2021

University of Piraeus
Department of Digital Systems
Information Systems and Services
Big Data and Analytics

DEMOKRITOS

Community Detection in Signed & Directed Graphs

Master Thesis

NIKOS PSYLLAKOS, MTN 1916

SUPERVISOR: MARIA HALKIDI, ASSOCIATE PROFESSOR

npsyllakos@gmail.com
OTE
27-Jun-21

Table of Contents

Acknowledgements

I would first like to thank my supervisor, Maria Halkidi, Associate Professor in the Department of Digital Systems, University of Piraeus, for giving me the opportunity to work on such an exciting project for my master thesis.
Your expertise was crucial in formulating the research questions and methodology and your persistent feedback coached my thinking and directed my work to a higher level.
Finally, I must express my very profound gratitude to my wife for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without her.

Nikos Psyllakos
June 2021

Abstract

In recent 2 decades, one of the most popular methods for processing data , is to convert it to a network, where data units are vertices and their relations are the links that connects them.
Such Networks in a clustering problem are usually represented as graphs where each element to be clustered is represented as a node and the distance between two elements is modeled by a certain weight on the edge linking the nodes. Graph clustering aims at partitioning a set of nodes into different groups called clusters or communities that share some form of similarity, similarity score can be formulated by using a distance-based criterion, a topology structural criterion or an “attitude” (polarity) criterion. These clusters will help us to explore information hidden in the data , there is a wide area of applications as e.g. Social Network Analysis, Statistical Data Analysis, Machine Learning, Data mining, Consuming Behavior Analysis, VLSI design, Computer graphics and Gene analysis.

This particular Thesis is putting under the spotlight Signed and Directed graphs and examines two community detection algorithms that cluster nodes with similar characteristics, the goal of the procedure is to emerge the Similarity score as the main and most important factor of Graph Clustering between nodes based on the connectivity pattern they follow including both directionality and polarity of the links which connects them.

The initial step is to work with the negative and positive subgraph of the examined Data Set. For each Subgraph we apply **Coupling (Reference) and Cocitation** as the Methodologies by which Directionality dominates criteria to reach Similarity and we calculate the appropriate co-citation and co-reference matrices. The *Reference or bibliographic coupling Matrix* of two nodes in a directed network is the number of nodes to which both these two nodes point **on the other hand** the *Cocitation Matrix* of two nodes in a directed network is the number of nodes that point to these both exact nodes. The procedure continues with the similarity Matrix calculation normalized by taking account of the degree of each node and under the influence **of Amsler [Amsler, 1972]** who pointed out that Co-citation and Bibliography coupling can be combined. And finally we apply two clustering algorithms affinity propagation and spectral clustering, both standalone and combined.

In this Thesis, we provide a concluded implementation of these proposed algorithms in Python,
and the datasets that were used to evaluate in practice their precision and certainty.
We display that these two algorithms have the theoretically expected results for both random
Generated signed directed graphs and real networks data sets.

Keywords

Signed and Directed Graphs, Social Network Analysis, Graph Clustering, Adjacency Matrix, Community Detection, Similarity Score, Co-Citation, Co-Reference, Affinity Propagation, Spectral Clustering, Python, Directionality, Polarity.

Περίληψη

Τις τελευταίες 2 δεκαετίες μια από τις δημοφιλέστερες μεθοδολογίες για την επεξεργασία δεδομένων είναι η μετατροπή τους σε δικτύωμα όπου οι μονάδες δεδομένων ταυτίζονται με τους κόμβους και οι σχέσεις που τις συνδέουν είναι οι ακμές του δικτυώματος.
Τέτοια Δίκτυα σε ένα πρόβλημα συσταδοποίησης συνήθως αντιπροσωπεύονται από Γράφους όπου κάθε συστατικό στοιχείο αντιπροσωπεύεται από κόμβους και οι σύνδεσμοι ανάμεσα στα συστατικά στοιχεία μοντελοποιούνται με συγκεκριμένης βαρύτητας ακμές. Η συσταδοποίηση σε γράφους στοχεύει στην κατηγοριοποίηση των κόμβων ενός γράφου σε ομάδες (συστάδες) ή κοινότητες, οι κόμβοι αυτοί μοιράζονται κάποιο βαθμό ομοιότητας, ο οποίος υπολογίζεται με κριτήρια την απόσταση ανάμεσα στους κόμβους του γράφου την τοπολογική δομή αλλά και την «συμπεριφορά» ανάμεσα στους κόμβους (φίλοι ή εχθροί). Οι συστάδες αυτές μας βοηθούν να εξερευνήσουμε τις πληροφορίες που κρύβονται μέσα στα σύνολα δεδομένων. Η μεθοδολογική αυτή προσέγγιση χρησιμοποιείται ευρέως σε επιστημονικά πεδία όπου είναι απαραίτητη η μελέτη και ανάλυση συμπεριφορικών μοντέλων η κοινωνιολογικών στερεοτύπων , όπως η ανάλυση κοινωνικών δικτύων, η στατιστική ανάλυση δεδομένων, η Εξόρυξη Δεδομένων, η Μηχανική Μάθηση, ο σχεδιασμός ολοκληρωμένων κυκλωμάτων, η Βιολογία και άλλα.

Η παρούσα Διπλωματική εργασία βάζει στο μελετητικό επίκεντρο τους Υπογεγραμμένους και Κατευθυντικούς γράφους και εξετάζει δύο αλγορίθμους εντοπισμού κοινοτήτων που συσταδοποιούν κόμβους με παρόμοια χαρακτηριστικά , ο σκοπός της διαδικασίας είναι η ανάδειξη του βαθμού ομοιότητας ως το κύριο και πιο σημαντικό παράγοντα συσταδοποίησης γράφων ανάμεσα σε κόμβους με βάση το μοτίβο συνδεσιμότητας ακολουθώντας ταυτόχρονα και την κατευθυντικότητα αλλά και την πόλωση των συνδέσμων που τους διασυνδέουν.

Στο αρχικό στάδιο εργαστήκαμε παράλληλα με τους θετικούς και αρνητικούς υπογράφους συγκεκριμένων συνόλων δεδομένων για κάθε υπογράφο εφαρμόσαμε τις μεθοδολογικές προσεγγίσεις **Coupling (Reference) και Cocitation** αναλύοντας αναφορές και συνδέσεις μεταξύ των κόμβων για υπολογίσουμε τους ανάλογους πίνακες η διαδικασία συνεχίστηκε με τον υπολογισμό του Πίνακα ομοιότητας και τις αναγκαίες κανονικοποιήσεις παίρνοντας υπόψη το βαθμό κάθε κόμβου αλλά και την επιστημονική εργασία του Αμσλερ που ανέδειξε ότι οι μεθοδολογικές προσεγγίσεις **Coupling (Reference) και Cocitation μπορούν να συνδυαστούν.** Τέλος εφαρμόσαμε δύο αλγορίθμους συσταδοποίησης affinity propagation και spectral clustering, τόσο ανεξάρτητα όσο και σε συνδυασμό.

Στη Διπλωματική παρέχουμε μια ολοκληρωμένη εφαρμογή των προτεινόμενων αλγορίθμων σε Python και τα σύνολα δεδομένων πάνω στα οποία εργαστήκαμε για να αξιολογήσουμε την δουλειά που έγινε σε επίπεδο ακρίβειας και ορθότητας. Δείξαμε ότι οι αλγόριθμοι αυτοί απέδωσαν τα αναμενόμενα από την θεωρία αποτελέσματα τόσο σε τυχαία παραγόμενους γράφους αλλά και σε δεδομένα πραγματικών δικτυωμάτων.

Λέξεις-Κλειδιά
Υπογεγραμμένοι και Κατευθυντικοί γράφοι, Ανάλυση Κοινωνικών Δικτύων, Συσταδοποίηση Γράφων,

Ανίχνευση Κοινοτήτων, Μέτρο Ομοιότητας, Affinity Propagation, Spectral Clustering, Python, κατευθυντικότητα, πόλωση.

1. Introduction to Graphs

1.1 Challenge Representation

“A picture speaks a thousand words” is one of the most commonly used phrases. But a graph speaks so much more than that. We can name Graphs **the mathematical structures used to study pairwise relationships between objects and entities.**

The visual representation of data, in the form of graphs, assist us to gain actionable insights and make by far better data oriented decisions based on them. To categorize and understand the wide usage of Graphs/Networks we must adopt terminology from **graph theory**. Graph theory is a specific sector of mathematics that emerged in the 19th and 20th century. It was a need of times as it allowed experts to describe phenomena from various fields: communication infrastructures, drawing and coloring maps, scheduling tasks and social networks mainly because it provides a better way of dealing with abstract concepts like relationships and interactions.

Every graph is a collection of nodes that can be connected to each other by means of edges. Each edge of a graph connects exactly two nodes. In a formal notation, a graph G is defined as follows:

A graph G consists of a collection N of nodes and a collection E of edges, for these elements we write G = (N, E).

Each edge e belongs to E collection is said to join two nodes, which are called its end points. If the edge e joins u, v which belongs to N collection of nodes, we represent it formally by writing e = (u, v).

Nodes u and v in this case are said to be adjacent. Edge e is said to be incident with nodes u and v, respectively. The nodes u and v are called the end nodes of the edge (u,v). If two edges have the same end nodes they are parallel. An edge of the form (v,v) is a loop.

A Graph is simple if it has no parallel edges and loops. A Graph is said to be Empty if it has no edges meaning E is empty. A Graph is a Null Graph if it has no nodes meaning N and E are empty.

Edges are Adjacent if they have a common node; nodes are Adjacent if they have a common edge. The degree of the node v, written as d(v), is the number of edges with v as an end node. We must acknowledge at this point that by convention, we count a loop twice and parallel edges contribute separately. Isolated nodes are nodes with degree 1.

A Graph is Complete if its edge set contains every possible edge between ALL of the nodes.

Any Walk in a Graph G = (N,E) is a finite, alternating sequence consisting of nodes and edges of the graph G. Any Walk is Open if the initial and final nodes are different. Any Walk is closed if the initial and final nodes are the same. Any Walk is a Trail if ANY edge is traversed at most once and consequently a Trail is a Path if ANY node is traversed at most once (Except for a closed walk). Any Closed Path is a Circuit. **[Wilson. 1986]**

There are many types of Graphs as shown in the following figure

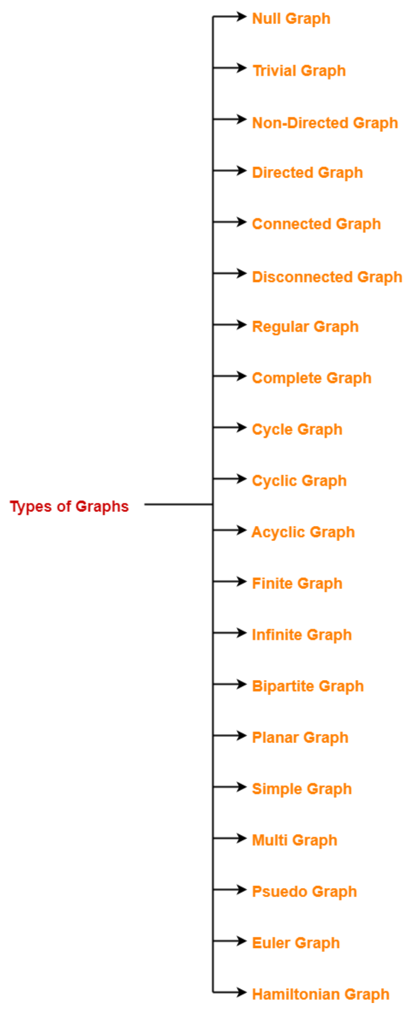


Figure 1.Graph Types

1.2 The Graph clustering Challenge

The Graph clustering challenge is the organized methodology of the detection of clusters or communities within a Graph. Cluster or community in a network is typically considered as a group of nodes with better connectivity (and/or stronger interactions) among its members than with the nodes of different communities**.**

The main types of Graphs that are used in Social Networks are summarized in the following categories: **Directed, Undirected, Signed** and **Unsigned**

Directed social networks are distinguished from undirected ones by the presence of directed edges between nodes. An example for **directed** network could be followership in Twitter where a user/node simply 'follows' another. Alternatively, undirected social networks consist of undirected edges between nodes. Facebook is an example for **undirected** networks with edges depicting only mutual friendships.

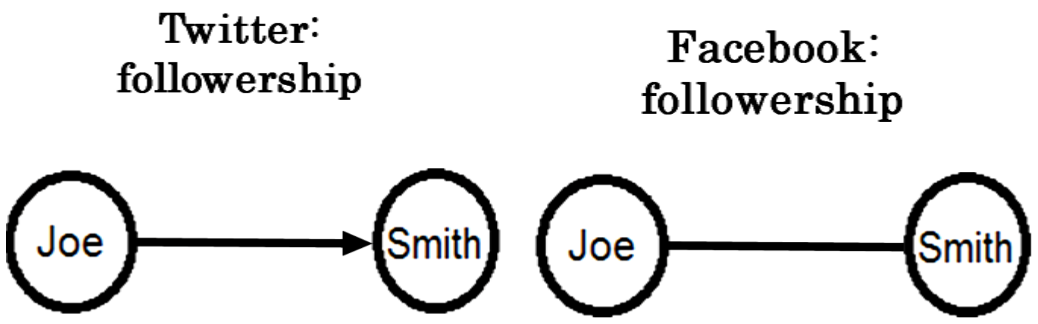
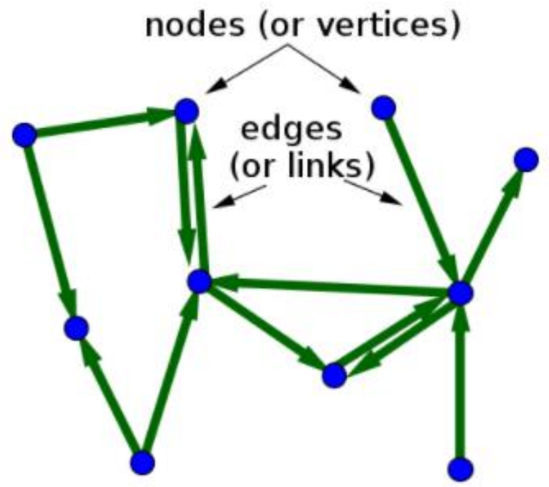


Figure 2.Followership

Another type of classification deals with nature of interactions (positive or negative) involved in social networks. In this type of classification, a social network could be categorized as either **signed** or **unsigned**. Unsigned networks are described by the presence of a single type of interaction, usually being positive in nature. In unsigned networks all nodes are the same, either the describe friends or strangers. But in the real world, relationships are not always positive by nature. Signed networks, capture this aspect of society allowing explicit show of trust or distrust among nodes. They can be designated as friends or foes. In this case, a node is said to be friends with the other if the node approves the opinion among themselves, or foes if a node disapproves the opinion among themselves.
E-opinions, Slashdot and Wiki are some of the examples of signed networks that indicate trust/friends or distrust/foes explicitly among themselves using an edge-weight of +1 and –1 correspondingly.

1.3 Type of Graphs

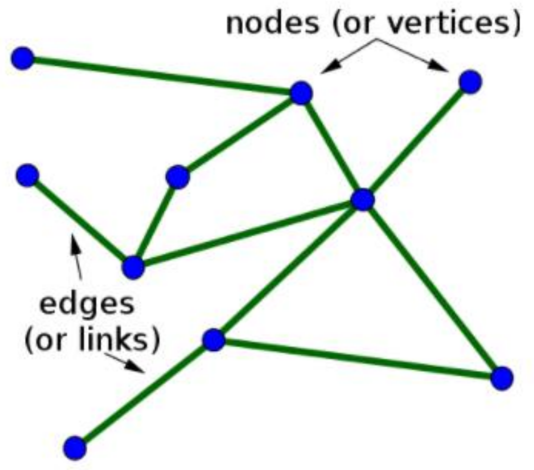
A Directed graph is a set of objects (called nodes) that are connected together, where all the edges are directed from one node to another. A directed graph is usually called as a digraph or a directed network. We can formally define a directed graph as G = (N,E), consisting of the set N of nodes and the set E of edges, which are ordered pairs of elements of N.



A directed graph with 10 vertices (or nodes) and 13 edges.

Figure 3.Directed Graph

An Undirected graph is a set of objects (called nodes) that are connected together, where all the edges are bidirectional. An undirected graph is usually called as an undirected network. We can formally define an undirected graph as G = (N,E), consisting of the set N of nodes and the set E of edges, which are unordered pairs of elements of N.



An undirected graph with 10 and 11 edges.

Figure 4.Undirected Graph

A signed Graph can be defined as a directed graph, G = (N, E) where N is the set of nodes in a network, $E \subseteq N \times N$ is the set of edges such that (u, v) indicates a link between $u \in N$ and $v \in N$
s: E → {+1, −1} assigns the edge weight. If node A is connected to node B as a friend, there should be a directed edge from node A to node B with a trust score of +1. Meanwhile, if A is connected to B as a foe, there should be an edge directed from A to B with a distrust score of −1.

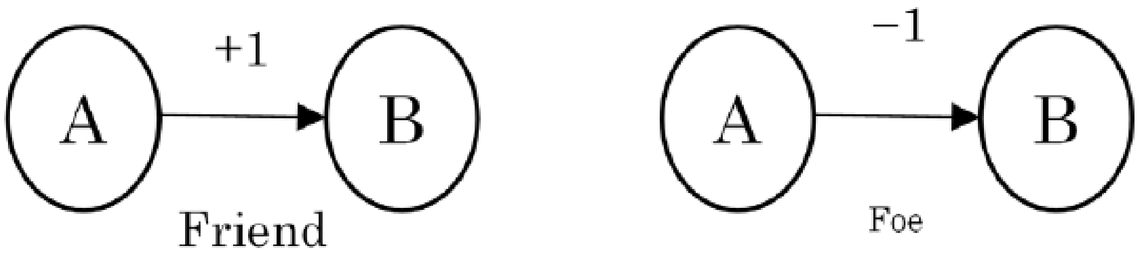


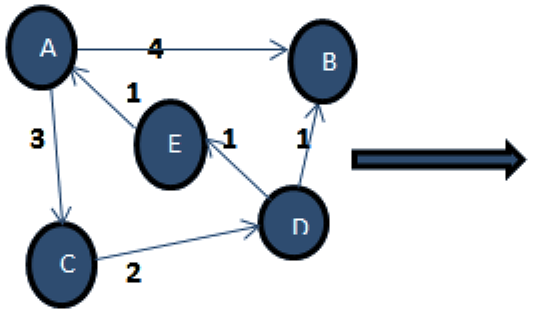
Figure 5.Signed Graph

The most basic kind of a signed network is a homogeneous signed network.
Formally, a homogeneous signed network is represented as a graph with the adjacency matrix
$A \in \{-1, 0, 1\}^{n \times n}$, which denotes the relationships between entities:

$$A_{ij} = \begin{cases} 1, & \text{if } i \text{ \& } j \text{ have positive relationship,} \\ -1, & \text{if } i \text{ \& } j \text{ have negative relationship,} \\ 0, & \text{if relationship between } i \text{ \& } j \text{ is unknown (or missing).} \end{cases}$$

Equation.1
A signed network can also be heterogeneous. In a heterogeneous signed network, there can be more than one kind of entity, and relationships between two, same or different, entities can be positive and negative. For example, in YouTube, there are two kinds of entities—users and videos, and every user can either like or dislike a video. Therefore, the YouTube network can be seen as a bipartite signed network, in which the positive and negative links are between users and videos. **[Chiang et al.2014]**

The most efficient way to **represent a Graph** is by using **an adjacency matrix. It is** a **matrix of size NxN where N is the number of nodes.**



$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 4 & 3 & 0 & 0 \\ B & 0 & 0 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 2 & 0 \\ D & 0 & 1 & 0 & 0 & 1 \\ E & 1 & 0 & 0 & 0 & 0 \end{array}$$

Figure 6.Adjacency Matrix

The above produced from a directed weighted graph and the adjacency matrix M that is corresponding to. The matrix is of size 5x5 as there are 5 nodes in total. The cost from node A to B is 4 and it's given in M[A][B]. Similarly, the cost from one node to itself is zero, so all the diagonal elements would always be zeros.

$$A = (a_{ij})_{n\times n} = \begin{cases} 1, & \text{if } (i,j) \in E, \quad \forall\, i,j \in 1, \ldots \ldots, n \\ 0, & \text{or else} \end{cases}$$

Equation.2

For an undirected graph, A is also symmetrical *A* = *A*^{T}. If the Graphs Edges are weighted, a Weight Matrix is produced in which the element is representing the weight of an edge connecting the nodes *i* and *j*.

The most common process of detecting communities in networks follows a twostep approach:
First, a quality measure (or objective function) needs to be specified, that captures the notion of community structure as groups of nodes with better internal connectivity than external (or more generally, an objective criterion which quantifies the desired properties of a community**.**
Then, using algorithmic techniques, the nodes of the network are assigned to specific communities, optimizing the objective function. The optimization process of the objective functions is usually based on the favored approach of employing heuristics or other approximation techniques**.** [**Vazirgiannis et al. 2013]**

1.4 Thesis Scope

The main object of our work is to **focus on Signed and Directed graphs** and to **propose community detection algorithms that cluster nodes with similar characteristics**, the goal of the procedure is to emerge the **Similarity score as the main and most important KPI** (key point indicator) of Graph Clustering between nodes based on the connectivity pattern they follow including both directionality and polarity of the links which connects them.
We initiate our research with the extraction of negative and positive subgraph of the examined Data Set and then we will attempt to apply Coupling (Reference) and Cocitation as the Methodologies by which Directionality dominates criteria to reach Similarity by calculating the appropriate co-citation and co-reference matrices. The procedure concludes with the similarity Matrix calculation under the theoretical influence of Amsler **[Amsler, 1972]** who pointed out that **Co-citation and Bibliography coupling can be combined** and the appliance of **two clustering algorithms affinity propagation and spectral clustering**, both standalone and combined.
On a practical level we provide a concluded implementation of these proposed algorithms in Python and the datasets that were used to evaluate in practice their precision and certainty.
We display that these two algorithms have the theoretically expected results for both random Generated signed directed graphs and real networks data sets.

2. Community Detection for Signed and Directed Graphs

2.1 Theoretical Basis

A signed graph can be used to model any system containing **two types of antithetical relationships,** such as **like/dislike, for/against, etc.** Such a graph is considered structurally if it can be partitioned into two or more mutually balanced hostile subgroups each having internal solidarity.

Signed social networks can be divided into two categories**: explicit networks and implicit networks. In the explicit networks**, **users can directly tag the polarity (positive or negative) to the relationship between two users.** For example, participants on Epinions can explicitly express trust or distrust of others; users on Slashdot can declare others to be either friends or foes**. In the implicit networks, users do not directly mark the polarities of relationships. However, the relationship polarities can be mined from the interaction data between users.** For example, in Twitter, we have very large-scale complex graphs where the nodes model users and Tweets, while the edges model interactions such as Replies, Retweets, or Mentions. A user may support some of users he follows (positive) and be against the others (negative). **So the relationship of "following" between users in Twitter can have polarity by two simple rules: "The friend of my enemy is my enemy" and "The enemy of my enemy is my friend"**

A positive link in a unsigned network just means a 'relationship', while a positive link in signed networks denotes a 'positive relationship', and a negative one denotes a 'negative relationship'. In a signed social network, the relationships between parties may be political alliances and oppositions, there are positive relationships–friendship, trust and like, as well as negative relationships–hostility, mistrust and dislike. The communities in signed networks are defined as the groups of nodes in which positive links are dense and between which negative links are also dense. A signed graph can be used to model any system containing two types of antithetical relationships, such as like/dislike, for/against etc. Such a graph is considered structurally balanced if it can be partitioned into two or more mutually balanced hostile subgroups each having internal solidarity. [Mendonca,2015]
A signed network G consists of a set of N nodes U = {u1, u2, . . . , uN}, a set of positive links E_p, and a set of negative links E_n. There are two major ways to represent a signed Network G Leskovec **[Leskovec et al. 2010a]** suggests that positive and negative links should be viewed as closely related features in signed networks. One way is to represent both positive and negative links into one adjacency matrix $A \in R^{N\times N}$, where $A_{ij} = 1$, $A_{ij} = -1$ and $A_{ij} = 0$ denote positive, negative, and missing links from u_i to u_j , correspondingly . Then we can proceed to independent networks analysis, we separate a signed network into a network with only positive links and a network with only negative links and then use
two adjacency matrices to represent these two networks, respectively. In particular, it uses $\mathbf{A}p \in R^{N\times N}$ to represent positive links where $\mathbf{A}^p_{ij}= 1$ and $\mathbf{A}^p_{ij} = 0$ denote a positive link and a missing link from u_i to u_j. Similarly, $\mathbf{A}^n_{ij} \in R^{N\times N}$ is used to represent negative links where $\mathbf{A}^n_{ij} = 1$ and $\mathbf{A}^n_{ij} = 0$ denote a negative link and a missing link from u_i to u_j. It is easy to convert one representation into the other with the following rules: $\mathbf{A} = \mathbf{A}^p - \mathbf{A}^n$ and $\mathbf{A}^p = (|\mathbf{A}|+\mathbf{A})/2$ and $\mathbf{A}^n = (|\mathbf{A}|-\mathbf{A})/2$, where $|\mathbf{A}|$ is the component-wise absolute
Value of **A**.

Relationships between users in Social Networks can be also Directed like in Twitter. Directed graphs are used to show asymmetrical relationship between users, while on the other hand Signed Networks are used to describe the Relationship's status between users.

In a Signed Directed network, positive/negative and in/out links determine four different types of connectivity. For example, celebrities in a social network tend to receive more in-links than out-links due to their popularity, while advertisers tend to form many out-links
in order to disseminate information. Because of the asymmetric and personalized nature, symmetric metric space is not adequate to model preferential attachment.

Based on these we can expand our two rule theory with two more conditions, the following four rules:

- **“A friend of my friend is my friend,”**
- **“A friend of my enemy is my enemy,”**
- **“An enemy of my friend is my enemy,”**
- **“An enemy of my enemy is my friend.”**

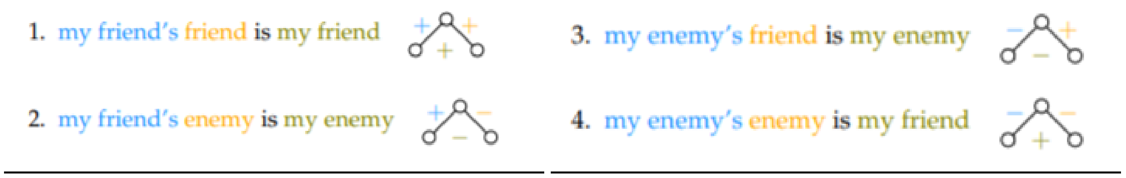

1. my friend’s friend is my friend
+ + +
2. my friend’s enemy is my enemy
+ − −
3. my enemy’s friend is my enemy
− + −
4. my enemy’s enemy is my friend
− − +

Figure 7a.Signed and Directed Social Network Rules

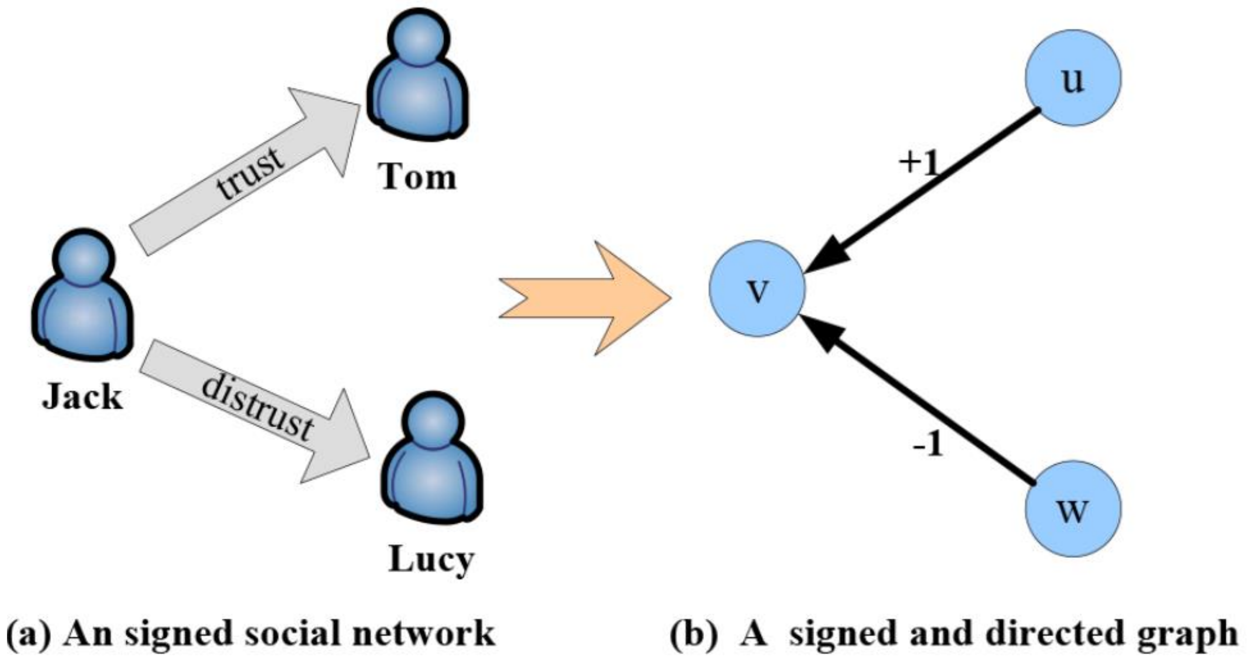


Figure 7b.Signed Social Network **Figure 7c.Signed and Directed Social Network**

So our challenge is to work following **hybrid logic of both the Directed Network Community** and the **Signed Network Community detection Algorithms.**

2.2 Similarity definition and configuration

In order to select the proper Methodology we have to demarcate our Theoretical and Experimental Environment. We must try to formulate a Method which will form in an optimized way clusters in Signed and Directed Network. Our Method will be based on Similarity Criteria and can be seen as a combination of already existing approaches.
A **signed graph** is a special type of graph that is useful for representing sentiment networks.
As we can see in Figure 8 a signed graph is **complete** because all the possible relations between nodes exist .A signed graph is also **directed** because each node is both a source and a destination node for directed **asymmetric ties**.

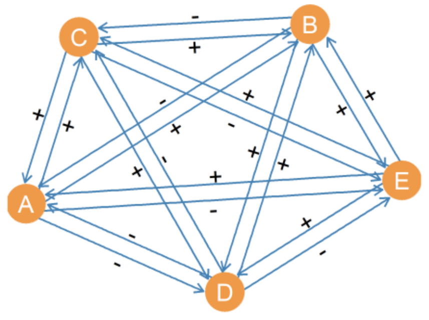


Figure 8.Signed Graph

Signed graphs have a number of unique properties. Some of them are the basis for entire network theories, such as **balance theory** or **status theory**. **Reciprocity**, just as in **weighted graphs**, takes on a different meaning in valued graphs. In the usual graph theory sense, all the relations in Graphs as in **Figure 8** are “reciprocal” because the graph is **complete** and thus all the dyads are **mutual**.
Reciprocity is better defined *as mutual dyads that have the same sentiment going from one node to the other*. In a signed graph mutual dyad, the relationship is reciprocal if both people think that they are friends or both people hate one another. A mutual dyad in a signed graph is non-reciprocal if one person likes the other person, but that person hates the first person. In the graph theoretic sense, reciprocal dyads in a signed graph are those that are connected by two asymmetric edges of the type: either positive or negative. A dyad is non-reciprocal if the two nodes are connected by asymmetric edges of different types.

Nodes *A* and *C* in **Figure 8** have a mutually positive relationship *A* likes *C* and *C* reciprocates by liking *A* back. Accordingly *A* and *D* have a mutually negative relationship, *A* hates *D* and *D* reciprocates by hating *A* back. While the notion of “reciprocity” in negative interactions like hating, or bullying seems **unreasonable**(because people tend to think of reciprocity as a genuine positive thing), **negative reciprocity** makes sense as a driver of human behavior, and may explain important phenomena like the escalation of violence among urban gangs or political parties through our political history. [**Papachristos, et.al 2013].**

But it is not oracular that all relationships will resemble **mutuality,** we must see that we can also identify nodes like **B and C** which shares a non-reciprocal sentiment relation **B likes C** but **C** does not reciprocate the sentiment and on the contrary **C dislikes B**. This brings us to another property of signed graphs, which is that this type of "imbalance" can drive us to think that there is something wrong with this dyadic state, and things must evolve to a new status either **C** will start to hate **B** (because of the received depreciation), or **C** ultimately convinces **B** to like them back.

The idea that there are some states in a signed graph that makes "more sense" than others (because the various sentiment relations are reciprocated) is the core backbone of the "balance" concept which concludes to the formulation of the **balance theory.**

This "Balance Concept" can be summarized in the formulation of the **idea of Similarity.** The Core Idea is to **assign *edge similarity scores* as well as *node similarity scores simultaneously*** that's why we are trying to describe in both levels **(Directionality and Polarity of Edges).**

We introduce a concept of $\mathbf{S_{\{similarity\}}}$ between vertices of **directed graphs.** Let say that G_A and G_B **are two directed graphs** with, respectively**,** n_A and n_B **vertices.**

We define a ***Similarity matrix* S** whose real entry s_{ij} expresses **how similar vertex *j* (in G_A)** is to **vertex *i* (in G_B).**We may define that s_{ij} is their **similarity score.**

The similarity matrix can be obtained as the limit of the normalized even iterates of

$$S_k +1 = BS_kA^T + B^TS_kA$$

Equation.3

Where *A* and *B* are adjacency matrices of the graphs and S_0 **is a matrix whose entries are all equal to 1. In the special case where $G_A = G_B = G$, the matrix S is square and the score s_{ij} is the similarity score between the vertices *i* and *j* of *G*.** We point out that Kleinberg's "hub and authority" method to identify web-pages relevant to a given query can be viewed as a special case of our definition in the case where one of the graphs has two vertices and a unique directed edge between them**. In analogy to Kleinberg, we show that our similarity scores are given by the components of a dominant eigenvector of a non-negative matrix. [Vincent D. Blondel .2003]**

The Polarity level comes into our **Similarity Concept by featuring two disjoints of Edges Positive Links E^+ and Negative Links E^-** so that our Signed Graph is a set of three sets**:**

$$G_s = (E^+, E^-, V)$$

Equation.4

Each "Polarized" Matrix can be treated as the previously analyzed G_A and G_B.

Our Process evolves by the computation of **Positive adjacency matrix A+** and **Negative adjacency matrix A-** and **then with the calculation of their Transpose Matrices.**

Transpose of positive adjacency matrix: $\mathbf{A^T+}$

Transpose of negative adjacency matrix: $\mathbf{A^T-}$

This leads us to the necessity of estimating our normalization factors based on degree which will symmetrize the coupling Matrices based on "balance" and eventually that will lead us to the final and Conclusive Similarity Matrix.

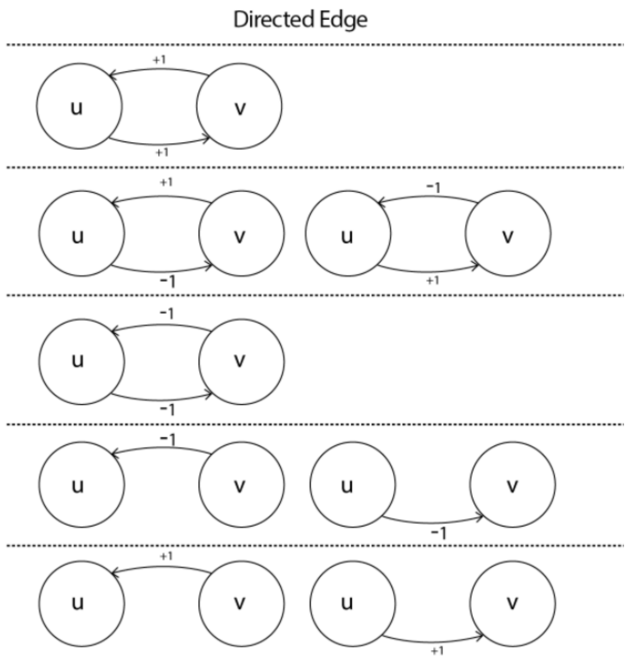


Figure 9.

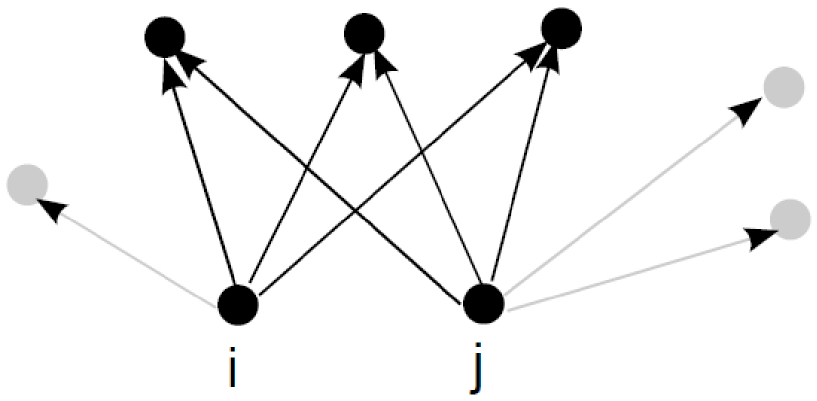
- **out-positive degree:The number of outgoing edges which have positive sign {Out_Degree(+)}**
- **out-negative degree:The number of outgoing edges which have negative sign {Out_Degree(-)}**
- **in-positive degree: The number of incoming edges which have positive sign {In_Degree(+)}**
- **in-negative degree: The number of incoming edges which have negative sign {In_Degree(-)}**

Notations	Descriptions
$\mathbf{A}$	Adjacency matrix
$\mathbf{A}^+(\mathbf{A}^-)$	Adjacency matrix of only positive(negative) links
$\lvert\mathbf{A}\rvert$	Absolute adjacency matrix
$\mathbf{R}$	Node relevance matrix
d_i	Degree of node u_i
$d_i^{in}(d_i^{out})$	Indegree (Outdegree) of node u_i
$d_i^{in+}(d_i^{out+})$	Indegree (Outdegree) of positive links of node u_i
$d_i^{in-}(d_i^{out-})$	Indegree (Outdegree) of negative links of node u_i
N_i	Set of neighbors for node u_i
$N_i^{in}(N_i^{out})$	Set of incoming (outgoing) neighbors for node u_i
$N_i^+(N_i^-)$	Set of positive (negative) neighbors for node u_i
$\mathbf{X}_{ij}$	the (i,j) entry of the matrix $\mathbf{X}$

2.3 Reference and Citation

Coupling (Reference) and Cocitation are the Methodologies by which Directionality emerges as criteria to Similarity.

The Reference or ***bibliographic coupling*** of two nodes i and j in a directed network is the number of other nodes to which both i and j point



The bibliographic coupling B_{ij} of i and j is

$$B_{ij} = \sum_{k=1}^{n} A_{ki}A_{kj} = \sum_{k=1}^{n} A^T_{ik}A_{kj}$$

$$\mathbf{B} = \mathbf{A}^T\mathbf{A}$$

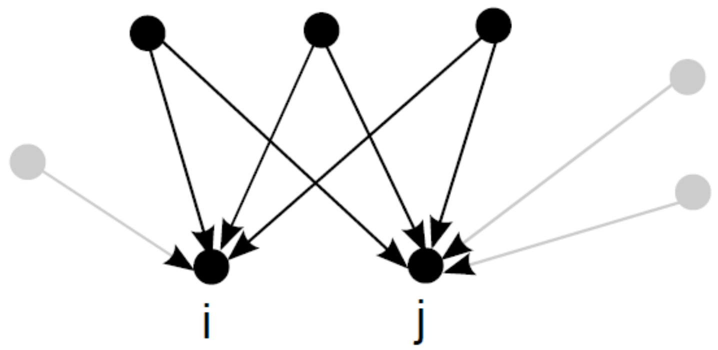
Figure 10.

Where:
$\mathbf{B}$ is a $n \times n$ matrix
$\mathbf{A}$ is the Adjacency Matrix of the initial Graph (Network)
It is symmetric since

$\mathbf{B}^T = (\mathbf{A}^T\mathbf{A}) = \mathbf{A}^T\mathbf{A} = B$ **Equation.5**

We define *bibliographic coupling network* in which there is a link if $Bij > 0$ for $i \neq j$

On the other hand the ***Cocitation* of two nodes i and j** in a directed network is the number of nodes **that point to both i and j**



The cocitation C_{ij} of i and j is

$$C_{ij} = \sum_{k=1}^{n} A_{ik}A_{jk} = \sum_{k=1}^{n} A_{ik}A_{kj}^{T}$$

$$\mathbf{C} = \mathbf{A}\mathbf{A}^{T}$$

Figure 11.

Where:
C is a $n \times n$ matrix
A is the Adjacency Matrix of the initial Graph (Network)
It is symmetric since
$\mathbf{C}^{T} = (\mathbf{A}\mathbf{A}^{T}) = \mathbf{A}\mathbf{A}^{T} = C$ **Equation.6**

We define *cocitation network* in which there is a link if $Cij > 0$ for $i \neq j$

2.3.1 Co-citation and Co-reference Matrices

By processing the above theoretical background we are forming **Bibliographic Coupling B Matrix for both Positive** and **Negative ties normalized with out-positive degree (**number of outgoing edges, with positive sign**)** and **out-negative degree (**number of outgoing edges, with negative sign**) in an attempt to factorize the connection between Directionality and Polarity.**

Positive Co-Reference Matrix (B+):

$$B^{+} = A^{+T} A^{+} \qquad \text{Equation.7a}$$

Negative Co-Reference Matrix (B-):

$$B^{-} = A^{-T} A^{-} \qquad \text{Equation.7b}$$

Where A^{+} and A^{-} are the Adjacency Matrices of positive and negative links (ties) exclusively and we formulate them as Positive Adjacency Matrix (A+) & Negative Adjacency Matrix (A-)

Likewise we are forming **Cocitation C Matrix** for both **Positive** and **Negative ties normalized with in-positive degree (**number of incoming edges, with positive sign**)** and **in-negative degree (**number of incoming edges, with negative sign**) in an attempt to factorize the connection between Directionality and Polarity.**

Positive Co-Citation Matrix (C+):

$$C^{+} = A^{+} A^{+T} \qquad \text{Equation.8a}$$

Negative Co-Citation Matrix (C-):

$$C^{-} = A^{-} A^{-T} \qquad \text{Equation.8b}$$

Where A^{+} and A^{-} are the Adjacency Matrices of exclusively positive and negative links (ties) and we formulate them as Positive Adjacency Matrix (A+) & Negative Adjacency Matrix (A-).

2.4 Similarity Measure – Similarity Matrices

Our next objective was to connect Citation and Reference of two nodes. **Our main influence was the work of Amsler [Amsler,1972] who pointed out that Co-citation and Bibliography coupling can be combined.** The relationship of co-citation between two nodes can be used to measure the number of common in-links of two nodes, whereas bibliographic coupling can be used to measure the number of common out-links of two nodes (Figure 12.).

According to Amsler, two nodes A and B are related if:

- **A and B are cited by the same node.**
- **A and B cite the same node.**
- **A cites a third node C that cites B.**

Our goal is to delimit the similarity measure, as the function that evaluates the similarity between two nodes of an examined Graph.

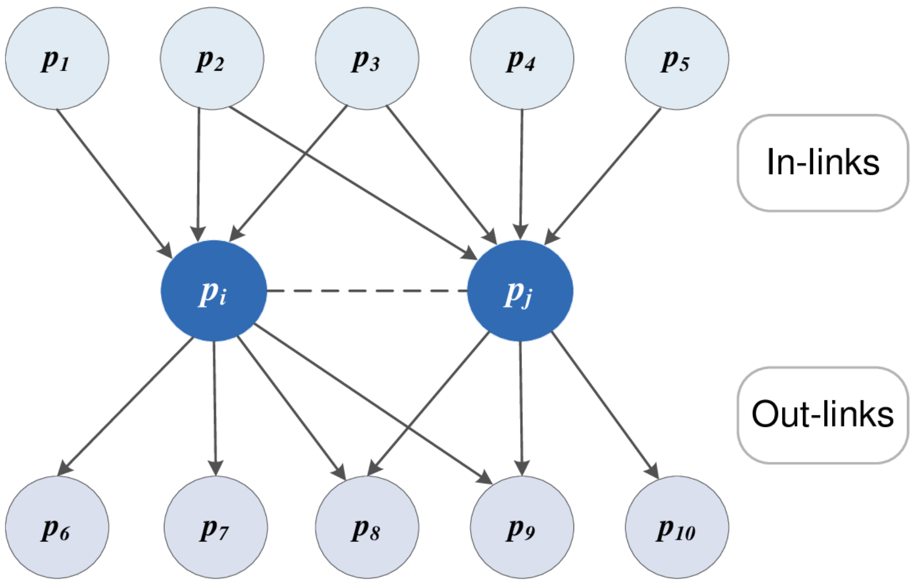
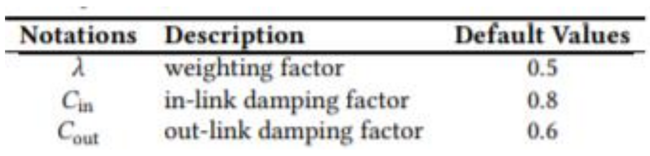


Figure 12.

Studying Co-citation (Small 1973), Bibliographic coupling (Kessler 1963) and Amsler (Amsler 1972) and the work on the **P (penetrating) – Rank formula by (WEIREN YU 2019),** we can express **Amsler Similarity** in our case as ***Amsler Similarity Modification***:

S = λ * Cin * Sim_in + (1-λ) * Cout * Sim_out **Equation.9**

where λ is the weight factor balancing the importance of in- and out-links; and Cin and Cout are in-and out-link damping factors.



Notations	Description	Default Values
λ	weighting factor	0.5
C_{in}	in-link damping factor	0.8
C_{out}	out-link damping factor	0.6

Figure 13.

Where Sim_in describes the incoming similarity as the sum of C^{+} , C^{-} co-citation matrices in reference with the in-degrees of nodes. Similarly, Sim_out describes outgoing similarity as the sum of B^{+} , B^{-} co-reference matrices in reference with the out-degrees of nodes.

On practical level in-similarity and out-similarity are frequency matrices. So Sim_in, Sim_out are the Similarity Matrices based on incoming and outgoing links which we can formulate with the above methodology:

"Incoming Link Similarity Matrix" = Positive Co-Citation Matrix (C+) + Negative Co-Citation Matrix (C-)

$$S_{in}(i,j) = (C^{+}[i,j] + C^{-}[i,j])$$

Equation.10a

"Outgoing Link Similarity Matrix" = Positive Co-Reference Matrix (B+) + Negative Co-Reference Matrix (B-)

$$S_{out}(i,j) = (B^{+}[i,j] + B^{-}[i,j])$$

Equation.10b

Whereas formulated in Equations 7 & 8 respectively:

Positive Co-Citation Matrix (C+):

$C^{+} = A^{+} A^{+T}$

Negative Co-Citation Matrix (C-):

$C^{-} = A^{-} A^{-T}$

And

Positive Co-Reference Matrix (B+):

$B^{+} = A^{+T} A^{+}$

Negative Co-Reference Matrix (B-):

$B^{-} = A^{-T} A^{-}$

Where A^{+} and A^{-} are the Adj Matrices of only positive and negative links (ties) and we can call these Positive Adjacency Matrix (A^{+}) & Negative Adjacency Matrix (A^{-}) respectively.

Finally we compute the Total Similarity Matrix as the normalized sum of “Incoming Link Similarity Matrix” and "Outgoing Link Similarity Matrix”:
“Total Similarity Matrix” = “Incoming Link Similarity Matrix” + "Outgoing Link Similarity Matrix” * Normalization Factor.

$$\mathit{Similarity}[i, j] = [S_{in}(i,j) + S_{out}(i,j)]$$

Equation.11a

We conclude to a normalization factor which is Degree Dominated where **In-degree is the number of head endpoints adjacent to a node and Out-degree is the number of tail endpoints adjacent to a node**. **A node has a large impact in a network when it has a large number of neighbors.** The importance of a node could be reflected **by the node degree,** which is the sum of the **positive degree** and the **absolute value of the negative degree.**

$$deg(v) = deg_P(v) + |deg_N(v)|$$

Equation.11b

Where $deg(v)$ represents the node degree, and $deg_{P}(v)$ and $deg_{N}(v)$ are the positive degree and the negative degree of the node, respectively. It is legitimate to assume that **if the degree of a node is larger than those** of its neighbors**, then the node is more likely to be a center of a community** than its neighbors.
Based on the above we formulated the Normalization Factor as the inverse maximum of the node degrees, i.e., Deg(i) and Deg(j) which denote the total sum of edges for nodes i and j respectively.

$$\mathit{Similarity}[i, j] = [S_{in}(i,j) + S_{out}(i,j)]/\max(Deg(i), Deg(j))$$

Equation.12

During our research we experimented allot with the Normalization issue and we worked in parallel with 2 major cases, **normalizing our Similarity at the final stage of its formation** or **normalizing the main components** that it consists of**.**

With that in mind in an alternate scenario that we will also represent in our experiments evaluation sector we normalized **Positive Co-Reference Matrix (B+), Negative Co-Reference Matrix (B-), Positive Co-Citation Matrix (C+)** and **Negative Co-Citation Matrix (C-)** using as benchmark **the out-positive degree** (number of outgoing edges, which have positive sign), **out-negative degree** (number of outgoing edges, which have negative sign), **in-positive degree** (number of incoming edges, which have positive sign) **and in-negative degree** (number of incoming edges, which have negative sign). **[Zadeh, Goel 2012]**

Positive Co-Reference Matrix (B+) = Positive Co-Reference Matrix (B+)/ (number of outgoing edges, which have positive sign)

Equation.13a

Negative Co-Reference Matrix (B-) = Negative Co-Reference Matrix (B-)/ (number of outgoing edges, which have negative sign)

Equation.13b

Positive Co-Citation Matrix (C+) = Positive Co-Citation Matrix (C+)/ (number of incoming edges, which have positive sign)

Equation.13c

Negative Co-Citation Matrix (C-) = Negative Co-Citation Matrix (C-)/ (number of incoming edges, which have negative sign)

Equation.13d

In any case our main tool don't change and that is **Similarity, incoming similarity** and **outgoing similarity** signify the number of common edges between nodes i and j based on the in-coming and out-going links respectively**. So their sum can only mean the total number of edges these two nodes have in common.** If these nodes have in common all their edges, that summarization is always equal to the degree Deg(i) and the degree Deg(j), and similarity will always reach one (1) which is its maximum value. **So whenever we achieve similarity score between two nodes equal to 1 it's an indication that these two nodes have 100% similarity.**

3. Proposed Approach for Community Detection in Signed and Directed Graphs

3.1 Community detection - Graph clustering algorithms

Clustering , is a key element of data analysis and plays a significant role regarding the **data structure partitioning** in a broad range of applications, from segmenting customers for more effective advertising to Telecommunication Networks optimization**. Distance and similarity** are the basis of the most important clustering algorithms in our case **.Distance** in order to recognize the **relationship** among data and **similarity** when you focus on certain dealing **data features.**
So the implementation of **specific algorithms** makes **Clustering** possible**. There is no magic wand, a panacea algorithm that can solve every problem that's why the acknowledgement of the problem we are facing is critical. We must fully understand the problem we want to solve, pick after rigorous research the appropriate criteria (for example in our case Similarity) in order to choose cautiously the eligible algorithm.** Implementing a certain clustering algorithm crudely means that we're going to give the algorithm a lot of input data with no labels and let it find these groupings in the data that is designed to find. Those groupings are the partitions widely called *clusters*.

Partition's meaning, value and structure vary and we can assume that they are dependent on the type of the Graph which is the research's object. In our case we study about **Signed and Directed** graphs so we must **combine criteria found in Signed and Directed** networks separately and combined in order **to give materiality to the concept of Similarity**. Effective partitions for **Signed** Networks tend to have more **positive links** within communities and more **negative links** between communities while accordingly in a **Directed** network the **in- and out-connectivity** of the nodes contain information about the role of each node making a distinction between "leaders" and "followers", according to the predominance of their in- or out-degree. So in a brute manor in **Signed and Directed** Networks we are looking for clusters containing nodes which belong to a wide spectrum starting from positive leaders like Mahatma Gandhi and ending with Scapegoats like Judas Iscariot.

Given the fact that we compute the Similarity Matrix of the initial Graph (Network) in order to reach the Similarity measure **we chose two algorithms to evaluate our Methodology. We chose algorithms that allow us to work directly and indirectly with the evaluated Similarity Matrix Affinity Propagation and Spectral Clustering.**

3.2 Affinity Propagation Clustering

3.2.1 Affinity Propagation Algorithm

Affinity propagation (AP) was first published in 2007 by Brendan Frey and Delbert Dueck in Science, is an algorithm that identifies centers of clusters, also called *exemplars* to form its clusters around them. This algorithm simultaneously considers all the points in the set as probable candidates to become *exemplars* and propagates repeatedly exchange of messages between the nodes of the Graph until the emergence of good exemplars and clusters **[Frey and Dueck.2007].**

In contrast to other traditional clustering methods, Affinity Propagation does not require any predefined information about the under process network which means that you do **not require to specify the number of clusters**.

In an attempt to depict the Affinity Propagation mechanism in plain English we can detail that each data point sends messages to all other points informing its targets of each target's relative attractiveness to the sender. Each target then responds to all senders with a reply informing each sender of its availability to associate with the sender, given the attractiveness of the messages that it has received from all other senders. Senders reply to the targets with messages informing each target of the target's revised relative attractiveness to the sender, given the availability messages it has received from all targets. The message-passing procedure proceeds until a consensus is reached. **Once the sender is associated with one of its targets, that target becomes the point's exemplar. All points with the same exemplar are placed in the same cluster. Affinity Propagation let us work directly with the Similarity Matrix**

Let's see how all of this gets formulated:

There are two kinds of messages exchanged in every iteration of the AP algorithm**: Responsibility and Availability.**

The responsibility message r(i , k), quantifies how well-suited element k is, to be an exemplar for element i , taking into account the nearest contender k' to be an exemplar for i.

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k} \{a(i,k') + s(i,k')\}$$

Equation.14

The similarity $s(i,k)$ indicates how well the data point with index *k* is suited to be the exemplar for data point *i*.

The Responsibility matrix R is initialized with all of its elements set to zero, r(i , k) can be thought of as relative similarity between i and k. It makes countable how similar is i to k, compared to some k', taking into account the availability of k'. In terms of Matrix elements i refers to the row and *k* refers to the column of the related matrix.

The availability message a(i ,k), sent from candidate exemplar point k to point i, reflects how appropriate it would be for point i to choose point k as its exemplar. A separate equation is used for updating the elements on the diagonal of the **availability matrix A** rather than the elements off the diagonal of the availability matrix.

$$a(k,k) \leftarrow \sum_{i' \text{ such that } i' \neq k} \max\{0, r(i',k)\},$$

Equation.15

The above equation is indicating that if you want to understand each iteration phase you have to sum all the values above 0 along the column except for the row whose value is equal to the column in question. In each iteration, the responsibility matrix is primarily updated using the availability matrix of the previous iteration. Then the availability matrix is updated. Availability is self-responsibility of k plus the positive responsibilities of k towards elements other than i. This procedure may be terminated after a fixed number of iterations, after changes in the values obtained fall below a threshold, or after the values stay constant for some number of iterations. (Figure 14.)

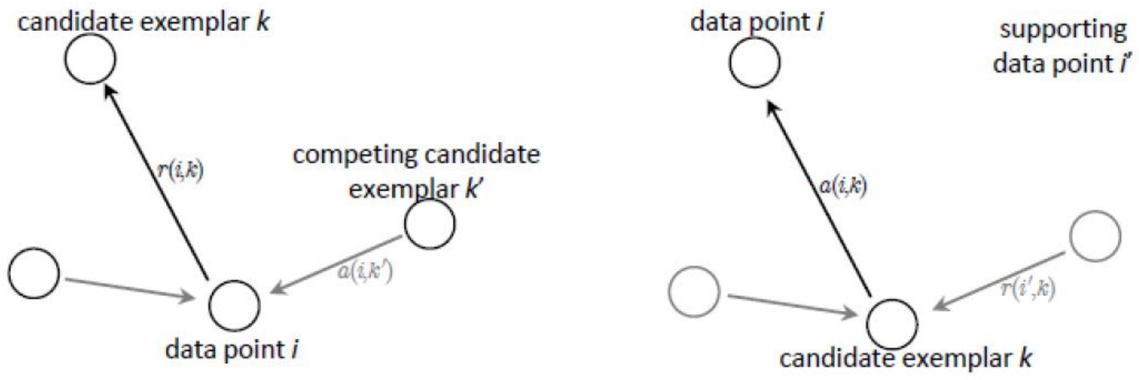


Figure 14. Exemplars procedure visualization

Finally the **Criterion Matrix** is calculated after the updating is terminated**. Criterion matrix C is the sum of Responsibility matrix R and Availability matrix A.**

$$c(i,k) \leftarrow r(i,k) + a(i,k)$$

Equation.16

The element with the highest criterion value in each row would be designated to be an exemplar. Elements corresponding to the rows which share the same exemplar are clustered together.

3.2.2 Affinity Propagation Parameters Tuning

There are certain parameters in Affinity propagation algorithm that can be tinkered and eventually tuned in order to achieve maximum performance.

These are some parameters that we focused and experimented on:

- **Damping:** Computing responsibilities and availabilities according to simple update rules will often lead to oscillations caused by "overshooting" the solution, so the responsibility and availability messages are "damped" like this: msg_new = (damping factor)(msg_old) + (1-damping factor)(msg_new)
- **Iteration:** In affinity propagation, a single iteration involves computing all responsibility messages based on the current availability messages, the input similarities and the input preferences, and then computing all availability messages based on the responsibility messages, which were just updated.
- **max_iter:** The maximum number of iterations. When the number is reached iterations mechanism halts. Default value is 200.
- **convergence_iter:** Affinity propagation iteratively computes responsibilities and availabilities, if decisions for the exemplars and the cluster boundaries are unchanged the algorithm terminates and the number of convergence iterations is reached. Default value is 15
- **Affinity**: Defines whether the distance metric we will choose to use is going to be Euclidean or Precomputed. In cases like ours where our input is Similarity Matrix affinity is set to Precomputed.

- **Preference**: This preference value indicates how strongly a data point thinks itself should be an exemplar. The number of exemplars, ie of clusters, is influenced by the input preference value. Default value is the median of the similarity matrix so if we want Affinity Propagation to be less eager in splitting clusters we must set the preference value lower.

3.3 Spectral Clustering

3.3.1 Spectral Clustering Methodology

Spectral clustering [Luxburg.2006] methods are based on different types of input matrixes, such as the adjacency matrix, the standard Laplacian matrix, and the normalized Laplacian matrix. **The standard Laplacian matrix is defined as *L = D - A* ,** where *A* is adjacency matrix and D is a diagonal matrix with elements D_{ii} being the degree of the i_{th} node.
In the case of directed graphs the elements of L are given by:

$$L_{i,j} := \begin{cases} \deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

where $\deg(v_i)$ is the degree of the vertex i.

Equation.17

The diagonal elements l_{ij} of L are therefore equal the degree of node v_i and off-diagonal elements l_{ij} are -1 if vertex v_i is adjacent to v_j and 0 otherwise.
The normalized Laplacian matrix is defined as:

$$NL = D^{-1/2} L D^{1/2} = D^{-1/2} (D-A) D^{1/2} = I - D^{-1/2} A D^{1/2}$$

Equation.18

Where *A* is adjacency matrix and D is a diagonal matrix with elements D_{ii} being the degree of the i_{th} node.
And its elements are similarly defined as:

$$L_{i,j} := \begin{cases} 1 & \text{if } i = j \text{ and } \deg(v_i) \neq 0 \\ \frac{1}{\sqrt{\deg(v_i)\deg(v_j)}} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise.} \end{cases}$$

Equation.19

Almost all above matrixes are constructed with the adjacency matrix and diagonal matrix of networks. These matrixes are able to illustrate only the local relationship between a node and its direct neighbors. This is why; the community number must be set in advance for the spectral clustering method based on standard Laplacian matrix. The normalized Laplacian matrix can solve this problem to some extent, because it has nontrivial eigenvalues close to the biggest eigenvalue 1. The eigenvector elements corresponding to these eigenvalues present ladder distribution. The proper community number of communities can be estimated by the ladders. However, when the community structure of network is not clear, the eigenvector elements cannot show obvious ladder distribution but an approximately continuous curve. In this case, we cannot get the proper community number from the ladder

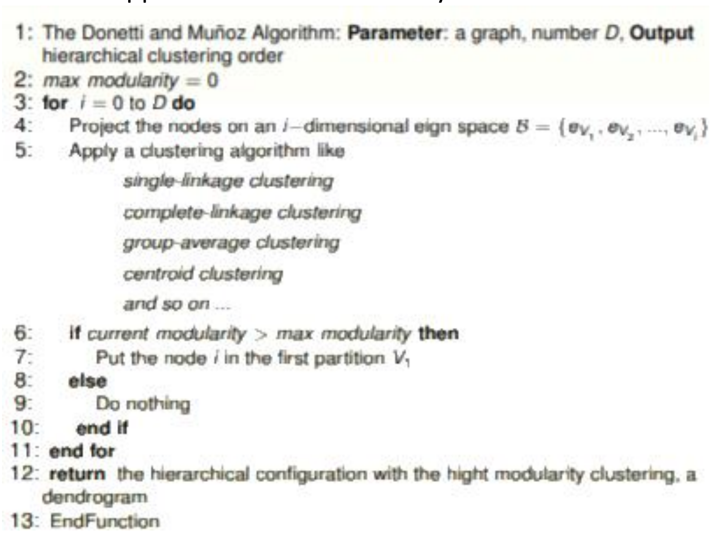
distribution of eigenvector elements. One way is to perform Eigenvalue decomposition we extract the eigenvector corresponding to the 2nd lower eigenvalue which is named algebraic connectivity.
The eigenvector corresponding to the algebraic connectivity is named after Fiedler (Fiedler Vector) and then we perform Spectral Clustering. The smallest non-zero eigenvalue is called spectral gap and the corresponding eigenvector is used for the task of spectral clustering.

The resulting Modularity is Q

$$Q = \sum^{n_c} \left[\frac{l_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right]$$

Equation.20

A similar approach can be reached by the Donetti & Muñoz Algorithm **[Donetti & Muñoz .2004]**



```
1: The Donetti and Muñoz Algorithm: **Parameter**: a graph, number $D$, **Output**
   hierarchical clustering order
2: *max modularity* = 0
3: for $i = 0$ to $D$ do
4:     Project the nodes on an $i$-dimensional eign space $B = \{ev_{1}, ev_{2}, \dots, ev_{i}\}$
5:     Apply a clustering algorithm like
           single-linkage clustering
           complete-linkage clustering
           group-average clustering
           centroid clustering
           and so on ...
6:     if current modularity > max modularity then
7:         Put the node $i$ in the first partition $V_1$
8:     else
9:         Do nothing
10:    end if
11: end for
12: return the hierarchical configuration with the hight modularity clustering, a
    dendrogram
13: EndFunction
```

The following Steps lead to successful partioning:
-First few eigenvectors of Laplacian are computed (let’s say k)
-Eigenvector components used to represent vertices as points in k dimensional Euclidean space
-Hierarchical clustering used to group points
-Modularity is used to pick best partition of resulting dendrogram

3.3.2 Spectral Clustering in Signed & Directed Graphs

Spectral clustering is an EDA technique that reduces complex multidimensional datasets into clusters of similar data in rarer dimensions**. The main outline is to cluster the all spectrum of unorganized data points into multiple groups based upon their uniqueness and is one of the most popular forms of multivariate statistical analysis.**

'Spectral Clustering uses the connectivity approach to clustering', wherein communities of nodes (i.e. data points) that are connected or immediately next to each other are identified in a graph. The nodes are then mapped to a low-dimensional space that can be easily segregated to form clusters. Spectral Clustering uses information from the eigenvalues (spectrum) of special matrices (i.e. Affinity Matrix, Degree Matrix and Laplacian Matrix) derived from the graph or the data set.

In Spectral Clustering we can't work directly on the Similarity Matrix therefore we must create a Similarity Graph from the Similarity Matrix G = (V,E). Each vertex v_i in this graph represents a data point x_i. Two vertices are connected if the similarity s_{ij} between the corresponding data points x_i and x_j is positive or larger than a certain threshold, and the edge is weighted by s_{ij}. The problem of clustering can now be reformulated using the similarity graph: we want to find a partition of the graph such that the edges between different groups have very low weights (which means that points in different clusters are dissimilar from each other) and the edges within a group have high weights (which means that points within the same cluster are similar to each other).

So we can work accordingly like in any other Graph.

We begin by performing the Eigen-decomposition procedure on our previously evaluated Similarity matrix, following these steps:

1. Construct the normalized Similarity matrix: **L = D−1/2ADˆ −1/2**.
2. Find the eigenvalues and their associated Eigen vectors
3. Identify the maximum gap which corresponds to the number of clusters by Eigen-gap heuristic. And there we compute the optimal number of Spectral Clustering clusters. One major sign to evaluate our methodology is Silhouette Coefficient Value, when it is near 1.0 it indicates that the sample is far away from the neighboring clusters. A value of 0.0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values indicates that those samples might have been assigned to the wrong cluster.

4. Experimental Evaluation

4.1 Environment and Tools

We will proceed to a thorough evaluation of our experimental procedure initially by analyzing the tools we used and estimating the functionality of the process stages.

We selected **Python 3.6** for the implementation as our programming language. Upon this we worked with specific libraries:

- **Networkx** for the Graph Modeling
- **Numpy** for the matrices calculations
- **Scikit-Learn (sklearn)** for the clustering algorithms
- **csgraph** for the Eigen Decomposition
- **Matplotlib** for creating two-dimensional plots
- **Graph Objects** for creating three-dimensional plots

Python 3.6

We concluded to this programming language after we evaluated the advantages.

We were in a dilemma between Python and R and although a simple research on the net will show up many R libraries for Signed and Directed Networks, Python has efficient high-level data structures and a simple but effective approach to object-oriented programming. Contrary to programming languages that are VM oriented (virtual machines) and consume a lot of memory resources, python doesn't have any of these requirements and thus its suitable for running in devices such as a medium specification laptop. Last but not least the existence of a large and dynamic programming community with mature developers publishing their work was the source of information essential to overleap many coding difficulties.

Networkx

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. This package provides data structures which it makes it able to represent many types of networks, or graphs such as simple graphs and directed graphs. The nodes in NetworkX graphs can be any Python object and edges may contain inconsistent data. This particular feature makes NetworkX perfect in representing networks from a variety of scientific fields. In addition to this many graph algorithms are implemented able to calculate network properties and structure measures, such as shortest path, centrality, clustering, and degree distribution. NetworkX can read and write various graph formats, and provides generators for many classic graphs and graph models, such as the Erdos-Renyi, Small World, and Barabasi-Albert models.

Numpy

Numpy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array. Numeric was the ancestor of NumPy and it was developed by Jim Hugunin. Another package Numarray with additional functionalities was developed in parallel. In 2005, Travis Oliphant formulated NumPy package by incorporating those supplemental features of Numarray into Numeric package.

A scientist who is working with Numpy a can perform Mathematical and logical operations on arrays, Fourier transforms and routines for shape manipulation and operations related to linear algebra.

NumPy is often used in collaboration with packages such as SciPy (Scientific Python) and Matplotlib (plotting library). This collaboration can be traced widely in this Thesis.
This combination is widely used as a replacement for MatLab.

Matplotlib

Matplotlib is a multi-platform data visualization package built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. It is specially designed to visualize 2D plots of arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also. The utilization of Matplotlib package allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.
Let's see some examples of the usage of Matplotlib package in different stages of this Thesis:

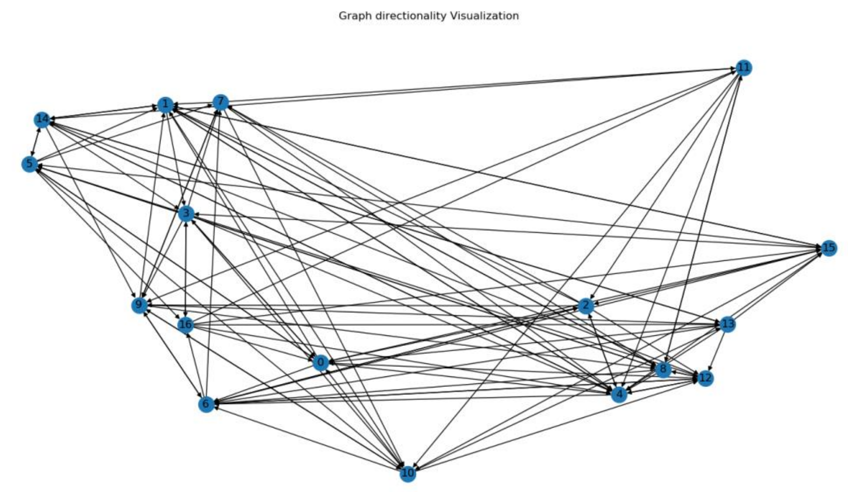


Figure 15. Graph Directionality Visualization

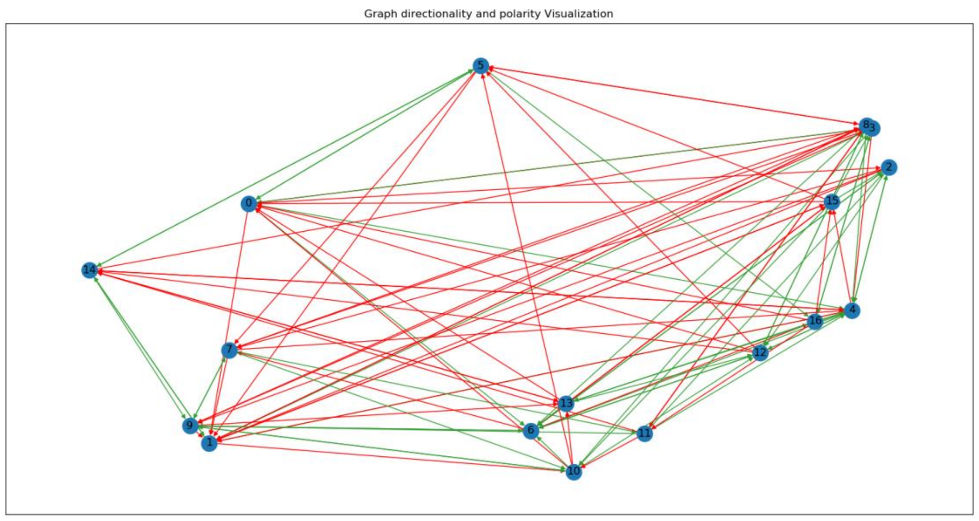


Figure 16. Graph Directionality & Polarity Visualization (green edges manifest positive connections while red edges manifest negative connections)

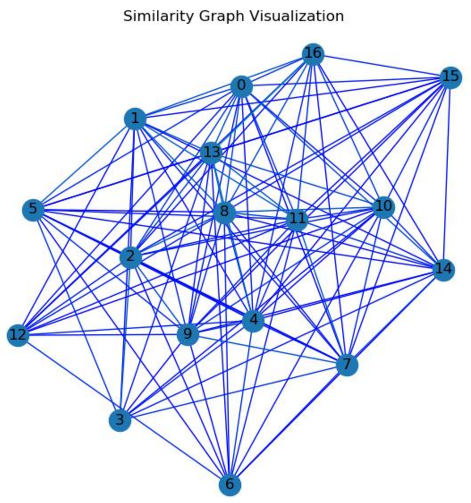


Figure 17. Similarity Graph Visualization

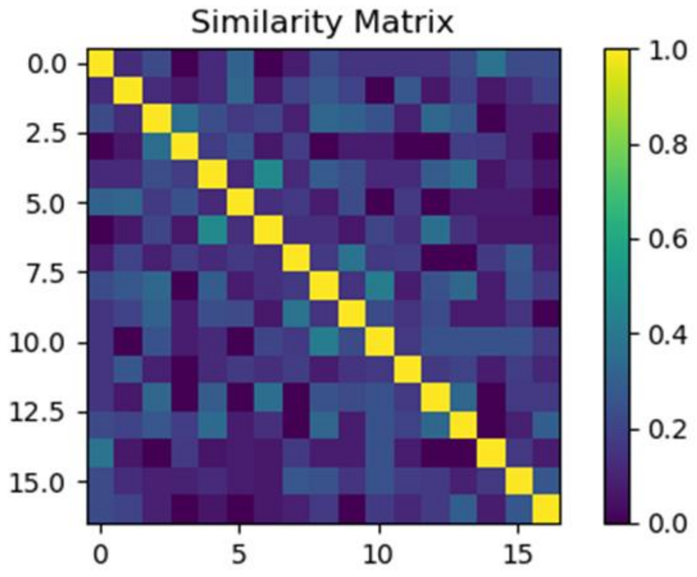


Figure 18. Similarity Matrix Heatmap

<u>**Scikit-Learn (sklearn)**</u>

Scikit-learn was initially developed by David Cournapeau as a Google summer of code project in 2007. Later Matthieu Brucher joined the project and started to use it as a part of his thesis work. In 2010 INRIA got involved and the first public release (v0.1 beta) was published in late January 2010.

Scikit-learn exposes a wide variety of machine learning algorithms, both supervised and unsupervised, using a consistent, task-oriented interface, thus enabling easy comparison of methods for a given application. Since the package is built upon the SciPy (Scientific Python) ecosystem, it can easily be integrated into applications outside the traditional range of statistical data analysis.

Some popular groups of models provided by Scikit-learn include *(Jason Brownlee* PhD, Python Machine Learning):

- **Clustering** is used for grouping unlabeled data, e.g. KMeans.
- **Cross-validation** is used for performance estimation of supervised models on the unseen data.
- **Datasets** are used for test datasets and for the generation of datasets with particular properties for the investigation of model behavior.
- **Dimensionality Reduction** is used for reducing the number of attributes in data for summarization, visualization and feature selection such as principal component analysis.
- **Ensemble methods** are used for combining the predictions of multiple supervised models.
- **Feature extraction** that is used for defining attributes in image and text data.

- **Feature selection** is used for identifying meaningful attributes from which supervised models are created.
- **Parameter Tuning** is used for getting the most out of supervised models.
- **Manifold Learning** that is used for summarizing and depicting complex multi-dimensional data.
- **Supervised Models** is a vast array not limited to generalize linear models, discriminant analysis, naive Bayes, lazy methods, neural networks, support vector machines and decision trees.

Csgraph

Csgraph stands for Compressed Sparse Graph, which focuses on Fast graph algorithms based on sparse matrix representations. The Csgraph module is a very important feature when dealing with graphs in SciPy. We can perform the functions on sparse matrices. We then concert those matrices into sparse graphs. It provides functions to represent the graph in different forms. It also consists of features to help traverse the matrices either directly or indirectly. In our case it was essential in order to perform Eigen decomposition during Spectral Clustering phase were it was not possible to work directly with the similarity matrix

Graph Objects

The **plotly Graph Objects** module provides an automatically-generated hierarchy of classes called "graph objects" that may be used to represent figures, with a top-level class figure.
Graph objects have several benefits compared to plain Python dictionaries.

- Graph objects provide precise data validation. If you provide an invalid property name or an invalid property value as the key to a graph object, an exception will be raised with a helpful error message describing the problem. This is not the case if you use plain Python dictionaries and lists to build your figures.
- Graph objects contain descriptions of each valid property as Python docstrings, with a full API reference available. You can use these docstrings in the development environment of your choice to learn about the available properties as an alternative to consulting the online Full Reference.
- Properties of graph objects can be accessed using both dictionary-style key lookup (e.g. fig ["layout"]) or class-style property access (e.g. fig.layout).
- Graph objects support higher-level convenience functions for making updates to already constructed figures (.update_layout (), .add_trace () etc.) as described below.
- Graph object constructors and update methods accept "magic underscores" (e.g. go.Figure (layout_title_text="The Title") rather than dict (layout=dict (title=dict (text="The Title")))) for more compact code, as described below.
- Graph objects support attached rendering (.show ()) and exporting functions (.write_image ()) that automatically invoke the appropriate functions from the plotly.io module.

In this Thesis we used Graph Objects to visualize Graphs and their clusters in a 3D format. It also gave us the ability to visualize rotating models.

Let's see some examples of the usage of Graph Objects package in different stages of this Thesis:

In Figure 19 we can see a simple 3D visualization of our examined Graph whereas in Figures 20a and 20b we can see the 3D visualization of Affinity Propagation (AF) and Spectral Clustering (SC) respectively.

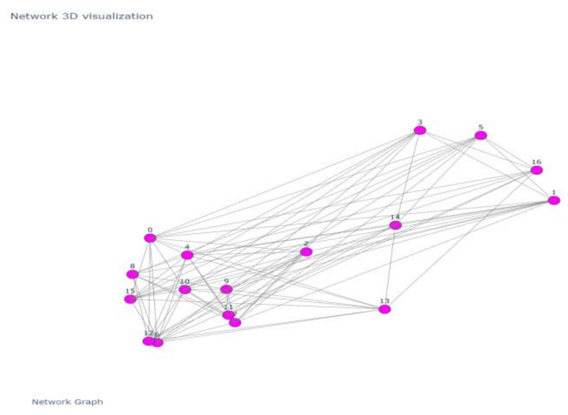
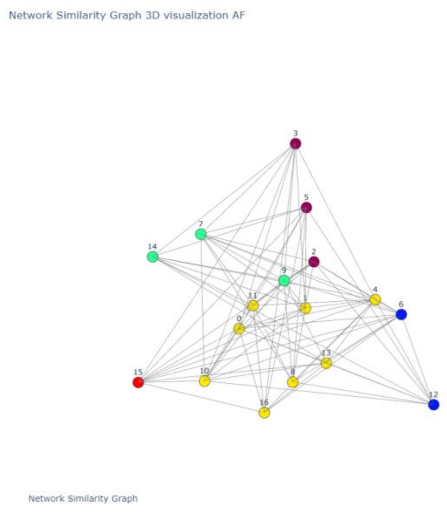


Figure 19.



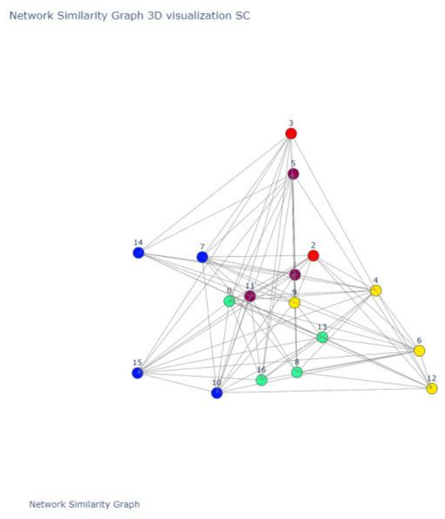


Figure 20a. **Figure 20b.**

4.2 Datasets (Real World and Generated)

We have two alternatives of workflow to check our algorithm, either to work with real world datasets or with technically generated Synthetic data.

Firstly we used a series of real world datasets, starting with **Bitcoin Alpha trust weighted signed network** this is a who-trusts-whom network of people who trade using Bitcoin on a platform called Bitcoin Alpha. Since Bitcoin users are anonymous, there is a need to maintain a record of users' reputation to prevent transactions with fraudulent and risky users. Members of Bitcoin Alpha rate other members in a scale of -10 (total distrust) to +10 (total trust) in steps of 1. This dataset contains 3.783 traders /nodes forming 24.186 trading relationships. It is a Signed and Directed network in which nodes

represent Traders and edges represent transactions. Transactions can be either positive, negative or neutral on the field of credibility of the trading partners. We run our algorithm but due to technical limitations of our CPU, it converged up to a certain limited number of nodes and it stopped processing.
We then downgraded our effort to smaller datasets we found on Snap i.e. Fraternity (regarding the relations between members of US Universities Frat Houses) and Tribes (regarding the relations between tribe members) and our Methodology proved functional with the expected results.
But the loose end (working in large Datasets) remained and we should find a way to tie it up. We turned to Synthetic data as a solution of forming data sets with widely calibrated parameters so we can show the progressively validation of our Methodology.
A NetworkX generator that produces random, directed, signed graphs proved was able to be constructed in only 2 lines of code in python.
We used the gnp_random_graph() function which returns a random graph, also known as Erdos-Renyi model graph (Gilbert, 1959; Erdős & Rényi, 1961), where we are in control to parametrize willingly the number of nodes and positive/negative weights. The model chooses each of the possible edges with an already selected probability (referring to the connectivity feature of the graph) and weights are randomly picked from a list of possible values with positive weight representing a positive sign (visualized as a green link) while negative weight representing a negative sign (visualized as a red link).

```
NEW = nx.gnp_random_graph(n, p, directed=True)
weights = [-1, 1]
for (edge_src, edge_dst, d) in NEW.edges(data=True):
    edge_weight = d['weight'] = random.choice(weights)
```

Where n is the number of nodes and p is the measure of the preselected connectivity probability.
This procedure gave us the ability to **escalate progressively** the size and connectivity of the produced graph in order to ratify the **correctness** of our methodology and the functionality of our algorithm.

4.3 Experimental Procedure – Results

The capability to compound synthetic data gave us the mean to verify our methodology on large datasets with a variety of size and node connectivity**.** We were based **on two important metrics the number of clusters and the ability or not of our algorithms to converge.** We worked in a two level approach; we escalated gradually the size and node connectivity while we established two algorithmic pipelines simultaneously by using Affinity Propagation and Spectral Clustering in both stand alone and connected manor. Convergence was described "binary" with a yes or no, based on whether the algorithm converged or not. Except from the special case where the Affinity Propagation algorithm "bursts" with unexpected results as if all nodes appear to have mutually equal similarity scores, in this case we refer to Convergence status as "random".
The Results are presented in tables resembling the escalation we described before as we were gradually increasing the complexity of the examined Graph.

Nodes (No)	Connectivity (%)	Edges (No)	Clusters (No)			Convergence (Bin)
		AF	AF	SC		
		Initial Dataset Graph	Exp	SpGap	Exp	
10	10	8	0	0	0	no
50	10	248	11	18	11	yes
100	10	997	19	19	19	yes
250	10	6318	37	43	37	yes
500	10	25192	71	69	71	yes
1500	10	224540	178	150	178	yes
5000	10	2498026	461	nan	0	random

Table 1.

Nodes (No)	Connectivity (%)	Edges (No)	Clusters (No)			Convergence (Bin)
		AF	AF	SC		
		Initial Dataset Graph	Exp	SpGap	Exp	
10	25	21	0	0	0	no
50	25	569	10	14	10	yes
100	25	2456	14	20	14	yes
250	25	15534	34	35	34	yes
500	25	62072	67	74	67	yes
1500	25	562528	159	170	159	yes
5000	25	6248403	473	nan	473	yes

Table 2

Nodes (No)	Connectivity (%)	Edges (No)	Clusters (No)			Convergence (Bin)
		AF	AF	SC		
		Initial Dataset Graph	Exp	SpGap	Exp	
10	50	44	4	2	4	yes
50	50	1236	10	21	10	yes
100	50	4934	16	17	16	yes
250	50	31149	34	57	34	yes
500	50	124772	66	93	66	yes
1500	50	1123498	166	182	166	yes
5000	50	9999569	0	0	0	no

Table 3.

Nodes (No)	Connectivity (%)	Edges (No)	Clusters (No)			Convergence (Bin)
		AF	AF	SC		
		Initial Dataset Graph	Exp	SpGap	Exp	
10	75	70	3	2	3	yes
50	75	1813	10	18	10	yes
100	75	7388	18	29	18	yes
250	75	46751	34	54	34	yes
500	75	187474	61	96	61	yes
1500	75	1685752	141	161	141	yes
5000	75	18755785	0	0	0	no

Table 4.

Nodes (No)	Connectivity (%)	Edges (No)	Clusters (No)			Convergence (Bin)
		AF	AF	SC		
		Initial Dataset Graph	Exp	SpGap	Exp	
10	90	83	2	6	2	yes
50	90	2201	11	22	11	yes
100	90	8809	15	34	15	yes
250	90	55935	37	75	37	yes
500	90	224553	67	115	67	yes
1500	90	2023970	182	207	182	yes
5000	90	33234562	0	0	0	no

Table 5.

As we described above we used two algorithmic pipelines simultaneously both stand alone and connected. Firstly **AF (Affinity Propagation)** and **SC (Spectral Clustering)** worked independently and produced No of Clusters through **Exemplars (Exp)** and Eigen Decomposition plus **Spectral Gap (SpGap) respectively.**
And then we used both clustering algorithms in the same pipeline: the **AF clustering algorithm** calculated a number of clusters for each given graph, while the **Spectral clustering algorithm** used the output of the **AF** as input to tune more refined results.
In order to evaluate our clustering efficiency we calculated the Silhouette Coefficient Value in various of the generated graphs and as we can see there are samples that are very close to the decision boundary between two neighboring clusters and we can observe an overlapping tendency specially during SC as the nodes number rises which is suppressed when the two clustering algorithms function in cooperation rather than stand alone.

Nodes Connectivity Degree

250 25%

Spectral Silhouette_Score = -0.0015337561006091534

Affinity Silhouette_Score = -0.0013716720251524723

250 50%

Spectral Silhouette_Score = 0.0007317356029707193

Affinity Silhouette_Score = -0.003342205973886501

250 75%

Spectral Silhouette_Score = -0.005294103188455384

Affinity Silhouette_Score = 0.0006393376337654699

250 90%

Spectral Silhouette_Score = 0.003983825698198112

Affinity Silhouette_Score = -0.002386037845357535

500 10%

Spectral Silhouette_Score = -0.010126577591512228

Affinity Silhouette_Score = 0.0004381557057284825

500 50%

Spectral Silhouette_Score = 0.0083421385758766

Affinity Silhouette_Score = -0.006408910374657886

500 75%

Spectral Silhouette_Score = -0.006438492356243044

Affinity Silhouette_Score = 0.004788414228141365

500 90%

Spectral Silhouette_Score = 0.008750706898741047

Affinity Silhouette_Score = 0.004073566179062481

1500 10%

Spectral Silhouette_Score = -0.016922260950355397

Affinity Silhouette_Score = 0.002943362045551687

1500 25%

Spectral Silhouette_Score = 0.016325495267873556

Affinity Silhouette_Score = -0.005110335100224544

1500 50%

Spectral Silhouette_Score = -0.01761335882858793

Affinity Silhouette_Score = 0.007413029975112354

1500 75%

Spectral Silhouette_Score = 0.014215445485424192

Affinity Silhouette_Score = -0.006849787098136669

1500 90%

Spectral Silhouette_Score = 0.020073556433615684

Affinity Silhouette_Score = -0.005782768303717352

4.4 Clustering Validation

4.4.1 Similarity Validation – Normalization Issues

The only indisputable criteria to validate the accuracy of our methodology were to check for the values in the diagonal of the similarity matrix. The KPI (key point indicator) was the appearance of the value of 1 in the Matrices Diagonal, when this occurred it made undeniable to verify the fact that each node was 100% similar with itself.
To achieve this KPI came through the experiment with various normalization methods with different characteristics. This proved essential because when you choose to not apply any normalization the produced Similarity Matrix has nothing to do with our needs.

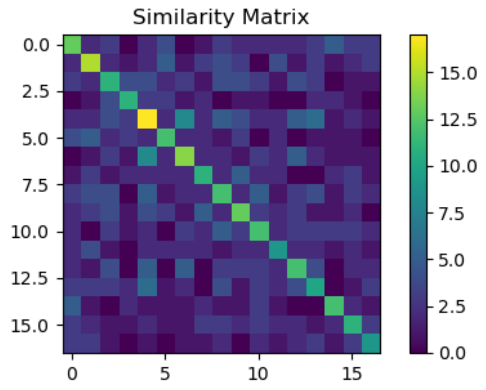


Figure 21. Non Normalized Similarity Matrix
Let’s view the methods we examined in order to choose the normalization formula which can optimize our algorithmic procedure.
<u>Normalization Methodology Explanation:</u>
Max Degree Normalization:

```
normalization_factors = np.zeros((size, size))
for i in range(size):
  for j in range(size):
    normalization_factors[i, j] = 1 / max(node degrees[i]['final'],
                                  node degrees[j]['final'])
```

We designate the normalization factor as the inverse maximum of the node degrees, D(i) and D(j) which are the final holistic sum of edges. **Degreed normalization approach incorporates the in-degrees and out-degrees of each node in the process**. The concept is that when two nodes i and j commonly point to a third node k, the similarity of i and j should be inversely related to in-degree of k. Similarly, when node h pointed by nodes i and j, the out-degrees of node h should be inversely to the similarity of node i and node j. The similarity between two nodes must be high in the same cluster and low in different clusters.

So we are successful if we could place high degree (number of edges) between nodes of the same community and set low degree between nodes that in different communities.
In some scenarios we are forming Bibliographic Coupling B Matrix for both Positive and Negative ties normalized with out-positive degree(number of outgoing edges, with positive sign) & out-negative

degree(number of outgoing edges, with negative sign) and Cocitation C Matrix for both Positive and Negative ties normalized with in-positive degree(number of incoming edges, with positive sign) & in-negative degree(number of incoming edges, with negative sign) in an attempt to factorize the connection between Directionality and Polarity. The nodes with higher out-degree are more central (choices made). The nodes with higher in-degree are more prestigious (choices received).

Pairwise Normalization

```
# Apply Pairwise Similarity Normalisation
from sklearn.metrics import pairwise_distances
from scipy.spatial.distance import cosine

final_NormSimilarity = 1-pairwise_distances(similarity, metric="cosine")
final_NormSimilarity = np.round(final_NormSimilarity, decimals=accuracy)
```

Pairwise Normalization with cosine metric works with normalized vectors. We perform it at the final stage of the similarity matrix formation in order to ordain the already achieved superiority of the main diagonal values at the scale of 0 to 1.

Amsler Similarity normalization

As we have seen in previous chapters we expressed Amsler Similarity in our case as Amsler Similarity Normalization:

S = λ * Cin * Sim_in + (1-λ) * Cout * Sim_out **Equation.9**

Where λ is the weight factor balancing the importance of in- and out-links; and Cin and Cout are in- and out-link damping factors. After a series of tests on a series of graphs we concluded that we may optimize these factors as **λ=0.5 Cin=0.4 and Cout=0.6**.

We perform Amsler before Pairwise during the formation of Similarity Matrix with the use of the following formulas:

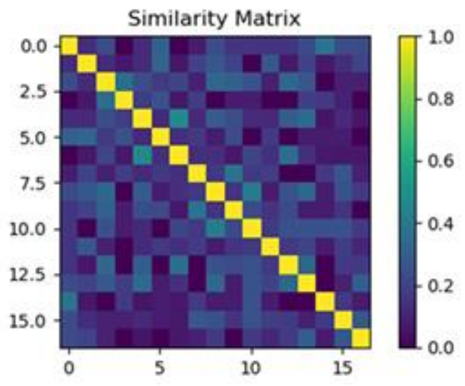
Incoming Link Similarity	$S_{in}(i,j) = (C^+[i,j] + C^-[i,j])$
Outgoing Link Similarity	$S_{out}(i,j) = (B^+[i,j] + B^-[i,j])$
Positive Co-Citation Matrix (C+)	$C^+ = A^+ A^{+T}$
Negative Co-Citation Matrix (C-)	$C^- = A^- A^{-T}$
Positive Co-Reference Matrix (B+)	$B^+ = A^{+T} A^+$
Negative Co-Reference Matrix (B-)	$B^- = A^{-T} A^-$
Adj Matrix of only positive links	A^+
Adj Matrix of only negative links	A^-

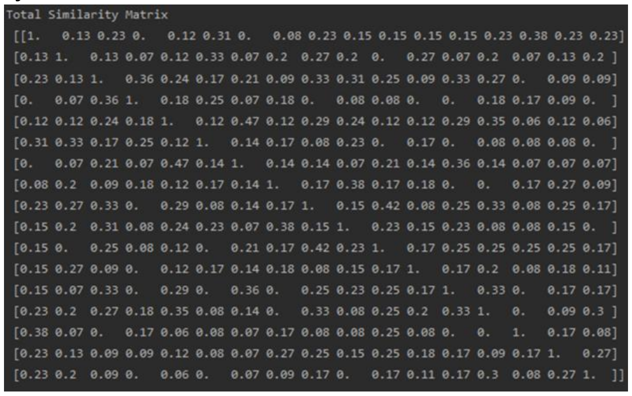
Table 6. Matrices Formulas

We worked in a small real world dataset with only 17 nodes in order to show in a better way the differences of every case.

These are the most dominant scenarios: We describe the combination of tools (Equations and formulas) we used in each case and we visualized our outcome in the form of the Similarity matrix Heatmap where we can show plainly the importance of the diagonal of the Similarity Matrix.

1st Scenario: Based on Equation 12. We decided to **normalize the Similarity** only at a **final Stage and only according to max Degree** where In-degree is the number of head endpoints adjacent to a node *and* Out-degree is the number of tail endpoints adjacent to a node.

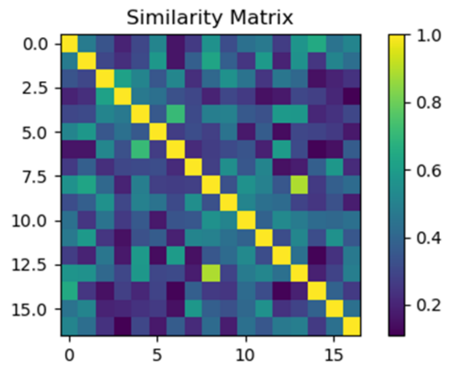


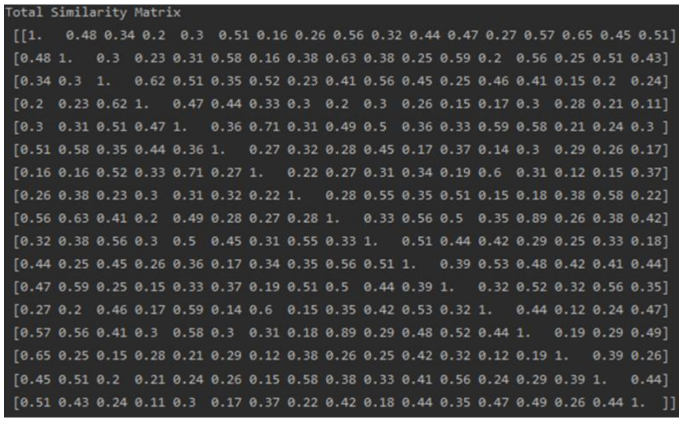


```
Total Similarity Matrix
 [[1.   0.13 0.23 0.   0.12 0.31 0.   0.08 0.23 0.15 0.15 0.15 0.15 0.23 0.38 0.23 0.23]
 [0.13 1.   0.13 0.07 0.12 0.33 0.07 0.2  0.27 0.2  0.   0.27 0.07 0.2  0.07 0.13 0.2 ]
 [0.23 0.13 1.   0.36 0.24 0.17 0.21 0.09 0.33 0.31 0.25 0.09 0.33 0.27 0.   0.09 0.09]
 [0.   0.07 0.36 1.   0.18 0.25 0.07 0.18 0.   0.08 0.08 0.   0.   0.18 0.17 0.09 0.  ]
 [0.12 0.12 0.24 0.18 1.   0.12 0.47 0.12 0.29 0.24 0.12 0.12 0.29 0.35 0.06 0.12 0.06]
 [0.31 0.33 0.17 0.25 0.12 1.   0.14 0.17 0.08 0.23 0.   0.17 0.   0.08 0.08 0.08 0.  ]
 [0.   0.07 0.21 0.07 0.47 0.14 1.   0.14 0.14 0.07 0.21 0.14 0.36 0.14 0.07 0.07 0.07]
 [0.08 0.2  0.09 0.18 0.12 0.17 0.14 1.   0.17 0.38 0.17 0.18 0.   0.   0.17 0.27 0.09]
 [0.23 0.27 0.33 0.   0.29 0.08 0.14 0.17 1.   0.15 0.42 0.08 0.25 0.33 0.08 0.25 0.17]
 [0.15 0.2  0.31 0.08 0.24 0.23 0.07 0.38 0.15 1.   0.23 0.15 0.23 0.08 0.08 0.15 0.  ]
 [0.15 0.   0.25 0.08 0.12 0.   0.21 0.17 0.42 0.23 1.   0.17 0.25 0.25 0.25 0.25 0.17]
 [0.15 0.27 0.09 0.   0.12 0.17 0.14 0.18 0.08 0.15 0.17 1.   0.17 0.2  0.08 0.18 0.11]
 [0.15 0.07 0.33 0.   0.29 0.   0.36 0.   0.25 0.23 0.25 0.17 1.   0.33 0.   0.17 0.17]
 [0.23 0.2  0.27 0.18 0.35 0.08 0.14 0.   0.33 0.08 0.25 0.2  0.33 1.   0.   0.09 0.3 ]
 [0.38 0.07 0.   0.17 0.06 0.08 0.07 0.17 0.08 0.08 0.25 0.08 0.   0.   1.   0.17 0.08]
 [0.23 0.13 0.09 0.09 0.12 0.08 0.07 0.27 0.25 0.15 0.25 0.18 0.17 0.09 0.17 1.   0.27]
 [0.23 0.2  0.09 0.   0.06 0.   0.07 0.09 0.17 0.   0.17 0.11 0.17 0.3  0.08 0.27 1.  ]]
```

Figure 22. Similarity Matrix Visualization

2nd Scenario: We applied Equation.12 in order to perform **Max-Degree normalization** during the formation of **Co-Citation and Bib Coupling Matrices** and **Pairwise Normalization** at the final stage of the formation of Similarity Matrix.

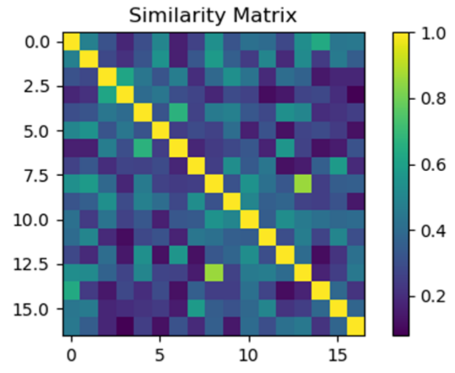


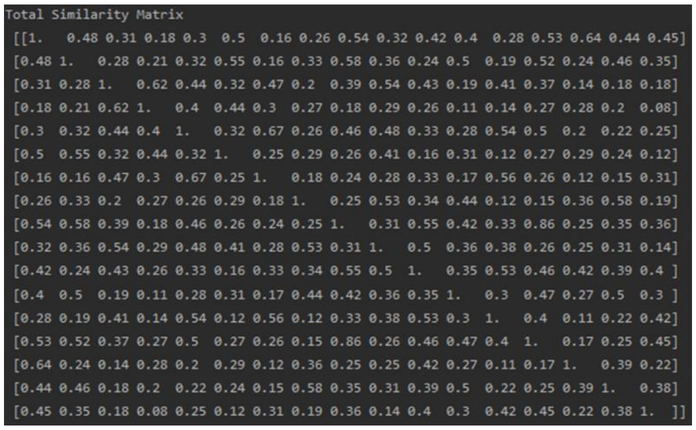


```
Total Similarity Matrix
 [[1.   0.48 0.34 0.2  0.3  0.51 0.16 0.26 0.56 0.32 0.44 0.47 0.27 0.57 0.65 0.45 0.51]
 [0.48 1.   0.3  0.23 0.31 0.58 0.16 0.38 0.63 0.38 0.25 0.59 0.2  0.56 0.25 0.51 0.43]
 [0.34 0.3  1.   0.62 0.51 0.35 0.52 0.23 0.41 0.56 0.45 0.25 0.46 0.41 0.15 0.2  0.24]
 [0.2  0.23 0.62 1.   0.47 0.44 0.33 0.3  0.2  0.3  0.26 0.15 0.17 0.3  0.28 0.21 0.11]
 [0.3  0.31 0.51 0.47 1.   0.36 0.71 0.31 0.49 0.5  0.36 0.33 0.59 0.58 0.21 0.24 0.3 ]
 [0.51 0.58 0.35 0.44 0.36 1.   0.27 0.32 0.28 0.45 0.17 0.37 0.14 0.3  0.29 0.26 0.17]
 [0.16 0.16 0.52 0.33 0.71 0.27 1.   0.22 0.27 0.31 0.34 0.19 0.6  0.31 0.12 0.15 0.37]
 [0.26 0.38 0.23 0.3  0.31 0.32 0.22 1.   0.28 0.55 0.35 0.51 0.15 0.18 0.38 0.58 0.22]
 [0.56 0.63 0.41 0.2  0.49 0.28 0.27 0.28 1.   0.33 0.56 0.5  0.35 0.89 0.26 0.38 0.42]
 [0.32 0.38 0.56 0.3  0.5  0.45 0.31 0.55 0.33 1.   0.51 0.44 0.42 0.29 0.25 0.33 0.18]
 [0.44 0.25 0.45 0.26 0.36 0.17 0.34 0.35 0.56 0.51 1.   0.39 0.53 0.48 0.42 0.41 0.44]
 [0.47 0.59 0.25 0.15 0.33 0.37 0.19 0.51 0.5  0.44 0.39 1.   0.32 0.52 0.32 0.56 0.35]
 [0.27 0.2  0.46 0.17 0.59 0.14 0.6  0.15 0.35 0.42 0.53 0.32 1.   0.44 0.12 0.24 0.47]
 [0.57 0.56 0.41 0.3  0.58 0.3  0.31 0.18 0.89 0.29 0.48 0.52 0.44 1.   0.19 0.29 0.49]
 [0.65 0.25 0.15 0.28 0.21 0.29 0.12 0.38 0.26 0.25 0.42 0.32 0.12 0.19 1.   0.39 0.26]
 [0.45 0.51 0.2  0.21 0.24 0.26 0.15 0.58 0.38 0.33 0.41 0.56 0.24 0.29 0.39 1.   0.44]
 [0.51 0.43 0.24 0.11 0.3  0.17 0.37 0.22 0.42 0.18 0.44 0.35 0.47 0.49 0.26 0.44 1.  ]]
```

Figure 23. Similarity Matrix Visualization

3rd **Scenario:** We apply **Max Degree normalization** by using Equation 12. **everywhere during the formation of Co-Citation, Bib Coupling Matrices** and at the pre-final stage of the formation of Similarity Matrix. Then we also apply **Pairwise normalization from page 41** at the **final stage of the Similarity Matrix formation.**

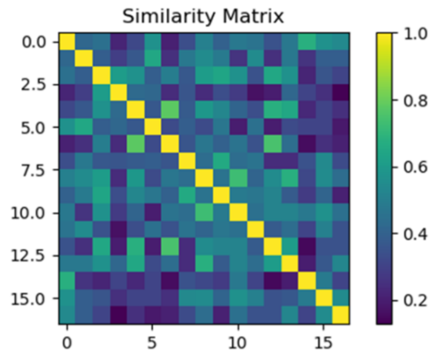


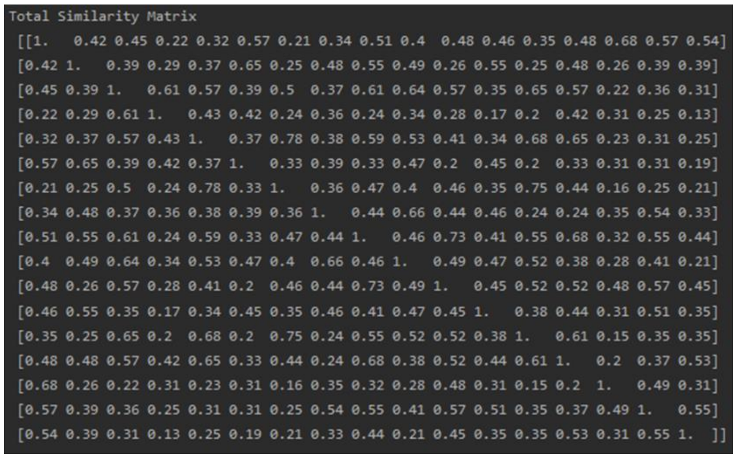


```
Total Similarity Matrix
 [[1.   0.48 0.31 0.18 0.3  0.5  0.16 0.26 0.54 0.32 0.42 0.4  0.28 0.53 0.64 0.44 0.45]
 [0.48 1.   0.28 0.21 0.32 0.55 0.16 0.33 0.58 0.36 0.24 0.5  0.19 0.52 0.24 0.46 0.35]
 [0.31 0.28 1.   0.62 0.44 0.32 0.47 0.2  0.39 0.54 0.43 0.19 0.41 0.37 0.14 0.18 0.18]
 [0.18 0.21 0.62 1.   0.4  0.44 0.3  0.27 0.18 0.29 0.26 0.11 0.14 0.27 0.28 0.2  0.08]
 [0.3  0.32 0.44 0.4  1.   0.32 0.67 0.26 0.46 0.48 0.33 0.28 0.54 0.5  0.2  0.22 0.25]
 [0.5  0.55 0.32 0.44 0.32 1.   0.25 0.29 0.26 0.41 0.16 0.31 0.12 0.27 0.29 0.24 0.12]
 [0.16 0.16 0.47 0.3  0.67 0.25 1.   0.18 0.24 0.28 0.33 0.17 0.56 0.26 0.12 0.15 0.31]
 [0.26 0.33 0.2  0.27 0.26 0.29 0.18 1.   0.25 0.53 0.34 0.44 0.12 0.15 0.36 0.58 0.19]
 [0.54 0.58 0.39 0.18 0.46 0.26 0.24 0.25 1.   0.31 0.55 0.42 0.33 0.86 0.25 0.35 0.36]
 [0.32 0.36 0.54 0.29 0.48 0.41 0.28 0.53 0.31 1.   0.5  0.36 0.38 0.26 0.25 0.31 0.14]
 [0.42 0.24 0.43 0.26 0.33 0.16 0.33 0.34 0.55 0.5  1.   0.35 0.53 0.46 0.42 0.39 0.4 ]
 [0.4  0.5  0.19 0.11 0.28 0.31 0.17 0.44 0.42 0.36 0.35 1.   0.3  0.47 0.27 0.5  0.3 ]
 [0.28 0.19 0.41 0.14 0.54 0.12 0.56 0.12 0.33 0.38 0.53 0.3  1.   0.4  0.11 0.22 0.42]
 [0.53 0.52 0.37 0.27 0.5  0.27 0.26 0.15 0.86 0.26 0.46 0.47 0.4  1.   0.17 0.25 0.45]
 [0.64 0.24 0.14 0.28 0.2  0.29 0.12 0.36 0.25 0.25 0.42 0.27 0.11 0.17 1.   0.39 0.22]
 [0.44 0.46 0.18 0.2  0.22 0.24 0.15 0.58 0.35 0.31 0.39 0.5  0.22 0.25 0.39 1.   0.38]
 [0.45 0.35 0.18 0.08 0.25 0.12 0.31 0.19 0.36 0.14 0.4  0.3  0.42 0.45 0.22 0.38 1.  ]]
```

Figure 24. Similarity Matrix Visualization

4th **Scenario:** We didn't apply Max Degree normalization anywhere during the formation of Co-Citation and Bib Coupling Matrices .**But then we applied Modified Amsler v2.0 formula from Equation.8 , max Degree normalization from Equation.12 and Pairwise Normalization at the final stage of the formation of Similarity Matrix.**





```
Total Similarity Matrix
 [[1.   0.42 0.45 0.22 0.32 0.57 0.21 0.34 0.51 0.4  0.48 0.46 0.35 0.48 0.68 0.57 0.54]
 [0.42 1.   0.39 0.29 0.37 0.65 0.25 0.48 0.55 0.49 0.26 0.55 0.25 0.48 0.26 0.39 0.39]
 [0.45 0.39 1.   0.61 0.57 0.39 0.5  0.37 0.61 0.64 0.57 0.35 0.65 0.57 0.22 0.36 0.31]
 [0.22 0.29 0.61 1.   0.43 0.42 0.24 0.36 0.24 0.34 0.28 0.17 0.2  0.42 0.31 0.25 0.13]
 [0.32 0.37 0.57 0.43 1.   0.37 0.78 0.38 0.59 0.53 0.41 0.34 0.68 0.65 0.23 0.31 0.25]
 [0.57 0.65 0.39 0.42 0.37 1.   0.33 0.39 0.33 0.47 0.2  0.45 0.2  0.33 0.31 0.31 0.19]
 [0.21 0.25 0.5  0.24 0.78 0.33 1.   0.36 0.47 0.4  0.46 0.35 0.75 0.44 0.16 0.25 0.21]
 [0.34 0.48 0.37 0.36 0.38 0.39 0.36 1.   0.44 0.66 0.44 0.46 0.24 0.24 0.35 0.54 0.33]
 [0.51 0.55 0.61 0.24 0.59 0.33 0.47 0.44 1.   0.46 0.73 0.41 0.55 0.68 0.32 0.55 0.44]
 [0.4  0.49 0.64 0.34 0.53 0.47 0.4  0.66 0.46 1.   0.49 0.47 0.52 0.38 0.28 0.41 0.21]
 [0.48 0.26 0.57 0.28 0.41 0.2  0.46 0.44 0.73 0.49 1.   0.45 0.52 0.52 0.48 0.57 0.45]
 [0.46 0.55 0.35 0.17 0.34 0.45 0.35 0.46 0.41 0.47 0.45 1.   0.38 0.44 0.31 0.51 0.35]
 [0.35 0.25 0.65 0.2  0.68 0.2  0.75 0.24 0.55 0.52 0.52 0.38 1.   0.61 0.15 0.35 0.35]
 [0.48 0.48 0.57 0.42 0.65 0.33 0.44 0.24 0.68 0.38 0.52 0.44 0.61 1.   0.2  0.37 0.53]
 [0.68 0.26 0.22 0.31 0.23 0.31 0.16 0.35 0.32 0.28 0.48 0.31 0.15 0.2  1.   0.49 0.31]
 [0.57 0.39 0.36 0.25 0.31 0.31 0.25 0.54 0.55 0.41 0.57 0.51 0.35 0.37 0.49 1.   0.55]
 [0.54 0.39 0.31 0.13 0.25 0.19 0.21 0.33 0.44 0.21 0.45 0.35 0.35 0.53 0.31 0.55 1.  ]]
```

Figure 25. Similarity Matrix Visualization

The Case scenarios that are qualified by giving best results are the 1st and the 4th.

1st Scenario	
Affinity Silhouette Score	**0.10063702927459384**
Spectral Silhouette Score	**0.06529194576007039**
4th Scenario	
Affinity Silhouette Score	**0.19011786311015436**
Spectral Silhouette Score	**0.05312464113541116**

Table 7. Dominant Case Scenarios Comparison

We then compared those test cases on the level of clustering:

1st Scenario

Affinity Clustering:

Number of Affinity Propagation clusters: 6
Affinity Propagation Cluster centers: [0 1 3 6 7 8]
Affinity Propagation Clusters:[[0, 14, 16], [1, 5, 11], [2, 3], [4, 6, 12], [7, 9, 15], [8, 10, 13]]

Spectral Clustering Stand Alone:

Optimal (Spectral Gap) number of Spectral Clustering clusters [1 3 6 14 10] : 5

Spectral Clustering with AF input:

Let's Compare Spectral Clustering where we use as input the Affinity Propagation output VS Spectral Clustering where we use optimal number of Clusters input
Spectral Clustering Clusters: [[0, 3, 5], [4, 6, 12], [2, 8, 13, 16], [10, 14, 15], [7, 9],[1, 11]]
Spectral Clustering Optimal Clusters: [[1, 3, 5],[2, 4, 6, 12], [0, 14, 10], [8, 13, 16], [7, 9, 11, 15]]

The above procedure was visualized in Figure 26. Where we can observe for the first time that the use of the combinational pipeline tunes our clustering results.

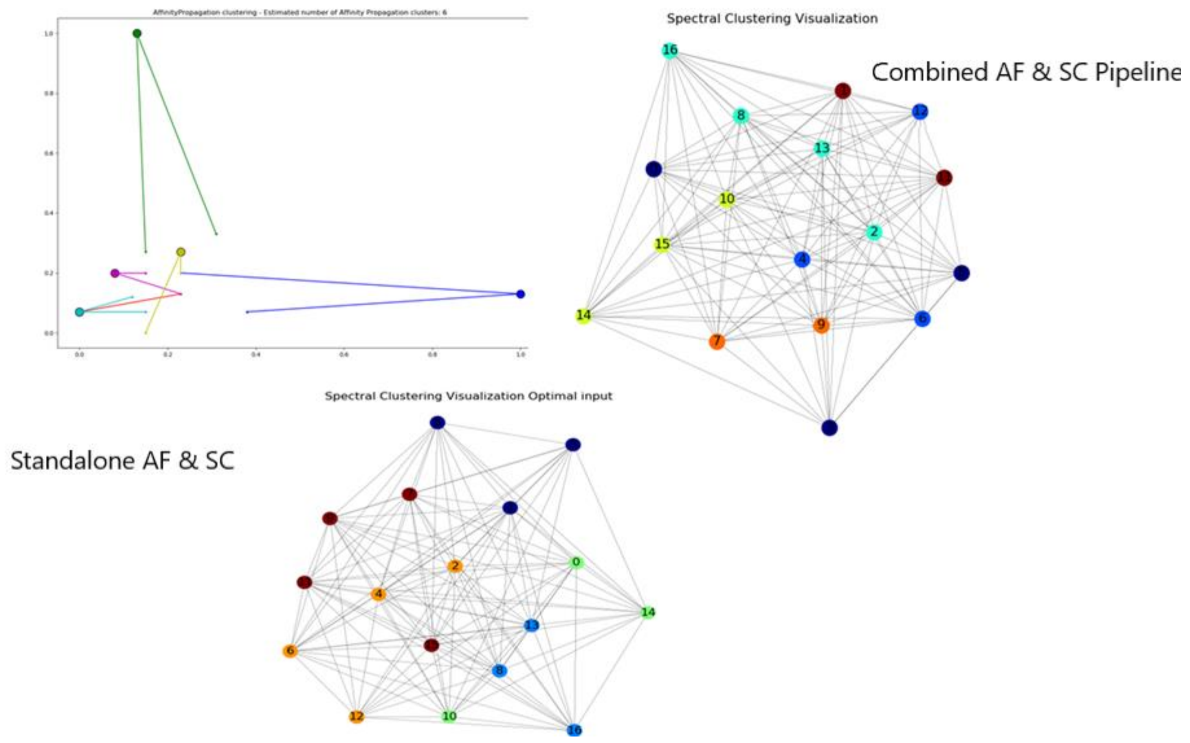


Figure 26. Visualization of Clusters by using AF and SC standalone and combined algorithmic Pipelines

4th Scenario

Affinity Clustering:

Number of Affinity Propagation clusters: 5

Affinity Propagation Cluster centers: [0 1 2 6 8]

Affinity Propagation Clusters: [[0, 14, 15, 16], [1, 5, 7, 11], [2, 3, 9], [4, 6, 12], [8, 10, 13]]

Spectral Clustering Stand Alone:

Optimal number of Spectral Clustering clusters [1 6 2 10 3]

Spectral Clustering with AF input:

Here Optimal (Spectral Gap) number and Spectral Clustering where we use input the Affinity Propagation produce the same number of Clusters.

Spectral Clustering Clusters: [[1, 13, 8], [2, 4, 6, 12], [3, 5], [0, 14, 10, 16], [9, 11, 15, 7]] **visualized in Figure.27**

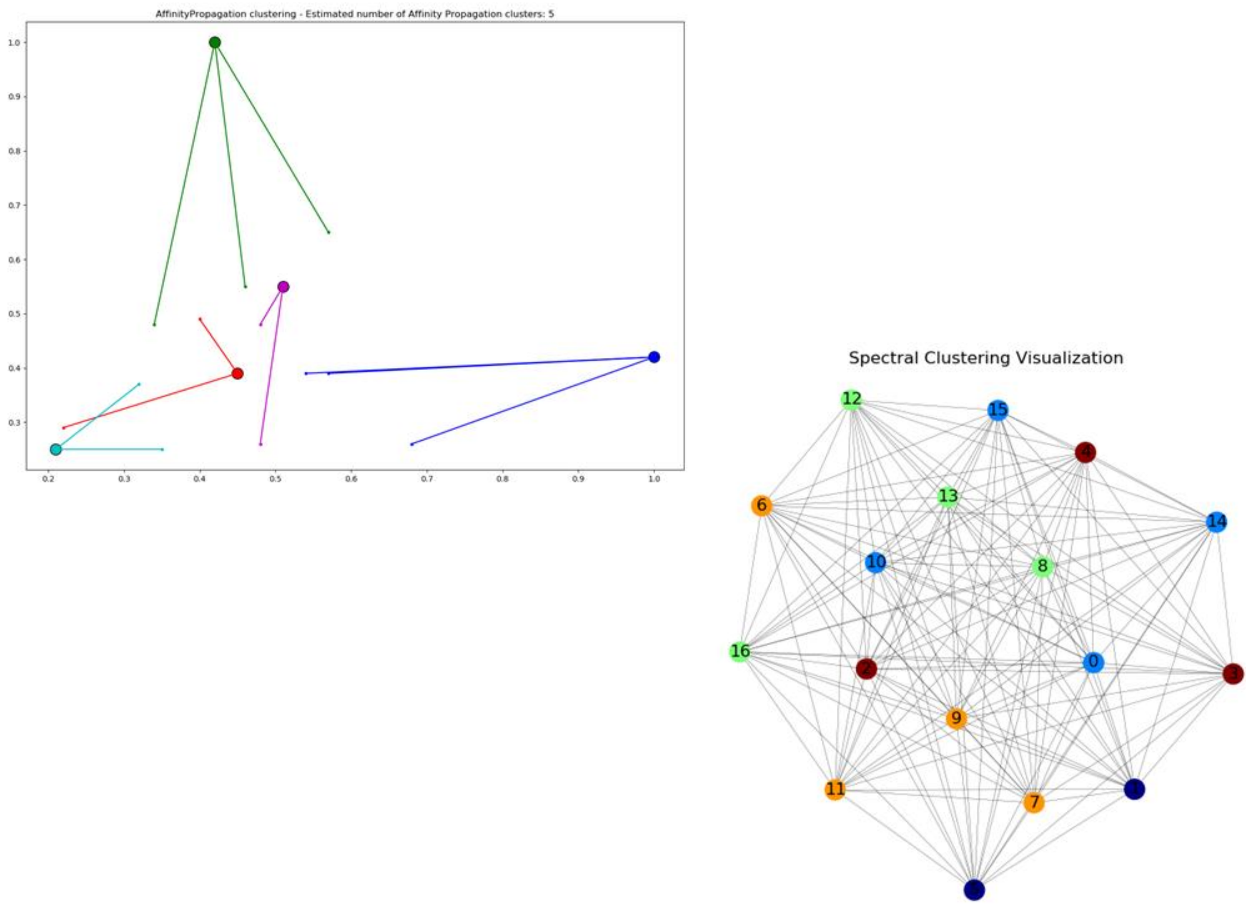


Figure 27. Visualization of Clusters by using AF and SC standalone and combined algorithmic Pipelines

The resemblance degree of the produced clusters within each scenario and the comparison of the **Similarity Matrix** in every case proved us that the normalization method that will optimize our experimental procedure is to **apply the Max Degree factor on the final stage of the formation of the Similarity measure.**

5. Conclusion – Future Work

5.1 Conclusion – Methodology evaluation

In all the experimental procedure our main concern was the values of the main Diagonal of the Similarity Matrix so that we could evaluate our method and verify that the normalization procedure was the optimal. They had to be equal to 1 in order to claim that every node shows 100% similarity with itself.

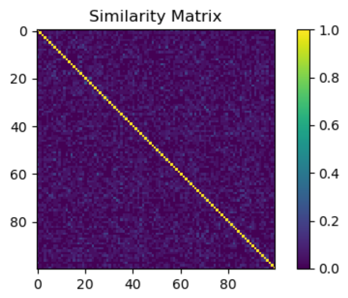
We used three main normalization procedures:

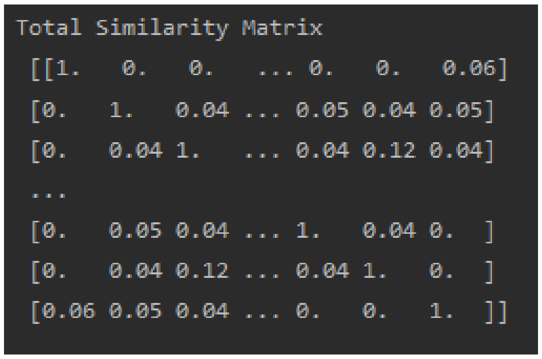
1. **Normalization of the Similarity at the final Stage of its formation and only according to max Degree.**
2. **Max-Degree normalization during the formation of Co-Citation and Bib Coupling Matrices.**
3. **Modified Amsler and max Degree normalization at the final stage of the formation of Similarity Matrix.**

The optimal procedure was the 1st because it lead us to the valued 1 diagonal of the Similarity Matrix without any other methodological "boost". The 2nd and the 3rd though they confirmed the dominance of the diagonal they needed an additional pairwise normalization to reach values equal to 1.

Let's make a visual comparison of the above:

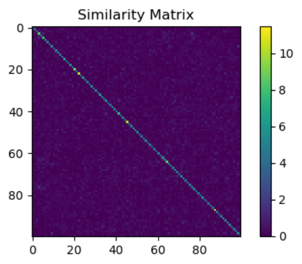
1. Max Degree Normalization on Similarity Matrix

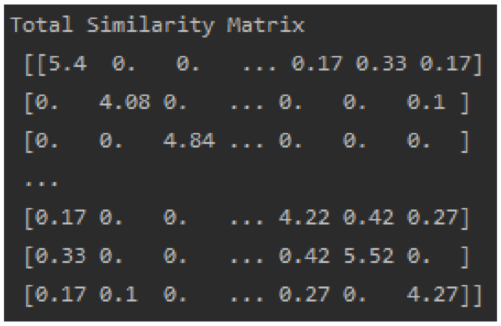




```
Total Similarity Matrix
 [[1.   0.   0.   ... 0.   0.   0.06]
 [0.   1.   0.04 ... 0.05 0.04 0.05]
 [0.   0.04 1.   ... 0.04 0.12 0.04]
 ...
 [0.   0.05 0.04 ... 1.   0.04 0.  ]
 [0.   0.04 0.12 ... 0.04 1.   0.  ]
 [0.06 0.05 0.04 ... 0.   0.   1.  ]]
```

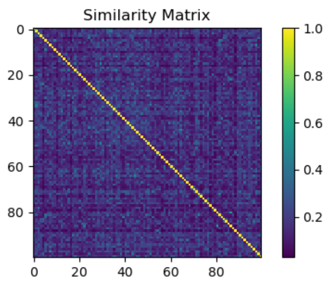
2. Max Degree Normalization on Components Matrices Before Pairwise Normalization "boost"

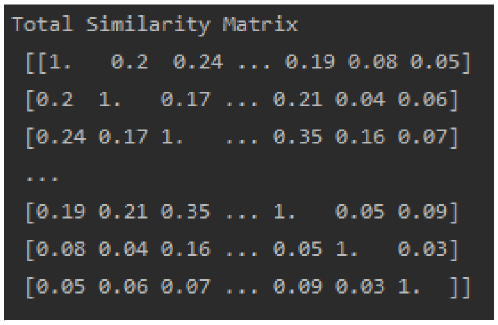




```
Total Similarity Matrix
 [[5.4  0.   0.   ... 0.17 0.33 0.17]
 [0.   4.08 0.   ... 0.   0.   0.1 ]
 [0.   0.   4.84 ... 0.   0.   0.  ]
 ...
 [0.17 0.   0.   ... 4.22 0.42 0.27]
 [0.33 0.   0.   ... 0.42 5.52 0.  ]
 [0.17 0.1  0.   ... 0.27 0.   4.27]]
```

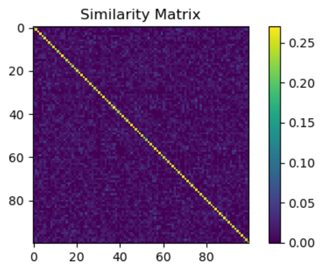
Max Degree Normalization on Components Matrices after Pairwise Normalization "boost"

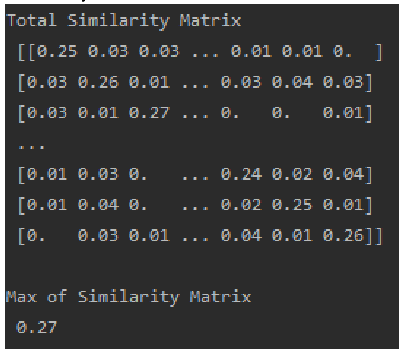




```
Total Similarity Matrix
 [[1.   0.2  0.24 ... 0.19 0.08 0.05]
 [0.2  1.   0.17 ... 0.21 0.04 0.06]
 [0.24 0.17 1.   ... 0.35 0.16 0.07]
 ...
 [0.19 0.21 0.35 ... 1.   0.05 0.09]
 [0.08 0.04 0.16 ... 0.05 1.   0.03]
 [0.05 0.06 0.07 ... 0.09 0.03 1.  ]]
```

3. Amsler and max Degree normalization on Similarity **before** Pairwise Normalization "boost"

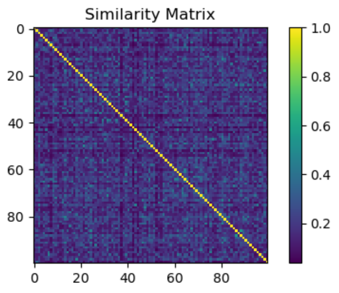


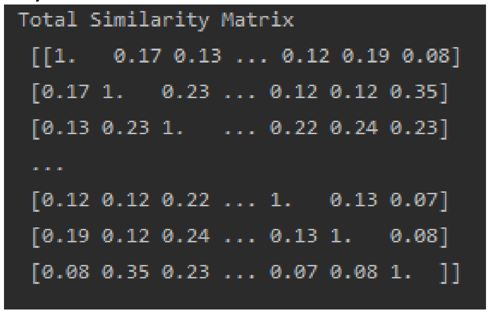


```
Total Similarity Matrix
 [[0.25 0.03 0.03 ... 0.01 0.01 0.  ]
 [0.03 0.26 0.01 ... 0.03 0.04 0.03]
 [0.03 0.01 0.27 ... 0.   0.   0.01]
 ...
 [0.01 0.03 0.   ... 0.24 0.02 0.04]
 [0.01 0.04 0.   ... 0.02 0.25 0.01]
 [0.   0.03 0.01 ... 0.04 0.01 0.26]]

Max of Similarity Matrix
 0.27
```

Amsler and max Degree normalization on Similarity **after** Pairwise Normalization "boost"





```
Total Similarity Matrix
 [[1.   0.17 0.13 ... 0.12 0.19 0.08]
 [0.17 1.   0.23 ... 0.12 0.12 0.35]
 [0.13 0.23 1.   ... 0.22 0.24 0.23]
 ...
 [0.12 0.12 0.22 ... 1.   0.13 0.07]
 [0.19 0.12 0.24 ... 0.13 1.   0.08]
 [0.08 0.35 0.23 ... 0.07 0.08 1.  ]]
```

It's obvious why the 1st methodological approach became dominant.
It is Simple Steady and functional even with large datasets. There are common findings during our experiment with every methodology referring datasets with low connectivity and escalating size that they display random clustering and difficulty to converge.

Another important finding is that the superiority of the combined algorithmic pipeline (output of AF as input in SC) it enhances the clustering results and this can help us in future research of this model's capabilities.

5.2 Future work – Possible Expansion

One of the most intriguing expansions of our implementations is to work with signed and directed graphs with pre-determined communities in massive scale. The sectors of appliance could be criminology, telecommunications or politics.
A large proportion of human trafficking is carried through criminal networks whose members have bonds with each other. These bonds can be family, race, color, religion or class oriented. The ability to predict the clusters of the succession of any criminal boss can be crucial in the battle against organized crime. We can also predict the evolvement of a political party where the predetermined communities also called factions rely on the political origin of its members.
Finally in my line of work in the Telecom Industry we can adjust the network's resources to service in high priority subscribers who are members of the same community not in the same way it is happening today. Nowadays every subscriber of every telecom provider around the globe is clustered in a scale of importance in fiscal terms. There is no network that has the ability to prioritize the communication between two subscribers in similarity terms. The definition of similarity can be tuned during the dataset preprocessing whether those two subscribers are possible friends or adversaries, have the same habits (seasonality of network's resources usage) or the same appetite for music, films or games. There also some other interesting criteria to form such a dataset that where inspired by the experimental procedure of this Thesis. In a Telecom subscribers (nodes) database we can easily define who initiates a call and who receives one (Nodes Directionality) and the call duration biased by the call seasonality.
For example a call over five minutes during working hours is most probable a professional call but a call with the same duration after 9 o clock or in weekends is a more intimate event (call between friends or lovers). All these parameters can help us to form a Signed and Directed Graph resembling a telecom provider subscriber's database and to perform clustering in order to define prioritization when an intra cluster call occurs. This can be feasible after 2023 when 5G core networks will be installed in providers and resources slicing can be possible in subscriber level.
The combined algorithmic pipeline can be an extremely useful tool to manage in a customized manner these multi-diverse networks.

6. Bibliography

Fragkiskos D. Malliaros and Michalis Vazirgiannis. 2013. Clustering and community detection in directed networks: A survey. Physics Reports 533, 4: 95–142. 10.1016/j.physrep.2013.08.002

Wei-Chung Liu, Liang-Cheng Huang, Chester Wai-Jen Liu, and Ferenc Jordán. 2020. A simple approach for quantifying node centrality in signed and directed social networks. Applied Network Science 5, 1. 10.1007/s41109-020-00288-w

Daniel López Sánchez, Jorge Revuelta, Fernando De la Prieta, Ana B. Gil-González, and Cach Dang. 2016. Twitter User Clustering Based on Their Preferences and the Louvain Algorithm. In Advances in Intelligent Systems and Computing. Springer International Publishing, 349–356. 10.1007/978-3-319-40159-1_29

Matthew Curran Benigni. 2017. Detection and Analysis of Online Extremist Communities. Carnegie Mellon University. 10.1184/R1/6715841.V1

Jiliang Tang, Yi Chang, Charu Aggarwal, and Huan Liu. 2016. A Survey of Signed Network Mining in Social Media. ACM Computing Surveys 49, 3: 1–37. 10.1145/2956185

Pouya Esmailian and Mahdi Jalili. 2015. Community Detection in Signed Networks: the Role of Negative ties in Different Scales. Scientific Reports 5, 1. 10.1038/srep14339

Symeon Papadopoulos, Yiannis Kompatsiaris, Athena Vakali, and Ploutarchos Spyridonos. 2011. Community detection in Social Media. Data Mining and Knowledge Discovery 24, 3: 515–554. 10.1007/s10618-011-0224-z

P. Anchuri and M. Magdon-Ismail. 2012. Communities and Balance in Signed Networks: A Spectral Approach. In 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. 10.1109/asonam.2012.48

Zhaoyue Zhong, Xiangrong Wang, Cunquan Qu and Guanghui Wang.2020. Efficient algorithm based on non-backtracking matrix for community detection in signed networks. arXiv:2006.15471v3

Yuemeng Li, Shuhan Yuan, Xintao Wu, and Aidong Lu. 2018. On spectral analysis of directed signed graphs. International Journal of Data Science and Analytics 6, 2: 147–162. 10.1007/s41060-018-0143-9

Q. Zheng, D. B. Skillicorn, and O. Walther. 2015. Signed Directed Social Network Analysis Applied to Group Conflict. In 2015 IEEE International Conference on Data Mining Workshop (ICDMW). 10.1109/icdmw.2015.107

Saeed Ahmadizadeh, Iman Shames, Samuel Martin, and Dragan Nešić. 2017. On eigenvalues of Laplacian matrix for a class of directed signed graphs. Linear Algebra and its Applications 523: 281–306. 10.1016/j.laa.2017.02.029

Nikos Koufos and Thanos Pappas.2017. Extracting positive and negative linked networkshttp://cs.uoi.gr/~apappas/projects/SocialNetworks/

Junghwan Kim, Haekyu Park, Ji-Eun Lee, and U Kang. 2018. SIDE. In Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18. 10.1145/3178876.3186117

Cheng-Shang Chang, Duan-Shin Lee, Li-Heng Liou, and Sheng-Min Lu. 2017. Community detection in signed networks: An error-correcting code approach. In 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). 10.1109/uic-atc.2017.8397662

Dong Li, Zhi-Ming Xu, Nilanjan Chakraborty, Anika Gupta, Katia Sycara, and Sheng Li. 2014. Polarity Related Influence Maximization in Signed Social Networks. PLoS ONE 9, 7: e102199. 10.1371/journal.pone.0102199

Chenlong Liu, Jing Liu, and Zhongzhou Jiang. 2014. A Multiobjective Evolutionary Algorithm Based on Similarity for Community Detection From Signed Social Networks. IEEE Transactions on Cybernetics 44, 12: 2274–2287. 10.1109/tcyb.2014.2305974

Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Predicting positive and negative links in online social networks. In Proceedings of the 19th international conference on World wide web - WWW '10. 10.1145/1772690.1772756

Pascal Pons and Matthieu Latapy. 2005. Computing Communities in Large Networks Using Random Walks. In Computer and Information Sciences - ISCIS 2005. Springer Berlin Heidelberg, 284–293. 10.1007/11569596_31

Jierui Xie and Boleslaw K. Szymanski. 2011. Community detection using a neighborhood strength driven Label Propagation Algorithm. In 2011 IEEE Network Science Workshop. 10.1109/nsw.2011.6004645

Darong Lai, Hongtao Lu, and Christine Nardini. 2010. Finding communities in directed networks by PageRank random walk induced network embedding. Physica A: Statistical Mechanics and its Applications 389, 12: 2443–2454. 10.1016/j.physa.2010.02.014

Wang, Zhe & Liang, Yingbin & Ji, Pengsheng. (2020). Spectral Algorithms for Community Detection in Directed Networks.

Luca Donetti. 2005. Improved spectral algorithm for the detection of network communities. In AIP Conference Proceedings. 10.1063/1.2008598

Zhou, Jianlin & Li, Lingbo & Zeng, An & Fan, Ying & Di, Zengru. (2018). Random walk on signed networks. Physica A: Statistical Mechanics and its Applications. 508. 10.1016/j.physa.2018.05.139.

Patrick Doreian and Andrej Mrvar. 1996. A partitioning approach to structural balance. Social Networks 18, 2: 149–168. 10.1016/0378-8733(95)00259-6

Fabián Riquelme and Pablo González-Cantergiani. 2016. Measuring user influence on Twitter: A survey. Information Processing & Management 52, 5: 949–975. 10.1016/j.ipm.2016.04.003

Mert Ozer, Nyunsu Kim, and Hasan Davulcu. 2016. Community detection in political Twitter networks using Nonnegative Matrix Factorization methods. In 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). 10.1109/asonam.2016.7752217

Tang, J., Chang, Y., Aggarwal, C., & Liu, H. (2016). A Survey of Signed Network Mining in Social Media. ACM Computing Surveys, 49(3), 1–37. 10.1145/2956185

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, 30(1–7), 107–117. 10.1016/s0169-7552(98)00110-x

Robin J Wilson. 1986. Introduction to graph theory. John Wiley & Sons, Inc., USA.

Kai-Yang Chiang, Cho-Jui Hsieh, Nagarajan Natarajan, Inderjit S. Dhillon, and Ambuj Tewari. 2014. Prediction and clustering in signed networks: a local to global perspective. J. Mach. Learn. Res. 15, 1 (January 2014), 1177–1213.

Girvan, M., & Newman, M. E. J. (2002). Community structure in social and biological networks. Proceedings of the National Academy of Sciences, 99(12), 7821–7826. 10.1073/pnas.122653799

Newman, M. E. J. (2006). Finding community structure in networks using the eigenvectors of matrices. Physical Review E, 74(3). 10.1103/physreve.74.036104

Rosvall, M., & Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences, 105(4), 1118–1123. 10.1073/pnas.0706851105

Raghavan, U. N., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. Physical Review E, 76(3). 10.1103/physreve.76.036106

Luxburg, U. von. (2006, August). A tutorial on spectral clustering. In (MPITechnical Reports No. 149). Tubingen: Max Planck Institute for Biological Cybernetics

Mendonca, I., Figueiredo, R., Labatut, V., & Michelon, P. (2015, September). Relevance of Negative Links in Graph Partitioning: A Case Study Using Votes from the European Parliament. 2015 Second European Network Intelligence Conference. 2015 Second European Network Intelligence Conference (ENIC). 10.1109/enic.2015.25

Page, Lawrence and Brin, Sergey and Motwani, Rajeev and Winograd, Terry (1999) *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report. Stanford InfoLab.

Kleinberg JM (1999) Authoritative sources in a hyperlinked environment. Journal of the ACM, 46(5):604-632.

Kunegis, J.; Stephan, S.; Lommatzsch, A.; Lerner, J.; Luca, E. D.; and Albayrak, S. (2010). Spectral Analysis of Signed Graphs for Clustering, Prediction and Visualization. In SDM, 559–570.

Jiawei Zhang, Qianyi Zhan, Lifang He, Charu Aggarwal, and Philip Yu. (2016). Trust hole identification insigned networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery*.

Täube, V. G. (2018). Cartwright/Harary (1956): Structural Balance: A Generalization of Heider's Theory. In Schlüsselwerke der Netzwerkforschung (pp. 101–104). Springer Fachmedien Wiesbaden. 10.1007/978-3-658-21742-6_24

Davis, J. A. (1967). Clustering and Structural Balance in Graphs. Human Relations, 20(2), 181–187. 10.1177/001872676702000206

Yang, B., Cheung, W., & Liu, J. (2007). Community Mining from Signed Social Networks. IEEE Transactions on Knowledge and Data Engineering, 19(10), 1333–1348. 10.1109/tkde.2007.1061

Gilbert, E. N. (1959). Random graphs. *Ann. Math. Statist.*, *30*, 1141–1144.

Erdős, P., & Rényi, A. (1961). On the evolution of random graphs. *Bull. Inst. Internat. Statist.*, *38*, 343–347.

Peixiang Zhao, Jiawei Han, and Yizhou Sun. (2009). P-Rank: a comprehensive structural similarity measure over information networks. In Proceedings of the 18th ACM conference on Information and knowledge management (CIKM '09). Association for Computing Machinery, New York, NY, USA, 553–562. 10.1145/1645953.1646025

Yu W., Le J., Lin X., Zhang W. (2012) On the Efficiency of Estimating Penetrating Rank on Large Graphs. In: Ailamaki A., Bowers S. (eds) Scientific and Statistical Database Management. SSDBM 2012. Lecture Notes in Computer Science, vol 7338. Springer, Berlin, Heidelberg. 10.1007/978-3-642-31235-9_15

Papachristos, Hureau, and Braga(2013) ."The Corner and the Crew: The Influence of Geography and Social Networks on Gang Violence." *American Sociological Review* 78 (3): 417–47.).

Vincent D. Blondel and Paul Van Dooren. (2003). Similarity matrices for pairs of graphs. In Proceedings of the 30th international conference on Automata, languages and programming (ICALP'03). Springer-Verlag, Berlin, Heidelberg, 739–750.