G_{T'-1})P(G_{T'-2})\dots P(G_0)x^0 \end{aligned} \tag{2.17}$$

Here the notation $G_{T'-1}$ is the graph at time $T'-1$. Using (2.17) and the fact that $x(T')$ must be an element of $H_{IJ}(G^\infty)$ it turns out that

$$x^0 = P(G_0)^{-1}P(G_1)^{-1}\dots P(G_{T'-1})^{-1}x(T') \in \bigcup_{Q\in\mathcal{Q}_t} QH_{IJ}(G^\infty)$$

and hence that $x^0 \in X^0$, which is a set of Lebesgue measure zero.

In short recap, it has been proven that two agents that asymptotically agree, yet are asymptotically disconnected, have a corresponding arbitrary initial opinion vector. It followed that this initial opinion vector must be an element of a set with Lebesgue measure zero. This implies that for almost all initial opinion vectors x^0 it must hold that asymptotically agreement of two agents implies asymptotically connectivity, which proves Theorem 2.3.3. □

2.4 Normalized Laplacian matrix

The model of Morărescu and Girard, which will be shown in the next section, belongs to the category of spectral clustering, since the model uses the eigenvalues of the normalized Laplacian matrix of the graph G. In this section it will be explained how the (normalized) Laplacian matrix and their eigenvalues can be used to interpret graphs and to reformulate the main problem of clustering.

Recall the multi-agent system G. Consider the adjacency matrix $A(G)$ and the diagonal matrix $D(G)$. The elements on the diagonal of $D(G)$ are equal to the degree of the agents, hence this matrix is denoted $D(G) = \text{diag}(d_1, d_2, \dots, d_n)$. The Laplacian matrix of G,

$$L(G) = D(G) - A(G)$$

can be normalized. This normalization is not necessary, but Von Luxburg mentions several arguments which support the use of the normalized Laplacian over the unnormalized for finding clusters in [22]. Moreover, the normalized Laplacian matrix is less sensitive to the size of graph G compared with the unnormalized Laplacian according to Morărescu and Girard. The normalized Laplacian matrix of graph G, which is row-stochastic [8], is defined as

$$\mathcal{L}(G) = D^{-\frac{1}{2}}(G)L(G)D^{-\frac{1}{2}}(G)$$

The elements of $\mathcal{L}(G)$ satisfy

$$\mathcal{L}_{ij} = \begin{cases} 1, & \text{if } i = j \text{ and } d_i(G) \neq 0. \\ \frac{-1}{\sqrt{d_i(G)d_j(G)}}, & \text{if } (i,j) \in E. \\ 0, & \text{otherwise.} \end{cases} \tag{2.18}$$

The eigenvalues of the (normalized) Laplacian matrix are used to measure connectivity of a graph. This algebraic connectivity tells how well a graph is connected and is defined as the second smallest eigenvalue of the $L(G)$. This value reflects the number of agents that needs to be removed from V to make the graph disconnected. Intuitively this implies, that the algebraic connectivity and connectivity of a graph have a positive relation; if the former increases, so does the density of the latter. For the second smallest eigenvalue of the normalized Laplacian matrix the relation remains the same, even though this does not represent the algebraic connectivity directly. Furthermore, this second smallest eigenvalue of the normalized Laplacian matrix gives insight into graph G. For example,

$$\mu_2(\mathcal{L}(G)) \begin{cases} = 0 & \text{if } G \text{ has 2 distinct components.} \\ = \frac{n}{n-1} & \text{if } G \text{ is a complete graph.} \\ \in (0,1] & \text{otherwise.} \end{cases} \tag{2.19}$$

For simplicity, the eigenvalues of $\mathcal{L}(G)$ are assumed to be ordered, $\mu_i(\mathcal{L}(G)) \leq \mu_{i+1}(\mathcal{L}(G))$, with $\mu_1(\mathcal{L}(G)) = 0$ being the smallest eigenvalue. The set of eigenvalues of $\mathcal{L}(G)$ is contained in $[0, 2]$ and all eigenvalues are real. Morărescu and Girard defined a new measure of connectivity

$$\underline{\mu_2}(\mathcal{P}) = \min_{I \in \mathcal{P}, |I| \geq 2} \mu_2(\mathcal{L}(G_I))$$

which represents the smallest second eigenvalue of the least connected subset of partition $\mathcal{P} = \{I_1, I_2, \ldots, I_p\}$, with p being the number of clusters found. If this value turns out to be greater than δ, the partition is valid, leading to a reformulation of the clustering problem.

Problem 1. *Given a graph $G = (V, E)$ and real number $\delta \in (0, 1]$, find a partition $\mathcal{P}$ of V. This partition must satisfy $\underline{\mu_2}(\mathcal{P}) > \delta$.*

The larger δ is chosen, the more densely connected the clusters in the partition will be. Problem 1 has many different solutions, e.g the trivial partition $\mathcal{P} = \{\{1\}, \ldots \{n\}\}$, which depend on the initial opinion vector.

2.5 Opinion dynamics model

Finally, the model of Morărescu and Girard, which leads to a solution of Problem 1, can be formulated. The opinion dynamics model, with a decaying confidence bound and $\alpha \in (0, \frac{1}{2})$, considered, is

$$x_i(t+1) = \begin{cases} x_i(t) & \text{if } N_i(t) = \emptyset \\ x_i(t) + \frac{\alpha}{|N_i(t)|} \sum_{j \in N_i(t)} (x_j(t) - x_i(t)) & \text{if } N_i(t) \neq \emptyset \end{cases} \tag{2.20}$$

with $N_i(t)$ as in (2.3). The dynamics (2.20) satisfies the weighted average as well as the properties for p_{ij}. If $N_i(t) \neq \emptyset$, (2.20) can be rewritten by means of simple algebraic steps.

$$\begin{aligned} x_i(t+1) &= x_i(t) + \frac{\alpha}{|N_i(t)|} \sum_{j \in N_i(t)} (x_j(t) - x_i(t)) \\ &= x_i(t) + \sum_{j \in N_i} \frac{\alpha}{|N_i(t)|} x_j(t) - \sum_{j \in N_i(t)} \frac{\alpha}{|N_i(t)|} x_i(t) \\ &= x_i(t) - \alpha x_i(t) + \sum_{j \in N_i(t)} \frac{\alpha}{|N_i(t)|} x_j(t) \\ &= (1 - \alpha) x_i(t) + \sum_{j \in N_i(t)} \frac{\alpha}{|N_i(t)|} x_j(t) \end{aligned} \tag{2.21}$$

This leads to the conclusion that matrix $P(t)$ has elements

$$p_{ij}(t) = \begin{cases} 1 - \alpha & \text{if } j = i \\ \frac{\alpha}{|N_i(t)|} & \text{if } j \neq i \end{cases} \tag{2.22}$$

for given $N_i(t)$. This satisfy the condition that for any i and j in V it holds that p_{ij} is non-zero if and only if $j \in \{i\} \cup \{N_i(t)\}$. Since α is chosen to be in the open interval $(0, \frac{1}{2})$, it follows that

$$\begin{aligned} p_{ii}(t) &> \frac{1}{2} \\ p_{ij}(t) &= \frac{\alpha}{|N_i(t)|} \end{aligned}$$

Looking at every row of matrix $P(t)$, this leads to

$$\sum_{j=1,i\neq j}^{n} |p_{ij}| = \alpha \in \left(0, \frac{1}{2}\right).$$

From this it can be concluded that matrix $P(t)$ is a strictly diagonally dominant and hence $P(t)$ must be invertible. Since $P(t) = P(G(t))$ by Assumption 2, it must also hold that matrix $P(G')$ is an invertible matrix. This matrix can be written as $P(G') = I - \alpha Q(G')$, with I being the identity matrix of size $n \times n$ and elements

$$Q_{ij}(G') = \begin{cases} 1 & \text{if } i = j \text{ and } d_i(G') \neq 0. \\ \frac{-1}{d_i(G')} & \text{if } (i,j) \in E'. \\ 0 & \text{otherwise.} \end{cases} \tag{2.23}$$

In order to show that this opinion dynamics model will lead to a solution of problem 1, the following lemma is used, for which the proof can be found in Appendix A.

Lemma 2.5.1. *Let $\mathcal{P}$ be a partition of the set V and let I be an element of $\mathcal{P}$, with $|I| > 2$. Then λ is an eigenvalue of $P_I(G_{\mathcal{P}})$ if and only if $\mu = \frac{1-\lambda}{\alpha}$ is an eigenvalue of $\mathcal{L}(G_I)$.*

Finally the problem of finding clusters can be solved, using this opinion dynamics model. The solution is formulated in the following corollary.

Corollary 1. *(Solution of problem 1)*
Let $\rho = 1 - \alpha\delta$, then for almost all initial vectors x^0 and under Assumption 3 the partition found, using (2.20)*, is a correct solution of Problem 1.*

3. Simulations

In this chapter, simulations conducted with the use of MATLAB will be discussed. To verify the model in Chapter 2, it will be tested on a problem with a known solution. Specifically, an example from the article of Morărescu and Girard will be reproduced. Next, two simple test cases will be considered to provide more insight in the clustering method.

3.1 Zachary Karate Club

Wayne Zachary illustrated in [24] a social network within a karate club. This network consists of 34 members, who interacted with one another outside of club meetings and classes depicted in Figure 3.1.

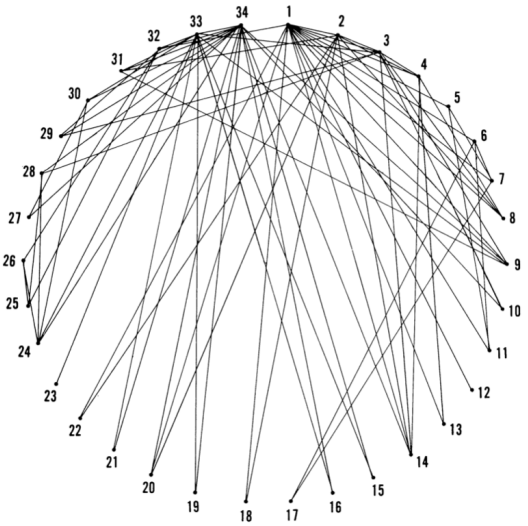
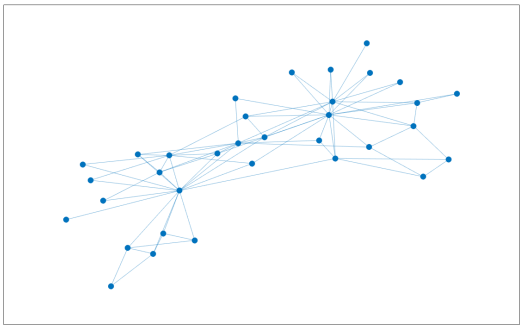
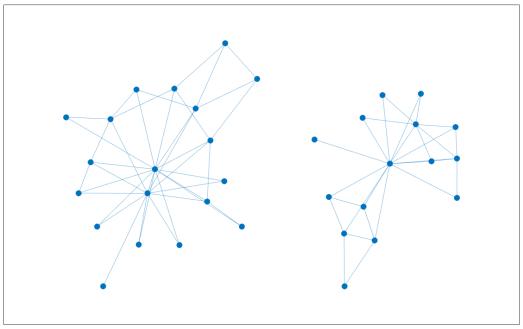


Figure 3.1: Network of the karate club [24]

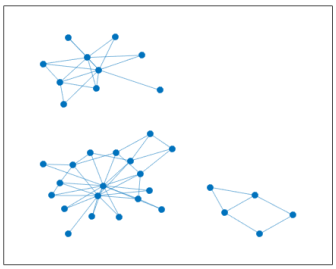
In this figure, the dots, the agents, represent the 34 members of the club. An edge connecting two agents represents the fact that these two members interacted with each other outside of the club. The adjacency matrix, representing this network, is copied from [24] and is written out in Listing C.1 in the Appendix. Variation of the parameters ρ, α and δ resulted in different clusters and a different number of clusters. When simulating this example, a random initial opinion vector is chosen with elements in the domain $[0, 1]$. The results show that not the same partitions were formed for every initial opinion vector. The most frequently found clusters for this example can be found in Figure 3.2. The results correspond to those of Morărescu and Girard.



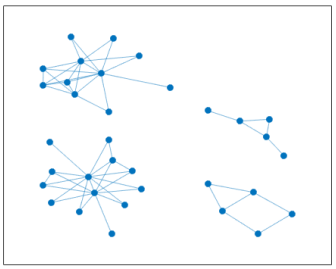
(a) $R = 10$, $\alpha = 0.1$, $\delta = 0.1$



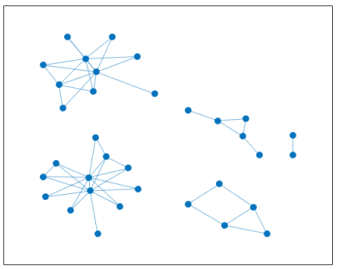
(b) $R = 10$, $\alpha = 0.1$, $\delta = 0.2$



(c) $R = 10$, $\alpha = 0.1$, $\delta = 0.3$



(d) $R = 10$, $\alpha = 0.1$, $\delta = 0.4$



(e) $R = 10$, $\alpha = 0.2$, $\delta = 0.4$

Figure 3.2: The clusters most frequently found for the Karate Club example with $R = 10$ and for different values of α and δ.

Recall that every agent has an area of confidence in which the opinion of an interacting neighbor should be. Since ρ is strictly smaller than 1, this area of confidence approaches zero, when time increases. For numerical reasons it is not possible to have infinite number of time-steps. Hence, a decision needs to be made on the value of ϵ, which will serve as restriction on the number of time-steps. Whenever ρ^t is bigger than this ϵ, the opinions will no longer updated and the process of removing edges is finished. Letting $\epsilon = 10^{-15}$ gave the desired results, when comparing with the results of Morărescu and Girard. This restriction will be used for the rest of the simulations conducted for this thesis.

3.2 Two complete graphs

Recall the solution to the problem of finding clusters. Corollary 1 states that for almost any initial opinion vector x^0, the partition found is a correct solution to Problem 1. However, the set of edges, $E(t)$, does depend on the initial opinion vector. This triggered curiosity and lead to the decision to look at what will happen if the initial opinions are decided in advance. What does this mean for the final clusters; Are they decided in advance, is it possible to break them apart, or could we even end up with one big cluster. To illustrate this problem, a network consisting of 8 agents is assumed, which can be divided in two complete subgraphs as below.

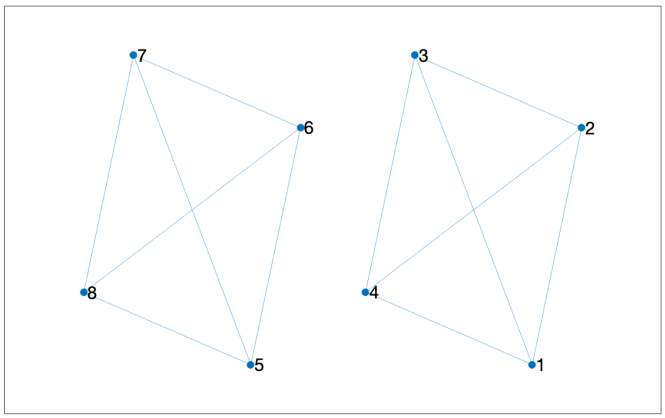


Figure 3.3: Network consisting of two complete subgraphs.

As mentioned, the initial opinion vector is no longer a random vector. Let the initial opinion vector be such that the agents in both subgraphs have the same opinion, as in (3.2). Again the program from Listing C.1 can be used, only with a different adjacency matrix, as in (3.1).

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \tag{3.1}$$

$$x^0 = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.55 \\ 0.55 \\ 0.55 \\ 0.55 \end{bmatrix} \tag{3.2}$$

Obviously, since the graph consists of two complete subgraphs disconnected from one another, the agents only interact with the agents within their subgraph and hence their opinion should not change. Testing this in MATLAB, gives indeed the predicted outcome. The final opinion vector is equal to the initial one and has not changed in between.

The next step is to add a link in this graph, to connect the two complete subgraphs, giving the graph from Figure 3.4, with corresponding adjacency matrix as in (3.3).

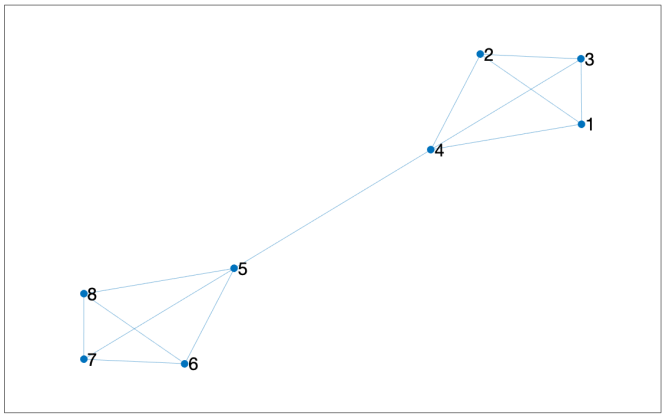


Figure 3.4: Network of Figure 3.3 with an added link between agent 4 and 5.

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \tag{3.3}$$

The idea is that adding this link will pull the two subgraphs together and that the opinions of the agents will converge to some opinion between 0.5 and 0.55. Intuitively, this one extra link is not likely to weigh up against the density of the agreement of opinions in the original two subgraphs. Hence this link will fail in most of the tested cases, as is confirmed by simulation results shown in Table 3.1.

$\boldsymbol{\alpha}$	$\boldsymbol{\delta}$	$\lvert\boldsymbol{\mathcal{P}}\rvert$
0.1	0.1	1
0.1	0.2	2
0.1	0.3	2
0.1	0.4	2
0.2	0.1	1
0.2	0.2	2
0.2	0.3	2
0.2	0.4	2

Table 3.1: Amount of clusters found for different values of α and δ.

Here $\mathcal{P}$ is equal to the partition of the set of agents and hence $|\mathcal{P}|$ is the number of clusters found. This led to the idea of adding a weight to this extra link between agent 4 and 5. This added weight can be seen as adding a second, third or even a fourth link between those two agents, depending on the weight of this link.

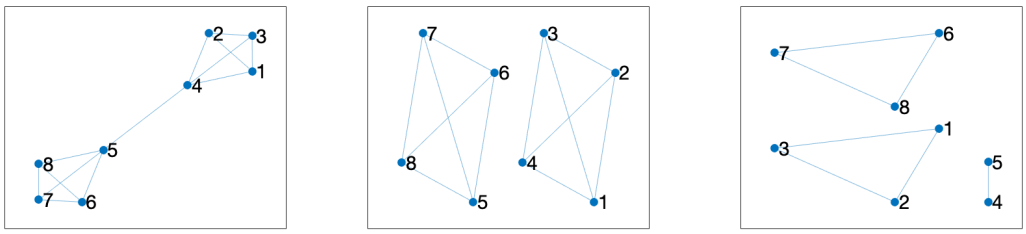
This weight, denoted as w_{45}, changes the adjacency matrix (3.3). Now, the entries a_{45} and a_{54} will no longer be equal to 1, but equal to w_{45}. If the link between the two subgraphs is stronger, then this link will probably hold longer when increasing time. Moreover, the graph could even stay one big cluster, even with increasing values of α and δ. Furthermore, this weighted link changes the weighted average. Previously, α was divided by the number of neighbors of an agent, $N_i(t)$. However, if the weight of a link is bigger than one, this link has respectively more influence on the weighted average than the other links. The model of Morărescu and Girard changes to

$$x_i(t+1) = \begin{cases} x_i(t) & \text{if } N_i(t) = \emptyset \\ x_i(t) + \frac{\alpha}{\sum_{j \in N_i(t)} w_{ij}} \sum_{j \in N_i(t)} w_{ij}(x_j(t) - x_i(t)) & \text{if } N_i(t) \neq \emptyset \end{cases} \tag{3.4}$$

where $\sum_{j \in N_i(t)} w_{ij}$ is equal to the sum of the weights of all neighbors of agent i at time $t \in \mathbb{N}$. When $N_i(t) \neq \emptyset$, the elements of matrix $P(t)$ are equal to

$$p_{ij}(t) = \begin{cases} 1 - \alpha & \text{if } j = i \\ \frac{w_{ij}\alpha}{\sum_{j \in N_i(t)} w_{ij}} & \text{if } j \neq i \end{cases} \tag{3.5}$$

This result has been found using a similar derivation as in Section 2.5. In this case the matrix $P(t)$ is still strictly diagonally dominant and hence invertible. Furthermore, the matrix $P(t)$ is still row-stochastic. While experimenting in MATLAB, a third, intuitively not expected, possibility for the final state appeared. The three possibilities that appeared can be found in the figure below.



(a) Initial situation remains (b) Link disappears (c) Third unexpected option

Figure 3.5: Possibilities for the final state after adding a weighted link between agent 4 and 5

Thinking more about the dynamics when the weight of the link, between agents 4 and 5, is increased, this third option does seem a likely possibility. If the link between agent 4 and 5 becomes really strong, it could result in a disproportionate convergence of opinions. The opinions of agent 4 and 5 grow fast towards one another and at the same time distance themselves from agents 1, 2, 3 and 6, 7, 8 respectively. This disproportionate convergence then results in the third option, with them forming a separate cluster.

Since α not only influences the parameter ρ, but also the matrix $P(t)$, it will be checked what partitions are formed for constant values α, while varying the values of δ and w_{45}. In Table 3.2 and Table 3.3 the number of clusters found for different weights and different values of δ have been organized for α is equal to 0.1 and 0.2 respectively. The program used for these results can be found in Listing C.2 in the Appendix.

w_{45}	$\delta = 0.1$	$\delta = 0.15$	$\delta = 0.2$	$\delta = 0.25$	$\delta = 0.3$	$\delta = 0.35$	$\delta = 0.4$	$\delta = 0.45$	$\delta = 0.5$
1	1	2	2	2	2	2	2	2	2
1.5	1	1	2	2	2	2	2	2	2
2	1	1	1	2	2	2	2	2	2
2.5	1	1	1	2	2	2	2	2	2
3	1	1	1	1	2	2	2	2	2
3.5	1	1	1	1	2	2	2	2	2
4	1	1	1	1	3	3	3	3	3
4.5	1	1	1	1	3	3	3	3	3
5	1	1	1	1	3	3	3	3	3
5.5	1	1	1	1	1	3	3	3	3
6	1	1	1	1	1	3	3	3	3

Table 3.2: Number of clusters found for different values of δ and w_{45} with $\alpha = 0.1$ and $R = 10$.

w_{45}	$\delta = 0.1$	$\delta = 0.15$	$\delta = 0.2$	$\delta = 0.25$	$\delta = 0.3$	$\delta = 0.35$	$\delta = 0.4$	$\delta = 0.45$	$\delta = 0.5$
1	1	2	2	2	2	2	2	2	2
1.5	1	1	2	2	2	2	2	2	2
2	1	1	1	2	2	2	2	2	2
2.5	1	1	1	2	2	2	2	2	2
3	1	1	1	2	2	2	2	2	2
3.5	1	1	1	1	2	2	2	2	2
4	1	1	1	1	3	3	3	3	3
4.5	1	1	1	1	3	3	3	3	3
5	1	1	1	1	3	3	3	3	3
5.5	1	1	1	1	1	3	3	3	3
6	1	1	1	1	1	3	3	3	3

Table 3.3: Number of clusters found for different values of δ and w_{45} with $\alpha = 0.2$ and $R = 10$.

Looking at these tables, The first thing that stands out is that the tables are exactly the same. This will be further addressed shortly, since this result was not expected. The results from the tables can be explained with some intuition. It seems that $\delta = 0.15$ can be seen as a threshold for the link breaking between agent 4 and 5. Looking at the influence of δ on the process, this makes sense. If δ increases, $\rho = 1 - \alpha\delta$ must decrease and the area of confidence of every agent gets a lot smaller in shorter time. This means, that agent 4 and 5 must converge to one another much faster in order to stay connected to each other. On the other hand, the $w_{45} = 4$ looks like a threshold to end up with 3 clusters. The link between agent 4 and 5 only influences agent 4 and 5 directly, all of the other agents indirectly. The higher the weight of the link between agent 4 and 5, the more disproportionate the convergence will be. In other words, the higher the value of w_{45}, the faster agent 4 and 5 approach each other and the faster the difference between agent 4, 5 and their initial neighbors will grow respectively. Since it seems hold that the tables for $\alpha = 0.1$ and $\alpha = 0.2$ are exactly the same, the same reasoning for Table 3.3 can be applied.

The observation that for $\alpha = 0.1$ and $\alpha = 0.2$ the tables completely overlap, was quite unexpected. By looking in smaller steps in δ and w_{45}, one can distinguish whether the outcome of these tables are independent of α or weakly dependent on α. The tables in Chapter B show that different number of clusters did occur for some combinations of δ and w_{45}. This implies that α has some influence forming a partition. However, this influence is only minimal since most of the results still overlapped for $\alpha = 0.1$ and $\alpha = 0.2$. An explanation for this result could a choice made earlier. It might be possible that applying the method for some time-steps extra, some of the values in the tables will change. Since a precise restriction has not been established,

the restriction on the time-steps might be the cause for the overlapping tables. However, since α influences the area of confidence of every agent, this influences how fast a link will break. For various results from Table 3.2 and Table 3.3, the number of time-steps have been found for which all the links remain.

$\boldsymbol{\alpha}$	$\boldsymbol{\delta}$	$\boldsymbol{w_{45}}$	$\lvert\boldsymbol{\mathcal{P}}\rvert$	Time-steps
0.1	0.2	1	2	638
0.2	0.2	1	2	313
0.1	0.3	3	2	623
0.2	0.3	3	2	302
0.1	0.4	6	3	404
0.2	0.4	6	3	194

Table 3.4: Number of time-steps for which no link breaks for different values of α and δ.

3.2.1 Eigenvalues of matrix $P(t)$

Another way of looking at the previous results, is looking at eigenvalues and eigenvectors of the matrix $P(t)$. Matrix $P(t)$, also referred to as the update-matrix, only changes when a link between agents disappears. This means that the eigenvalues and eigenvectors of matrix $P(t)$ are almost constant over time. Looking at the eigenvalues and -vectors of matrix $P(0)$ corresponding to the initial situation, when there is no link between agent 4 and 5, only two distinct eigenvalues appear. This matrix can be found with only knowing the adjacency matrix and value of α, since $P(t)$ is independent of δ. In the initial situation the adjacency matrix is a block diagonal matrix, since there is no link between agent 4 and 5. A block diagonal matrix has the property that its set of eigenvalues is the union of the sets of eigenvalues of the block matrices on the diagonal. In this specific case the set $N_i(t)$ will not be empty for any $i \in \{1, 2, \ldots, 8\}$, hence matrix $P(t)$ has elements as in (3.5). The matrix $P(0)$ is of the form

$$P(0) = \begin{bmatrix} 1-\alpha & \frac{\alpha}{|N_1(1)|} & \frac{\alpha}{|N_1(1)|} & \frac{\alpha}{|N_1(1)|} & 0 & 0 & 0 & 0 \\ \frac{\alpha}{|N_2(1)|} & 1-\alpha & \frac{\alpha}{|N_2(1)|} & \frac{\alpha}{|N_2(1)|} & 0 & 0 & 0 & 0 \\ \frac{\alpha}{|N_3(1)|} & \frac{\alpha}{|N_3(1)|} & 1-\alpha & \frac{\alpha}{|N_3(1)|} & 0 & 0 & 0 & 0 \\ \frac{\alpha}{|N_4(1)|} & \frac{\alpha}{|N_4(1)|} & \frac{\alpha}{|N_4(1)|} & 1-\alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1-\alpha & \frac{\alpha}{|N_5(1)|} & \frac{\alpha}{|N_5(1)|} & \frac{\alpha}{|N_5(1)|} \\ 0 & 0 & 0 & 0 & \frac{\alpha}{|N_6(1)|} & 1-\alpha & \frac{\alpha}{|N_6(1)|} & \frac{\alpha}{|N_6(1)|} \\ 0 & 0 & 0 & 0 & \frac{\alpha}{|N_7(1)|} & \frac{\alpha}{|N_7(1)|} & 1-\alpha & \frac{\alpha}{|N_7(1)|} \\ 0 & 0 & 0 & 0 & \frac{\alpha}{|N_8(1)|} & \frac{\alpha}{|N_8(1)|} & \frac{\alpha}{|N_8(1)|} & 1-\alpha \end{bmatrix} \tag{3.6}$$

Indeed, this is a block diagonal matrix and hence the eigenvalues of the two separate blocks will determine the eigenvalues of $P(0)$. We know that the matrix $P(t)$ is a row stochastic matrix at any given time. Since we have a block diagonal matrix, the blocks on the diagonal must be row-stochastic as well. Since these blocks are irreducible and row-stochastic, it follows by the Perron-Frobenius theorem that for each block $\lambda = 1$ is a simple eigenvalue of maximum modulus. Moreover, since every agent has the same number of neighbors, the two blocks are the same. Thus the eigenvalue of $P(0)$ equal to 1 has algebraic multiplicity equal to 2. Actually, the algebraic multiplicity of $\lambda = 1$ of $P(t)$ corresponds to the number of clusters at any given time. However, to go back to the situation with a link between the two complete graphs, the same results holds. If this link breaks down or agent 4 and 5 form a separate cluster, the irreducible adjacency matrix turns into a reducible one in block diagonal form. Recall the definition of irreducibility, which also means that any agent can be reached, when starting form an arbitrary agent in the network. This implies that once one or two links disappear, every block on the diagonal of matrix $P(t)$

has eigenvalue equal to 1 with algebraic multiplicity 1. Hence, the algebraic multiplicity of $P(t)$ corresponds to the number of clusters at that time.

To summarize, the matrix $P(t)$ gives conformation of the number of clusters we see at that time. However, looking at the first matrix that is produced, $P(0)$, might gives us an idea of what will happen during the process. For this reason the choice has been made to look closer at the first update-matrix. The matrix

$$P(0) = \begin{bmatrix} 1-\alpha & \frac{\alpha}{\sum_{j=1}^n a_{1j}} & \frac{\alpha}{\sum_{j=1}^n a_{1j}} & \frac{\alpha}{\sum_{j=1}^n a_{1j}} & 0 & 0 & 0 & 0 \\ \frac{\alpha}{\sum_{j=1}^n a_{2j}} & 1-\alpha & \frac{\alpha}{\sum_{j=1}^n a_{2j}} & \frac{\alpha}{\sum_{j=1}^n a_{2j}} & 0 & 0 & 0 & 0 \\ \frac{\alpha}{\sum_{j=1}^n a_{3j}} & \frac{\alpha}{\sum_{j=1}^n a_{3j}} & 1-\alpha & \frac{\alpha}{\sum_{j=1}^n a_{3j}} & 0 & 0 & 0 & 0 \\ \frac{\alpha}{\sum_{j=1}^n a_{4j}} & \frac{\alpha}{\sum_{j=1}^n a_{4j}} & \frac{\alpha}{\sum_{j=1}^n a_{4j}} & 1-\alpha & \frac{\alpha \cdot w_{45}}{\sum_{j=1}^n a_{4j}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\alpha \cdot w_{45}}{\sum_{j=1}^n a_{5j}} & 1-\alpha & \frac{\alpha}{\sum_{j=1}^n a_{5j}} & \frac{\alpha}{\sum_{j=1}^n a_{5j}} & \frac{\alpha}{\sum_{j=1}^n a_{5j}} \\ 0 & 0 & 0 & 0 & \frac{\alpha}{\sum_{j=1}^n a_{6j}} & 1-\alpha & \frac{\alpha}{\sum_{j=1}^n a_{6j}} & \frac{\alpha}{\sum_{j=1}^n a_{6j}} \\ 0 & 0 & 0 & 0 & \frac{\alpha}{\sum_{j=1}^n a_{7j}} & \frac{\alpha}{\sum_{j=1}^n a_{7j}} & 1-\alpha & \frac{\alpha}{\sum_{j=1}^n a_{7j}} \\ 0 & 0 & 0 & 0 & \frac{\alpha}{\sum_{j=1}^n a_{8j}} & \frac{\alpha}{\sum_{j=1}^n a_{8j}} & \frac{\alpha}{\sum_{j=1}^n a_{8j}} & 1-\alpha \end{bmatrix} \tag{3.7}$$

with a_{ij} elements of matrix A, has eigenvalues and eigenvectors obviously depending on α and w_{45}. It turns out that for any value of α and w_{45}, the eigenvalues of $P(0)$ follow a certain pattern, in which the fourth up to and including the seventh eigenvalue are equal to one another, if the eigenvalues are ordered in descending order. Assume that we have eigenvalues in descending order, then the eigenvalues are of the following order;

$$1 = \lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4 = \lambda_5 = \lambda_6 = \lambda_7 \geq \lambda_8.$$

Hence, for any α there are five different eigenvalues; $\lambda_1 = 1$ is always an eigenvalue due to the Perron-Frobenius theorem, one eigenvalue with algebraic multiplicity 4 and three other distinct eigenvalues. An increase of α, leads to a decrease of the eigenvalues that are not equal to 1. Looking at the eigenvectors corresponding to the eigenvalues, there is a pattern as well. We will only look at the eigenvectors corresponding to the eigenvalues with algebraic multiplicity 1. The eigenvector corresponding to the simple eigenvalue equal to 1 is the constant eigenvector. The numerical eigenvector actually had a maximum difference between elements of order $\mathcal{O}(10^{-15})$. However, this must be due to numerical imprecision, since every P-matrix is row-stochastic and hence $P(0)\mathbb{1} = \mathbb{1}$ must be true. This eigenvector represents the average behaviour of our system. The other interesting eigenvectors are the eigenvectors corresponding to λ_2, λ_3 and λ_8 respectively are of the form

$$u_2 = \begin{bmatrix} a \\ a \\ a \\ b \\ -b \\ -a \\ -a \\ -a \end{bmatrix} \tag{3.8}$$

$$u_3 = \begin{bmatrix} -c \\ -c \\ -c \\ d \\ d \\ -c \\ -c \\ -c \end{bmatrix} \tag{3.9}$$

$$u_8 = \begin{bmatrix} -e \\ -e \\ -e \\ f \\ -f \\ e \\ e \\ e \end{bmatrix} \tag{3.10}$$

with $|a| > |b|$, $|d| > |c|$ and $|f| > |e|$ respectively for a small weight. These eigenvectors can be verified analytically by means of simple steps. The goal of this subsection is to figure out what these eigenvectors tell us about the dynamics and how they can be interpreted.

From the previous subsection it could be concluded that the value of α only had very little impact on the final partition, but it did influence when a link gets lost. If α increases, the

eigenvalues of $P(0)$, except for the eigenvalue equal to one, all decrease in absolute value. Unfortunately, the eigenvectors did not change with an increase of α, so the eigenvalues and -vectors of $P(0)$ can not be used to determine how fast a link will break. The next step was to look at what happens to the eigenvalues and -vector when the value of w_{45} increases.

Recall the situation of two complete subgraphs, both consisting of four agents, that are connected to one another via an extra link. This extra link can have a weight, w_{45}, which means that this weighted link increases the impact that the agents, connected by this link, have on one another. Increasing w_{45} hence increases the attraction between agent 4 and 5 and also the attraction between the two subgraphs. Keeping α constant, it turns out that increasing w_{45} decreases λ_2 and λ_8, but increases λ_3. Based on observations of the change of eigenvectors in combination with the knowledge on the influence of w_{45} on the process, we will explain the meaning of these eigenvectors without convincing mathematical proof.

Looking at the second eigenvector, the absolute value of a increases whereas the absolute value of b decreases when w_{45} increases. For a low value of w_{45}, it holds that the absolute value of a is strictly bigger than the absolute value of b. This implies that with w_{45} increasing, the difference between a and b in absolute value increases as well. Due to the form of this eigenvector as in (3.8), it is implied that the difference between the fourth and fifth element decreases, while the difference between element 3 and 4 and between 5 and 6 increases. Based on these observations, this second eigenvector represents how agent 4 and 5 pull away from the rest of the network.

The second interesting eigenvector is the third eigenvector. For this eigenvector it holds that for a low value of w_{45}, the absolute value of c is strictly smaller than the absolute value of d. If the weight of the link between agent 4 and 5 increases, $|c|$ increases, while $|d|$ decreases. These developments of c and d with w_{45} increasing, implies that there is no difference between the fourth and fifth element of u_3. However, the difference between element 3 and 4, and hence also between element 5 and 6, decreases a little when w_{45} increases. From these observations, one could say that (3.9) represents the need of the network to stay one cluster. The fact that the difference between the inner two elements with the elements on the outside decrease, implies that agent 4 and 5 pull the original two subgraphs harder towards one another with increasing value of w_{45}.

The last eigenvector, u_8, has the property that with a small weight between agent 4 and 5, $|f| > |e|$ holds. When w_{45} increases, $|f|$ increases and $|e|$ decreases. Hence the difference between element 4 and 5 increases. On the other hand, the difference between element 3 and 4, which is approximately equal to the difference between element 5 and 6, stays almost constant, while w_{45} increases. These observations can link this eigenvector to the two original subgraphs we started with, pulling away from one another.

To conclude this subsection, the eigenvalues of the first P-matrix can be linked to properties of the dynamics. However, the hope was to gain insight on how these eigenvalues and eigenvectors could be used to predict which links are the weakest and would disappear first. In this specific example it was clear in advance which links could disappear and which links would never disappear. For this example it is possible to see which links are the weaker links that could disappear. All of the eigenvectors corresponding to the eigenvalues of multiplicity 1 not equal to one had the form that the inner two elements differ from the ones on the outside. Combining this observation with the dynamical meaning of these eigenvectors, one could conclude that for this specific example the eigenvectors can be used to find the weaker links. However, the representations of these eigenvectors are based on observations. Moreover, it turned out that the final partition formed depends heavily on the value of δ, while δ has no influence on the eigenvalues or -vectors of $P(0)$. Hence, it is not possible to state that these eigenvectors can predict which exact links will disappear.

3.3 Network of a line

For the second test to gain more intuition how this algorithm works, a different network is considered. The network considered here, consists of 5 agents and 4 edges as Figure 3.6.

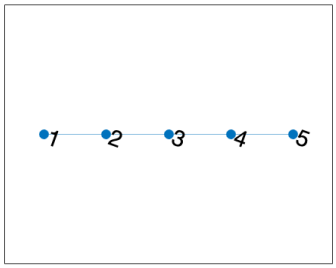


Figure 3.6: Network with 5 agents and 4 edges

The adjacency matrix corresponding to this network is the symmetrical tridiagonal matrix with zero entries on the diagonal and 1's on the off-diagonals;

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.11}$$

As we have seen before, the matrix $P(t)$ is almost constant over time and only changes if a link between two agents disappears. Also for this test case we look at matrix $P(0)$, which is given as

$$P(0) = \begin{bmatrix} 1-\alpha & \alpha & 0 & 0 & 0 \\ \frac{1}{2}\alpha & 1-\alpha & \frac{1}{2}\alpha & 0 & 0 \\ 0 & \frac{1}{2}\alpha & 1-\alpha & \frac{1}{2}\alpha & 0 \\ 0 & 0 & \frac{1}{2}\alpha & 1-\alpha & \frac{1}{2}\alpha \\ 0 & 0 & 0 & \alpha & 1-\alpha \end{bmatrix}. \tag{3.12}$$

Using an eigenvalue decomposition of this matrix, this matrix can be rewritten in the following form

$$P(0) = U\Lambda U^{-1}, \tag{3.13}$$

with

$$U = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & u_5 \end{bmatrix}, \qquad \Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 & 0 \\ 0 & 0 & 0 & \lambda_4 & 0 \\ 0 & 0 & 0 & 0 & \lambda_5 \end{bmatrix}.$$

Here λ_i is the i'th eigenvalue of $P(0)$ and u_i its corresponding eigenvector. Now, we assume that the matrix $P(t)$ will not change over time at all. Using the weighted average and the eigenvalue decomposition, the development of the opinion vector $x(t)$ changes to

$$x(t+1) = Px(t) = P^2x(t-1) = \ldots = P^{t+1}x^0 = U\Lambda^{t+1}U^{-1}x^0. \tag{3.14}$$

The initial condition will be chosen in such a way that one particular eigenvalue in combination with its corresponding eigenvector characterises the development of the opinions and hence the dynamics considered here. This eigenvalue is the eigenvalue that is closest to 1 in absolute value. Assume we have ordered eigenvalues of the matrix $P(0)$;

$$1 = \lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4 \geq \lambda_5.$$

which makes the second eigenvalue the eigenvalue that is closest to 1. Now x^0 needs to be chosen such that $U^{-1}x^0 = \mathrm{e}_2$, with e_2 being the second standard basis vector, leading for any $t \in \mathbb{N}$ to

$$x(t+1) = U\Lambda^{t+1}U^{-1}x^0 = U\Lambda^{t+1}e_2 = u_2\lambda_2^{t+1} \tag{3.15}$$

This second eigenvalue is also referred to as the slowest eigenvalue, since the $\lim_{t\to\infty} \lambda_2^t$ approaches zero the slowest, compared to the other eigenvalues. The initial opinion vector is now equal to the eigenvector corresponding to this slowest eigenvalue and the opinions will not change direction.

The eigenvector corresponding to this slowest eigenvalue has the form

$$u_2 = \begin{bmatrix} a \\ b \\ 0 \\ -b \\ -a \end{bmatrix}.$$

To look at what happens in this system, the program from Listing C.3 in the Appendix is used. Even though the eigenvectors are independent of α, α does influence the matrix P, its eigenvalues and hence the dynamics. To simplify the problem, we only want positive eigenvalues that are not to close to zero, hence for the first test the choice has been made to set $\alpha = 0.1$. If the the chosen eigenvalue gets closer to zero, then the opinion vector reaches the zero vector faster, however we want to check what happens to the links and hence delaying that all of the opinions reach zero is preferable. To keep the parameters in all of the simulations as constant as possible, the parameter R is set equal to 10. Since the clusters formed depend on the opinions and also on the initial opinion vector, looking at this initial opinion vector gives an indication of what will happen. In the initial opinion vector, u_2, the difference in opinion between agent 3 and its neighbors is initially the biggest. This gives the impression that the links connecting agent 3 to its neighbors are the weakest links and are the most likely to disappear. Performing some tests in MATLAB, this was indeed the case.

Even though the value of α is set to be fixed at first, the parameter $\rho = 1 - \alpha\delta$ is still a free variable. For different values of ρ it has been checked whether or not any links get lost. For $\alpha = 0.1$ this slowest eigenvalue is equal to $\lambda_2 \approx 0.97071$. After some trial and error tries, the domain for ρ, containing the critical value of ρ for which the links connecting agent 3 to its neighbors will be lost is $[0.967, 0.968]$. If these links have disappeared, there will be three instead of only one cluster as final situation.

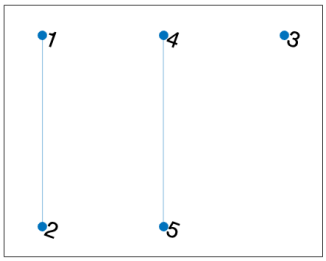


Figure 3.7: The three clusters after the disappearance of the links connecting agent 3 to its neighbors.

In the figure below it has been checked for 100 different values of ρ inside this domain whether or not the links eventually disappeared. In this first test α is set to be equal to 0.1.

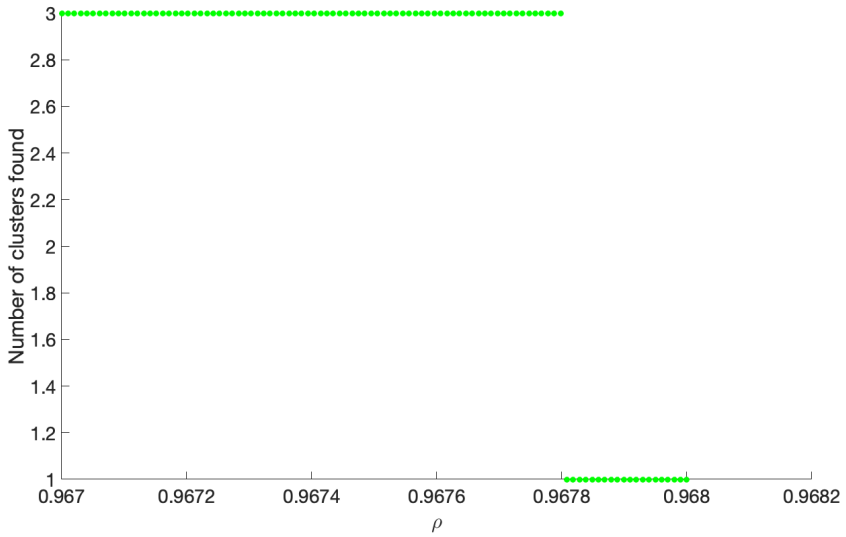


Figure 3.8: Number of clusters formed for different values of ρ with $\alpha = 0.1$

Even though it is nice to know that the critical value for ρ in this case is approximately equal to 0.96782, this does not tell us much about its relation to the considered eigenvalue. Next, since only α influences the value of the eigenvalues, for different values of α, the first ρ for which no link gets lost has been found. To plot the following results the program in Listing C.4 from the Appendix has been used. The correlation between the parameters α and λ_2 is a negative linear correlation, as can be seen in the following figure. The decrease of the slowest eigenvalue λ_2 is proportional to α.

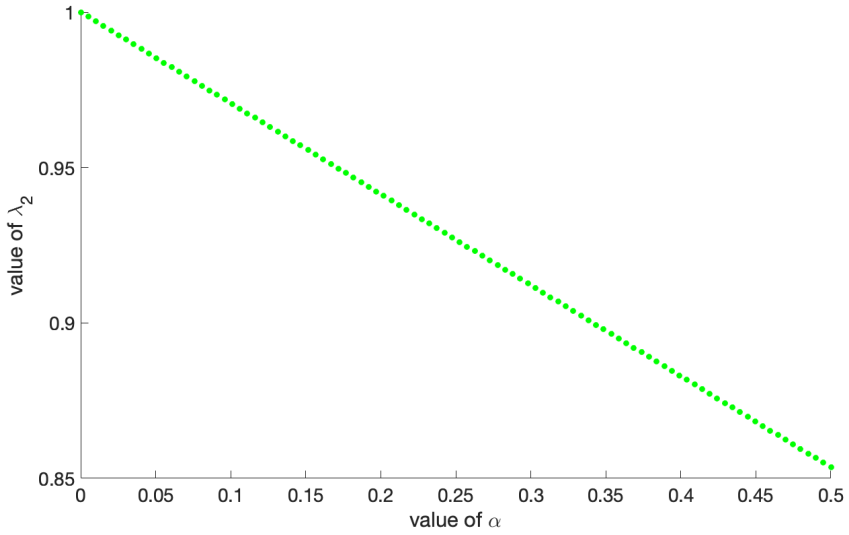
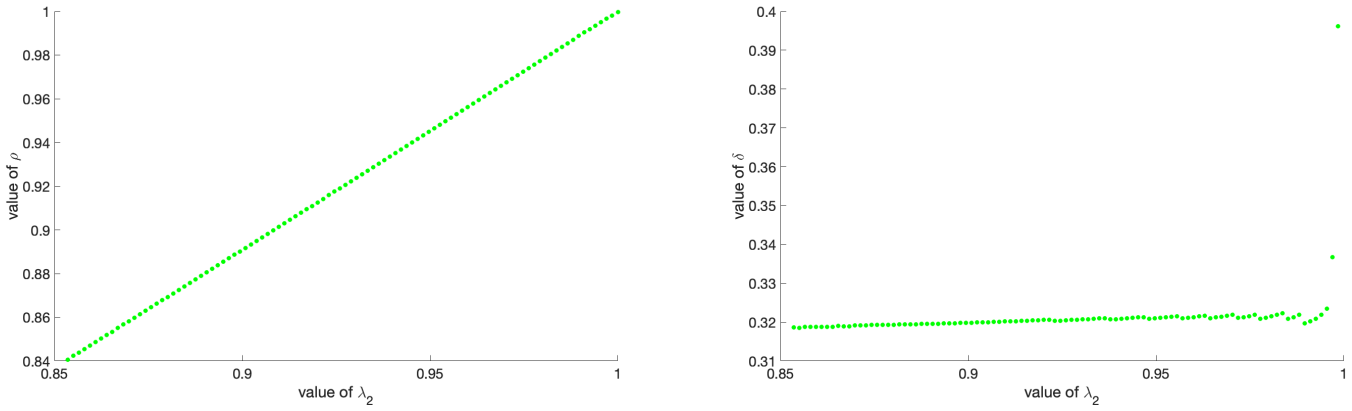


Figure 3.9: Relation between value of α and the slowest eigenvalue λ_2

The next step is to find a critical value of ρ for different values of the slowest eigenvalue. This critical value of ρ is defined as the the smallest value of ρ for which no link will be lost. In Figure 3.10 the slowest eigenvalue has been plotted against the critical value of ρ and the corresponding value of δ.



(a) Relation between λ_2 and critical value of ρ

(b) Relation between λ_2 and critical value of δ

Figure 3.10: Relation between λ_2 and critical values of ρ and δ

From Figure 3.10(a) it can obviously be concluded that there is a linear relation between the slowest eigenvalue and the critical value for ρ which is the first value of ρ for which no links disappear. Since there is also a linear relation between α and λ_2, there must be a linear relation between the value of α and the critical value of ρ as well. Due to the definition of $\rho = 1 - \alpha\delta$, it follows that with increasing value of λ_2, or decreasing value of α, in combination with all of the linear relations, the value of the critical δ, corresponding to the critical ρ, should be approximately constant. This is indeed the case as can be seen in Figure 3.10(b). However, if λ_2 approaches one, which is equivalent to α approaching zero, the value of the critical value of δ starts to fluctuate significantly. When α approaches zero, there might not be enough points in the domain of ρ in which the critical value of ρ is found. This could explain the more chaotic behaviour of the plot when λ_2 approaches 1.

This result is not an obvious result, since in this method, the area of confidence of the agents decreases exponentially.

4. Illustrative Example

As a final visualization of this methodology, a data set found on the internet will be used to perform this method on. This data set is found on *GitHub* on [1] used for recommender systems. Recommender systems are widely used in society. These systems are based on preferences of real people who rate and write comments on any type of product online. Websites like *Netflix* use these systems to advertise the right movies and series to the right people. Ratings specify one's preferences and if one liked a specific series in the category 'English costume drama', this user probably also like other series or movies from this category. Some feedback systems are less explicit; websites like *Amazon.com* also use the act of buying an item as sign of approval of that item [2]. The advantage of collecting data this way is that it does not require any work of the customer. Whether or not it is ethically correct to collect data of users without them knowing it is a totally different question, which is outside the scope of this thesis.

With this feedback system, the recommendation analysis is based on interactions between items and users, since the interests of the past function as proper indication of choices in the present and the future. Choices of users can be combined as a basis of the recommender system; buying an item in combination of the rating and comments of this item is another way to build a recommender system. However, there are many more characteristics that can be taken into account. For example users can leave comments to read for perspective buyers. These comments can be liked by other users, leaving a sign to new buyers that those comments were useful in their choice of buying. In this way recommender system can filter the ratings or comments. If a user gets many likes on his or her comment, their rating and view of an item is probably trustworthy and hence the website can recommend the right items to perspective buyers to increase their sales. The methodology of Morărescu and Girard is not able to cluster on many characteristics. Nevertheless, in the way of defining the link between agents, it will be possible to cluster a big data set of items on some characteristics. In order to show this, a big data set will be analyzed. This set contains items, which can be bought on *Amazon.com*, such as songs or albums. Every item in this set has a certain ASIN-number and will function as agents in the network. This ASIN number is a unique number assigned to an item that can be bought. Furthermore, in this data set many characteristics are taken into account. For example, categories of genres, whether or not the item belongs to the category of baby products, is it an item connected to Easter, is it a piece of clothing and many other characteristics *Amazon.com* uses for their items. Obviously, since all of the items considered are songs or albums, the only characteristics these items have are whether or not they belong to a category of music, if it is an album and whether or not it is vinyl or a CD. Since we already know that the algorithm of Morărescu and Girard works, the only things necessary to cluster this set is to define when two items are linked and to make the adjacency matrix after the set has been imported into MATLAB.

The decision has been made to look at the number of overlapping characteristics between the items. The weight of the link between two agents is equal to the number of characteristics these items have in common. The way adjacency matrix has been formed is defined in the MATLAB program stated in Listing C.5. Testing the clusters for only 100 out of 6931 items with $\alpha = 0.1$ and $\delta = 0.1$ a very chaotic output appeared.

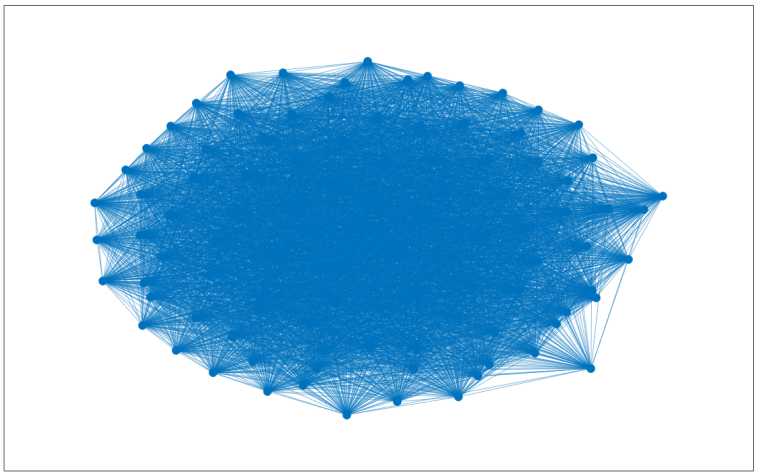
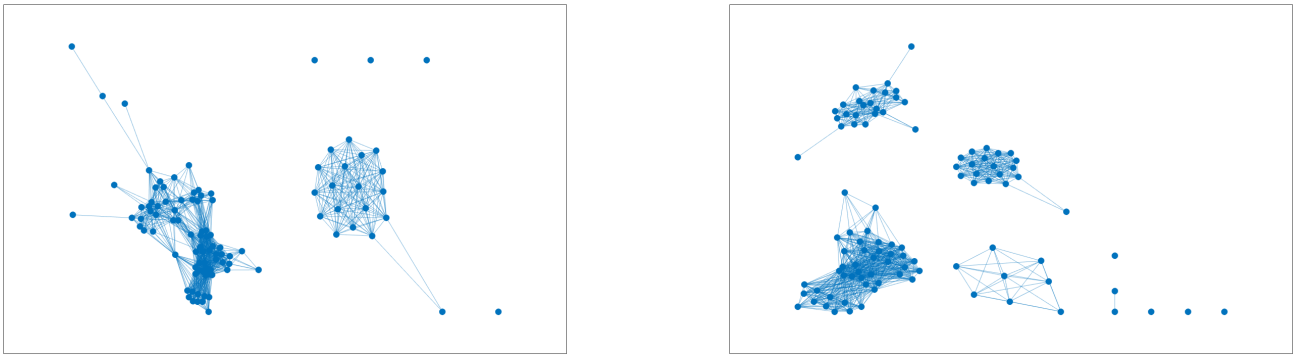


Figure 4.1: Clusters of first 100 items with $\alpha = 0.1$ and $\delta = 0.1$

From previous tests and results, it became clear that increasing the values of α and δ, more links are likely to disappear when applying this method. However, even increasing these two values give a similar chaotic output. Looking at the data set and hence the corresponding adjacency matrix this chaotic output makes sense. It turns that there are some characteristics that almost all of the items have. This lead to the decision of redefining when there exists a link between two agents. Hence, we will consider the case that there exists a link between two agents if they have at least four characteristics in common and that the weight of this link will be equal to the number of characteristics they have in common. If a cluster will be formed, in this cluster there must be agents that have at least four characteristics in common. For items in another clusters the same holds. The idea is now, that items from different clusters have none, or at most a few characteristics in common. In theory, these clusters could be used to recommend items. If a prospective buyer looks at an item, the items belonging to the same cluster could be items that this user likes as well. Testing this new definition of a link for the same 100 items as before had a more clear output as can be seen below.



(a) $\alpha = 0.1$ and $\delta = 0.1$

(b) $\alpha = 0.1$ and $\delta = 0.4$

Figure 4.2: Clusters found when there exists a link between two agents if they have at least four characteristics in common.

The higher the value of δ, the more clusters will form, which are more dense than the ones found for $\delta = 0.1$. With this restriction the clusters are more organised and can indeed be useful to recommend items. This is a plot for only the first 100 items. In Figure 4.3, the adjacency matrix for the first 1000 items has been formed and clustered by means of this algorithm.

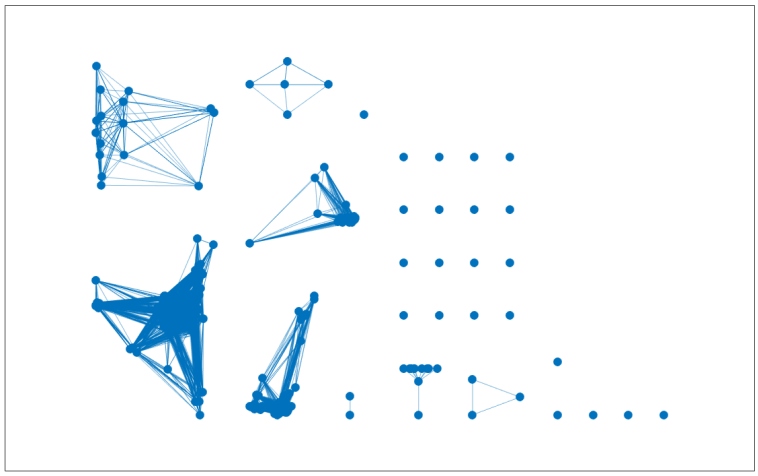
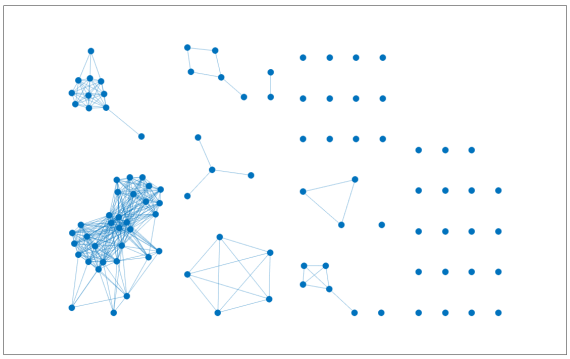
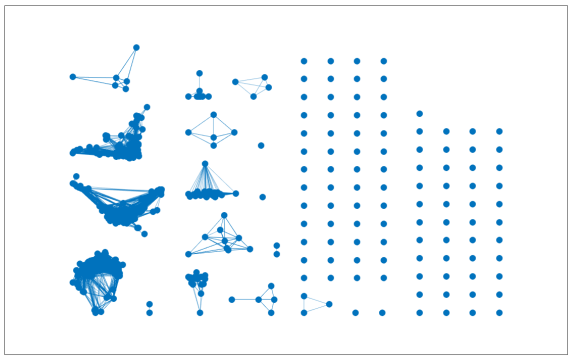


Figure 4.3: Clusters formed out of the first 1000 items with $\alpha = 0.1$ and $\delta = 0.4$.

However, in the plots in Figure 4.2 and in Figure 4.3 there are still many items contained in each cluster. If the definition of a link is even more strict, it is most likely that more and smaller clusters appear. In Figure 4.4 below a link between two agents exists if those two agents have at least six characteristics in common.



(a) Cluster for the first 100 items of the data set with $\alpha = 0.1$ and $\delta = 0.4$



(b) Cluster for the first 1000 items of the data set with $\alpha = 0.1$ and $\delta = 0.4$

Figure 4.4: Clusters of the data set. Here a link between two agents exists if those two agents have at least 6 characteristics in common. The results are plotted with $\alpha - 0.1$ and $\delta = 0.4$

As can be seen in the figures, the outcomes remain chaotic, even when a link has a more strict definition. Therefore, the methodology of Morărescu and Girard may not be the best choice of clustering method for recommendation systems with data sets this big.

5. Discussion

The article written by Morărescu and Girard, which was the main article for this thesis, explained the use of opinion dynamics in their methodology very well. The text was written clearly, and proofs haven been repeated, after learning more about background theorems such as the Perron-Frobenius theorem. However, while writing this thesis, some problems came up. When simulating one of the examples used to illustrate their methodology, it seemed that some information was missing. When simulating the example of the Karate club, it turned out that there was no clear restriction on the number of time-steps for which their simulation was still reliable and accurate. Eventually a substantiated choice for this restriction was made, which is also used in all of the other simulations in this research. Furthermore, in their results of this specific example it became clear that out of 1000 tries, not always the same clusters appear. Why this happens, could be due to some instability of the method. Performing the model on more examples could give more insight into why or when this happens.
Secondly their method belongs to the category of spectral clustering. In their article, the eigenvalues of the normalized Laplacian matrix of a partition is used to define a new measure of connectivity. This measure is used to determine if a partition is a correct solution to the problem of finding clusters and to say something about the density of the clusters. In this thesis, no research has been done on this part of the method. However, this could be an interesting approach to look at this methodology. It is already known that the higher this value of connectivity, the more densely connected these clusters are. The article implies that when clusters are more densely connected these clusters are automatically more stable, since the link are less likely to be removed. This stability could also be an interesting topic of research.

5.1 Further research

Some possibilities for further research have been mentioned already. However, the list of options for further research is not done. A different way of determining the competence of this methodology is looking at the computational effort of this algorithm. The advantage of this method is that it is not necessary to know the number of clusters in advance. Nevertheless, since Morărescu and Girard defined a dynamical system, MATLAB has to perform many calculations. But, there are more methods and algorithms that have this same advantage. For example, the method of Girvan and Newman, who focus on the least central links [9, 10]. What is the advantage of using the methodology of Morărescu and Girard over for example that of Girvan and Newman?
Furthermore, Morărescu and Girard mention that it is possible to define how well a partition reflects the community structure of a graph using modularity of partitions. This modularity uses the degree of agents and the predicted number of edges between two agents. Furthermore, it measures how well a partition reflects community structure of graphs. Hence, the higher the modularity, the better the community structure of a graph is reflected. However, modularity is not able to capture clusters of different sizes [14]. Hence, conducting research to find methods to check the stability of partitions, is very useful.

6. Conclusion

The intention of this thesis was to understand the methodology, the model and the mathematical proofs behind the model from the article of Morărescu and Girard. Their model relies on opinion dynamics, which induces a dynamical discrete-time system, and its goal is to form clusters in networks. The method has been explained and theorems and statements, substantiating the method, have been proved. The mathematics in their article was written clearly and well-explained.
Secondly, based on their article, two test cases have been formulated which have been used to develop intuition on what this algorithm does. On these specific cases, tests have been conducted to compare observations on the network with specific eigenvalues and eigenvectors. Based on a specific example, the behaviour of the links have been connected to the eigenvectors corresponding to the first matrix representing how the initial opinions would be updated. Furthermore, the impact of a weighted link on this network has been explained. For the second test case, the dynamics were manipulated by a choice of initial opinions, in order to find a critical value of ρ. This critical value of ρ represents the smallest value of ρ, for which no links in the test case were broken. This value has been compared to the eigenvalue characterising the dynamics of opinions and a linear relation has been found.
Finally, to illustrate the algorithm on a existing example, a data set found online has been converted to an input for the algorithm. This data set has been clustered based on different restrictions. These clusters could be used in simple recommendation systems of online web shops such as *Amazon.com* or streaming services like *Netflix* or *HBO*.
This method is probably not the best method to use on big data sets, since it is not able to cluster on multiple characteristics. However, for smaller examples this methods works quite well. Converting this problem into a dynamical system seems to work nicely to reformulate and solve the problem of clustering.

A. Appendix: Proofs

A.1 Proof of Proposition 2.2.1

Proposition 2.2.1 states that every agents' opinion will converge to a asymptotic opinion. Moreover, if agents will come to an agreement this agreement will be reached no slower than $\mathcal{O}(\rho^t)$.

Proof. Take $i \in V$ and $t \in \mathbb{N}$. Now using (2.1), the fact that elements p_{ij} are only non zero if and only if $j \in \{i\} \cup \{N_i(t)\}$ and Assumption 1,

$$\begin{aligned}
|x_i(t+1) - x_i(t)| &= \left| \left(\sum_{j=1}^{n} p_{ij}(t) x_j(t) \right) - x_i(t) \right| \\
&= \left| \sum_{j=1}^{n} p_{ij}(t)(x_j(t) - x_i(t)) \right| \\
&= \left| \sum_{j \in N_i(t)} p_{ij}(t)(x_j(t) - x_i(t)) \right| \\
&\leq \sum_{j \in N_i(t)} p_{ij}(t) |x_j(t) - x_i(t)|.
\end{aligned}$$

Here, the second equality holds since matrix $P(t)$ is right stochastic and hence $\sum_{j=1}^{n} p_{ij} = 1$. Next, from the definition of the confidence neighborhood $N_i(t)$ it follows that

$$|x_i(t+1) - x_i(t)| \leq \sum_{j \in \mathbb{N}} p_{ij}(t) R \rho^t.$$

Assumption 1 implies that for all $t \in \mathbb{N}$

$$|x_i(t+1) - x_i(t)| \leq (1 - p_{ii}(t)) R \rho^t \leq R \rho^t.$$

Now, take t, τ both in $\mathbb{N}$, then

$$|x_i(t+\tau) - x_i(t)| \leq \sum_{k=0}^{\tau-1} |x_i(t+k+1) - x_i(t+k)| \leq \sum_{k=0}^{\tau-1} R \rho^{t+k}$$

leading to

$$|x_i(t+\tau) - x_i(t)| \leq \frac{R}{1-\rho} \rho^t (1 - \rho^\tau) \leq \frac{R}{1-\rho} \rho^t$$

by taking τ to ∞ in the right hand side. Since the parameter ρ is chosen to be in the open interval $(0, 1)$, it follows that $(x_i(t))_{t \in \mathbb{N}}$ is a Cauchy sequence in $\mathbb{R}$ and hence a convergent sequence. □

A.2 Proof of Lemma 2.3.2

Lemma 2.3.2 states that under Assumption 1 and Assumption 3 there exists a $T' \in \mathbb{N}$ such that for all $t \geq T'$ it holds that $G(t) = G^\infty = G_\mathcal{C}$. To show this lemma it is useful to look at the definitions of these graphs. Since $G(t) = (V, E(t))$, $G^\infty = (V, E^\infty)$ and $G_\mathcal{C} = (V, E_\mathcal{C})$, proving Lemma 2.3.2 is equivalent to proving

$$E(t) = E^\infty = E_\mathcal{C} \tag{A.1}$$

for all $t \geq T'$, which will be done in three steps:

1. $E(t) \subseteq E^\infty$ for all $t \geq T'$;
2. $E^\infty \subseteq E_\mathcal{C}$;
3. $E_\mathcal{C} \subseteq E(t)$ for all $t \geq T'$;

Proof. Let $(i, j) \in E(t)$, with $t \geq T'$, and take $T_1 \geq T$, where T is as in (2.4). Then for all $t \geq T_1$ it is trivial that $(i, j) \in E^\infty$. Hence, it holds that $E(t) \subseteq E^\infty$.
Take $(i, j) \in E^\infty$, which implies that agents i and j are asymptotically connected. Now Proposition 2.3.1 says that agents i and j must asymptotically agree and hence that $x_i^* = x_j^*$. By the definition of $E_\mathcal{C}$ it can be concluded that $(i, j) \in E_\mathcal{C}$, which proves $E^\infty \subseteq E_\mathcal{C}$. To show the last step, let $(i, j) \in E_\mathcal{C}$, which means by definition that agents i and j must asymptotically agree. For all $t \in \mathbb{N}$ the following inequalities can be used.

$$\begin{aligned} |x_i(t) - x_j(t)| &= |x_i(t) - x_i^* + x_i^* - x_j^* + x_j^* - x_j(t)| \\ &\leq |x_i(t) - x_i^*| + |x_i^* - x_j^*| + |x_j^* - x_j(t)| \\ &= |x_i(t) - x_i^*| + |x_j(t) - x_j^*| \\ &\leq 2M\hat{\rho}^t. \end{aligned} \tag{A.2}$$

In the final step of (A.2) the fast convergence assumption, Assumption 3, has been used. Knowing that $\hat{\rho} < \rho$, lead to the statement that $\exists\, T_2 \in \mathbb{N}$ such that for all $t \geq T_2$ it holds that $2M\hat{\rho}^t \leq R\rho^t$. Form this we can conclude that for all $t \geq T_2$ it holds that $(i, j) \in E(t)$ and hence that $E_\mathcal{C} \subseteq E(t)$.

Now, taking $T' = \max\{T_1, T_2\}$, in combination with the above inclusions will prove (A.1) and hence Lemma 2.3.2. □

A.3 Proof of Lemma 2.5.1

Lemma 2.5.1 states a relation between the eigenvalues of $P_I(G_{\mathcal{P}})$ and $\mathcal{L}(G_I)$.

Proof. Let $P_I(G_{\mathcal{P}}) = I - \alpha Q(G_I)$ with elements $Q_{ij}(G_I)$

$$Q_{ij}(G_I) = \begin{cases} 1 & \text{if } i = j \text{ and } d_i(G_I) \neq 0. \\ \frac{-1}{d_i(G_I)} & \text{if } (i,j) \in E_I. \\ 0 & \text{otherwise.} \end{cases}$$

In order to prove the lemma two new matrices will be introduced.

$$R_{ij}(G_I) = \begin{cases} \frac{1}{\sqrt{d_i(G_I)}} & \text{if } i = j \text{ and } d_i(G_I) \neq 0. \\ \frac{-1}{d_i(G_I)\sqrt{d_j(G_I)}} & \text{if } (i,j) \in E_I. \\ 0 & \text{otherwise.} \end{cases} \tag{A.3}$$

$$D_{ij}(G_I) = \begin{cases} \sqrt{d_i(G_I)} & \text{if } i = j. \\ 0 & \text{otherwise.} \end{cases} \tag{A.4}$$

The products of the two matrices in (A.3) and (A.4) lead to

$$\begin{aligned} \mathcal{L}(G_I) &= D(G_I)R(G_I) \\ Q(G_I) &= R(G_I)D(G_I) \end{aligned}$$

and hence $\mathcal{L}(G_I)$ and $Q(G_I)$ have the same set of eigenvalues; $\sigma(\mathcal{L}(G_I)) = \sigma(Q(G_I))$. This leads to

$$\begin{aligned} \sigma(P_I(G_{\mathcal{P}})) &= 1 - \alpha \cdot \sigma(Q(G_I)) \\ \sigma(P_I(G_{\mathcal{P}})) &= 1 - \alpha \cdot \sigma(\mathcal{L}(G_I)) \\ \sigma(\mathcal{L}(G_I)) &= \frac{1 - \sigma(P_I(G_{\mathcal{P}}))}{\alpha} \end{aligned} \tag{A.5}$$

implying $\lambda \in \sigma(P_I(G_{\mathcal{P}}))$ if and only if $\mu = \frac{1-\lambda}{\alpha} \in \sigma(\mathcal{L}(G_I))$ □

B. Appendix: Tables

w_{45}	$\delta = 0.25$	$\delta = 0.26$	$\delta = 0.27$	$\delta = 0.28$	$\delta = 0.29$	$\delta = 0.3$
4.5	1	1	1	1	3	3
4.6	1	1	1	1	3	3
4.7	1	1	1	1	2	3
4.8	1	1	1	1	1	3
4.9	1	1	1	1	1	3
5.0	1	1	1	1	1	3
5.1	1	1	1	1	1	3
5.2	1	1	1	1	1	3
5.3	1	1	1	1	1	3
5.4	1	1	1	1	1	1
5.5	1	1	1	1	1	1

Table B.1: Number of clusters found for different values of δ and w_{45} with $\alpha = 0.1$ and $R = 10$.

w_{45}	$\delta = 0.25$	$\delta = 0.26$	$\delta = 0.27$	$\delta = 0.28$	$\delta = 0.29$	$\delta = 0.3$
4.5	1	1	1	1	3	3
4.6	1	1	1	1	3	3
4.7	1	1	1	1	3	3
4.8	1	1	1	1	1	3
4.9	1	1	1	1	1	3
5.0	1	1	1	1	1	3
5.1	1	1	1	1	1	3
5.2	1	1	1	1	1	3
5.3	1	1	1	1	1	2
5.4	1	1	1	1	1	3
5.5	1	1	1	1	1	1

Table B.2: Number of clusters found for different values of δ and w_{45} with $\alpha = 0.2$ and $R = 10$.

C. Appendix: MATLAB Programs

Matlab program to simulate opinion dynamics model from [14]

```
clear all
close all
A = [0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 ;
    1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 ;
    1 1 0 1 0 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 ;
    1 1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
    1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
    1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
    1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
    1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
    1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 ;
    0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 ;
    1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
    1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
    1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
    1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 ;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 ;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 ;
    0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
    1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 ;
    1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 ;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 ;
    1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 ;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 1 1 ;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 ;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 ;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 ;
    0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 ;
    0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 ;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 ;
    0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 ;
    1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 1 1 ;
    0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 1 1 1 0 1 ;
    0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 1 0 0 1 0 0 1 1 1 1 1 1 1 0 ];  %Adjacency
     matrix
d = 0.1;                                        %Parameter \delta
R = 10;                                         %Parameter R
a = 0.2;                                        %Parameter \alpha
r = 1-a*d;                                      %Parameter \rho

for t = 1:80000
        if r^t<=1e-15                           %Find tend for MATLAB's accuracy
         break
```

```
        end
end
tend = t;

X(:,1) = rand(length(A),1);                     %Initial opinion vector
for t = 1:tend                                  %Time-loop
    M = zeros(length(A));
    P = zeros(length(A));
    for i = 1:length(A)                         %Loop over every individual column
        for l = 1:length(A)                     %Check which agents exchange opinion
            if l~=i && A(l,i)==1 && abs(X(i,t) - X(l,t))<=R*r^t  %N_i(t)
                M(l,i) = 1;
            else
                M(l,i) = 0;
            end
        end
    colsumM = sum(M);
    N(i,t) = colsumM(i);                        %#agents interacting with i at time t
    end
    for n = 1:length(A)                         %Filling matrix P(t) at time t
        if N(n,t)==0                            %If i has no interacting neighbors
            P(n,n) = 1;                         %x_i(t+1) = x_i(t)
        else                                    %If i has interacting neighbors
            P(n,n) = 1 - a;                     %p_{ii} = 1-a
                for b = 1:length(A)
                    if M(n,b)==1 && b~=n        %If j is interacting neighbor of i
                        P(n,b) = a/N(n,t);      %p_{ij} = a/(N_i(t))
                    elseif M(n,b)==0 && b~=n
                        P(n,b) = 0;
                    end
                end
        end
    end
    X(:,t+1) = P*X(:,t);                        %Weighted average: x(t+1)=P(t)x(t)
end
G = graph(M);                                   %Graph of final adjacency matrix M
plot(G,'NodeLabel',{},'MarkerSize',12)          %Plot the final state

conn = conncomp(G)';                            %Find which agent belongs to which
    cluster

I1 = [];
I2 = [];
for i = 1:length(conn)
    if conn(i)==1                               %Create G_{I_1}
        I1 = [I1 i];
    elseif conn(i)==2                           %Create G_{I_1}
        I2 = [I2 i];
    end
end

```

```
for k = 1:length(I1)
    for l = 1:length(I1)
        if k==l
            M1(k,l)=0;
        elseif k~=l && A(I1(k),I1(l))==1        %Make E_{I_1}
            M1(k,l)=1;
        end
    end
end

for k = 1:length(I2)
    for l = 1:length(I2)
        if k==l
            M2(k,l)=0;
        elseif k~=l && A(I2(k),I2(l))==1        %Make E_{I_2}
            M2(k,l)=1;
        end
    end
end

G1 = graph(M1);                                 %G_{I_1}
L1 = laplacian(G1);                             %L(G_{I_1})
D1 = diag(degree(G1));
NLM1 = (D1^(-0.5))*L1*(D1^(-0.5));              %Normalized Laplacian
e1 = sort(eig(NLM1));
m1 = e1(2);                                     %2nd smallest eigenvalues of
    normalized Laplacian matrix

G2 = graph(M2);                                 %G_{I_2}
L2 = laplacian(G2);                             %L(G_{I_2})
D2 = diag(degree(G2));
NLM2 = (D2^(-0.5))*L2*(D2^(-0.5));              %Normalized Laplacian
e2 = sort(eig(NLM2));
m2 = e2(2);                                     %2nd smallest eigenvalues of
    normalized Laplacian matrix

m = min([m1 m2]);                               %Algebraic connectivity according to
    Morarescu and Girard
logical(m>d)
```

Listing C.1: Simulation program

Matlab program to find the amount of clusters that arise for different values of α, δ and w_{45}.

```
clear all
close all
A = [0 1 1 1 0 0 0 0 ; 1 0 1 1 0 0 0 0 ;
     1 1 0 1 0 0 0 0 ; 1 1 1 0 0 0 0 0 ;
     0 0 0 0 0 1 1 1 ; 0 0 0 0 1 0 1 1 ;
     0 0 0 0 1 1 0 1 ; 0 0 0 0 1 1 1 0 ];           %Adjacency matrix

R = 10;                                             %Parameter R
a = 0.1;                                            %Parameter \alpha

W = zeros(length(A));

weights = [1:0.5:6];                                %Vector of weights
delta = [0.1:0.05:0.5];                             %Vector of different \delta's
C = zeros(length(weights),length(A));
clusters = [weights'];


X(:,1) = [0.5 0.5 0.5 0.5 0.55 0.55 0.55 0.55]';    %Initial opinion vector
for d = 1:length(delta)
    r(d) = 1-a*delta(d);
    for t = 1:6000
        if r(d)^t<=1e-15                            %Find tend for MATLAB's accuracy
         break
        end
    end
    tend = t;

    for w = 1:length(weights)                       %Weight-loop
        A = [0 1 1 1 0 0 0 0 ; 1 0 1 1 0 0 0 0 ;
        1 1 0 1 0 0 0 0 ; 1 1 1 0 weights(w) 0 0 0 ;
        0 0 0 weights(w) 0 1 1 1 ; 0 0 0 0 1 0 1 1 ;
        0 0 0 0 1 1 0 1 ; 0 0 0 0 1 1 1 0 ];

    for t = 1:tend                                  %Time-loop
            M = zeros(length(A));
            P = zeros(length(A));
            for i = 1:length(A)                     %Loop over every individual column
                for l = 1:length(A)                 %Check which agents exchange
    opinion
                    if l~=i && A(l,i)~=0 && abs(X(i,t) - X(l,t))<=R*r(d)^t  %N_i(t)
                        M(l,i) = A(l,i);
                    else
                        M(l,i) = 0;
                    end
                end
            colsumM = sum(M);
            W(i,t) = colsumM(i);                    %Sum of weights
            end
```

```
49
50            for n = 1:length(A)                          %Filling matrix P(t) at time t
51                if W(n,t)==0                             %If i has no interacting neighbors
52                    P(n,n) = 1;                          %x_i(t+1) = x_i(t)
53                else                                     %If i has interacting neighbors
54                    P(n,n) = 1 - a;                      %p_{ii} = 1-a
55                    for b = 1:length(A)
56                        if M(n,b)~=0 && b~=n             %If j is interacting neighbor of i
57                            P(n,b) = (M(n,b)*a)/W(n,t); %p_{ij} = w*a/(sum(weights))
58                        elseif M(n,b)==0 && b~=n
59                            P(n,b) = 0;
60                        end
61                    end
62                end
63            end
64        X(:,t+1) = P*X(:,t);                             %Weighted average: x(t+1)=P(t)x(t)
65        end
66
67    G = graph(M);                                        %Graph of final adjacency matrix M
68    C(w,:) = conncomp(G);                                %Finds which agent belongs to
    which cluster
69    end
70    clusters = [clusters max(C,[],2)];
71 end
72 clusters;
73 Table = array2table(clusters, 'VariableNames', {'weight', 'delta = 0.1', 'delta = 0.15
    ', 'delta = 0.2', 'delta = 0.25','delta = 0.3', 'delta = 0.35', 'delta = 0.4', '
    delta = 0.45', 'delta = 0.5'})
```

Listing C.2: Program to produce a table with amount of clusters arisen

Matlab program to plot ρ against number of clusters found

```
clear all
close all
format long
A = [0 1 0 0 0 ; 1 0 1 0 0 ; 0 1 0 1 0 ; 0 0 1 0 1 ; 0 0 0 1 0];    %Adjacency Matrix

R = 10;                                          %Parameter R
a=0.1;                                           %Parameter \alpha
n = 100;                                         %Number of points in linspace
rho = linspace(0.967,0.968,n);                   %Parameter \rho

wtot = sum(A,2);                                 %Find #neighbors of every agent
P = eye(size(A))+ a*inv(diag(wtot))*(A - diag(wtot));   %Matrix P(0)
[u,d] = eig(P);                                  %Eigenvectors & eigenvalues
[d, ind] = sort(diag(d), 'descend');             %Sort eigenvalues in descending order
u = u(:, ind);                                   %Sort eigenvectors in same order

e2 = [0 1 0 0 0]';                               %2nd standard basis vector
X(:,1) = u*e2;                                   %Initial opinion vector

clusters = [];
Rho = [];
TRUE = zeros(size(A));
T = [];

for r = 1:length(rho)
    Rho = [Rho ; rho(r)];
    for t = 1:6000
        if rho(r)^t<=1e-15                       %Find tend for MATLAB's accuracy
         break
        end
    end
    tend = t;
    for t = 1:tend                               %Time-loop
        for k = 1:size(A)
            for l = 1:size(A)
                if A(k,l) ==1 && abs(X(k,t)-X(l,t))<=R*rho(r)^t   %Check difference in
     opinion
                    TRUE(k,l) = 1;               %If within area of confidence, then
    there is an edge connecting those agents
                else
                    TRUE(k,l) = 0;
                end
            end
        end
    T = [T TRUE];
        if TRUE(3,:)==zeros(1,5)                 %If links around agent 3 break
          break                                  %The time-loop stops
        end

    X(:,t+1) = u(:,2)*(d(2)^t);                  %Update of opinions
```

```
    end
clusters= [clusters ; max(conncomp(graph(TRUE)))];
end
L = [Rho clusters];
Table = array2table(L, 'VariableNames', {'\rho', 'clusters found'})
figure
scatter(L(:,1),L(:,2),80,'g','filled')      %Plot \rho against #clusters found
xlabel('\rho')
ylabel('Number of clusters found')
set(gca,'Fontsize',28)
```

Listing C.3: Program to plot ρ against number of clusters

Program to plot critical values of ρ and δ against parameters α and λ_2, for which no link disappears

```
clear all
close all
format long
A = [0 1 0 0 0 ; 1 0 1 0 0 ; 0 1 0 1 0 ; 0 0 1 0 1 ; 0 0 0 1 0];    %Adjacency Matrix

R = 10;                                          %Parameter R
alpha = linspace(0,0.5);                         %Linspace for parameter \alpha

Rho = [];
for a = 1:length(alpha)                          %Loop over \alpha
    Alpha=alpha(a);                              %Parameter \alpha
    rho = linspace(0.7,1.0,1000);                %Linspace for parameter \rho

    wtot = sum(A,2);                             %Find #neighbors of every agent
    P = eye(size(A))+ Alpha*inv(diag(wtot))*(A - diag(wtot));   %Matrix P(0)
    [u,d] = eig(P);                              %Eigenvectors & eigenvalues
    [d, ind] = sort(diag(d), 'descend');         %Sort eigenvalues in descending order
    u = u(:,ind);                                %Sort eigenvectors in same order

    e2 = [0 1 0 0 0]';                           %2nd standard basis vector
    X(:,1) = u*e2;                               %Initial opinion vector

    TRUE = zeros(size(A));

    for r = 1:length(rho)                        %Loop over all values of \rho
        for t = 1:6000
            if rho(r)^t<=1e-15                   %Find tend for MATLAB's accuracy
             break
            end
        end
        tend = t;
        for t = 1:tend                           %Time-loop
            for k = 1:size(A)
                for l = 1:size(A)
                    if A(k,l) ==1 && abs(X(k,t)-X(l,t))<=R*rho(r)^t     %Check
   difference in opinion
                        TRUE(k,l) = 1;           %If k in area of confidence of l, then
    there is an edge connecting those agents
                    else
                        TRUE(k,l) = 0;
                    end
                end
            end
            if TRUE(3,:)==zeros(1,5)             %If links around agent 3 break
              break                              %The time-loop stops
            end

        X(:,t+1) = u(:,2)*(d(2)^t);              %Update of opinions
        end
```

```
48
49          if max(conncomp(graph(TRUE)))==1        %If #clusters found is equal to 1
50            break                                 %The loop over \rho stops
51          end
52
53      end
54  Rho = [Rho rho(r)];                             %Add critical value of \rho
55  end
56
57  for h = 1:length(alpha)
58      K = eye(size(A))+ alpha(h)*inv(diag(wtot))*(A - diag(wtot));
59      val = sort(eig(K),'descend');
60      lambda(h) = val(2);                         %lambda_2 for all values of alpha
61      delta(h) = (1-Rho(g))/alpha(g);             %corresponding critical \delta to
        critical \rho and \alpha
62  end
63  scatter(alpha,lambda,80,'g','filled')                       %Plot \lambda_2 as
        function of \alpha
64  xlabel('value of \alpha')
65  ylabel('value of \lambda_2')
66  set(gca,'Fontsize',28)
67
68  figure
69  scatter(alpha,Rho,80,'g','filled')              %Plot critical \rho as function of \
        alpha
70  xlabel('value of \alpha')
71  ylabel('value of \rho')
72  set(gca,'Fontsize',28)
73
74  figure
75  scatter(lambda,Rho,80,'g','filled')             %Plot critical \rho as function of \
        lambda_2
76  xlabel('value of \lambda_2')
77  ylabel('value of \rho')
78  set(gca,'Fontsize',28)
79
80  figure
81  scatter(lambda,delta,80,'g','filled')           %Plot corresponding critical \delta as
         function of \lambda_2
82  xlabel('value of \lambda_2')
83  ylabel('value of \delta')
84  set(gca,'Fontsize',28)
```

Listing C.4: Program to find relations between parameters

Program to build adjacency matrix from the data set

```
clear all
close all


T = readtable('amazon_music_metadata.csv'); %Importt data set
A = table2array(T(1:end,3:end));            %Convert table to matrix
H = zeros(size(A,1),size(A,2));
E = zeros(size(A,1));
for i = 1:size(A,1)                         %Loop over all items
    for j = 1:size(A,2)                     %Loop over all characteristics
        if A(i,j)==1                        %If item i has characteristic j
            for q = 1:size(A,1)             %Loop over all items
                if q==i
                    H(q,j)=0;
                elseif A(q,j)==1 && q~=i    %If item q has the same characteristic
                    H(q,j)=1;               %H(q,j)=1
                end
            end
        elseif A(i,j)== 0                   %If item i does not have characteristic j
            for e = 1:size(A,1)
                H(e,j) = 0;                 %j'th column of H is zero
            end
        end
    end
   S = sum(H,2);                            %Column vector with sum of each row
   for w = 1:size(A,1)                      %Loop over all items
       if S(w)>=4                           %If item i en w have 4 or more
    characteristics in common as restriction
       E(i,w) = S(w);                       %E(i,w)=#in common = weight of link
       else
           E(i,w) = 0;
       end
   end
end
E;                                          %Adjacency matrix

d = 0.4;                                    %Parameter \delta
R = 10;                                     %Parameter R
a = 0.2;                                    %Parameter \alpha
r = 1-a*d;                                  %Parameter \rho

for q = 1:8000
    if r^q<=1e-15
        break
    end
end
tend = q;

X(:,1) = rand(length(E),1);                 %Initial opinion vector
for t = 1:tend                              %Time-loop
```

```
    M = zeros(length(E));
    P = zeros(length(E));
    for i = 1:length(E)                     %Loop over every individual column
        for l = 1:length(E)                 %Check which agents exchange opinion
            if l~=i && E(l,i)~=0 && abs(X(i,t) - X(l,t))<=R*r^t  %N_i(t)
                M(l,i) = E(l,i);
            else
                M(l,i) = 0;
            end
        end
    colsumM = sum(M);
    N(i,t) = colsumM(i);                    %#agents interacting with i at time t
    end
    for n = 1:length(E)                     %Filling matrix P(t) at time t
        if N(n,t)==0                        %If i has no interacting neighbors
            P(n,n) = 1;                     %x_i(t+1) = x_i(t)
        else                                %If i has interacting neighbors
            P(n,n) = 1 - a;                 %p_{ii} = 1-a
                for b = 1:length(E)
                    if M(n,b)~=0 && b~=n    %If j is interacting neighbor of i
                        P(n,b) = (M(n,b)*a)/N(n,t);      %p_{ij} = a/(N_i(t))
                    elseif M(n,b)==0 && b~=n
                        P(n,b) = 0;
                    end
                end
        end
    end
    X(:,t+1) = P*X(:,t);                    %Weighted average: x(t+1)=P(t)x(t)
end
G = graph(M);                               %Graph of final adjacency matrix M
plot(G,'NodeLabel',{},'MarkerSize',12)      %Plot clusters
```

Listing C.5: Making adjacency matrix from the data set

Bibliography

[1] https://github.com/caserec/Datasets-for-Recommender-Systems/blob/master/Processed%20Datasets/AmazonMusic.tar.xz, September 2019. Accessed: 2020-06-16.

[2] C.C. Aggerwal. *Reccomender Systems: The Textbook*. Springer, 2016.

[3] V.I. Bogachev. *Measure Theory, Volume 1*. Springer, 2007.

[4] F.O. Farid. Notes on matrices with diagonally dominant properties. *Linear Algebra and its Applications*, 435(11):2793–2812, 2011.

[5] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):289–305, 1973.

[6] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(4):619–633, 1975.

[7] M. Fiedler. Laplacian of graphs and algebraic connectivity. *Banach Center Publications*, 25(1):57–70, 1989.

[8] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[9] M. Girvan and M.E.J. Newman. Structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826, 2002.

[10] M. Girvan and M.E.J. Newman. Finding and evaluating community structure in networks. *Physical Review*, 69(2), 2004.

[11] R.A. Horn and C.R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

[12] E. Jacobsen and R.S. Weiss. A method for the analysis of the structure of complex organizations. *American Sociological Review*, 20(6):661–668, 1955.

[13] R. Kumar and P.K. Pattnaik. *Graph Theory*. Bengaluru: Laxmi Publications, 2018.

[14] I. Morarescu and A. Girard. Opinion dynamics with decaying confidence: Application to community detection in graphs. *IEEE Transactions on Automatic Control*, 56(8), 2011.

[15] M.E.J. Newman. Modularuty and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23):8577–8582, 2006.

[16] T. Paoletti. Leonard Euler's solution to the Koningsberg bridge problem. https://www.maa.org/press/periodicals/convergence/leonard-eulers-solution-to-the-konigsberg-bridge-problem, May 2011. Accessed: 2020-06-12.

[17] S.U. Pilai, T. Suel, and S. Cha. The Perron-Frobenius theorem. *IEEE Signal Processing Magazine*, 22(2):62–75, 2005.

[18] S.A. Rice. The identification of blocs in small political bodies. *The American Political Science Review*, 21(3):619–627, 1927.

[19] D.A. Smith. Convergent sequences (and the squeeze theorem). https://www.directknowledge.com/convergent-sequences/, January 2010. Accessed: 2020-04-22.

[20] D.A. Spielman and S.H. Teng. Spectral partitioning works: Planar graphs and finite element meshes. *Linear Algebra and its Applications*, 421(2-3):284–305, 2007.

[21] A.L. Traud, E.D. Kelsic, P.J. Mucha, and M.A. Porter. Comparing community structure charachteristics in online collegiate social networks. *SIAM Review*, 53(3):526–543, 2011.

[22] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[23] R. Xing and B. Zhou. Sharp bounds for the spectral radius of nonnegative irreducible matrices. *Linear Algebra and its Applications*, 449:194–209, 2014.

[24] W.W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.