Karlsruhe Institute of Technology

Experiments on Simmelian Backbones for Clustering Social Networks

Master Thesis of

Michael Hamann

At the Department of Informatics
Institute of Theoretical Informatics (ITI)

Reviewer:	Prof. Dr. Dorothea Wagner
Second reviewer:	Prof. Dr. Peter Sanders
Advisor:	Dipl.-Inform./Dipl.-Math. Tanja Hartmann

Duration: 3rd December 2013 – 2nd June 2014

KIT – University of the State of Baden-Wuerttemberg and National Laboratory of the Helmholtz Association www.kit.edu

Danksagung

Als erstes möchte ich mich bei Frau Prof. Wagner für ihre Einladung, an ihrem Lehrstuhl meine Masterarbeit zu schreiben, bedanken. Nicht weniger möchte ich auch meiner Betreuerin Tanja Hartmann ganz herzlich für die hervorragende Betreuung dieser Arbeit danken. In den wöchentlichen Treffen nahm sie sich stets Zeit, auf meine Ideen einzugehen und wusste aber auch, wann der Rahmen der Masterarbeit erreicht war und ich mich eher der Dokumentation des bereits erreichten widmen sollte. Für die Unterstützung hierbei mit hilfreichen Korrekturen und Kommentaren möchte ich mich ganz besonders bedanken.

Ebenfalls bedanken möchte ich mich bei meinen Eltern, ohne deren Unterstützung ich dieses Studium wohl kaum so absolvieren hätte können. Dank gebührt auch meiner Freundin, die stets Rücksicht nahm, wenn ich an meiner Masterarbeit arbeiten wollte und keine Zeit für sie hatte und mich außerdem ebenfalls mit Korrekturhinweisen unterstützt hat.

Selbständigkeitserklärung

Hiermit versichere ich, dass die vorliegende Arbeit sowie der verwendete Quelltext meine eigene Arbeit sind. Arbeiten anderer wurde durch Quellenverweise oder Nennung der Autoren (Quelltext) eindeutig kenntlich gemacht.

Statement of Authorship

I hereby declare that this document and the used code have been composed by myself and describe my own work, unless otherwise acknowledged in the text.

Karlsruhe, 2. Juni 2014

Abstract

Regarding the task of detecting communities in a social network a problem for some community detection algorithms is the very dense network structure. We are exploring ways to solve this problem by applying the community detection algorithms on top of *Simmelian Backbones*, a concept introduced by Nick et al. [NLCB13] that identifies strong connections in the network and thus allows to filter the network.

We are experimentally evaluating these algorithms on a set of 100 Facebook networks from universities and colleges in the US that was introduced by Traud et al. [TMP11] and also used for the evaluation of Simmelian Backbones by Nick et al. [NLCB13]. The dataset also contains attributes like major, gender, high school, graduation year and dormitory. In a benchmark of four different community detection algorithms on these networks Lee et al. [LC13] tested how well these algorithms could detect the dormitory and graduation year structure using just the friendship information of the 40 smallest networks in the dataset.

In this work we will use one of these community detection algorithms, the *Louvain algorithm* [BGLL08] that maximizes *modularity*, a density-based quality measure, on all Facebook networks in the dataset. We restrict the networks to the current students and analyze the resulting clusterings by comparing them to the dormitories and the graduation years using an evaluation based on the *Jaccard index*.

We confirm the results of Lee et al. [LC13], which show that the Louvain algorithm is much better at the detection of the graduation years than at the detection of the dormitories, which only works for two networks according to our results. We show that with the help of Simmelian Backbones the Louvain algorithm can better identify the dormitory structure on some of the networks. We also introduce edge weights based on the definition of Simmelian Backbones that lead to similar results as the backbones, but avoid the problem that in the backbones some nodes are not connected anymore and we are thus getting a lot of small communities.

We also explore the possibility of speeding up the calculation of communities by executing the Louvain algorithm on the Simmelian Backbone. We show that the calculation of the backbone scales linearly with the number of edges and using the backbone dramatically improves the speed of the Louvain algorithm. Nevertheless measured on the original network there is a significant drop in the modularity of the resulting clustering and our implementation of the Simmelian Backbone algorithm cannot beat the running time of the Louvain algorithm on the original network.

Deutsche Zusammenfassung

Clusteralgorithmen, deren Ziel es ist, Communities in sozialen Netzwerken zu identifizieren, d.h. das Netzwerk in Teile zu teilen, die im Verhältnis zum Rest des Netzwerkes besonders stark verbunden sind, bereiten typischerweise die sehr dichte Verbindungsstruktur der sozialen Netzwerke Probleme. In dieser Arbeit wird versucht diese Probleme zu lösen, indem der Algorithmus zum Finden der Communities nicht auf das ursprüngliche Netzwerk sondern auf einem *Simmelian Backbone* des Netzwerkes angewendet wird. Simmelian Backbones sind ein von Nick et al. [NLCB13] eingeführtes Konzept zur Identifikation besonders wichtiger Verbindungen in einem sozialen Netzwerk um das Netzwerk dann auf die wichtigsten Verbindungen einzuschränken.

Die experimentellen Untersuchungen zu dieser Thematik werden in dieser Arbeit auf einem Datensatz von 100 Facebook-Netzwerken von US-amerikanischen Universitäten und Colleges die von Traud et al. [TMP11] präsentiert wurden durchgeführt. Dieser Datensatz

wurde auch schon von Nick et al. für die Evaluierung der Simmelian Backbones verwendet. Neben den Freundschaftsbeziehungen enthält der Datensatz auch anonymisierte Attribute wie Studienfach, die ehemalige Schule, das Abschlussjahr oder die Wohngruppe. Auf den 40 kleinsten Netzwerken dieses Datensatzes testeten Lee et al. [LC13] vier verschiedene Algorithmen zum Finden von Communities und überprüften, wie gut diese die Abschlussjahre oder die Wohngruppen finden, da diese eine Community-Struktur darzustellen scheinen.

Einer der getesteten Algorithmen ist der *Louvain-Algorithmus* [BGLL08], ein Algorithmus, der inkrementell versucht, eine Community-Struktur zu finden, die eine möglichst hohe *Modularity* aufweist. Modularity ist ein dichtebasiertes Qualtitätsmaß, das sich großer Popularität erfreut und auch recht gut der menschlichen Intuition entspricht was eine gute Community-Struktur ist [GGHW10]. Der Louvain-Algorithmus ist sehr populär, da er zum einen sehr schnell ist, zum anderen aber die gefundenen Community-Strukturen trotzdem eine sehr gute Qualität bezüglich Modularity aufweisen.

In dieser Arbeit wird die Struktur der 100 Facebook-Netzwerke anhand der Ergebnisse des Louvain-Algorithmus und der Wohngruppen- und Abschlussjahr-Attribute untersucht. Die gefundenen Communities werden dabei mit Hilfe des *Jaccard-Indexes* mit den Wohngruppen und Abschlussjahrgängen verglichen. Da die Netzwerke neben den aktuellen Studierenden auch die Daten von ehemaligen Studierenden und anderen Universitätsangehörigen enthalten, werden die Netzwerke auf die aktuellen Studierenden beschränkt um eine sinnvolle Analyse zu ermöglichen.

Die Ergebnisse von Lee et al. [LC13] werden dabei insofern bestätigt, als dass der Louvain-Algorithmus auf den ursprünglichen Netzwerken eher die Struktur der Jahrgänge erkennt, lediglich auf zwei Netzwerken entspricht die gefundene Struktur den Wohngruppen.

Die Anwendung des Louvain-Algorithmus auf dem Simmelian Backbone zeigt deutliche Verbesserungen für die Erkennung der Wohngruppen. Eine besondere Rolle spielt dabei der Jahrgang der Studierenden, die zum Zeitpunkt, als die Daten gesammelt wurden, gerade ihr Studium angefangen haben. Sie werden bei den meisten Netzwerken auch auf dem Backbone als eigenständige Community erkannt und scheinen auch sonst eine vom Rest der Studierenden eher abgetrennte Gruppierung zu bilden.

Aus dem Simmelian Backbone werden außerdem Definitionen für gewichtete Netzwerke abgeleitet, auf denen der Louvain-Algorithmus ähnliche Ergebnisse liefert wie auf dem Simmelian Backbone. Während auf dem Simmelian Backbone allerdings einige der Studierenden nicht mehr mit dem Rest der Studierenden verbunden sind und daher zwangsweise als eigenständige Gruppen erkannt werden, wird dieses Problem mit den gewichteten Netzwerken vermieden.

Abgesehen von der semantischen Qualtität der gefundenen Communities wird auch untersucht, ob Simmelian Backbones dazu verwendet werden können, den Louvain-Algorithmus zu beschleunigen, indem man ihn auf den Backbones ausführt. Die Ergebnisse zeigen, dass die Performance des Louvain-Algorithmus auf dem Simmelian Backbone deutlich zunimmt, häufig um einen Faktor zwischen 10 und 20, allerdings nimmt auch die Modularity der berechneten Communities auf dem ursprünglichen Netzwerk betrachtet ab, so dass nicht klar ist, wie praktikabel diese Beschleunigung ist, wenn der Fokus auf der Modularity liegt. Dazu kommt, dass die in der Arbeit verwendete Implementierung von Simmelian Backbones zwar linear in der Anzahl der Verbindungen in dem Netzwerk skaliert, allerdings trotzdem die Geschwindigkeit des Louvain-Algorithmus auf dem ursprünglichen Netzwerk nicht schlagen kann. Trotzdem könnte der Ansatz insbesondere für langsamere Algorithmen interessant sein.

Contents

1. Introduction

Regarding the analysis of a network, *clustering* algorithms help understanding the structure of the network by identifying parts of the network that are especially densely connected while sparsely connected to the other parts. These parts are also known as *clusters*, a set of such clusters that cover the whole network form a *clustering*. A clustering can be a partition of the network into distinct clusters but there are also algorithms that calculate clusters that may overlap. There are many applications for clustering algorithms for example in machine learning or in bioinformatics. In this work we consider social networks.

In a social network we have people as entities and social relationships between them. Clustering a social network thus means identifying groups of people/friends that are particularly closely connected, which means that they form a community in the network. This is why the task of clustering is also known as community detection.

A problem of social networks in the context of community detection is that social networks are usually very densely connected and everybody knows everybody via a short list of relationships. This is also known as *small world phenomenon*. This makes it hard to identify parts that are particularly densely connected compared to the rest of the network as basically everything is connected.

Not just in the context of social networks but also in general, a difficulty with clustering algorithms is, that there is no universal definition of what a good clustering is, as it is not clear how to evaluate a clustering. In a certain context there can be many clusterings which might make sense from a semantical point of view. Even though social networks are already a relatively specific application domain it is not clear how a social network should be clustered as each person can participate in many groups of friends – a person could have friends from their old school, from university, from their jobs, hobbies, neighbors, and so on. Each of these groups could be a valid cluster in the social network that makes sense semantically.

However, many clustering algorithms only take the network structure into account. Such clustering algorithms do not use semantical information which means that they need to derive the clustering from the structure of the network itself. As the structure of a real world network usually does not imply a specific clustering but many possible clusterings there is also not just one true clustering. There are many approaches for calculating a clustering, some algorithms model the network as a system of flows and search for bottlenecks in order to find cluster borders while others use ideas from information theory or density-based values that are optimized.

There are several measures for evaluating the quality of a clustering. A rather popular one is *modularity*, which is density-based. There are several algorithms that try to greedily optimize modularity, as optimizing modularity is NP-complete [BDG+08] and for larger networks it is thus computationally too expensive to calculate the clustering with the maximum modularity value. In this work we will focus on the *Louvain community detection algorithm* [BGLL08] which is very fast also for large networks and still achieves good modularity values.

In a benchmark, Lee et al. [LC13] tested how well four different community detection algorithms can identify the dormitory and graduation year structure in the 40 smallest of a set of 100 social networks from Facebook of US universities and colleges from September 2005. They compared the found communities to the dormitories and to the graduation years using a classifier. Their results show that very often either way too small or way too large communities are found. The Louvain community detection algorithm, which was also part of that benchmark, usually only found too large communities compared to the dormitories, which might be explained by the large number of connections.

Nick et al. [NLCB13] took a different approach for making the structure of social networks more obvious: By looking at the local structure of the people to which a person is connected they are able to determine the strength of a connection in the network. The *Simmelian Backbone* algorithm that they presented allows to remove the less important connections from the networks which makes the structure of the networks more obvious.

As this simplified network should be easier to cluster than the original network we decided to apply the Louvain community detection algorithm on top of the Simmelian Backbone. With this approach we hope to improve the correspondence of the resulting clusterings to the dormitories or graduation years while at the same reducing the running time.

We will first introduce our test dataset and our notation. In Section 2 we describe the Simmelian Backbone algorithm and its difficulties, in Section 2.2 the Louvain community detection algorithm is presented and we will also discuss the task of evaluating clusterings.

For the experimental evaluation we will describe the used implementations before we present a series of experiments in which we will examine the effect of the Simmelian Backbone on the Louvain community detection algorithm. In terms of quality of the clustering we will – similar to the experiments of Lee et al. – compare the communities that are found to the dormitory and graduation year structure. On the other hand we will also consider the differences in modularity of the resulting clusterings. Another aspect that we will discuss is the possibility of performance improvements by clustering the backbone instead of the original network.

1.1 Social Networks

When people interact with each other they build relationships, which can be expressed in a network that consists of a set of social actors with a set of dyadic ties between them. During life, humans get to know a lot of people – people from their school, neighborhood, their jobs, hobbies and so on.

As it is difficult and expensive to gather information about such social networks directly from everyday life, we will instead look at social networks that are represented by digital communication and online communities like Facebook.

In 2011, Traud et al. [TMP11] introduced a set of 100 Facebook networks that contain the complete inner-university friendship network of 100 universities and colleges in the United States. Facebook was widely and intensively used by the students at the time of the data collection (September 2005) and the data is complete in the sense that it is not just a sample of the networks. The networks do not only contain data of the current students but also of a subset of the former students that graduated in 2005 and the years before 2005. In addition to that also data from some faculty members is contained. The data contains several attributes from the profiles: gender, year of graduation, dormitory, primary academic major, secondary academic major, high school, and student/faculty status. This attribute data is not complete as not every person has provided these details in his or her Facebook profile. Apart from the graduation year and the gender, the attributes are anonymized in the sense that they are just numeric values without further meaning. The size of the networks differs significantly and ranges from 769 to 41554 people with 16656 to 1590655 friendships in between.

In Section 3.4 we will further explore the details of the dataset and how we dealt with the incomplete data.

These Facebook networks have been the basis of previous studies. Lee et al. [LC13] benchmarked variants of four community detection algorithms on the 40 smallest of these networks. Nick et al. [NLCB13] introduced Simmelian Backbones on the example of these networks.

1.2 Preliminaries

We use the terms network and *graph* interchangeably. A graph is a tuple $G = (V, E)$ with a set V of $n = |V|$ vertices that represent the entities (people in our case) and a set $E \subset V \times V$ of $m = |E|$ undirected edges that represent symmetric relationships between them. We use the notation of an unordered set $e = \{u, v\}$ for the representation of these edges. For one algorithm we also need a *directed graph* with directed edges which are represented as ordered pair $e = (u, v)$.

In some cases we will also consider a weighted, undirected graph $G = (V, E, \omega)$ where $\omega : E \to \mathbb{R}_0^+$ assigns weights to all edges. In order to simplify the following notations we define $\omega(e) := 1$ for all $e \in E$ for unweighted graphs.

A *clustering* $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ of a graph $G = (V, E)$ is a partition of V into non-empty clusters $C_1, C_2, \ldots, C_k$ with $\cup_{i=1,\ldots,k} C_i = V$.

We define the *neighborhood* $N(v)$ of a node $v \in V$ as $N(v) := \{u \in V \mid \{u, v\} \in E\}$. Then the *weighted degree* $\omega(v)$ of a node $v \in V$ is defined as

$$\omega(v) := \sum_{u \in N(v)} \omega(\{u, v\})$$

For a cluster $C \in \mathcal{C}$ let $\mathrm{in}(C) := \{\{u, v\} \in E \mid u, v \in C\}$ denote the set of edges inside C, the so-called *intra-cluster edges* of C. We define the *weight* $\omega(C)$ *of a cluster* $C \in \mathcal{C}$ as the weight of all intra-cluster edges:

$$\omega(C) := \sum_{e \in \mathrm{in}(C)} \omega(e)$$

We define the *weight* $\omega(\mathcal{C})$ *of a clustering* $\mathcal{C}$ as the sum of the weights of all clusters in the clustering:

$$\omega(\mathcal{C}) := \sum_{C \in \mathcal{C}} \omega(C)$$

Regarding the whole graph G as a single cluster which means that all edges are considered as intra-cluster edges, we define the weight $\omega(G)$ analogous to the weight of a cluster:

$$\omega(G) := \sum_{e \in E} \omega(e)$$

A very simple quality measure for a clustering is *coverage*, which is defined as the fraction of intra-cluster edges divided by the total number of edges. For weighted graphs we can simply replace the number of edges by their total weight:

$$\operatorname{cov}(\mathcal{C}) := \frac{\omega(\mathcal{C})}{\omega(G)}$$

Coverage has a range from 0 to 1, however the optimum value of 1 can be trivially obtained by using the whole graph as a single cluster, which means that for finding a good clustering simply optimizing coverage is unsuitable.

A widely used quality measure for a clustering is *modularity*, introduced by Newman and Girvan [NG04]. Newman [New04] also generalized the measure for weighted graphs. The idea of modularity is to use coverage, but to subtract the expected value of the coverage of the same community structure in a graph with the same nodes but random connections between them.

Let $\mathcal{C}$ denote a clustering of an undirected, weighted graph $G = (V, E, \omega)$. Then the *modularity* of the clustering $\mathcal{C}$ is defined as (see also [Gö10], Section 1.2.2):

$$\operatorname{mod}_{\omega}(\mathcal{C}) := \frac{\omega(\mathcal{C})}{\omega(G)} - \frac{1}{4 \cdot \omega(G)^2} \sum_{C \in \mathcal{C}} \left(\sum_{v \in C} \omega(v) \right)^2$$

We will discuss again modularity and the task of finding clusterings with good modularity in more detail in Section 2.2.

For comparing two sets A and B (for example two clusters) the Jaccard Index J is defined as

$$J(A, B) := \frac{|A \cap B|}{|A \cup B|}$$

In Section 2.4.2 we will extend this definition for comparing clusterings.

2. Clustering Simmelian Backbones of Facebook Data

The goal of *Simmelian Backbones* as introduced by Nick et al. [NLCB13] is to measure the strength of ties in a social network in order to restrict the network to the strong ties, the backbone. The definition of the strength of a tie that is used by Nick et al. is based on the sociological theory of Georg Simmel.

In his book "Soziologie: Untersuchungen über die Formen der Vergesellschaftung" [Sim08] (English translation included in "The sociology of Georg Simmel" [Sim50]), Georg Simmel states that there is a fundamental difference between a relationship between two people and a group of three or more people which is much more important than the number of people in the group. This means that triangles play a very important role when studying the structure of a social network.

Each group two individuals share has an impact on their relationship as each group has its own social rules and expectations. From this observation, Dekker [Dek06] derives different Simmelian Tie measures that are based on different interpretations of the importance of group number and size. The interpretation that is the basis of Simmelian Backbones is the interpretation that the most important factor is the number of individuals with whom two individuals share a group. This does not regard the number or the size of the groups these individuals are part of.

A different formulation is that the important factor for a tie between two people is the number of common friends they have. This is also based on the idea that each common friend enforces the similarity between the two people by observing them and comparing their behavior with norms. A common friend could tell others if there should be any inconsistencies. Each common friend could introduce a new norm that both of them need to fulfill which further constrains the two people and thus underlines their similarity.

In a more formal way this means that we are looking at the number of triangles $T(e)$ an edge in an undirected graph G is part of, this number of triangles is the *strength* of the tie:

$$T : E \to \mathbb{N}_0, \{u, v\} \mapsto |N(u) \cap N(v)|$$

Based on these concepts, Nick et al. [NLCB13] introduce the concept of the *rank-ordered neighborhood* of a node which are the neighbors of a node sorted from strongest to weakest

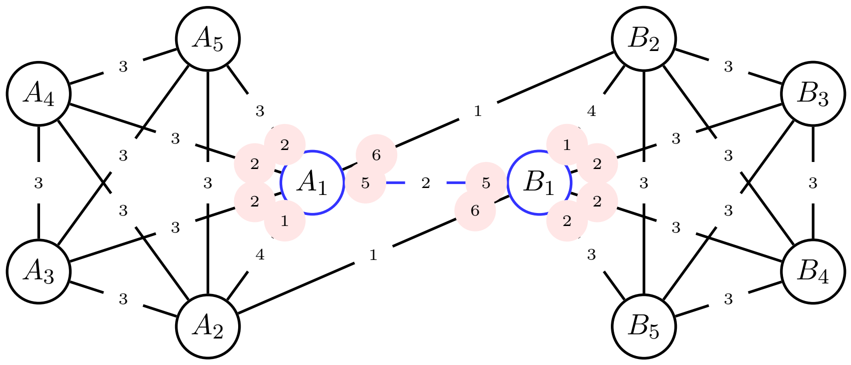


Figure 2.1: Example graph with tie strength as labels in the middle of the edges and the ranks of the neighbors of A_1 and B_1 as edge labels next to the two nodes.

tie between the node and the neighbor. They refer to this graph with the rank-ordered neighborhoods as the *ranked neighborhood graph.*

Each neighbor is assigned a rank where the smallest number is the highest ranked neighbor. The rank of a neighbor is its position in the rank-ordered neighborhood unless it is connected via a tie of the same strength as the previous neighbor in which case it gets the same rank as the previous neighbor. Note that this means that there can be zero, one, or more than one neighbors with a certain rank. Mathematically, the function $\text{rank}(u, v)$ that returns the rank of a neighbor v of a node u is defined as

$$\text{rank} : V \times V \to \mathbb{N}^+, (u, v) \mapsto |\{w \in N(u) \,|\, T(\{u, v\}) < T(\{u, w\})\}| + 1$$

An example of this ranking can be seen in Figure 2.1 for the neighbors of the nodes A_1 and B_1, the ranks are the numbers next to these nodes, the strength of the ties is denoted by the labels in the middle of the edges.

In order to be able to filter the network for obtaining the backbone Nick et al. [NLCB13] introduce a measure for the tie redundancy for which they use the term *embeddedness*.

In the ranked neighborhood graph, an edge is considered to be *(relatively) strongly embedded* if the two incident nodes have many relatively strongly connected common neighbors. In other words this means that an edge is strongly embedded if there is enough overlap between the top-ranked neighbors of the two incident nodes.

The Simmelian Backbone of a graph is then defined as the graph that consists of the original set of nodes and all edges of the original graph that are strongly embedded.

For determining the degree of embeddedness of an edge Nick et al. present a parametric and a nonparametric variant.

The parametric variant uses a parameter k that specifies the maximum rank that is included. This means that only edges that connect a node to one of its top k neighbors are considered. For an edge $e = \{u, v\}$ this means that the equation

$$\min(\text{rank}(u, v), \text{rank}(v, u)) \leq k$$

must hold.

In order to simplify the notation let $N_k(v)$ denote the top k neighbors of v:

$$N_k(v) := \{w \in N(v) \,|\, \text{rank}(v, w) \leq k\}$$

The degree of the embeddedness $\mathrm{emb}_k(e)$ of an edge $e = \{u, v\}$ is then defined as the overlap among the top k neighbors of the two incident nodes u and v. For the calculation of the overlap, u and v are considered the same, or in other words: If u has v and v has u as top k neighbor this counts as a common neighbor. This means that for example in Figure 2.1 the nodes A_1 and B_1 have three common neighbors among their top 6 neighbors and one common neighbor among their top 5 neighbors. The formal definition of the embeddedness in the parametric variant is:

$$\mathrm{emb}_k(\{u, v\}) = \left| \left\{ \begin{cases} u & \text{if } w = v \\ w & \text{otherwise} \end{cases}, w \in N_k(u) \right\} \cap N_k(v) \right|$$

The nonparametric variant is to determine for each edge the value k that maximizes the overlap among the top k neighbors of the two incident nodes normalized by the size of the union of the two neighborhoods. In other words the value for k is determined that maximizes the Jaccard index for the top k neighbors of the two nodes. The two nodes u and v are again identified with each other for the calculation of the overlap. This means that the nonparametric embeddedness $\mathrm{emb}(e)$ of an edge $e = \{u, v\}$ is defined as

$$\mathrm{emb}(\{u, v\}) = \max_{k \in \mathbb{N}} J \left(\left\{ \begin{cases} u & \text{if } w = v \\ w & \text{otherwise} \end{cases}, w \in N_k(u) \right\}, N_k(v) \right)$$

Note that the values of k that need to be considered are limited by the maximum of the degrees of u and v.

In order to obtain a Simmelian Backbone the graph can be filtered by specifying a minimum value for the embeddedness, in the case of the parametric variant this is a required overlap value, in the case of the nonparametric variant a required Jaccard index.

Furthermore, the parametric variant can be separated into two steps: Filtering the graph by only including the edges that connect a node to one of its top k neighbors, and filtering the graph by removing the edges whose incident nodes do not have enough common neighbors among their top k neighbors.

A variant of the parametric approach is to skip the first step and to only filter the edges by the number of common neighbors among the top k neighbors of the two incident nodes. This has the advantage that for the value of 0 common neighbors this is equivalent to the original graph such that for a fixed value of k we can have a smooth transition from the original graph to the backbone by modifying just one parameter. In this work we will use this variant of Simmelian Backbones. This variant was not presented by Nick et al. [NLCB13] but is a possible configuration in their reference implementation that is available as part of the Visone application [Nic13]. The variant can be executed by deselecting the "conditioned" parameter.

Nick et al. [NLCB13] show at the example of the dataset of the 100 Facebook networks that Simmelian Backbones make it easier to see the community structure that is given by the dormitories in visualizations. They show that especially for the attributes dormitory, graduation year and gender more edges between two different attribute values than between the same attribute value are removed. This means that the homophily is increased in the backbones compared to the original networks. Nick et al. [NLCB13] used the parametric variant requiring an overlap of 5 among the top 10 neighbors which are also the default parameters in their implementation in Visone [Nic13].

As we would like to apply clustering algorithms on top of Simmelian Backbones, an important question is if the backbone still contains all necessary intra-cluster edges or, if it is maybe even possible that subgraphs that should be contained in one cluster according to the original graph structure could be split into multiple connected components of the backbone.

2.1 Connected Components

As soon as we require more than zero matches among the top-ranked neighbors of the nodes that are incident to an edge, the graph can be split into multiple connected components. If among the top-ranked neighbors at least i neighbors need to be in common, edges that are incident to a node of degree less than i will be removed and thus nodes of degree less than i will become isolated nodes. Also a node of a higher degree can become isolated, for example if its neighbors have a low degree.

While on the one hand isolated nodes could mean that these nodes were not really belonging to a group of other nodes it could also mean that the graph is very sparse in a certain area, but still has a community structure, or that the community structure contains very small communities. An extreme example is a fully connected subgraph of four nodes that will be split into isolated nodes if at least four common neighbors are required.

In general we expect that Simmelian Backbones will destroy parts of the graph that have a low connectivity and will be beneficial for larger communities that have a lot of intra-community connections. In first experiments we noticed that indeed especially for higher values for the required overlap like an overlap of 5 among the top 10 neighbors a lot of small connected components and singletons appear in the backbone. The more the overlap parameter is increased, the worse the problem gets even though semantically the kept edges actually seem to make sense as also Nick et al. [NLCB13] showed.

For the evaluation of a clustering that is calculated on the Simmelian Backbone of a network but evaluated on the original network isolated nodes that were not isolated in the original network are a negative factor. The same is true for small components of two or more nodes which are also usually not what one is looking for.

As this cannot be avoided when applying the Simmelian Backbone algorithm before clustering we decided to derive edge weights from the ranked neighborhood graph and to cluster the weighted graph. We introduce two kinds of edge weights: The first one is simply the overlap between the top k neighbors increases by one in order to avoid edges with weight zero. The second one is the square of this overlap again increased by one. The intuition behind the squaring of the overlap is that higher overlaps are considerably more important than lower overlaps, and we want to significantly increase the cost of separating two nodes with a very high overlap while keeping the cost of separating two nodes with a relatively low overlap low.

A different approach that we will also evaluate experimentally is to post-process the clustering that was calculated on the backbone by merging clusters that consist of a single node into larger clusters according to the graph structure in the original network.

2.2 Clustering

The main goal of clustering algorithms is to identify parts of the graph that are, compared to the rest of graph, particularly well connected. In the context of this work we will only consider clusterings where clusters must not overlap, this means that a clustering partitions the graph into distinct clusters. In this setting, the nodes of a good cluster should be more connected to the nodes in their own cluster than to the nodes in other clusters.

An algorithm that tries to find the maximum ratio between intra-cluster and inter-cluster edges will return the connected components of the graph or even the whole graph as result, which is obviously not what we want so we need another formal definition.

Unfortunately there is no perfect definition of what a good clustering is as this also depends a lot on the data that shall be clustered, for example there can be very different cluster sizes and if the graph has an actually overlapping cluster structure an algorithm that calculates distinct clusters needs to choose one of them. Nevertheless there is a measure that is very popular: Modularity, for a definition we refer to Section 1.2. A problem with modularity optimization is that modularity has a resolution limit (see also Fortunato et al. [FB07]) which means that in very large networks it is not possible to identify smaller clusters by optimizing modularity.

Because of the popularity of modularity and our hope to address the problem of the resolution limit with the help of Simmelian Backbones by improving the ratio of intra-community links to inter-community links we will focus on modularity-based clustering.

As optimizing modularity is NP-complete [BDG$^+$08] it is in practice not possible to calculate the clustering with the exact maximum modularity value for larger graphs. This means that the algorithms that are used in practice are heuristics that try various greedy optimization techniques in order to calculate a clustering with a good modularity value.

2.3 Louvain

A technique that performs rather well both in terms of performance on large graphs as well as in terms of quality is the Louvain method of community detection that optimizes modularity iteratively in a greedy algorithm. The Louvain method of community detection was introduced by Blondel et al. in 2008 [BGLL08] and got its name from the catholic university of Louvain where all authors were when they published the algorithm.

The algorithm starts with a clustering in which each node is in its own cluster and then iteratively tests for each node if the modularity can be improved by moving it into a neighboring cluster, and moves the node if the modularity can be improved. When the modularity cannot be improved anymore by moving nodes into neighboring clusters the clusters are contracted to a new graph where the same process is executed again. In order to keep the modularity value the same for the contracted graph, a self-loop is added to each node with the weight of the intra-cluster edges and edges between two clusters get the sum of the weights of the edges between the two clusters.

2.4 Evaluation

For the evaluation of a clustering the first question we have to answer is, what we expect of a good clustering. With a community detection method like the Louvain modularity optimization method one could, of course, use the modularity of the resulting clustering as optimization criteria. Even though Görke et al. found in their evaluation [GGHW10] that modularity complies rather well with human intuition of clustering quality, it is still the question if it actually matches what we expect in the application domain which are social networks in our case.

Lee et al. [LC13] chose a different approach in their evaluation: They compared the found communities to the communities that are induced by the dormitory and graduation year attribute. As Nick et al. [NLCB13] show that on the Simmelian Backbone the edge structure correlates even more with these attributes, we decided to use an evaluation similar to the one of Lee et al.

For comparing the found clusters to the ground truth communities that are induced by the dormitory and graduation year attributes Lee et al. [LC13] used a classifier. Since they also evaluated clustering algorithms that calculate overlapping clusterings, a node can be possibly assigned to multiple clusters. Using these assignments of nodes to clusters their classifier tries to infer the ground truth. The quality measure that Lee et al. evaluated is how well it was able to execute this task.

The Simmelian Backbone gets very sparse for larger overlap values and contains a lot of small connected components and singletons. They are all becoming individual clusters as we will apply the Louvain algorithm on top of the Simmelian Backbone. This means that the classifier will get a lot of features which increases the running time. This is a problem that already Lee et al. [LC13] had to cope with, as one of the clustering algorithms they evaluated produced a lot of small clusters, their benchmark needed months to be evaluated. Furthermore detected clusters that are smaller than the ground truth communities are not negatively evaluated by the classifier, as these small clusters allow the classifier to infer the ground truth communities just as well as a cluster that perfectly matches a ground truth community. As this is a property that we do not want for our evaluation and we did not want to spend months running the experiments, we decided against using this classifier-based approach.

Another approach that is also mentioned by Lee et al. [LC13] is to compare the found clusters to the ground truth communities using *normalized mutual information* (NMI). The original definition of normalized mutual information cannot be used for overlapping clusterings. There are definitions for overlapping clusterings that are for example discussed by McDaid et al. [MGH11] but Lee et al. were not satisfied with them, as they do not take interactions between community memberships into account. This is why they decided against using normalized mutual information for their evaluation. As the Louvain algorithm calculates non-overlapping clusterings we decided to give normalized mutual information a try.

2.4.1 Normalized Mutual Information

Our first attempt at comparing the detected clusters to the ground truth communities was using normalized mutual information. Normalized mutual information as for example defined by Ana et al. [AJ03] can be used for comparing a clustering A to a clustering B. Using the notation of Kuncheva et al. [KH04] where N denotes the total number of nodes, $N_{i.}$ the number of nodes in cluster i in the first clustering with c_A clusters, $N_{.j}$ the number of nodes in cluster j in the second clustering with c_B clusters and N_{ij} the number of nodes that are both in cluster i in the first clustering and in cluster j in the second clustering it is defined as follows:

$$\mathrm{NMI}(A,B) = \frac{-2\sum_{i=1}^{c_A}\sum_{j=1}^{c_B} N_{ij}\log\left(\frac{N_{ij}N}{N_{i.}N_{.j}}\right)}{\sum_{i=1}^{c_A} N_{i.}\log\left(\frac{N_{i.}}{N}\right) + \sum_{j=1}^{c_B} N_{.j}\log\left(\frac{N_{.j}}{N}\right)}$$

However, during our experiments, we noticed that the values of normalized information for clusterings that consisted (almost) exclusively of singletons were much higher than we expected, and in many graphs also much higher than the values that we got for non-degenerated clusterings.

For example if we take a graph G with n nodes with a ground truth community structure T that consist of k equally sized clusters and have a clustering $\mathcal{C}$ that consists of n singletons we get the following normalized mutual information:

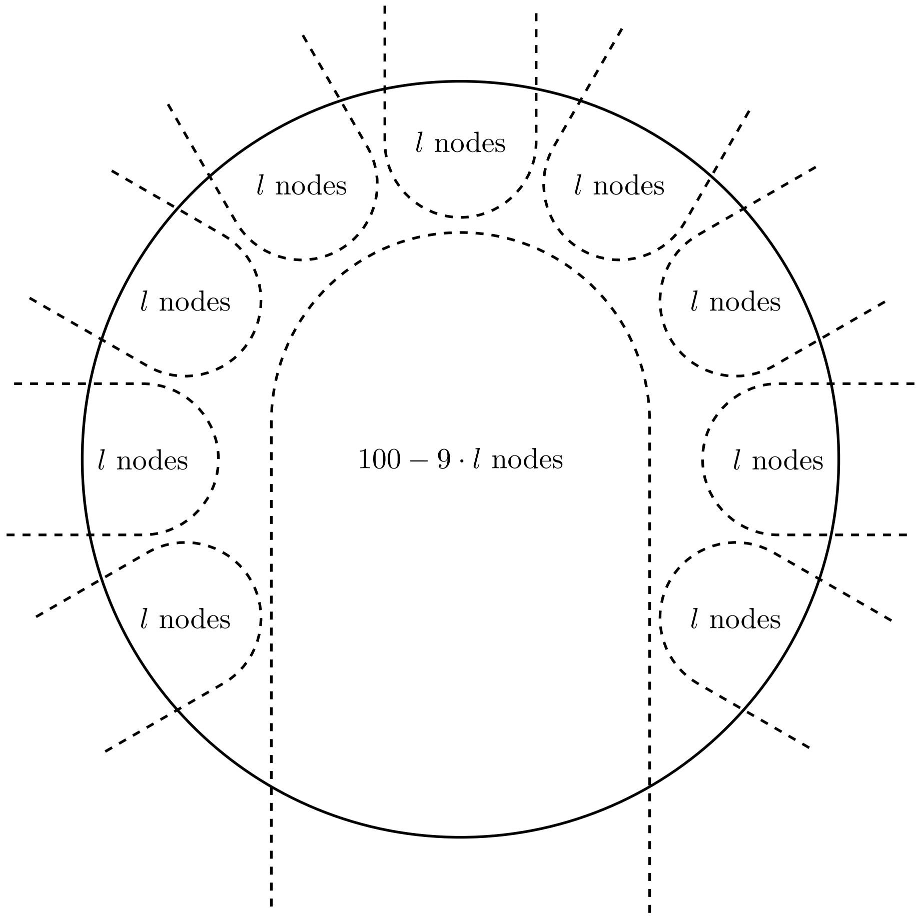


Figure 2.2: Schematic drawing of a ground truth community with all overlapping clusters (dashed) in the second scenario.

$$\begin{aligned}
\mathrm{NMI}(T, \mathcal{C}) &= \frac{-2 \sum_{i=1}^{k} \sum_{j=1}^{n/k} 1 \log\left(\frac{1 \cdot n}{n/k \cdot 1}\right)}{\sum_{i=1}^{k} n/k \log\left(\frac{n/k}{n}\right) + \sum_{j=1}^{n} 1 \log\left(\frac{1}{n}\right)} \\
&= \frac{-2 \cdot k \cdot n/k \log(k)}{k \cdot n/k \log\left(\frac{1}{k}\right) + n \log\left(\frac{1}{n}\right)} \\
&= \frac{-2 \cdot n \log(k)}{n \log\left(\frac{1}{k}\right) + n \log\left(\frac{1}{n}\right)} \\
&= \frac{2 \log(k)}{\log(k) + \log(n)}
\end{aligned}$$

If we choose $n = 1000$ and $k = 10$, we get 0.5 as normalized mutual information even though intuitively one would rather expect a value close to 0.

On the other hand, as a more realistic scenario, we consider the same ground truth clustering T with 10 communities with 100 nodes each and compare them to a clustering $\mathcal{C}'$ with 10 clusters of 100 nodes each. We assign each cluster in $\mathcal{C}'$ to a ground truth community in T but the overlap is not perfect. In Figure 2.2 such a ground truth community is symbolized by the circle. It is overlapped by l nodes in each cluster in $\mathcal{C}'$ that is not assigned to it, the other $100 - 9 \cdot l$ nodes are in the assigned cluster. For example for $l = 1$ each ground truth community overlaps with 91 nodes of a cluster in $\mathcal{C}'$ and each of the

l	0	1	2	3	4	5	6	7	8	9	10
NMI	1	0.783	0.624	0.489	0.373	0.272	0.185	0.113	0.055	0.016	0

Table 2.1: The NMI values for the different values of l

remaining 9 nodes is in a different cluster in $\mathcal{C}'$. This gives us the following equation for the normalized mutual information of T and $\mathcal{C}'$:

$$\begin{aligned}
\mathrm{NMI}(T,\mathcal{C}') &= \frac{-2\sum_{i=1}^{10}\left((100-9\cdot l)\log\left(\frac{(100-9\cdot l)\cdot 1000}{100\cdot 100}\right)+9\cdot l\log\left(\frac{l\cdot 1000}{100\cdot 100}\right)\right)}{\sum_{i=1}^{10}100\log\left(\frac{100}{1000}\right)+\sum_{j=1}^{10}100\log\left(\frac{100}{1000}\right)} \\
&= \frac{-2\cdot 10\left((100-9\cdot l)\log\left(\frac{100-9\cdot l}{10}\right)+9\cdot l\log\left(\frac{l}{10}\right)\right)}{2\cdot 10\cdot 100\log\left(\frac{1}{10}\right)} \\
&= \frac{(100-9\cdot l)\log\left(\frac{100-9\cdot l}{10}\right)+9\cdot l\log\left(\frac{l}{10}\right)}{100\log(10)}
\end{aligned}$$

For the different values of l we get the values in Table 2.1. This shows that even if we assume that 73 of the 100 nodes in a ground truth community are matched by a cluster and just 27 are assigned to the wrong cluster and the number of clusters is correct, we only get a normalized mutual information of 0.489, which is less than the value we achieve if we simply use singletons as clustering. As for higher overlap values in the backbone we expect more and more singletons in the clusterings that are calculated on the backbone, we would like to have a measure for the evaluation that gives us lower values for singletons than for clusterings that match at least partially. This is why we decided against using normalized mutual information.

2.4.2 Jaccard Index

After discovering these problems we wondered what should be properties of a good clustering also in the context of Simmelian Backbones. A general property should be that for each ground truth community there should be one cluster that matches this community. It could happen that this cluster is too large or too small, these are the cases where the quality measure should get worse.

A possible measure is thus to find for each ground truth community a cluster that has the highest value according to a certain quality measure. Possible measures for this include the *Jaccard index* which is the size of the intersection of cluster and ground truth community divided by the size of the union of them. Another possible measure is the *F1 score* which is defined as the geometric mean between *precision* (the size of the intersection divided by the size of the cluster) and *recall* (the size of the intersection divided by the size of the ground truth community). We decided to use the Jaccard index because of its simplicity.

When combining the Jaccard index of the different ground truth communities one can use the average of the individual values. The problem with this is, that for ground truth communities of very different sizes changing the assignment of a node to a different cluster has very different effects depending on the size of the cluster. As the ground truth communities that we will use – the dormitories and the graduation years of the Facebook data – have sometimes very different sizes, so we decided to use a weighted average that is weighted by the size of the ground truth communities. For a ground truth community structure $\mathcal{C}$ and a clustering $\mathcal{C}'$ we thus define the *(unidirectional) Jaccard index* $J(\mathcal{C},\mathcal{C}')$:

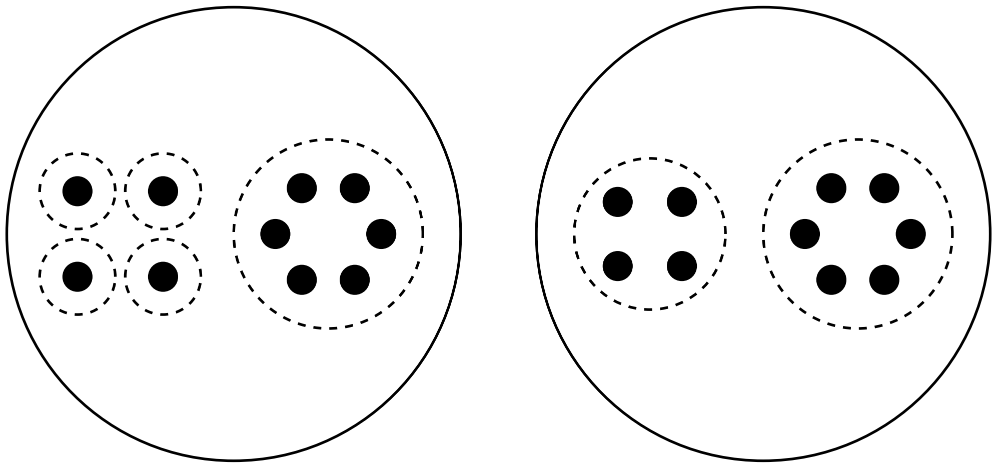


Figure 2.3: Two times the same ground truth community with ten nodes matched by two different clusterings (dashed).

$$J(\mathcal{C}, \mathcal{C}') := \frac{1}{|V|} \sum_{C \in \mathcal{C}} |C| \max_{C' \in \mathcal{C}'} J(C, C')$$

Note that $J(\mathcal{C}, \mathcal{C}')$ is in most cases not the same as $J(\mathcal{C}', \mathcal{C})$. While this measure is very intuitive, as it measures directly how well the best-matching clusters cover the ground truth communities, it has a problem: It ignores all clusters that are not the best-matching cluster for a ground truth community. For example consider the example in Figure 2.3: The two ground truth communities, which consist of ten nodes each, both overlap with a cluster of six nodes, and, additionally, on the left side with four singleton clusters and on the right side with another cluster of four nodes. Both times the Jaccard index is 0.6 even though intuitively the clustering on the right side is a better match.

This problem gets even worse when one also wants to consider overlapping clusters – this is possible with our definition of the unidirectional Jaccard index but one can easily ignore a lot of clusters. By using the power set of the nodes as clustering one can even get the maximum Jaccard index of 1 without providing a meaningful clustering. While this is no problem for us, as we only consider distinct clusters, we are still also interested in taking the clusters into account that are not selected as best-matching cluster of a ground truth community.

A method for avoiding these problems,which we adopted from Yang et al. [YL13, YML14], is to use the same procedure also in the other direction, i.e. with swapped roles of the clusters and the ground truth communities and to use the average of both values as measure. While Yang et al. used the unweighted average of the Jaccard indexes of the clusters we will use the weighted average in both directions. We decided to use the weighted average as we want the measure to be relative to the number of nodes that do not match, and not to the number of clusters, as the number of clusters can quickly increase when clustering on the backbone. We call this measure that uses the Jaccard index in both directions the *bidirectional Jaccard index* J_{bidi}:

$$J_{\text{bidi}}(\mathcal{C}, \mathcal{C}') := 0.5 \left(J(\mathcal{C}, \mathcal{C}') + J(\mathcal{C}', \mathcal{C}) \right)$$

For the evaluation we will use both the unidirectional and bidirectional Jaccard index in order to see if the clusters that are not selected as best-matching clusters for the ground truth communities match worse or not. In the case where the detected clusters are larger than the ground truth communities this can also be the other way around, in that case the roles of the ground truth communities and the detected clusters are swapped.

3. Implementation Details and Experiments

In the next section we describe the implementation of the Simmelian Backbone algorithm, in Section 3.2 the clustering algorithm and the setup of the evaluation. In Section 3.3 we will present the experimental setting and in the following sections our results.

3.1 Implementation of Simmelian Backbones

A reference implementation of the Simmelian Backbone algorithm is available as part of the Visone application [Nic13].

As it is not possible to automate the execution of the Simmelian Backbone extraction in Visone, which is crucial in order to be able to perform tests with a larger number of graphs and parameters, we decided against using Visone.

Mark Ortmann kindly provided us the source code of the Simmelian Backbone implementation in Visone which is based on yFiles [WEK02], a commercial graph library that cannot be used for free. As we prefer free and open source software solutions we decided to implement the Simmelian Backbone calculation ourselves using C++ and the LEMON [LEM] graph library.

For the Simmelian Backbone algorithm we first need to count for each edge the number of triangles it is part of. Ortmann et al. [OB14] presented an overview of state-of-the art triangle counting algorithms. The variant "L+n" which is based on the triangle counting algorithm that was presented by Chiba and Nishizeki in 1985 [CN85] scored best in their benchmark. Mark Ortmann also provided us the (C++) source code of their implementation of the variant "L+n", but it could not be used directly as it uses specialized data structures and only calculates the total number of triangles, and not the number of triangles for each edge.

In Algorithm 3.1 we present the triangle counting algorithm in the variant "L+n" as it was presented by Ortmann et al. [OB14]. The main idea of the algorithm is to create a directed acyclic graph by sorting the nodes by the degree of the node and to direct all edges into the direction of the node with the higher degree. That way, as shown in Figure 3.1, each triangle has a unique node, t_3 in the figure, that has only incoming edges.

Instead of sorting the nodes one can also simply direct the edges into the direction as specified by the sorting order. In line 1 we define this directed graph. Note that in order

Algorithm 3.1: Triangle counting algorithm

```
Input: Graph $G = (V, E)$ with node ids
Output: Triangle counts $T : E \to \mathbb{N}$
1  $G' \leftarrow \left(V, \left\{ \begin{cases} (u,v) & \text{if } \deg(u) < \deg(v) \text{ or } \deg(u) = \deg(v) \text{ and } \mathrm{id}(u) < \mathrm{id}(v) \\ (v,u) & \text{if } \deg(u) > \deg(v) \text{ or } \deg(u) = \deg(v) \text{ and } \mathrm{id}(u) > \mathrm{id}(v) \end{cases} \middle| \{u,v\} \in E \right\}\right)$;
2  $M : V \to \{0,1\}, v \mapsto 0$;
3  foreach $v \in V$ do
4      foreach $(u,v) \in E'$ do
5          $M(u) \leftarrow 1$;
6      foreach $(u,v) \in E'$ in ascending degree/node id order of $u$ do
7          foreach $(u,w) \in E'$ where $w$ is before $v$ in degree/node id order do
8              if $M(w) = 1$ then
9                  $T(\{u,v\}) \leftarrow T(\{u,v\}) + 1$;
10                 $T(\{u,w\}) \leftarrow T(\{u,w\}) + 1$;
11                 $T(\{w,v\}) \leftarrow T(\{w,v\}) + 1$;
12         $M(u) \leftarrow 0$;
```

to get the correct edge directions one needs a total order and thus needs to take the node id into account, as node degrees are not necessarily unique.

In the directed graph we then iterate over all nodes. For each node we iterate over all incoming edges and mark the source nodes of these edges, in the algorithm this is in line 5. In Figure 3.1 this means that for example when we reach t_3 we mark t_1 and t_2. We then iterate again over these incoming neighbors and for each such neighbor we check for all outgoing neighbors if they are marked (line 8 in the algorithm) and if yes we found a triangle. In Figure 3.1 this means that from t_1 we find t_2. Because of the unique roles of the nodes in the triangles we will find each triangle only once.

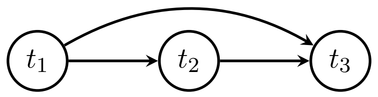


Figure 3.1: In the directed graph, each node in a triangle has a unique role.

If we order all nodes from lower to higher degree from left to right we only mark nodes that are left of the current node v. This is why in line 7, when we iterate over the outgoing edges, we do not need to check nodes that are in the sort order after v and thus right of v. As we iterate over the incoming edges in line 6 in ascending order and all outgoing edges point in the same direction, we will never check the current neighbor again, which is why in line 12 we can directly delete the mark.

There are further optimizations that Ortmann et al. applied in their implementation. The last node in the loop over the outgoing edges in line 7 does not have any marked neighbors anymore, as there were no marks between that node and v, which means that we can skip the check for the mark in line 8 for that last node. Also the first node u in the order does not need to be marked in line 5 as it can never be the target of an outgoing edge because all outgoing edges point to later nodes in the order.

Our implementation is a simplified version of this algorithm, we implemented the general algorithm but we did not sort the edge iterators. For the sake of simplicity we also omitted the previously described optimizations and we also needed to put the step where the mark is deleted in line 12 into an extra loop after the counting of the triangles. We also simply copied the whole graph in order to get an optimized directed graph structure instead of manually maintaining just the lists of in- and outgoing edges for each node. As the triangle counting executed on a normal computer takes less than ten seconds for all graphs in our benchmark, and for most of them even less than a second, we did not try to apply further optimizations.

The implementation of the filtering of the edges for obtaining the Simmelian Backbone is directly based on the implementation in Visone. It uses the same parameters and provides the exactly same results.

3.2 Used Clustering Algorithm and Implementation of the Evaluation

We used the implementation of the Louvain method of modularity optimization [BGLL08] that is provided by the authors.

We developed some C++ programs that use the Simmelian Backbone and Louvain implementation in order to generate various statistics in the form of JSON files and also very simple text files with the backbone and cluster information.

For processing these output files we developed various Python scripts that store various properties of the different backbones and clusterings in a SQLite database and use matplotlib [Hun07] for generating plots of them. This allows us to create plots of the different results by just changing an SQL query and a few parameters.

Furthermore we generated GEXF graph files for Gephi [BHJ09] including properties like the cluster id for various clusterings, the dormitory and graduation data and the overlap of the top 10 neighbors from the ranked neighborhood graph, which we used for exploring and visualizing individual graphs.

3.3 Experimental Setting

For our experiments we have used all 100 Facebook networks and executed the Simmelian Backbone algorithm with parameter $k = 10$ and required an overlap from 0 to 14 among the top ten ranks of the neighbors of adjacent nodes. Note that even though we only consider the top ten ranks this can include more than ten neighbors if enough neighbors have the highest rank below eleven. On all these backbones we have executed the Louvain clustering algorithm and calculated various statistics. In addition to the Louvain clusterings we also regarded the dorm and year assignments as clusterings both on the backbones and on the original networks.

For the analysis of the results we will mainly present a series of plots that we generated from the statistics that we collected of the examined clusterings. Each plot has the list of networks at its x axis and the particular value for each network is plotted along the y axis. As already discussed in Section 2.2 modularity optimization is not independent of the size of the network. In order to give an idea how big each specific network is, we added a second y axis at the right side of the plot with a different scale, which we used for plotting the number of edges in each network.

In order to measure how well we achieve our goal of the detection of communities that match the dormitory or graduation year structure we will use the uni- and bidirectional Jaccard index that we introduced in Section 2.4.2. We will also visualize individual networks in order to better understand the actual differences between the different detected communities and the dormitory or graduation year structure.

Since the data contains some artefacts that are concealing the structure of the dormitories and graduation years, we will first restrict the dataset to a well-defined subset that allows us to execute this task. As we want to examine the effect of the Simmelian Backbone on the Louvain algorithm, we will first apply the Louvain algorithm directly on the network and then, in the second part of the experiments, we will use the Simmelian Backbone algorithm as preprocessing step.

3.4 The Dataset

Some of the 100 Facebook networks contain, apart from a giant component, also some small components of just two or three nodes. The people in these small connected components belong to groups like graduation years or dormitories that are not distinct from the giant component, which means that their separation is not meaningful from a semantical point of view with respect to the attributes graduation year/dormitory. As this only concerns very few nodes, we simply exclude them and only consider the giant component of every network. As we always applied this basic preprocessing step we always mean the giant component of a network if we speak of the "original" network.

The dataset contains an attribute that indicates the student/faculty status which can have different values, one is "active student", another one is "previous student" but there are also several other possible values whose meaning is not clear from the provided anonymized data. This means that we cannot be sure about the status of these people.

At the point of the data collection in September 2005 the graduation years of the current students are mainly 2006, 2007, 2008 and 2009. In one network (Northeastern19) there is also a large number of students with graduation year 2010. In all other networks there are, compared to the number of students who graduate for example in 2009, only very few students who have provided the year 2010 as their graduation year, in some networks even less than ten students. One can only speculate if these students just entered the wrong year or if they are actually in some special program. Looking at the networks themselves these students with graduation year 2010 do not form any kind of group but are rather randomly distributed among the other students when using a force-based layout that groups the current students in mostly homogeneous graduation years. The graduation year 2005 plays a special role, too, as most of these students are not marked as active students anymore but a minority still is. We assume that this is simply because they had not updated their data in Facebook yet when the data was collected.

Apart from a smaller but still relatively significant number of former students who graduated in the years before 2005 there are also some graduation years like 1968 with just one person in the dataset that is nevertheless well connected to the current students which might be a faculty member. This means that the Facebook networks do not only contain data of current students but also data of previous students and possibly faculty members. Hence for some dormitories we do not just have data of the students that were living in them at the point of the data collection but also data of those that lived there before.

We decided to restrict the data to the current students that graduate after 2005 and furthermore in all networks where the number of students that graduate in 2010 was not significant, we also excluded those.

As we wanted to keep as many students as possible that simply forgot to enter their data, we kept everybody who is either a student or whose student status is missing and whose graduation year is after 2005 and before 2010 if 2010 is not considered and just after 2005,

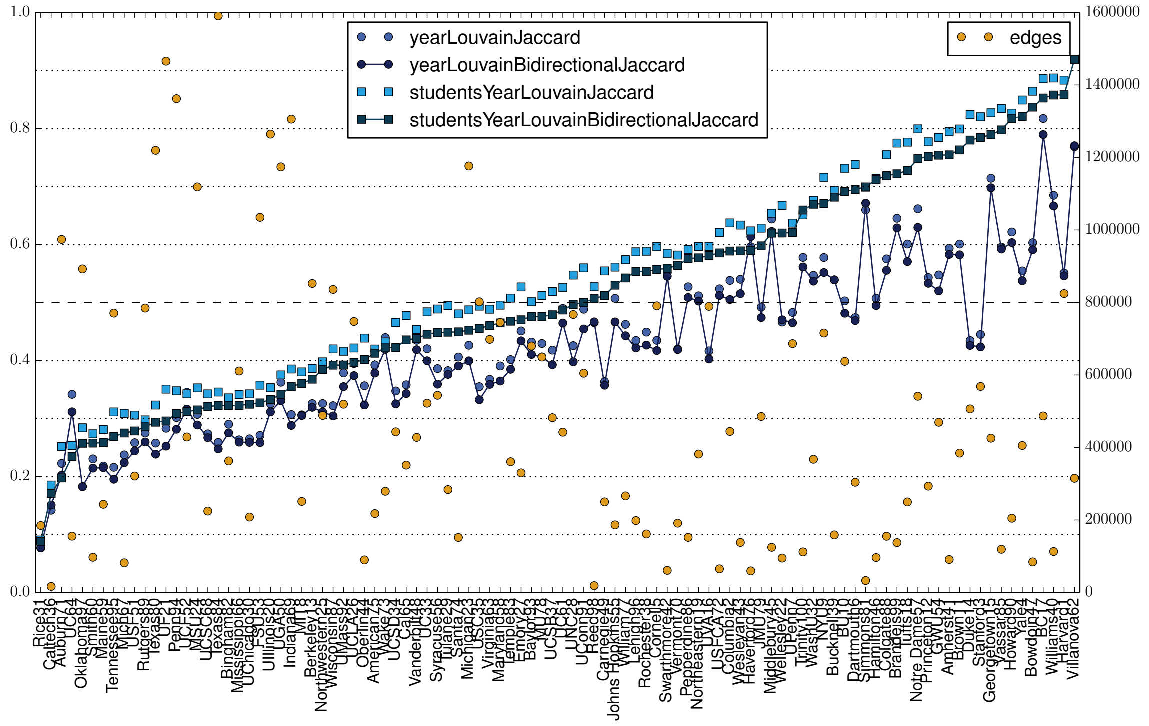


Figure 3.2: For all 100 Facebook networks (x-axis): The unidirectional Jaccard index of the graduation years and the Louvain clusters (yearLouvainJaccard) and the bidirectional Jaccard index of the graduation years and the Louvain clusters (yearLouvainBidirectionalJaccard) of the unfiltered networks compared to the same Jaccard indexes of the filtered networks that only include the current students (studentsYearLouvainJaccard and studentsYearLouvainBidirectionalJaccard). For comparison at the right y-axis the number of edges of the unfiltered networks. The networks are sorted by the bidirectional Jaccard index of the graduation years and the Louvain clusters calculated on the filtered networks.

otherwise. Again we chose the giant component of these networks as some of the current students have only connections to former students.

This also makes sense for the dormitories, as the students that lived in a dormitory a few years ago are not necessarily connected with those that are currently living in a dormitory.

We performed all experiments both on the giant components of the unfiltered networks and on the giant components of the networks that only include the active students. In the next section we will examine the differences between the filtered and unfiltered data when applying the Louvain community detection algorithm on them.

3.5 Clustering the Original Networks

In order to understand the effect of the filtering of the networks on our results we will compare the output of the Louvain community detection algorithm on the filtered and unfiltered networks.

Graduation Years

As a first step we want to examine the uni- and bidirectional Jaccard index of the clusters that are calculated by the Louvain algorithm and the graduation years, both using the unfiltered and the filtered networks.

For the comparison of the detected communities and the graduation year and dormitory structure using the Jaccard indexes we excluded all people with a value of 0 for the graduation year or dormitory respectively, but we included these people for the calculation of the clusters.

In Figure 3.2 we visualized the uni- and bidirectional Jaccard index of the Louvain clusters and the graduation years both for the clusters calculated on the filtered and on the unfiltered data. For comparison we also added the number of edges of the unfiltered networks. One can see that there is wide range of different values reaching from values close to 0 which indicate that graduation years and Louvain clusters do not match at all up to values close to 1, which indicate a very good match between Louvain clusters and graduation years.

As there are only four – or in one case five – graduation years in the filtered networks a Jaccard index of 0.25 (or 0.2) can be achieved by detecting just a single large cluster. Every detected clustering that correctly identifies one of the larger graduation years also automatically gets at least a Jaccard index of 0.25. This means that values below 0.25, and in fact probably even below 0.5, do not indicate a good match between graduation year clusters and Louvain clusters.

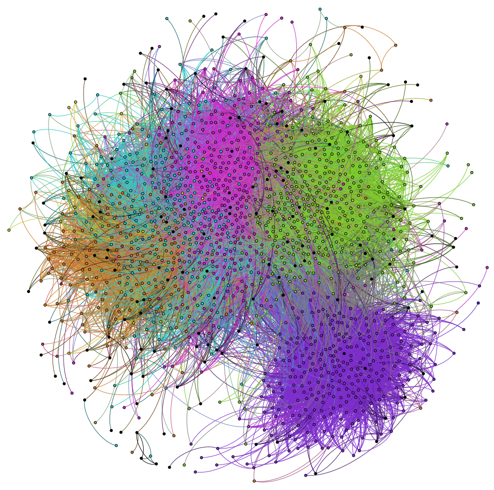
For most networks the unidirectional Jaccard index of the graduation years and the Louvain clusters gives higher values than the bidirectional Jaccard index and the difference is even larger for the filtered networks that are restricted to the current students. A possible explanation for this is, that in addition to the Louvain clusters that match the graduation year clusters rather well there are additional Louvain clusters that do not match the graduation year clusters. While the unidirectional Jaccard index ignores them they have a negative impact on the bidirectional Jaccard index.

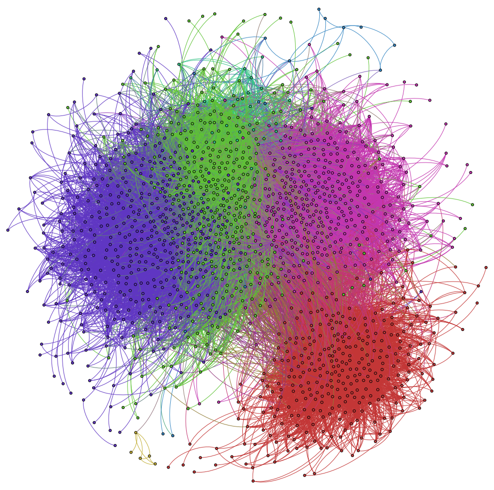
Apart from a few exceptions the Jaccard indexes of the graduation years and the Louvain clusters on the filtered networks are always higher than those of the unfiltered networks. This means that the Louvain algorithm can more accurately detect the graduation year clusters for the current students than for the former students.

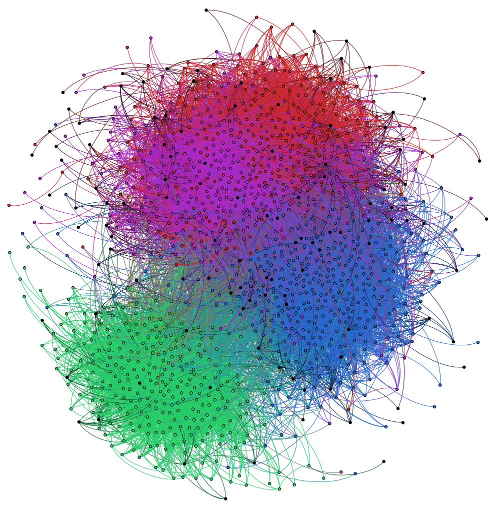
It is interesting to see that for the largest networks the Louvain clusters do not match the graduation year structure at all. We will have a closer look at this phenomenon in the next section.

In order to better understand the difficulties of the Louvain algorithm we examined the network Simmons81 in more detail. In Figure 3.2 it is the 21st from the right. The Jaccard index of 0.7 for the filtered network and a tiny bit less for the unfiltered network indicates, that the graduation years should somehow match the Louvain clusters but not perfectly.

In Figure 3.3 we show visualizations of this network using Gephi [BHJ09]. We computed the layout of the network using the Fruchterman-Reingold layout algorithm, a force-directed layout algorithm in Gephi [FR91]. This means that this layout visualizes the structure of the network and shows, to a certain extent, also the community structure of the network by grouping nodes that are especially densely connected. In Figure 3.3a and 3.3b we show the unfiltered network colored by graduation year and Louvain clustering while Figure 3.3c and 3.3d contain only the students again colored by graduation year and Louvain clusters respectively. The layouts of the former and the latter two visualizations are the same so the nodes can be directly compared while the colors do not match. Note that even though the nodes themselves are colored, one can primarily see the colors of the edges, which are colored by the mix of the colors of the incident nodes. The layout algorithm arranged the students sorted by graduation year in clockwise order from the first to the last graduation year. In the first two visualizations the graduation year 2009 is at the bottom right while in the third and the fourth visualization the graduation year 2009 is at the bottom left. The year 2009 is also visually separated from the rest of the students by a small gap which


(a) All people colored by graduation year


(b) All people colored by Louvain clusters


(c) Current students colored by graduation year

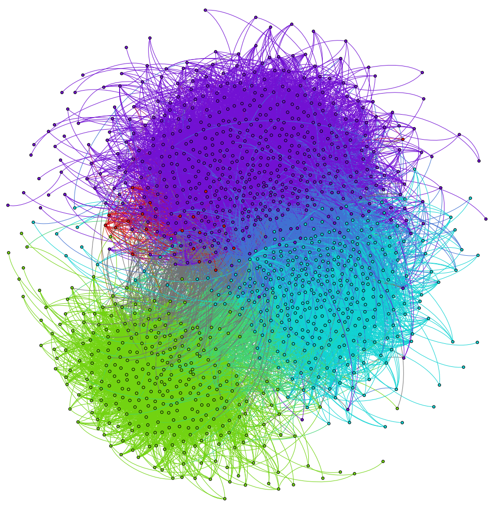

(d) Current students colored by Louvain clusters

Figure 3.3: Visualizations of the network Simmons81 using Gephi

shows that they are not yet that connected with the rest of the students, which can be easily explained by the fact that they just started studying in 2005 when the data was collected. This was already observed by Nick et al. [NLCB13] when they visualized the Simmelian Backbone of the network Caltech36 and we also observed this in other networks of the dataset.

Both in Figure 3.3a as well as in Figure 3.3c we can see that there is also a clear but smaller visual gap between the graduation years 2008 and 2007.

As we can see in Figure 3.3a most students are from the graduation years 2009, 2008 and 2007. Figure 3.3b shows that the Louvain algorithm is able to detect these larger graduation years relatively accurately which explains the Jaccard indexes of almost 0.7.

In the network that is restricted to the current students the Louvain algorithm was unable to differentiate between the graduation years 2007 and 2006, as we can see in Figure 3.3d the Louvain algorithm combined the two graduation years into one cluster. This explains why the Jaccard indexes for the filtered network are only slightly better than those for the unfiltered network, even though one could expect from the unfiltered network that the Louvain algorithm should achieve an almost perfectly matching clustering for the current students. In the filtered network the algorithm instead detects another small community that does not correspond to any graduation year, this small community is embedded in the large graduation year 2008. Even though one would expect this cluster to negatively influence the bidirectional Jaccard index this is not visible Figure 3.2 which can be easily explained by the use of the weighted average of the Jaccard values of the clusters and the small size of the cluster.

In summary we can see that filtering the networks has a positive effect on the results of the Louvain algorithm regarding the graduation years. Now we will discuss the effects of the filtering on the Louvain algorithm regarding the dormitories.

Dormitories

In the evaluation of Lee et al. [LC13] the Louvain algorithm performed a lot worse when comparing the clusterings to the dormitories, in their evaluation of the 40 smallest networks of the Facebook dataset the accuracy of the detection of the graduation years was 60 percent on average, while the accuracy for the detection of the dormitories was just 25.4 percent. Our data does not only confirm the worse accuracies but shows even bigger differences.

In Figure 3.4 we plotted the Jaccard indexes for the dormitories and the Louvain clusters as in Figure 3.2 for the graduation years. In contrast to the graduation years where the Louvain clusters matched the graduation years pretty good for a lot of the networks, we can see that there are in fact just two networks where the Louvain algorithm is able to detect a clustering that is similar to the dormitory structure: Caltech36 and Rice31. Most networks even have Jaccard indexes below 0.2.

Even though the Jaccard indexes for most networks are lower for the dormitories than for the graduation years, one must not forget that the Jaccard index for large Louvain clusters that do not match depends largely on the number and sizes of communities that shall be detected. As already mentioned if the algorithm detected the whole network as cluster the Jaccard index with four equally-sized communities would still be 0.25, while if we wanted to detect 20 equally-sized communities it would be 0.05. This means that a Jaccard indexes for the graduation years of 0.25 is not necessarily better than a Jaccard index of 0.1 for the dormitories. Nevertheless even if we take these differences into account the values for the graduation years are still much better than those for the dormitories.

In contrast to the graduation years there is no general difference between the data for the students and the other people in the dataset. This might be explained by the fact that we

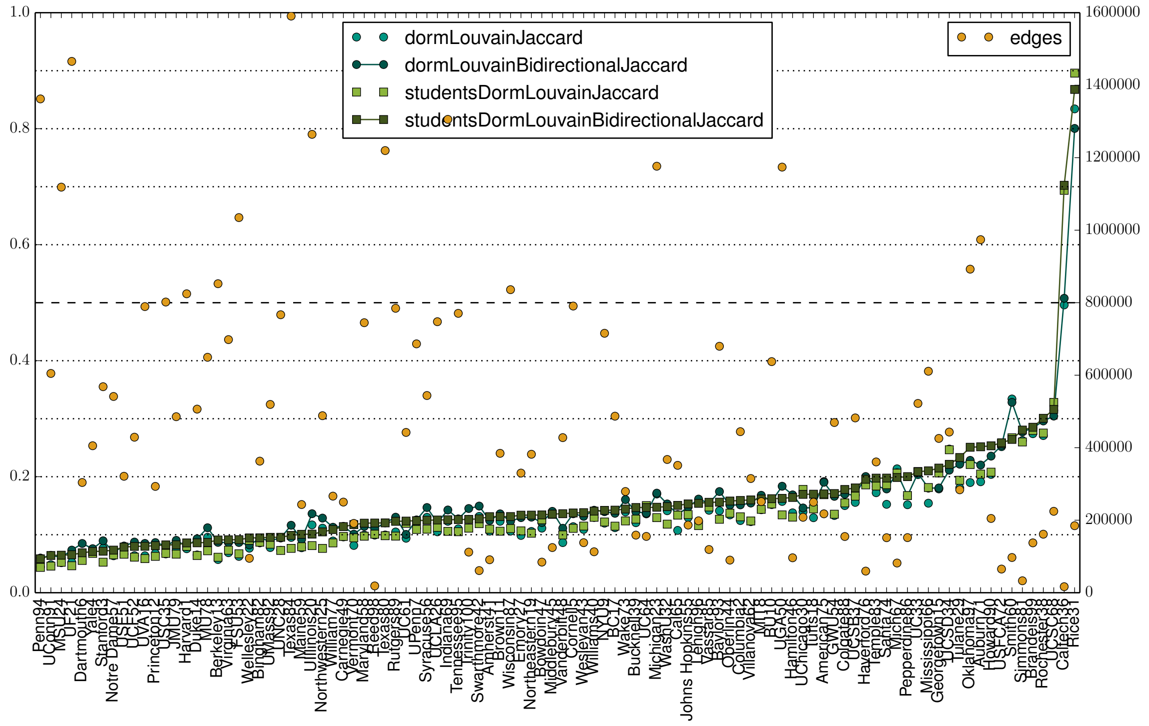


Figure 3.4: For all 100 Facebook networks (x-axis): The unidirectional Jaccard index of the dormitories and the Louvain clusters (dormLouvainJaccard) and the bidirectional Jaccard index of the dormitories and the Louvain clusters (dormLouvainBidirectionalJaccard) of the unfiltered networks compared to the same Jaccard indexes of the filtered networks that only include the current students (studentsDormLouvainJaccard and studentsDormLouvainBidirectionalJaccard). For comparison at the right y-axis the number of edges of the unfiltered networks. The networks are sorted by the bidirectional Jaccard index of the dormitories and the Louvain clusters calculated on the filtered networks.

only evaluated the indexes for those people that actually provided dormitory information and that their behavior in terms of friendships to other members of the dormitory does not vary significantly between the current and the former students.

In most cases the unidirectional Jaccard index performs worse than the bidirectional Jaccard index. As we can already assume from the good Jaccard indexes for the graduation years and as we will see in absolute numbers soon there are more dormitories than Louvain clusters in most networks. This means that in the case of the unidirectional Jaccard index we would compare each dormitory to a larger Louvain cluster while in the case of the bidirectional Jaccard index we would also compare each Louvain cluster to the best matching dormitory, which is possibly a much better match. It is likely that some dormitories match better than, others as there are sometimes significant differences in the sizes of the dormitories, in some universities they differ by a factor of ten.

It is interesting to see that for the two networks with good results in one case the network that is restricted to the students gives better results, while in the other case the full network gives the better results.

Therefore we can conclude that it is beneficial for the comparison with the graduation years to use the filtered data and as the evaluation of the dormitories is not impacted we will use the networks that are restricted to the students for all further evaluations also for the dormitories in order to keep the results comparable.

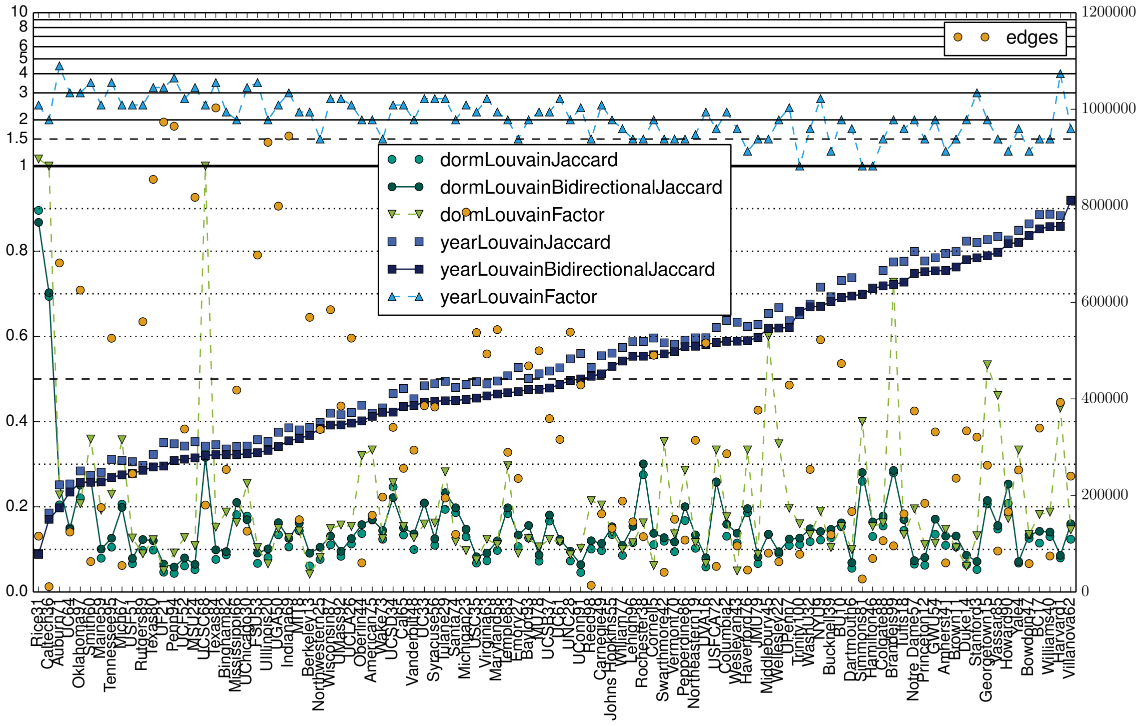


Figure 3.5: The Jaccard indexes are the same as in Figure 3.2 and Figure 3.4 but this time just for the filtered networks. Additionally the plot contains the number of edges in the filtered networks and the factor between the number of Louvain clusters and graduation years (yearLouvainFactor) and the factor between the number of Louvain clusters and the dormitories (dormLouvainFactor) which is the number of Louvain clusters divided by the number of graduation years/dormitories. The networks are sorted by the bidirectional Jaccard index of the graduation years and the Louvain clusters.

For the sake of readability especially in the context of the Simmelian Backbones from now on unless explicitly mentioned the “original” network is always the filtered network that only includes the current students.

Number of clusters

A very simple measure for the similarity between two clusterings is the number of clusters. In Figure 3.5 we have added the Jaccard indexes of the Louvain clusters and the dormitories to the Jaccard indexes of the Louvain clusters and the graduation years. Additionally the plot contains the factor between the number of Louvain clusters and the number of dormitories and the factor between the number of Louvain clusters and the number of graduation years. A value higher than one means that there are too many Louvain clusters, a value lower than one means that there are too few Louvain clusters compared to the number of dormitories/graduation years. Note that the lower part of the y-axis till 1.0 is linear while the upper part is logarithmic.

In Figure 3.5 we can see that for the graduation years there are too many clusters while for the dormitories there are too few clusters. There are three networks – Rice31, Caltech36, and UCSC68 – for which there are exactly as many clusters as there are dormitories and these are also the three networks with the highest Jaccard indexes for the dormitories and the Louvain clusters. For the graduation years there is one exact match of the number of clusters and graduation years which is the network Simmons81 but this is not the network with the highest Jaccard indexes, as we have already seen in the lower half of Figure 3.3 this is because the Louvain algorithm did not identify the separation between the graduation year 2006 and 2007. It is clear to see that the factor between the number of clusters and the graduation years decreases on average with the rising similarity of the clusters to the graduation years. One should also note that the number of clusters is on average much closer to the number of graduation years. The largest factor is less than 4 which means that there are at most four times as many Louvain clusters as there are graduation years, while we have sometimes more than ten or even twenty times more dormitories than there are Louvain clusters.

It is interesting to see that the factor between the number of Louvain clusters and the number of graduation years is not always linked to the match between Louvain clusters and graduation years according to the Jaccard indexes. For example in the case of the network with the highest Jaccard indexes, Harvard1, there are a lot more Louvain clusters than there are graduation years. Nevertheless there seem to be Louvain clusters among them that almost perfectly match the graduation year structure, and in addition to these almost perfectly matching Louvain clusters there are probably just a few additional small Louvain clusters that are of no consequence for the Jaccard indexes.

Lee et al. [LC13] claimed that the Louvain community detection algorithm cannot find the dormitory structure in the larger networks because of the resolution limit of modularity and thus finds the coarser graduation year structure. We cannot fully confirm this, there are also small networks with good results for the graduation year structure. In the largest networks (that were not included in the benchmark of Lee et al.) we are actually getting bad results both for the graduation years and the dormitories. Maybe this is indeed a result of the resolution limit, but maybe this is also because in large universities the students are not that connected within their graduation years. It would be interesting to see if in the larger institutions the detected community structure is correlated to another attribute, like for example majors or groups of related majors, but this is out of the scope of this work.

Nevertheless it is obvious that the community structure found by the Louvain algorithm is in most networks much coarser than the dormitory structure which means that here the Louvain algorithm is unable to find the fine-grained dormitory structure. An interesting question is if maybe in some of these networks the dormitory structure is simply not the basis of the structure of the network. This could be the subject of further research both using measures like the number of intra-dormitory connections versus the number of inter-dormitory connections and also using other community detection algorithms like greedy clique expansion [LRMH10]. Greedy clique expansion performed better than the Louvain algorithm according to the experiments of Lee et al. [LC13] but they only used the 40 smallest Facebook networks which means that these large networks were not included in the benchmark. As their benchmark used a different similarity measure it would also be interesting to repeat their experiments using an evaluation based on Jaccard indexes.

In this section we have seen that filtering the network to only include the current students is useful for the comparison of the Louvain clusters to the graduation years as otherwise the graduation year data is too messy. After filtering the networks, we can see that the match of the Louvain clusters and the graduation years ranges, according to the Jaccard index, from no match to almost perfect match with a tendency towards the latter, while the Louvain clusters match the dormitories in just two cases. For most networks the number of the Louvain clusters is way too low for the dormitories and a bit too high for the graduation years, though for the networks with the highest Jaccard indexes of Louvain clusters and dormitories the number of Louvain clusters matches the number of dormitories.

In the next section we will look at the effects of the Simmelian Backbone algorithm on these results.

3.6 Clustering on Simmelian Backbones

As the Simmelian Backbone algorithm filters the network and thus the backbone is much sparser we also expect to see more and smaller clusters on the backbone, and thus expect that the backbone will not help detecting the graduation year structure, as the graduation year structure requires less and not more clusters. As we need more Louvain clusters in order to match the number of dormitories we hope that the Simmelian Backbone algorithm will improve the match between Louvain clusters and dormitories. As in the previous section we will first compare the Louvain clusters to the graduation years and then to the dormitories.

For our experiments we used the parametric Simmelian Backbones with a maximum rank of 10 but we omitted the first step of filtering the edges, which means that an overlap of 0 implies that all edges are included in the backbone. We varied the required overlap among the neighbors of the nodes incident to an edge between 0 and 14.

Graduation Years

For comparison we plotted in Figure 3.6 again the uni- and bidirectional Jaccard index of the graduation years and the Louvain clusters calculated on the original network. In addition to that, we added the uni- and bidirectional Jaccard index of the graduation years and the Louvain clusters calculated on a Simmelian Backbone of the network that we chose per network and also separately for the uni- and bidirectional Jaccard index. We chose the minimum required overlap for the backbone calculation such that it gave the maximum Jaccard index. For most networks this was an overlap of 0 which means that none of the Simmelian Backbones that we calculated had a positive effect on the Jaccard index of the graduation years and the Louvain clusters calculated on that backbone. In the few cases where a backbone had a positive effect we annotated that value with the required overlap value. For comparison we also added the number of edges of the networks.

Figure 3.6 clearly shows that, as expected, for most networks the Simmelian Backbone does not improve the situation for the graduation years. In some cases, like for the network Yale4 also just the unidirectional Jaccard index indicates an improvement while the bidirectional Jaccard index does not see any improvement. A possible explanation for this is the fact that the Simmelian Backbone algorithm does not leave the nodes of the network connected and thus leads to a larger number of smaller clusters, which is something that we can see better by using the bidirectional Jaccard index. There is one network where the improvement as measured by the Jaccard index of the graduation years and the Louvain clusters was relatively significant: the network Swarthmore42. In order to see the effects of the Simmelian Backbone on the clusters in the context of the graduation year we visualized the network Swarthmore42 using Gephi. Figure 3.7a depicts the full (filtered) network with colors according to the graduation year. The layout of the network was calculated using the Fruchterman-Reingold algorithm [FR91] and is identical for all three visualizations in Figure 3.7. One can see again the clear separation of the graduation year 2009 compared to the rest of the network. Figure 3.7b shows the nodes colored by the Louvain clusters calculated on the original network. One can see that while the algorithm managed to identify the graduation year 2009 the other graduation years are only partially matched, though one can clearly see that each of the three remaining years (2008, 2007 and 2006) has one cluster but that the separation is not the same as for the years and two additional clusters appeared.

Figure 3.7c shows the Simmelian Backbone with overlap 1 of the network Swarthmore42. Note that we kept the layout of the original network in order to make the different visualizations easier to compare. This means that the layout possibly no longer represents

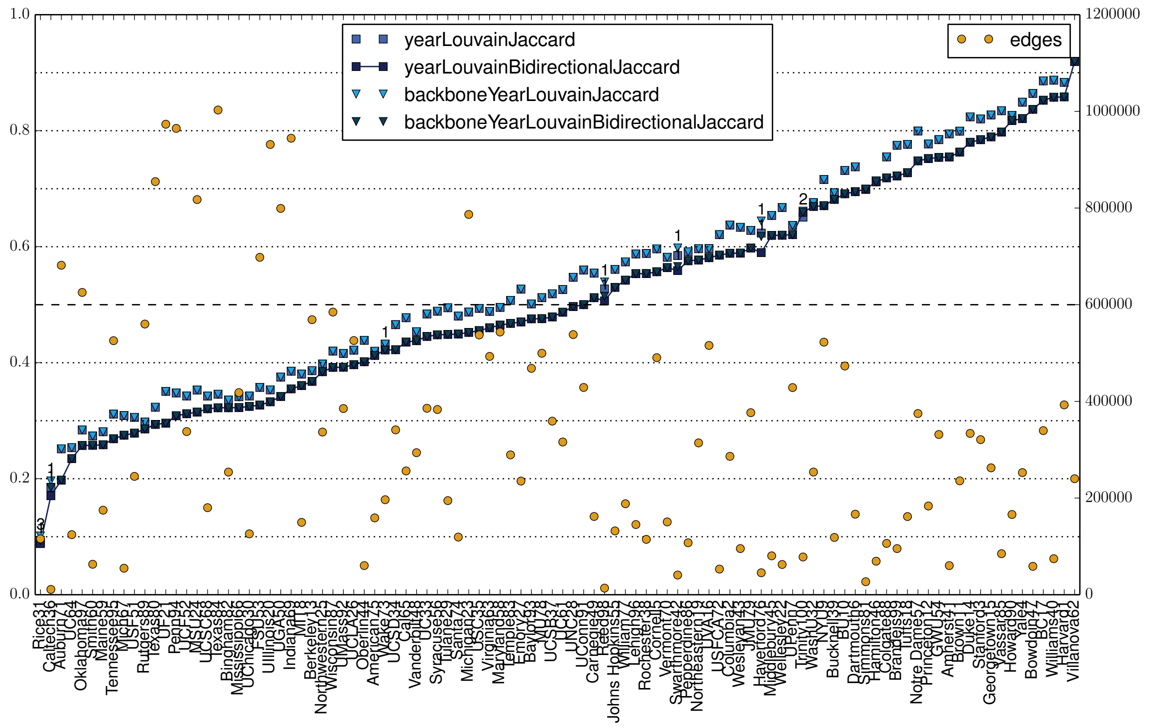


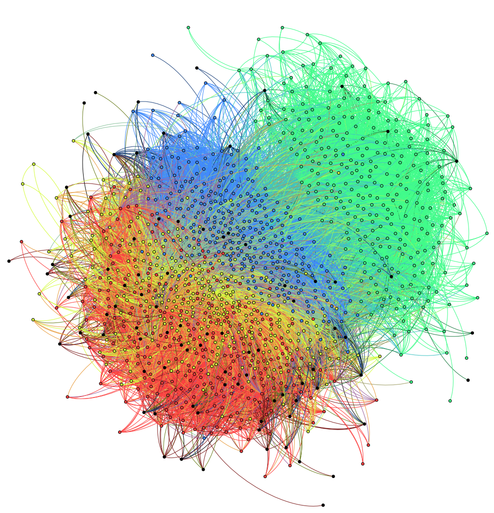
Figure 3.6: The uni- and bidirectional Jaccard index of the graduation years and the Louvain clusters calculated on the original network (yearLouvainJaccard and yearLouvainBidirectionalJaccard) in comparison with the uni- and bidirectional Jaccard index of the graduation years and the Louvain clusters calculated on a selected Simmelian Backbone (backboneYearLouvainJaccard and backboneYearLouvainBidirectionalJaccard). The backbone (including the option of the original network) was selected such that the respective Jaccard index was maximized. In the cases where a non-zero overlap was required and thus edges were actually filtered the Jaccard indexes are annotated by the required overlap that was used for the calculation of the backbone. For comparison the number of edges in the original network were added. The networks are sorted by the bidirectional Jaccard index of the graduation years and the Louvain clusters calculated on the Simmelian Backbone.

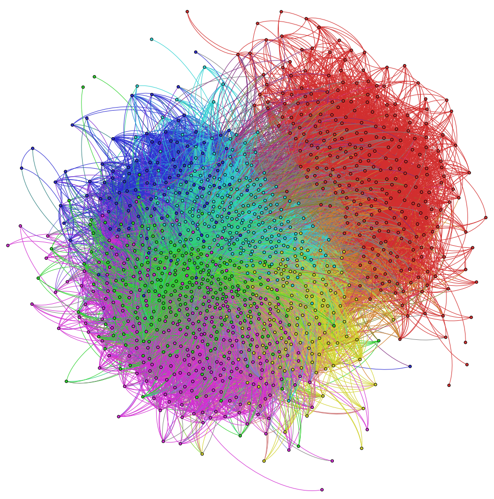
the structure of the network. We can see that there are much less edges between the graduation year 2009 and the rest of the network which means that now the separation between the graduation year 2009 and the rest of network is even stronger. Interestingly, the cluster structure of the older years is different. This means that if we apply the Louvain algorithm on top of the Simmelian Backbone we are actually able to identify a different structure than with the Louvain algorithm alone, but from these visualizations it is unfortunately not obvious why the Jaccard index actually improved. This means that even in this single example it is not clear that the Simmelian Backbone actually identified the graduation year structure and not another structure that has just a relatively high Jaccard index when compared to the graduation years.

In summary we can thus conclude that clustering on the backbone does not make sense if the goal is to detect the graduation year structure.

Dormitories

Similar to the plot of the Jaccard indexes of the graduation years and the Louvain clusters we plotted the analog values for the dormitories in Figure 3.8. As there is not enough


(a) Colored by graduation year


(b) Colored by Louvain clusters on the original network

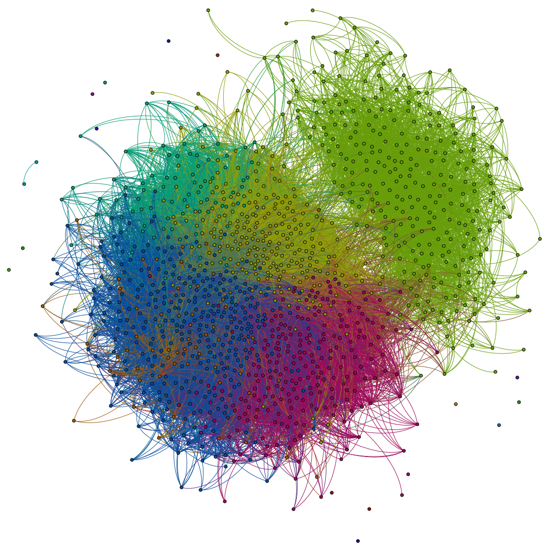

(c) The Simmelian Backbone with overlap 1 colored by the Louvain clusters on that backbone

Figure 3.7: Visualizations of the network Swarthmore42 using Gephi

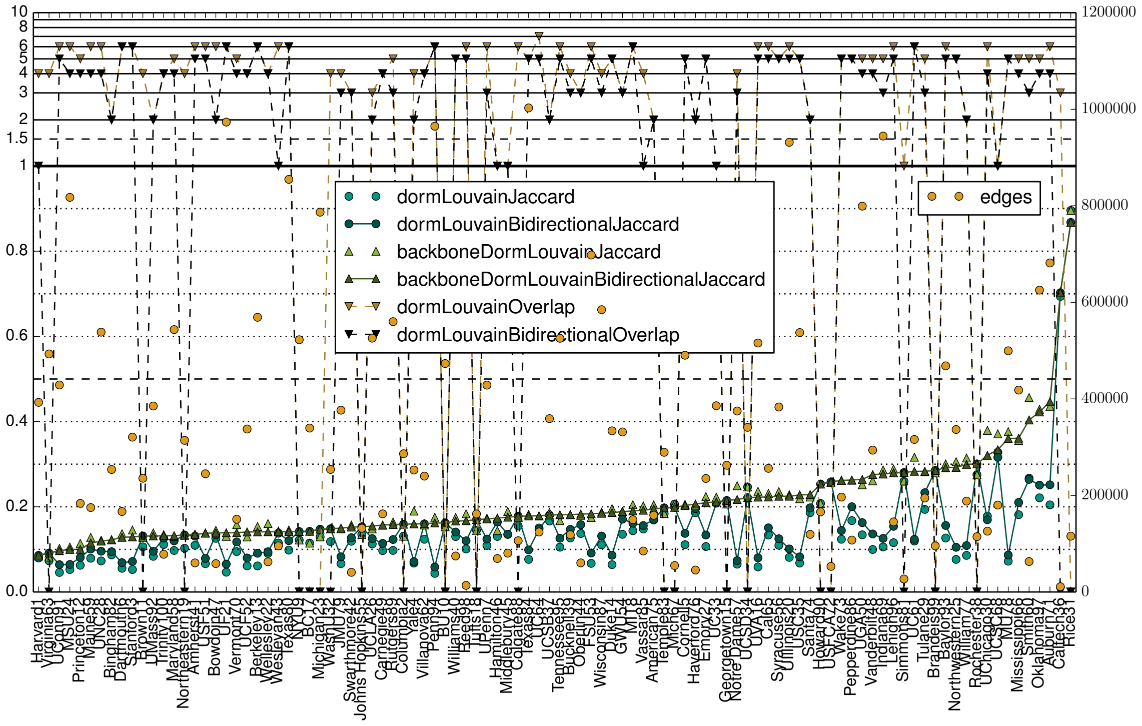


Figure 3.8: The uni- and bidirectional Jaccard index of the dormitories and the Louvain clusters calculated on the original network (dormLouvainJaccard and dormLouvainBidirectionalJaccard) in comparison with the uni- and bidirectional Jaccard index of the dormitories and the Louvain clusters calculated on a selected Simmelian Backbone (backboneDormLouvainJaccard and backboneDormLouvainBidirectionalJaccard). The backbone (including the option of the original network) was selected such that the respective Jaccard index was maximized. The used overlap values for the uni- and bidirectional Jaccard index are the values of dormLouvainOverlap and dormLouvainBidirectionalOverlap. For comparison the number of edges in the original network were added. The networks are sorted by the bidirectional Jaccard index of the dormitories and the Louvain clusters calculated on the Simmelian Backbone.

space for annotating the Jaccard indexes with the used overlap values for the backbone calculation we instead plotted the overlap values.

In Figure 3.8 we can see that for the dormitories there are significant improvements. For the majority of the networks the improvements are not very significant as they can be mostly explained by the fact that the Jaccard index of small ground truth communities with large clusters can be improved by decreasing the size of the clusters regardless whether or not the clusters match. Albeit none of the networks that had Jaccard indexes below 0.5 has a Jaccard index better than 0.5 on the Simmelian Backbone, there are three networks with Jaccard indexes above 0.4. Especially interesting is the result for the networks Auburn71 and Oklahoma97, as both are among the 12 largest networks in the dataset which means that for them the resolution limit of the Louvain algorithm might have been the main problem.

The three networks for which the number of clusters already matched the number of dormitories (Rice31, Caltech36 and UCSC68) have almost unchanged Jaccard indexes. Furthermore, only an overlap of 1 seems to make sense for these networks.

For most of the networks an overlap of 4 to 6 seems to give the best improvements for the dormitories.

Compared to the results on the original network the bidirectional Jaccard index of the dormitories and the Louvain clusters on the backbone is for a larger number of networks lower than the unidirectional Jaccard index. This can be explained by the fact that the backbones contain isolated nodes and small connected components that have a lower Jaccard index with the dormitory they are part of than the Louvain cluster that is selected for the unidirectional Jaccard index. This also explains why for a larger number of networks the bidirectional Jaccard index of the dormitories and the Louvain clusters on the Simmelian Backbone is better at a slightly lower overlap than the unidirectional Jaccard index, as for higher overlaps the number of isolated nodes and small connected components increases.

In Figure 3.9 we show various visualizations of both the original network Smith60 and the Simmelian Backbone with overlap 5. The layout of the two is fundamentally different, since their structure, as interpreted by the Fruchterman-Reingold layout algorithm, is fundamentally different. While the original network in Figure 3.9a is still structured by graduation years even though the Louvain clusters give a colored mix (apart from the year 2009 at the top) the Simmelian Backbone in Figure 3.9b is only structured by year insofar as the graduation year 2009 still forms a close group and the graduation year 2008 seems to be more connected between each other than the rest. Apart from that, one can clearly see the College's apparently successful efforts to mix people of the different graduation years. As one can see in Figure 3.9c, which has the same layout as Figure3.9b but is colored by dormitory, this structure of the network is in large parts the dormitory structure. In this obvious setting the Louvain algorithm can identify the clusters that are shown in Figure 3.9d that got a unidirectional Jaccard index of a bit more than 0.45.

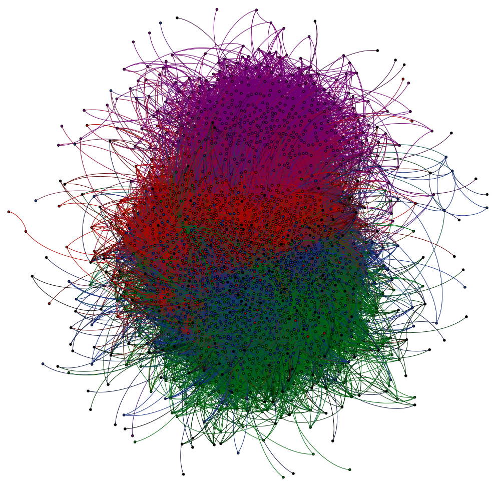
A possible explanation for the different structure depending on the graduation year is that Facebook was introduced in 2004, which means that only the students of the graduation years 2008 and 2009 were already able to connect themselves in Facebook when they started studying. As, at least in our experience, students of later semesters have less contact with other students of their year this might explain why there are fewer connections between the students in the earlier years. On the other hand, in a university or college where the dormitory structure is very important, it is easy to explain why people connect with each other in the dormitory and also preferably within their dormitory with the students of the same graduation year.

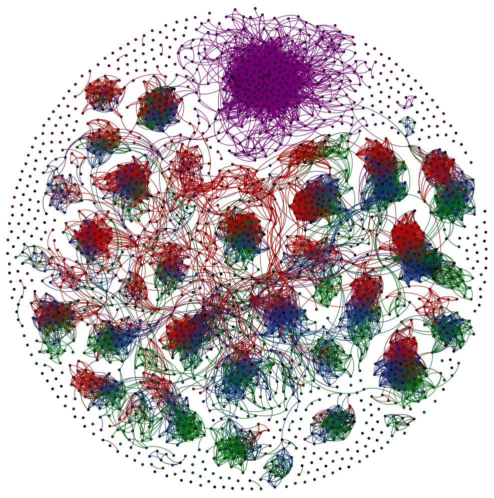
As this behavior of the graduation year 2009 does not seem to be a special case but something that can be easily explained basically for all networks we wanted to see the effect of excluding the graduation year 2009 from the evaluation.

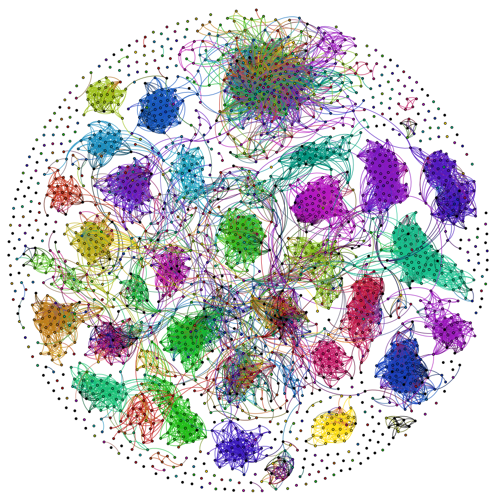
Excluding 2009

In the following, we excluded all students that will graduate in 2009 from the calculation of the Jaccard indexes. In Figure 3.8 we plotted these values both for the dormitories and the Louvain clusters on the original network, as well as for the dormitories and the Louvain clusters on the backbone. In Figure 3.11 we did the same for the graduation years. As the bidirectional Jaccard index provided no additional insights we restricted the plots to the unidirectional Jaccard indexes.

In Figure 3.10 one can see that for many of the networks where the Jaccard index was significantly increased by clustering on the backbone the graduation year 2009 seems to be an exception. Especially the two large networks Auburn71 and Oklahoma97 are very good examples of this behavior on the backbone. There the unidirectional Jaccard index is even above 0.6 for the students that do not graduate in 2009. On the original network this phenomenon also exists but is a lot less significant for most networks. For some networks like Wake73 the behavior is exactly the opposite, which means that actually the dormitory


(a) Colored by graduation year


(b) The Simmelian Backbone with overlap 5 colored by graduation year


(c) The Simmelian Backbone with overlap 5 colored by dormitory

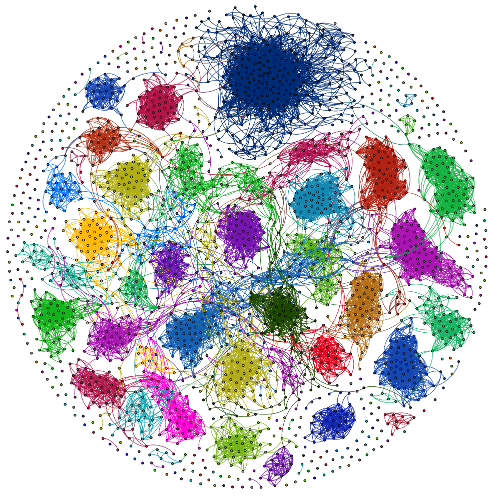

(d) The Simmelian Backbone with overlap 5 colored by Louvain clusters calculated on the backbone

Figure 3.9: Visualizations of the network Smith60 using Gephi

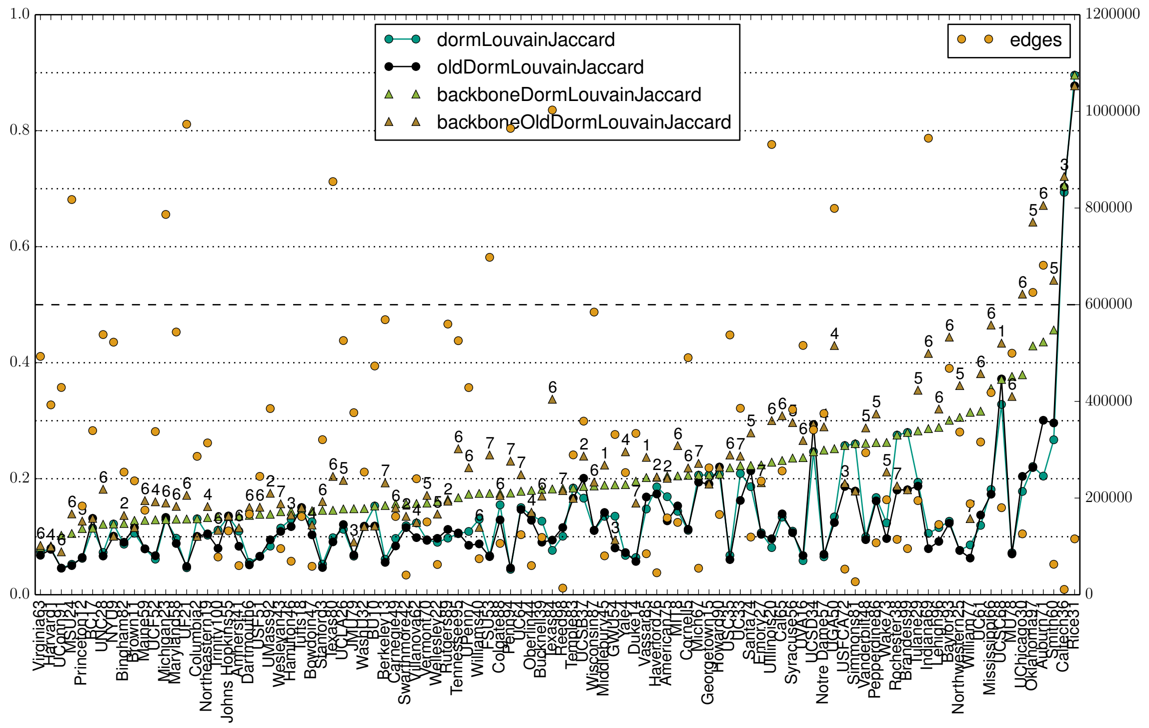


Figure 3.10: The same unidirectional Jaccard indexes as in Figure 3.8 but instead of the bidirectional Jaccard indexes the unidirectional Jaccard indexes that were calculated on all students that do not graduate in 2009 were added (oldDormLouvainJaccard and backboneOldDormLouvainJaccard).

structure was matched better in the community of the students of the graduation year 2009.

In the visualizations of the networks we have already seen that the year 2009 also plays a special role in the detection of the years, as in many network this seems to be the group that is most separated from the rest of the students. Looking at Figure 3.11 we can confirm this observation. Especially for networks with a Jaccard index between 0.3 and 0.6 there is a huge drop of the Jaccard index with the graduation years. It becomes obvious that for many networks with a Jaccard index around 0.4 the Louvain algorithm was actually not able to identify the graduation years apart from the graduation year 2009.

In this section we confirmed that the Simmelian Backbone helps with the detection of the dormitories but not the graduation years. While we showed large improvements for the detection of the dormitory clusters of the students that do not graduate in 2009, we also found that in most networks the graduation year 2009 is detected by the Louvain algorithm as a single large cluster, especially when applying the Louvain algorithm on top of a Simmelian Backbone.

Comparing the uni- and bidirectional Jaccard indexes and looking at visualizations of individual networks we saw that the Simmelian Backbone leads to many small Louvain clusters that quite often even consist of just a single node. As we already discussed in Section 2.1 this is an unavoidable effect of the Simmelian Backbone algorithm. In the following section we will thus discuss the possibility of post-processing the Louvain clustering in order to get rid of these singletons.

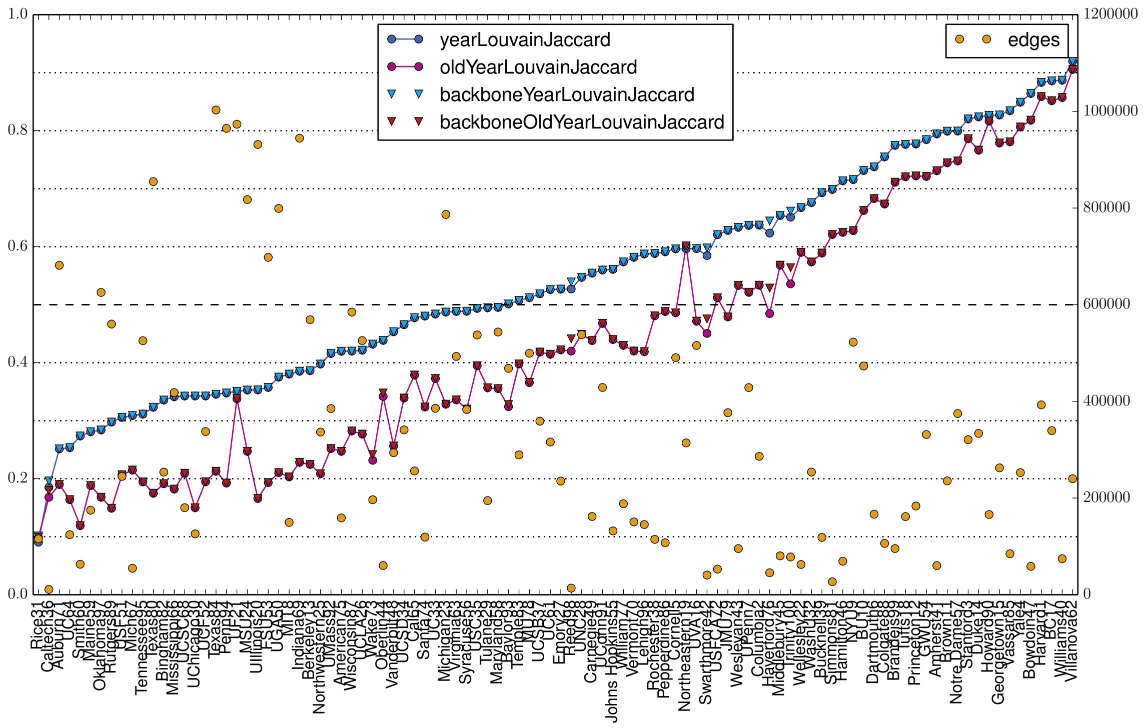


Figure 3.11: The same as Figure 3.10 but for graduation years instead of dormitories.

3.7 Merging the Singletons in Backbone Clusterings

From looking at the visualizations one can get the impression that most of the nodes that are singletons in the backbone actually belong to one of the larger clusters. This is the reason why we decided to merge the singleton clusters in larger clusters. In order to do this, we iterate repeatedly over the singletons and merge each of them into the cluster that gives the maximum (positive) modularity gain on the original network until no singletons can be merged anymore.

While this seems similar to the Louvain algorithm there are actually a couple of differences, as in our simple post-processing step the assignment of singletons to clusters is final and we do not check again if maybe another cluster would be better for a node. We never merge a singleton into another singleton cluster and we also never merge larger clusters into each other. The reason for this approach is that we want to keep the cluster structure that the Louvain algorithm calculated on the backbone as much as possible the same, which means that we do not want to modify the structure of the larger clusters but we just want to put the singletons into matching clusters.

Especially Simmelian Backbones with a high overlap parameter value also contain a lot of small connected components of more than one node (besides singletons) which also appear as small clusters in the Louvain clustering that was calculated on the backbone. Even though most of them do not make sense semantically regarding the dormitory and graduation year structure, we decided not to merge these small clusters as there is no obvious criterion when to stop merging clusters and we definitely did not want to reconstruct the Louvain clusters that we calculated on the original network.

Graduation Years

In Figure 3.12 we plotted the uni- and bidirectional Jaccard index of the graduation years and the Louvain clusters calculated on the backbone with and without the merging step. Note that the overlap parameter for the backbone is chosen separately for all four Jaccard

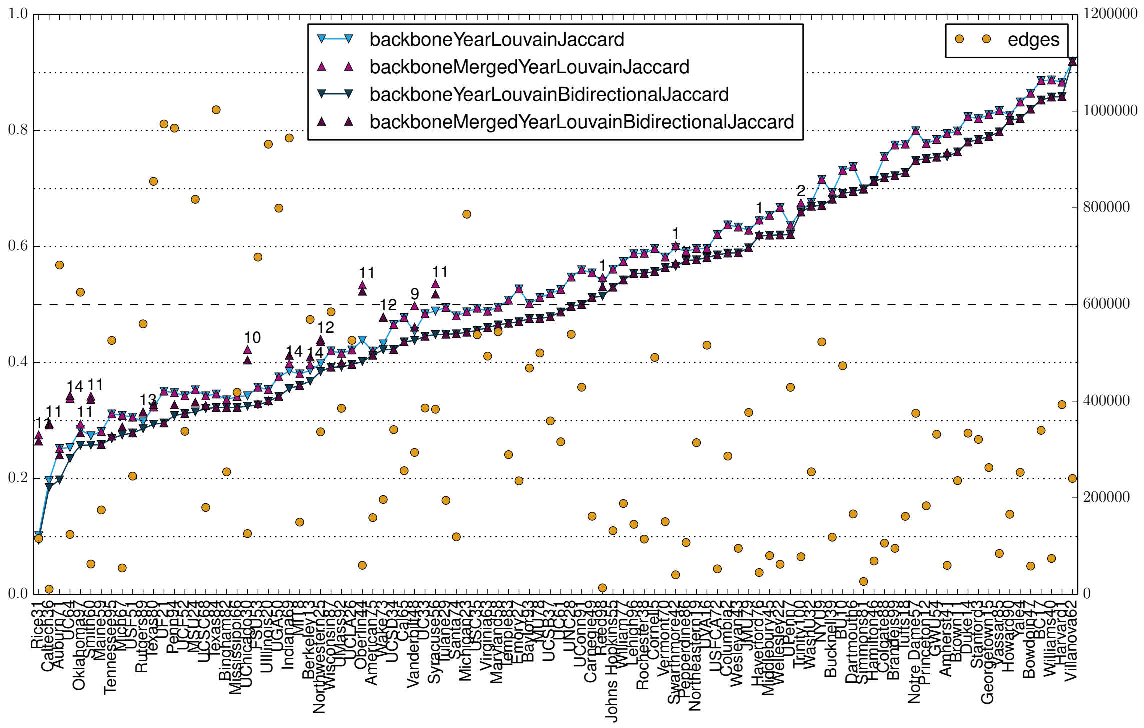


Figure 3.12: The uni- and bidirectional Jaccard index of the graduation years and the Louvain clusters on the backbone with and without merging as post processing.

indexes such that each of them is maximized. We also added the overlap values for the unidirectional Jaccard index of the graduation years and the merged Louvain clusters as annotations.

For most networks the results are very similar to the results for the Louvain clusters without post processing, which is not surprising as the backbones for a low overlap parameter do not contain that many singletons that we would expect a visible change of the Jaccard indexes. However for some networks we see really high overlap parameters of 10 to 13 which give moderate increases of the Jaccard indexes for some of these networks. If we require an overlap of 11 or even 13 this means that there are only the edges left where the incident nodes have more than 10 neighbors that are ranked with rank 10 or better, and that most or even all of them are the same. And apparently when we take the clusters that are defined by these edges as basis for a clustering and iteratively merge isolated nodes into them with the really trivial algorithm that we described before, we can get a clustering that matches the graduation year structure better than the original Louvain clusters. This could be an interesting starting point for developing new clustering algorithms that greedily expand clusters starting from edges that are strongly embedded in the sense of the Simmelian Backbone. However even though the resulting clusterings have higher Jaccard indexes than the Louvain clusters that are calculated on the original networks, the Jaccard indexes of this new clusterings are still not very high. We are not sure if this is due to the simple algorithm that we applied for expanding the clusters or if this is a sign that they might be a better basis but are actually not that good either.

Dormitories

For the dormitories we plotted the same values as for the graduation years in Figure 3.13. In order to increase the readability of the plot we did not annotate the individual values with the used overlap values but instead added the two overlap values as individual values to the plot, they are mainly in the upper part of the plot. For the dormitories the differences

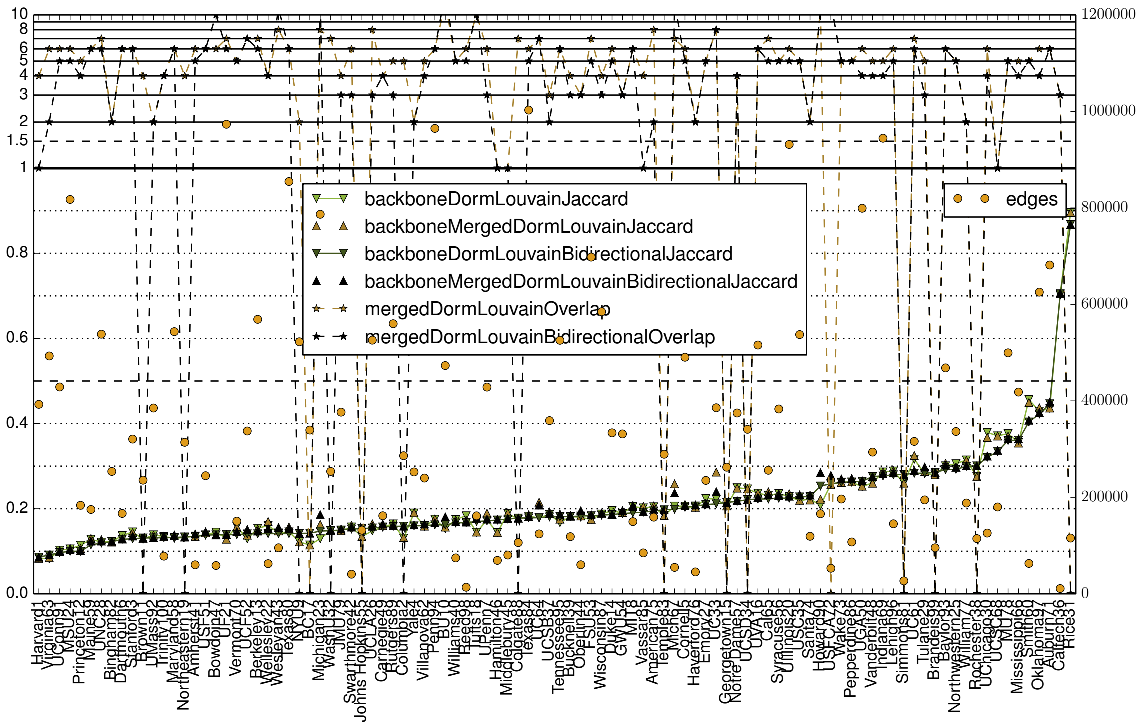


Figure 3.13: The uni- and bidirectional Jaccard index of the dormitories and the Louvain clusters on the backbone with and without merging as post processing.

of the Jaccard indexes are even smaller than for the graduation years, for some networks we see similar effects as for the graduation years with very high overlap values. In general the overlap values are higher also for the bidirectional Jaccard index but we can still see that for some networks the bidirectional Jaccard index is better for lower overlap values.

The effect that the bidirectional Jaccard index increased from merging singletons can be seen but the differences are really small. This is probably due to the fact that the number of singletons is not that high compared to the total number of nodes in the network. It would be interesting to see if the bidirectional Jaccard index can be further increased by also merging clusters of size two or three.

All in all it seems that merging the singletons has basically two effects: For some networks a really high overlap parameter leads to a new clustering algorithm that uses edges with really high overlap values as basis. For most other networks it has only small effects in terms of the Jaccard index but it is still an interesting approach if singletons are a problem as the quality of the clusterings stays almost the same.

3.8 Overlap as Edge Weight in the Original Network

From the networks where the merged clusters that were calculated on a very sparse Simmelian Backbone it seems that the edges that have really high overlap values are also those that are definitely within a dormitory or graduation year (or even both). This observation that we also made while examining individual networks with different overlap parameters brought us to the idea to use these overlap values as edge weights and to cluster on the weighted network in order to avoid the splitting of the network into multiple connected components as we already mentioned in Section 2.1. As already discussed in Section 2.1, we used two variants: In the first variant we used the overlap increased by one (in order to avoid edge weights of 0) and in the second variant we used the square of the overlap increased by one. We will call these two variants the (simple) weighted and

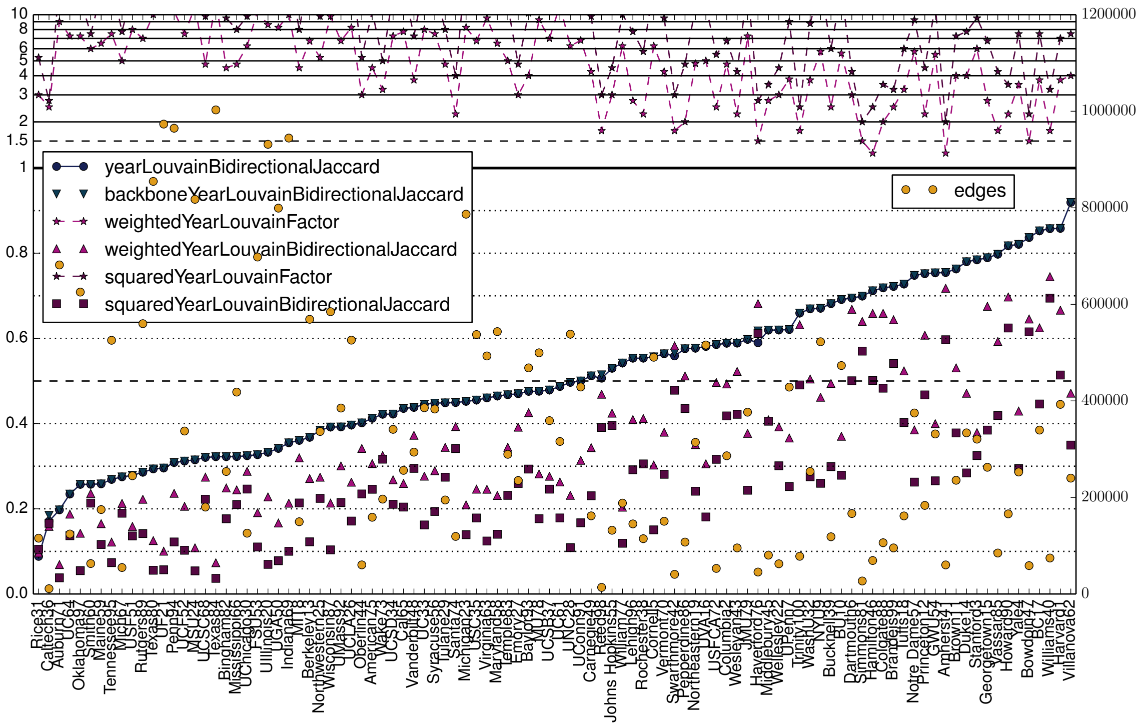


Figure 3.14: The bidirectional Jaccard index of the graduation years and the Louvain clusters calculated on the original network, the backbone that gives the highest Jaccard index and the weighted network with simple and squared weights. Additionally, the factor between the graduation years and the Louvain clusters, which is simply the number of the Louvain clusters divided by the number of graduation years, is shown both for the weighted and the squared weighted network.

the squared weighted network. In order to keep the plots readable we will only use the bidirectional Jaccard index in this section.

Graduation Years

In Figure 3.14 we plotted the bidirectional Jaccard indexes of the graduation years and the Louvain clusters calculated on different variants of the network: the original unweighted network, the backbone that gives the highest Jaccard index and the weighted and the squared weighted network. Apart from a few networks like Haverford76 where the weighted network leads to an increase of the Jaccard index we can clearly see that clustering on the weighted network is for most networks not useful when the goal is to detect the graduation years.

We also plotted the number of Louvain clusters divided by the number of graduation years and one can clearly see that for most networks the number of Louvain clusters is way too high. However, the cases where we get higher Jaccard indexes of graduation years and Louvain clusters calculated on the weighted or squared weighted networks are also cases where the number of Louvain clusters almost matches the number of graduation years. This means that, in contrast to the clusterings that were calculated on the backbone, we do not get a lot of small Louvain clusters.

Dormitories

In Figure 3.15 we plotted exactly the same values as in Figure 3.14 but for the dormitories instead of the graduation years. One can see that for the dormitories the number of Louvain

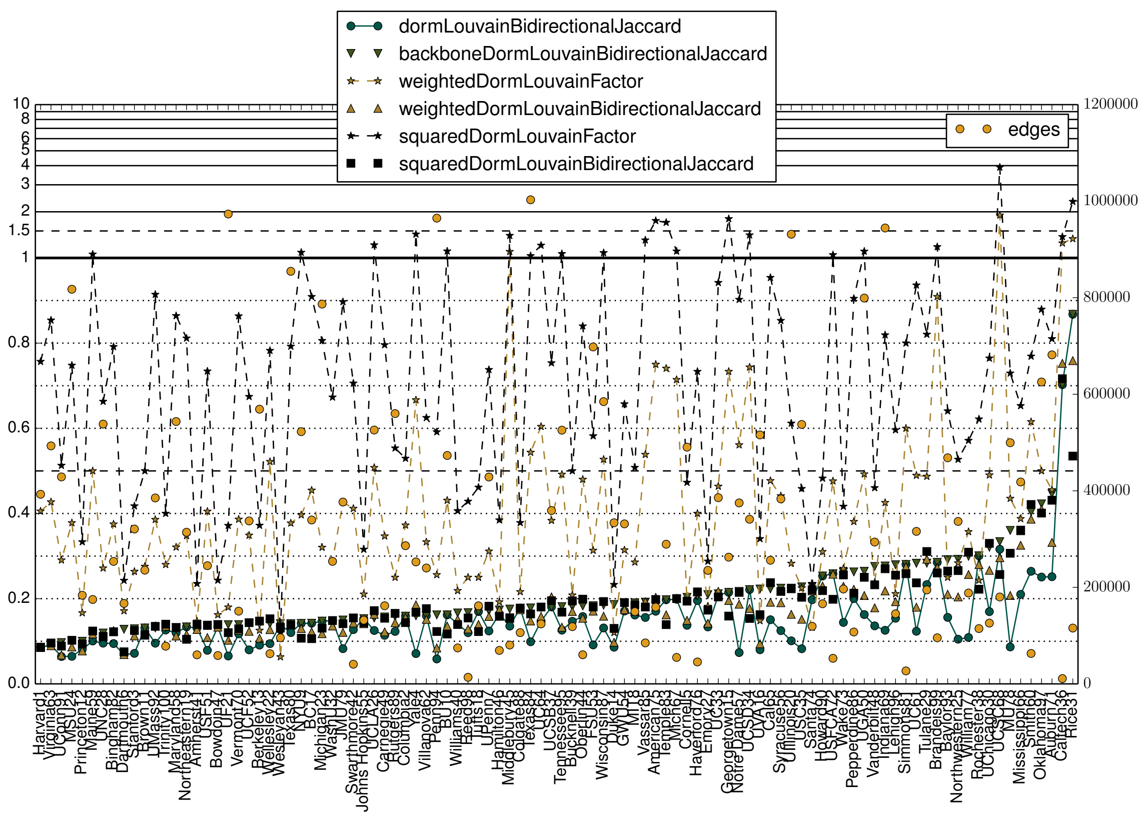


Figure 3.15: The bidirectional Jaccard index of the dormitories and the Louvain clusters calculated on the original network, the backbone that gives the highest Jaccard index and the weighted network with simple and squared weights. Additionally, the factor between the dormitories and the Louvain clusters which is simply the number of the Louvain clusters divided by the number of dormitories is shown both for the weighted and the squared weighted network.

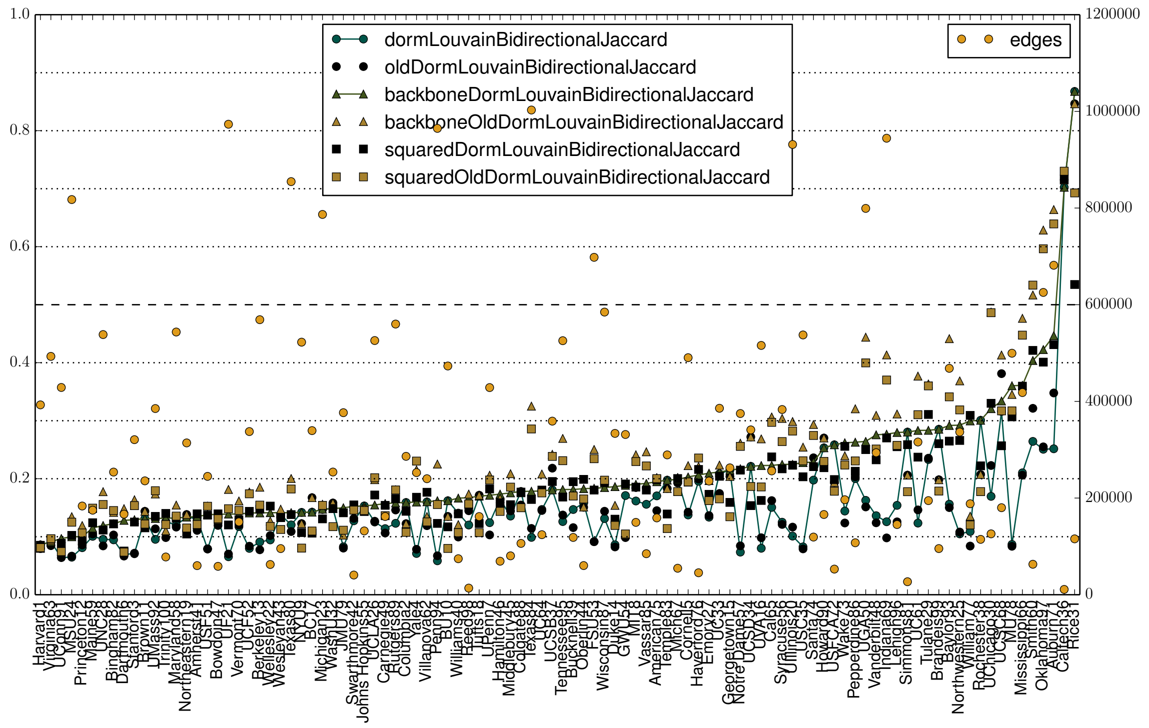


Figure 3.16: The same Jaccard indexes as in Figure 3.15 without the simple weighted backbone but with the values for the students that do not graduate in 2009.

clusters matches a lot better even though in many cases it is too low. Especially the number of the Louvain clusters on the squared weighted network comes in some cases really close to the number of dormitories.

The bidirectional Jaccard index of the dormitories and the Louvain clusters on the squared weighted network is for many networks similar to the bidirectional Jaccard index of the dormitories and the Louvain clusters on the best backbone and in some cases even exceeds it. The bidirectional Jaccard index of the dormitories and the Louvain clusters on the weighted network is in most cases a bit lower but in some cases even higher. It is interesting to see that even if the number of Louvain clusters on the squared weighted network almost matches the number of dormitories, the Jaccard index of these clusters can be very low.

As the Jaccard indexes of the Louvain clusters on the squared weighted networks matches the Jaccard index of the Louvain clusters on the backbones in many cases, we wondered if they also had the same behavior concerning the graduation year 2009. Therefore we also plotted the values for the students that do not graduate in 2009 in Figure 3.16. One can see that on many networks the Louvain algorithm behaves indeed similarly even though the actual Jaccard indexes do not match exactly.

We can conclude that for the quality in terms of the Jaccard index of the Louvain clustering and the dormitory clustering the squared weighted network is an interesting alternative to the backbone as it does not suffer from the problem that a lot of singletons and small clusters are generated but still leads to an increased Jaccard index of dormitories and Louvain clusters. For the graduation years the weighted and squared weighted networks are similar to the backbones only useful for very few networks.

3.9 Comparison by Modularity

In the previous sections our focus was the Jaccard index of Louvain clusters and the dormitories and graduation years. In this section we want to focus on modularity, the

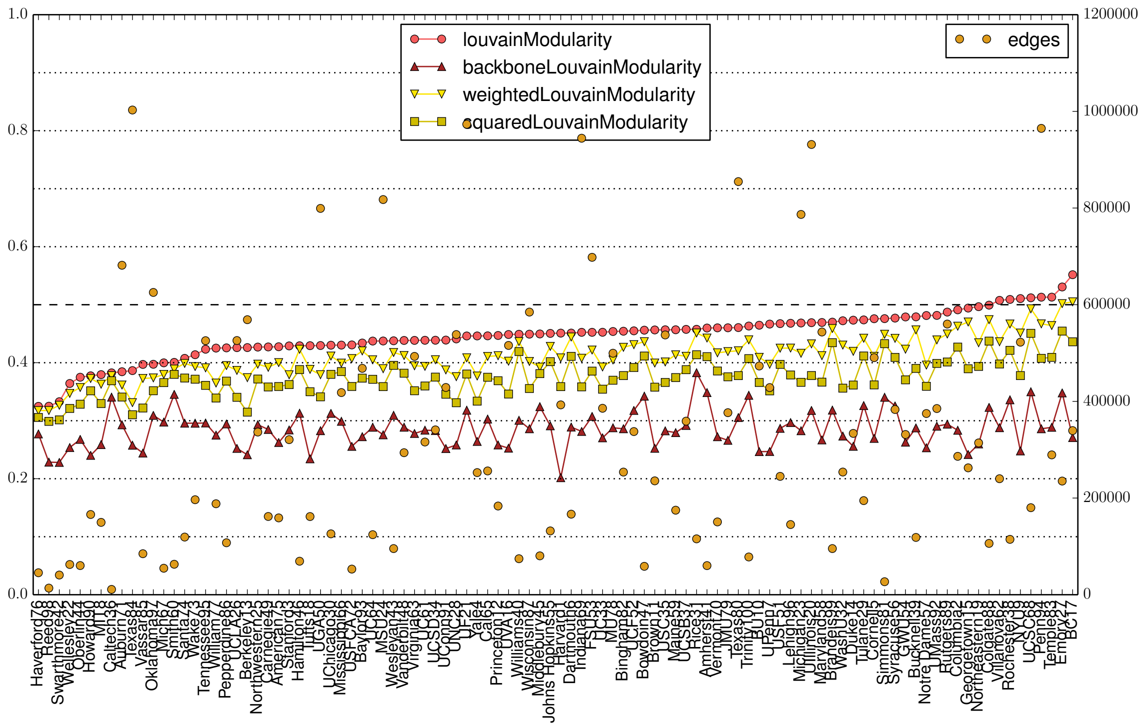


Figure 3.17: The modularity of the Louvain clusters calculated on the original network, on the weighted network, on the squared weighted network and on the backbone with overlap 5. All modularity values are measured on the original network. The networks are sorted by the modularity of the Louvain clusters that were calculated on the original network.

quality measure that is optimized by the Louvain algorithm. As a first step we will look at the modularity of the Louvain clusters that are calculated on the backbone and the weighted networks compared to the modularity of the Louvain clusters that are calculated on the original network. As a second step we will compare these values to the modularity of the dormitory and graduation year clusters in order to see if these values are a possible explanation why the Louvain algorithm sometimes finds the dormitory and sometimes the graduation year structure.

In Figure 3.17 we plotted the modularity of the Louvain clusters on the original network and the modularity of the Louvain clusters that were calculated on three variants of the networks: the Simmelian Backbone with overlap 5, the weighted and the squared weighted network. All modularity values are measured on the original network. In general the modularity values that the Louvain algorithm achieves on these networks are not very high with values between 0.4 and 0.5 for most networks (1 is the upper bound of modularity). When we execute the Louvain algorithm on the Simmelian Backbone the modularity on the original network drops to around 0.3. The modularity of the Louvain clusters that were calculated on the backbone does not seem to be correlated to the modularity of the Louvain clusters that were calculated on the original networks. On the weighted and the squared weighted networks the Louvain algorithm finds clusterings with modularity values that are closer to the modularity of the Louvain clusters that were calculated on the original network. This is not surprising as the weighted and the squared weighted network contain still all edges. That is, the Louvain algorithm is not constrained by the selection of the edges as done by the Simmelian Backbone algorithm and especially we do not get singletons and very small clusters since the network is not split into several connected components.

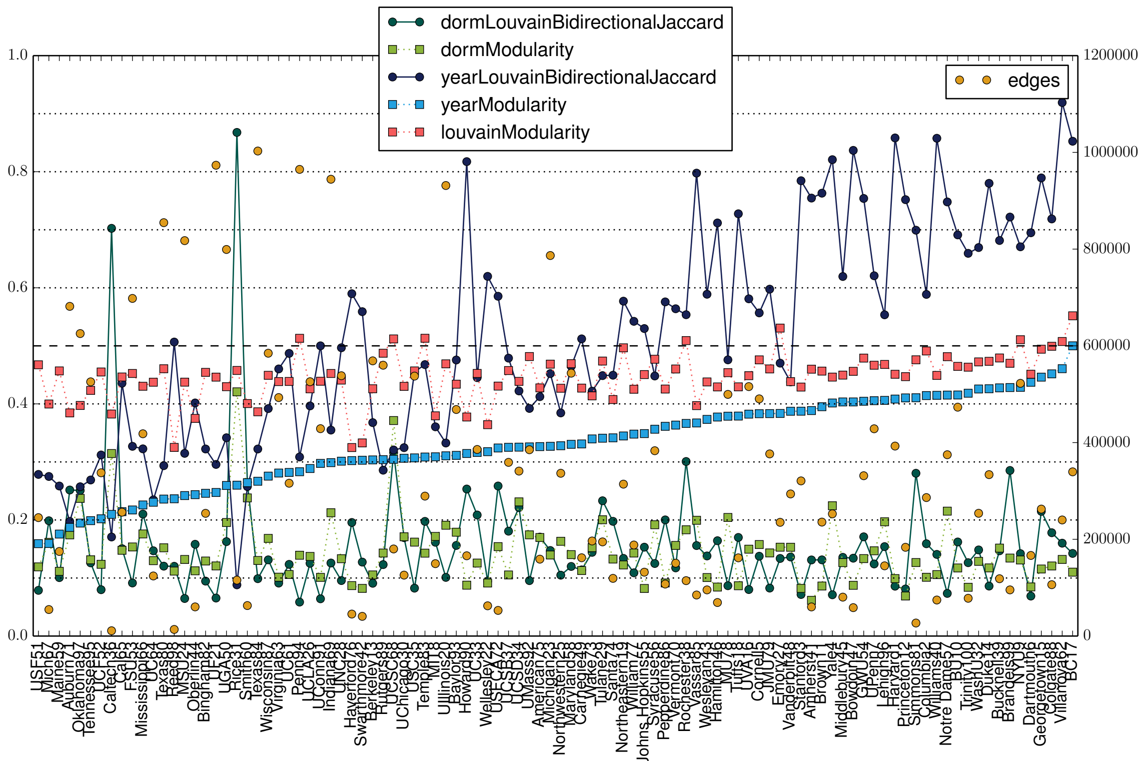


Figure 3.18: The bidirectional Jaccard index of the Louvain clusters calculated on the original networks and the dormitories and graduation years (dormLouvainBidirectionalJaccard and yearLouvainBidirectionalJaccard) compared to the modularity value of the dormitory, graduation year and Louvain clusters (dormModularity, yearModularity and louvainModularity). The networks are sorted by the modularity value of the graduation years.

We can thus conclude that the Simmelian Backbone algorithm is not suitable for increasing the quality of the clusterings of the Louvain algorithm in terms of modularity. This is also true for the weighted and squared weighted networks. It would be interesting to see if the modularity of the Louvain clusters that are calculated on the backbone gets higher when the modularity of the Louvain clusters that are calculated on the original network is significantly higher or if it remains at values below 0.4. If the modularity goes up, this would be a sign that the Simmelian Backbones gives a structure that is similar to the original network from the perspective of the Louvain algorithm but with less edges which could be interesting for developing highly scalable clustering algorithms for huge (social) networks.

In order to see if there is a correlation between the Jaccard indexes and the modularity of the clusterings on the original networks, we plotted the bidirectional Jaccard indexes of the Louvain clusters on the original networks and the dormitories and graduation years together with the modularity values of the dormitory, graduation year and Louvain clusterings in Figure 3.18.

For most networks the modularity of the graduation years is higher than the modularity of the dormitories but still below the modularity of the Louvain clusters. One can see a general tendency towards higher Jaccard indexes of graduation years and Louvain clusters when the modularity of the graduation years comes close to the modularity of the Louvain clusters, which is especially the case for the graduation years with higher modularity values.

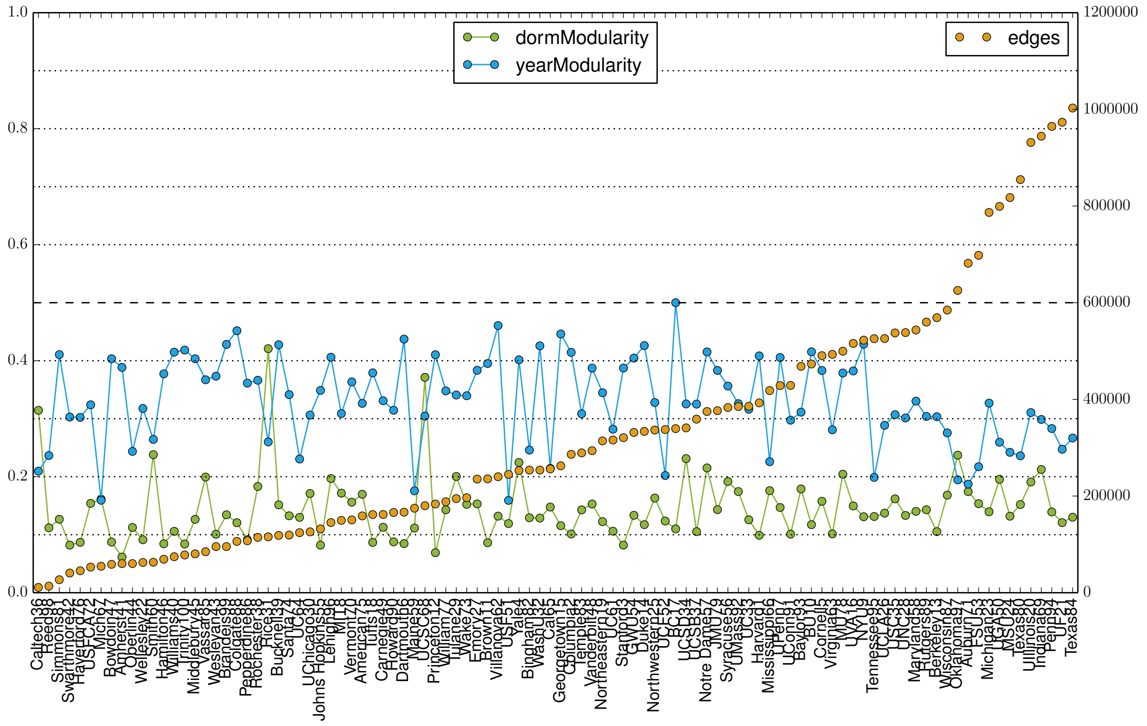


Figure 3.19: The modularity of the dormitory and the graduation year clusterings compared to the number of edges, the networks are sorted by the number of edges.

While the dormitories have in general much lower modularity values there are also four networks where the dormitories have a higher modularity value than the graduation years. Among them are Rice31 and Caltech36, the two networks with a high Jaccard index of the dormitories and the Louvain clusters, and also the other two networks have relatively high Jaccard indexes of the dormitories and the Louvain clusters.

In Figure 3.19 we plotted again the modularity values of the dormitories and the graduation years on the original networks, but this time sorted by the number of edges. While the values for the smaller networks are rather mixed, in none of the larger networks the graduation year clustering has a good modularity value. This is consistent with our earlier results that the Louvain algorithm was unable to find the graduation years in the large networks.

In Figure 3.20 we plotted only the modularity of the dormitories and the Louvain clusters and the Jaccard index of the Louvain clusters and the dormitories not only for the original network but also for the Simmelian Backbone with overlap 5. This time we calculated the modularity values on the backbone in contrast to Figure 3.17 where we calculated all modularity values on the original network. On the backbone, the Louvain algorithm detects clusterings which have a modularity value between 0.9 and 1 for most networks. The dormitories have higher modularity values than on the original network, but they only exceed 0.5 for five networks.

Even though high Jaccard indexes of the dormitories and the Louvain clusters calculated on the backbone seem to occur more frequently for networks where the dormitories have a high modularity value on the backbone there are also a lot of examples where the opposite is true. The large differences between the modularity values of the dormitories and the Louvain clusters on the backbone show that it is easy to find a clustering with good modularity values on the backbone, but this clustering will not match the dormitory structure. This is confirmed by the Jaccard indexes of below 0.5 for most networks.

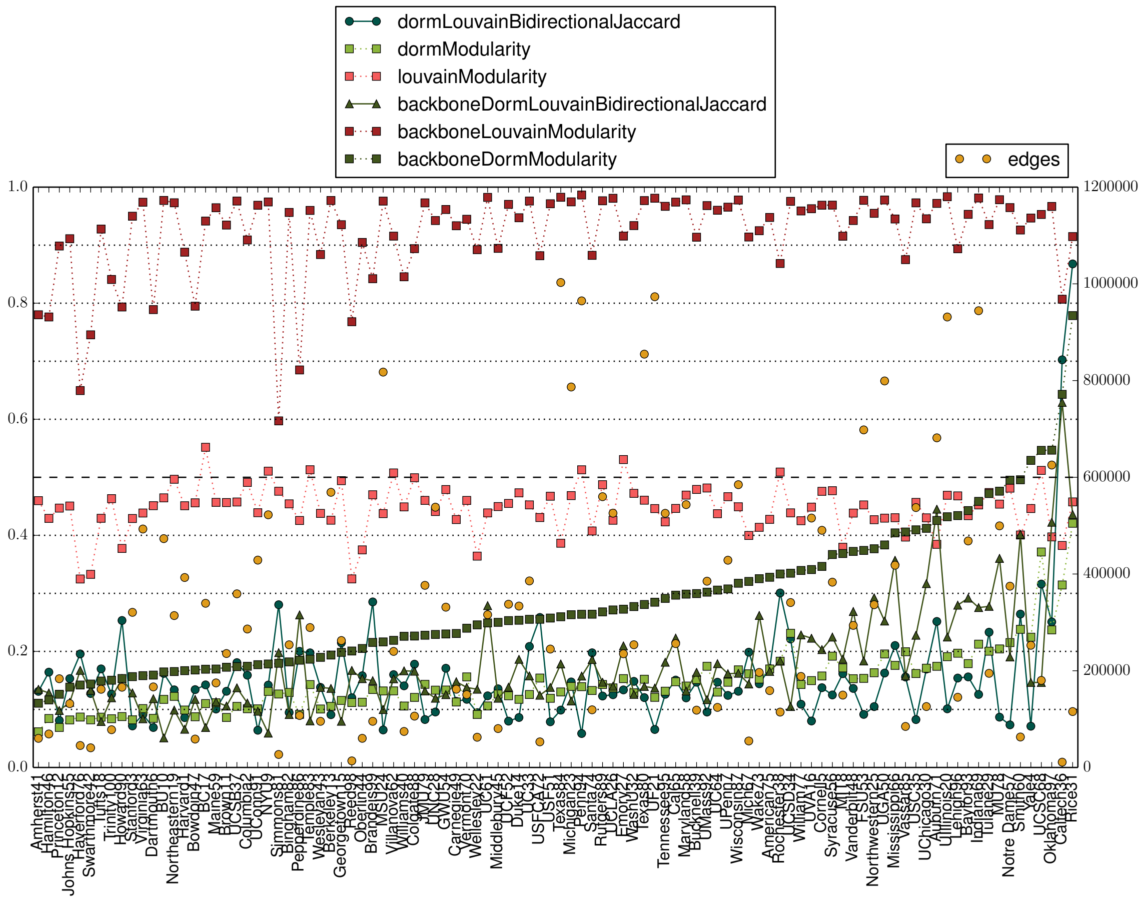


Figure 3.20: Comparison of the dormitories and the Louvain clusters on the original networks and the Simmelian Backbone with overlap 5 using the bidirectional Jaccard index and the modularity values. Note that the modularity for the Louvain clusters that are calculated on the backbone is also calculated on the backbone.

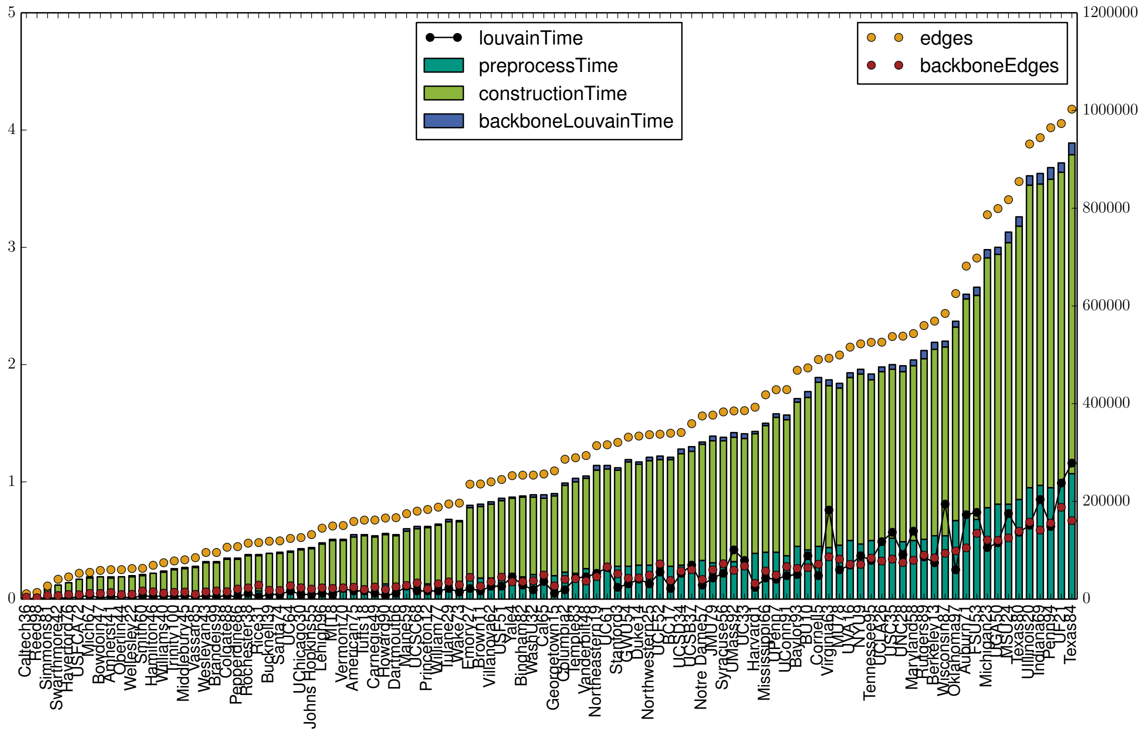


Figure 3.21: The running time of the backbone calculation and the Louvain algorithm both on the original network as well as one the Simmelian Backbone with overlap 5. The networks are sorted by the number of edges

We can conclude that on the original networks the Louvain clusterings whose modularity values match the modularity value of the dormitories/graduation years also have high Jaccard indexes with them. On the Simmelian Backbone with overlap 5 and also on the weighted and squared weighted networks the modularity of the Louvain clusters is lower than the modularity of the Louvain clusters that are calculated on the original network if we calculate all modularity values on the original network. The dormitories have, compared to the original network, higher but still relatively low modularity values on the Simmelian Backbones with overlap 5, but the Louvain clusters get very high modularity values on the backbones, which explains why the Louvain algorithm is for most networks not able to find the dormitories on the backbones.

3.10 Performance

Even though performance was not the primary focus of this work we briefly examine the running time of the Simmelian Backbone algorithm in contrast to the Louvain algorithm. In Figure 3.21 we plotted the running time of the Louvain algorithm on the original network and on the backbone with overlap 5 and also the preprocessing and construction time of the backbone. For comparison we also plotted the number of edges both in the original network as well as in the backbone. All times are in CPU seconds (user time) excluding graph reading and graph data structure construction time on a 2.66 GHz Intel®Core$^{\text{TM}}$2 Duo CPU.

The preprocessing time includes the triangle counting algorithm as well as constructing the ranked neighborhood of all nodes that can be used for all backbone parameters and variants, while the construction time consists of the calculation of the overlap. We did not optimize our implementation for maximum performance, using better data structures would definitely further reduce the running time.

One can easily see that the Simmelian Backbone algorithm even in our simplified and not so optimized version scales linearly in the number of edges.

Nevertheless the Louvain algorithm is faster than the Simmelian Backbone algorithm which means that for the Louvain algorithm the Simmelian Backbone algorithm does not make sense as preprocessing step from a performance point of view, letting alone the problem of the loss of quality of the resulting clustering.

However one can see that the Simmelian Backbone dramatically reduces the number of edges and that the Louvain algorithm on the backbone is a lot faster than on the original network (a factor of 10-20 sometimes). This means that the Simmelian Backbone algorithm could be interesting as preprocessing step for slower clustering algorithms that struggle especially with the high number of edges which can be found in social networks.

4. Conclusion

We examined 100 Facebook networks from US universities and colleges using the Louvain clustering algorithm and the Simmelian Backbone algorithm. In a first step we filtered the data to only include current students and not parts of the previous students and other university members, which increased the accuracy of the detection of the graduation years by the Louvain algorithm. Confirming the results of Lee et al. [LC13], we also found, with our evaluation using Jaccard indexes, that the clusters detected by the Louvain algorithm are in most cases more similar to the graduation years than to the dormitories. Only for two networks the Louvain algorithm was able to detect a clustering that is similar to the dormitories.

We found that the graduation year 2009 is different from the previous graduation years. These students, who started studying at the time of the data collection, are much better connected within their year than with the rest of the students. Our results show that for almost all networks the graduation year 2009 is much better matched by a cluster than the previous graduation years.

Executing the Louvain algorithm on a series of Simmelian Backbones, we found that the cluster structure on the backbones is different from the structure on the original networks, but that it is in most cases not beneficial for the detection of the graduation years. One reason for this is that clustering on the backbone yields more clusters, this is not surprising. The detection of the dormitories is significantly improved when clustering on a Simmelian Backbone, but in most cases only when we do not consider the students of the graduation year 2009 in the evaluation, which further confirms the special role of the graduation year 2009.

The Simmelian Backbone algorithm creates a lot of isolated nodes which are not isolated in the original network. This leads to Louvain clusters consisting of just one node. Therefore we did experiments with merging these singleton clusters into the cluster that gives the highest increase of modularity in the original network. For most networks this only led to a small increase of the accuracy of the detection of the dormitories and graduation years. For some networks, however, it seems that a clustering that is created by using the clusters on a very thin backbone as basis and then incrementally adding nodes to the clusters is more similar to the graduation years or sometimes also dormitories than the best matching Louvain clusters calculated on the backbones or the original networks.

Looking for a way to avoid the creation of singleton clusters we used the Simmelian Backbone algorithm in order to create edge weights instead of filtering the network and then executed the Louvain algorithm on the weighted networks which lead to similar results as clustering on the backbones, but without the singletons.

We also looked at the modularity of the resulting clusterings. The modularity on the original networks was decreased when the clustering was calculated on the Simmelian Backbone, but with the weighted networks the modularity values were closer to the clusters on the original networks. At least on the original networks we also found that the modularity of the dormitories and the graduation years as clusters in comparison to the modularity of the Louvain clustering is an indicator for the success of the detection of the dormitories or graduation years.

Concerning the performance we showed that the Simmelian Backbone scales linearly with the number of edges. On the Simmelian Backbone the Louvain algorithm was a lot faster, sometimes a factor of 10 or 20. Nevertheless in our implementation the Louvain algorithm executed on the original networks was still faster than the Simmelian Backbone calculation alone. This means that only with a more efficient Simmelian Backbone implementation it might be possible to use the Simmelian Backbone as speedup technique for the Louvain algorithm.

Alltogether this means that Simmelian Backbones are an interesting technique both for improving the quality of the clustering in certain aspects as well as for improving the performance of clustering algorithms.

Open Problems

There are many possible directions for future research in the area of clustering on filtered networks. The most obvious one is that more different parameters could be tried for the calculation of the Simmelian Backbones and that the experiments could be extended to other social networks.

However, apart from social networks, there are also other networks in which triangles can be observed. It might be interesting to examine different categories of networks in order to see if there are networks that are especially suited for using Simmelian Backbones. Another interesting question is which properties allow Simmelian Backbones to identify structures community detection algorithms are unable to identify.

As already mentioned, it seems that edges that are especially strongly embedded in the sense of the Simmelian Backbone could be used as seeds for a clustering. It would be interesting to develop clustering algorithms based on these seed edges.

Another interesting aspect is also the question which clustering algorithms are suitable for being executed on backbones in the sense that either the quality of the clustering is improved or that the use of the backbone decreases the running time or even both.

Concerning the quality, an open question is also if there are possibilities to further improve the quality of the clustering with a good post-processing algorithm that takes care of singletons that clearly belong to one of the clusters.

On a more fundamental level also different definitions of backbones could be tried or developed in order to maybe better select the included edges such that, for example, very small connected components are avoided.

Bibliography

[AJ03] L. Ana and A. Jain, "Robust data clustering," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2, June 2003, pp. II–128–II–133 vol.2.

[BDG$^+$08] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 2, pp. 172–188, 2008.

[BGLL08] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008. [Online]. Available: http://stacks.iop.org/1742-5468/2008/i=10/a=P10008

[BHJ09] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *ICWSM*, 2009, pp. 361–362. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154

[CN85] N. Chiba and T. Nishizeki, "Arboricity and subgraph listing algorithms," *SIAM Journal on Computing*, vol. 14, no. 1, pp. 210–223, 1985. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/0214017

[Dek06] D. Dekker, "Measures of simmelian tie strength, simmelian brokerage and the simmelianly brokered," *Journal of Social Structure*, vol. 7, no. 1, pp. 1–22, 2006.

[FB07] S. Fortunato and M. Barthélemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007. [Online]. Available: http://www.pnas.org/content/104/1/36.abstract

[FR91] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software: Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, 1991. [Online]. Available: http://dx.doi.org/10.1002/spe.4380211102

[GGHW10] R. Görke, M. Gaertler, F. Hübner, and D. Wagner, "Computational Aspects of Lucidity-Driven Graph Clustering," *Journal of Graph Algorithms and Applications*, vol. 14, pp. 165–197, 2010.

[Gö10] R. Görke, *An Algorithmic Walk from Static to Dynamic Graph Clustering.* Karlsruhe, 2010. [Online]. Available: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000018288

[Hun07] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[KH04] L. Kuncheva and S. Hadjitodorov, "Using diversity in cluster ensembles," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 2, Oct 2004, pp. 1214–1219 vol.2.

[LC13] C. Lee and P. Cunningham, “Benchmarking community detection methods on social media data,” *CoRR*, vol. abs/1302.0739, 2013.

[LEM] “Library for Efficient Modeling and Optimization in Networks (LEMON).” [Online]. Available: http://lemon.cs.elte.hu/trac/lemon

[LRMH10] C. Lee, F. Reid, A. McDaid, and N. Hurley, “Detecting highly overlapping community structure by greedy clique expansion,” *arXiv preprint arXiv:1002.1827*, 2010, implementation source and binaries available at http://sites.google.com/site/greedycliqueexpansion/.

[MGH11] A. F. McDaid, D. Greene, and N. Hurley, “Normalized Mutual Information to evaluate overlapping community finding algorithms,” *ArXiv e-prints*, Oct. 2011.

[New04] M. E. J. Newman, “Analysis of weighted networks,” *Phys. Rev. E*, vol. 70, p. 056131, Nov 2004. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.70.056131

[NG04] M. E. J. Newman and M. Girvan, “Finding and evaluating community structure in networks,” *Phys. Rev. E*, vol. 69, p. 026113, Feb 2004. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.69.026113

[Nic13] B. Nick, “Simmelian backbone extraction - visone user support,” 2013, [Accessed 31-May-2014]. [Online]. Available: http://visone.info/wiki/index.php?title=Simmelian_backbone_extraction&oldid=1360

[NLCB13] B. Nick, C. Lee, P. Cunningham, and U. Brandes, “Simmelian backbones: Amplifying hidden homophily in facebook networks,” in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ser. ASONAM ’13. New York, NY, USA: ACM, 2013, pp. 525–532. [Online]. Available: http://doi.acm.org/10.1145/2492517.2492569

[OB14] M. Ortmann and U. Brandes, “Triangle listing algorithms: Back from the diversion,” in *ALENEX*, C. C. McGeoch and U. Meyer, Eds. SIAM, 2014, pp. 1–8.

[Sim08] G. Simmel, *Soziologie: Untersuchungen über die Formen der Vergesellschaftung.* Duncker & Humblot, 1908.

[Sim50] ——, *The sociology of Georg Simmel.* Simon and Schuster, 1950, vol. 92892.

[TMP11] A. L. Traud, P. J. Mucha, and M. A. Porter, “Social structure of facebook networks,” *CoRR*, vol. abs/1102.2166, 2011.

[WEK02] R. Wiese, M. Eiglsperger, and M. Kaufmann, “yfiles: Visualization and automatic layout of graphs,” in *Graph Drawing*, ser. Lecture Notes in Computer Science, P. Mutzel, M. Jünger, and S. Leipert, Eds. Springer Berlin Heidelberg, 2002, vol. 2265, pp. 453–454. [Online]. Available: http://dx.doi.org/10.1007/3-540-45848-4_42

[YL13] J. Yang and J. Leskovec, “Overlapping community detection at scale: a nonnegative matrix factorization approach,” in *Proceedings of the sixth ACM international conference on Web search and data mining.* ACM, 2013, pp. 587–596.

[YML14] J. Yang, J. McAuley, and J. Leskovec, “Community detection in networks with node attributes,” *arXiv preprint arXiv:1401.7267*, 2014.