Karlsruhe Institute of Technology

Iterative Local Community Detection Combining Graph Structure and Attribute Similarity

Bachelor's Thesis of

Jonas Krautter

at the Department of Informatics
Institute for Program Structures and Data Organization (IPD)

Reviewer:	Prof. Dr.-Ing. Klemens Böhm
Second reviewer:	Prof. Dr. Dorothea Wagner
Advisor:	Dr. rer. nat. Emmanuel Müller
Second advisor:	Michael Hamann, M.Sc.
Third advisor:	Dipl.-Inform. Patricia Iglesias Sánchez

15th December 2014 – 10th April 2015

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 10th April 2015

. . . . . .

(Jonas Krautter)

Abstract

Graph clustering has received a lot of attention by scientists and many algorithms have been developed to approximate solutions to variants of the problem, which is usually NP-hard. Lately methods have been proposed to combine attribute information and graph topology into the clustering process. However, none of the available combining algorithms offers a local approach, able to detect a community around a given seed node or a set of seed nodes. Therefore, we present an iterative solution in this thesis: Learning attributes in a local environment, adding edges to represent attribute similarity and using a known local structural algorithm, to provide a method for local community detection that respects both graph structure and attribute data. Evaluation results of the method on synthetic data and real world data from the Facebook friendship network and the Amazon co-purchase network prove the usefulness of the method to improve the detection of communities and identify communities in overlapping structures.

Zusammenfassung

Eine Vielzahl von Wissenschaftlern hat sich der Zerlegung von Graphen in sogenannte Communities bereits gewidmet und die Zahl der Algorithmen, die für die Variationen des zumeist NP-schweren Problems bereits entwickelt wurden, ist ebenso groß. Neuerdings werden auch Algorithmen vorgeschlagen, die zusätzlich zur Struktur des Graphen auch die Attribute der Knoten in den Prozess mit einbeziehen. Jedoch gibt es noch keinen dieser Algorithmen, die Attribute und Struktur nutzen, der als lokaler Clustering-Algorithmus, der die Communities aus der Umgebung von gegebenen Startknoten heraussucht, genutzt werden kann. Deshalb präsentiert diese Arbeit einen iterativen Ansatz: Attribute werden in einer lokalen Umgebung erlernt, Kanten werden hinzugefügt, um die Attributsähnlichkeit zweier Knoten strukturell zu repräsentieren und ein bereits bekannter, lokaler, struktureller Algorithmus wird dann verwendet, um insgesamt einen lokalen Algorithmus zur Suche nach Communities, der sowohl Attribute als auch Graph-Struktur einbezieht, zu erhalten. Die Ergebnisse der experimentellen Anwendung auf künstlich erzeugten Daten sowie Daten aus dem Facebook-Freundschaftsgraphen und Daten des Amazon Online-Versandhauses zeigt die Nützlichkeit der Methode zur Verbesserung der Qualität von erkannten Communities und zur Identifikation von Communities in überlappenden Strukturen.

Contents

List of Figures

List of Tables

1. Introduction

A vast quantity of real world datasets and especially networks can be modelled as a graph, which is a collection of nodes together with relationships or connections, represented by edges in between them. Therefore, this generalized data structure has received a lot of attention by computer scientists, biologists, physicists and social scientists. Many algorithms have been developed to extract information from graph data in order to gain new insights into the real data behind the graph representation. In order to understand the structure especially of large graphs the user is often interested in finding dense subsets – the so called communities – of the nodes within the graph, where the nodes within the subset are densely interconnected but have only few edges to exterior nodes. Clustering algorithms often try to detect those subset by optimizing an objective function, that estimates the quality of detected communities. While a lot of objective functions and algorithms to provide clustering solutions exist, the problem is usually NP-hard. The interested reader may find an overview on the subject in [4].

In this thesis we will focus not on clustering an entire graph but rather on detecting different communities locally around a given seed node. A user might be interested in identifying a group of people around a given person in a social network or the functional subset a molecule belongs to within a biochemical network. A local method is preferred over global clustering in certain applications, for example, if the amount of data is too large to use a global method which has a running time dependent on the graph size. Further advantages of a local community detection method will be introduced in the next chapter. Today's local community detection algorithms use the edge structure of a given graph to identify a dense subset. But real world data often provides more than just structure to gather information about communities. Usually the data source provides attributes assigned to the nodes, such as personal information about a person in a social network or prices and product categories in a shop. Since attributes are often correlated with communities, the usage of attribute information during detection of a local community can lead to a significant increase in quality of the detected subsets. For example, in a social network the persons within a structural dense subset usually share similar interests or attributes like their place of residence or their age.

Thus we suggest a combination of known global clustering techniques for attributed graphs and local algorithms for non-attributed graphs in this thesis to combine edge structure and attribute information for a local detection of communities. Since the attributes have to be discovered first from the local view of a seed node, the approach is an iterative one. While performing only structural clustering on the first step of the algorithm, a second iteration will lead to different results in the detected community, since the algorithm now knows about attributes in the local environment of the seed node. Changing the attribute similarity functions can then lead to even more alternative discovered structures.

Another advantage of the iterative approach is the potential to detect overlapping community structures. A person in a social network might be a member of their high school soccer team but also a member of their high school class and both subsets are likely more dense than an arbitrary one. An algorithm only respecting graph topology might identify one of the communities or maybe both, but by using the available attributes of the person nodes one can direct the algorithm to identify one specific community by changing the similarity function respectively.

The actual proposed method is a combination of adding weighted edges or weight to existing edges to represent attribute similarity (inspired by the CODICIL-algorithm [12]) and a structural local community detection algorithm such as PageRank-Nibble [1] or Greedy Community Expansion [16]. Whenever the local structural algorithm performs a step on a previously unknown node in the graph, the attribute similarity of that node to all other nodes, that have been discovered so far, is computed and edges are added to the graph, whenever the similarity exceeds a user-given threshold. This leads to different results after the second iteration of the local algorithm. We named the developed method *Iterative Local Selective Community Detection (ILSCD).*

The thesis is structured as follows: In the next Chapter 2 we will present existing algorithms on local clustering and algorithms using both attribute similarity and graph structure. Chapter 3 will introduce the theoretical assumptions and notions used throughout the thesis and provide detailed information about graph structure and attribute vectors. Afterwards in Chapter 4 we will elaborate the algorithmic details and the time and space complexity of the algorithm will be discussed. Evaluation results of the method on synthetic data generated by an overlapping variant of the LFR-benchmark [6] together with an attribute generation model and also on real world data from the social network Facebook and a subset of the Amazon co-purchase network, which was used in [10], will be provided in Chapter 5. Finally the last Chapter 6 will summarize the results of this thesis and outline some ideas for further development and research.

2. Related work

A single definition of the graph clustering problem that is accepted in general does not exist and a great variety of objective functions and methods to optimize them have been proposed [4]. The most distinguishing feature of the ILSCD method is the combination of structural and attribute information for local clustering. However, efforts have been made to achieve attribute and topology combination earlier and using a local approach for graph clustering has become popular as well. Therefore, we will present a selection of available methods here. A tabular overview of all presented methods compared to the work in this thesis is available at the end of the chapter.

2.1. Local structural community detection

Various local graph clustering algorithms have been proposed and two of them shall be outlined in this thesis, since the iterative approach to include attribute similarity makes use of them. In the first part, however, we will explain, why a local algorithm is sometimes preferred over a global clustering variant. At the end of this section we will provide a tabular overview (Table 2.1) of the presented algorithms and methods, comparing the features of them in order to emphasize the new features introduced by this work.

2.1.1. Advantages of local community detection

A graph clustering algorithm is classified as a local algorithm, if it does not work on all nodes or edges of a graph but rather evolves communities from a given node subset, the so called seed nodes. That does not necessarily mean that the algorithm does not know the entire graph in a later state of execution, but at first no further information than the seed nodes and their neighbours is available and no global graph parameters – such as the number of nodes, the number of edges or the average degree – may be accessed. The easiest variant of a local algorithm is one that finds a single community around a given seed node, which is what will be considered in this thesis. In various applications a local method is preferred over a global algorithm.

Whenever a user is interested in a single community only, global clustering is wasting computation time and resources on information that is not relevant to the user afterwards. For example a web search engine might want to present the user a set of relevant other websites with respect to the current site the user is visiting. A locally evolved community around the current site as a seed node could be delivered as an adequate result. A global clustering of the web graph is certainly not possible to compute, which leads to another advantage of locally operating algorithms: The user does not need to know the entire graph for local community detection. The data source available to the user might not even

allow arbitrary access to nodes, edges and attributes of the graph. Once more the example of the web graph is fitting: A clustering algorithm would first require the user to extract a certain amount of nodes from the web and store them for later usage. Furthermore, speed is another argument why someone might chose the local variants, since local algorithms usually have a running time dependent on the size of the discovered community and its quality and not on the size of the entire graph. Networks have grown enormously lately and datasets like the social network Facebook with 1.19 billion monthly active users as of December 2014 [3] are difficult if not impossible to analyse globally.

All in all various reasons exist for a user to work with local community detection algorithms. Of course there is also disadvantages to them – usually in terms of quality of the discovered subsets – since a local algorithm has always less information than a global one.

2.1.2. PageRank-Nibble

The *PageRank-Nibble* algorithm [1] by Andersen et al. is an improved version of the Nibble algorithm [15] by Spielman et al. It can be used to detect a single community around a given seed node by approximating a local PageRank-vector and optimizing conductance for sets, selected by a PageRank-induced ordering. PageRank-Nibble has been introduced as a local algorithm on unweighted and undirected graphs. The *volume* of a node subset on an unweighted and undirected graph $G = (V, E)$ is defined as the sum of all node degrees within that subset $\text{vol}(C \subset V) = \sum_{u \in C} \deg(u)$ where $\deg(u) = |\{(u, v) \in E\}|$. *Conductance* on an unweighted undirected graph $G = (V, E)$ is defined as

$$\Phi(C) = \frac{\left|\{(u, v) \in E \mid u \in C \wedge v \notin C\}\right|}{\min\left(\text{vol}(C), \text{vol}(V \backslash C)\right)}.$$

Since the attribute similarity is represented as a weighted edge within the ILSCD method, the algorithm has been adapted to work on weighted graphs by extending the respective attributes such as node degree and volume to their counterparts on weighted graphs, which will be elaborated in detail in the next chapter and in Chapter 4. Its advantage is the almost linear time complexity, dependent on the size and quality of the desired community. A lazy random walk variant is used to approximate the PageRank-values and the algorithm takes two input parameters to control quality and size of a detected community. The parameter α defines the probability for the random walk to remain on the current node, while ϵ controls the amount of residual probability for each node, which is relevant to the approximation quality. The running time of PageRank-Nibble is $O\left(2^b \cdot \log_3(m)/\Phi^2\right)$ where 2^b is the minimum volume of the community to be found and Φ is the minimum conductance. Both variables affect the values of α and ϵ, but we have used a variant of PageRank-Nibble that can directly be controlled via α and ϵ, which will be explained later.

2.1.3. Greedy Community Expansion (GCE)

Another class of local community detection algorithms is that of *Greedy Community Expansion* [16]-algorithms. To find a local minimum or maximum of an objective function,

the algorithm starts with a set $C = \{v\}$, where v is a user-given seed node. Then the algorithm iterates over the neighbourhood of S to find a node which can be added to C to improve the value of the objective function of C. If no further improvement of the community quality can be achieved by including a node from the neighbourhood of C into the set, the algorithm terminates and outputs C as a local maximum/minimum of the objective function. A possible objective function on a graph $G = (V, E)$ could be the following one, which is the fraction of intra-community edges to inter-community edges "M" taken from [16]:

$$M(C) = \frac{|\{(u, v) \in E \mid u \in C \wedge v \in C\}|}{|\{(u, v) \in E \mid u \in C \wedge v \notin C\}|}$$

Dividing the sum of edges within C through the sum of outgoing edges from C, the GCE algorithm wants to maximize this objective function in order to find a dense subset. Since the ILSCD method proposed in this thesis works with any local detection algorithm, that is available for weighted graphs, both PageRank-Nibble and GCE have been used for evaluation. We have adapted both objective functions – conductance and "M" – to work on weighted, undirected graphs, which will be elaborated in Chapter 4.

2.2. Attribute respecting algorithms

Proposals for graph partitioning algorithms, that combine topology and attribute similarity, have been made and some examples will be outlined here. The CODICIL-algorithm [12] is to be emphasized, since the method we propose in this thesis adapts the idea of adding edges to a local variant.

2.2.1. Cohesive Pattern Miner (CoPaM)

The *Cohesive Pattern Miner* [9] by Moser et al. introduces the concept of cohesive patterns, which are subgraphs fulfilling the cohesive pattern constraints density, cohesion and connectivity. Cohesion represents the attribute similarity of the nodes, while the density constraint ensures the discovered subgraphs are densely connected. However the algorithm does not try to maximize/minimize cohesion or density but rather finds a maximum cohesive pattern (subgraph) that fulfils the cohesion and density constraints for user-given thresholds. CoPaM takes an attributed graph $G = (V, E, D, A)$ where V are the nodes, E are the edges, $D = \{D_1, D_2, ...D_a\}$ is the attribute space and A is a function $A : V \rightarrow D_1 \times D_2 \times ... \times D_a$ that assigns attributes from the attribute space to the nodes. A user also needs to input a cohesion function $s : 2^V \times 2^D \times \mathbb{R} \rightarrow \{\text{True}, \text{False}\}$ and thresholds $\alpha, \Theta_s, \Theta_d$. The cohesion function defines if a subset of the nodes $C \subset V$ and their attributes w.r.t. a subset of the feature space $A \subset D$ fulfil the cohesion constraint for the given threshold Θ_s. Thresholds α and Θ_d affect the density and size of the feature space subset of the patterns, which are to be discovered. This makes it difficult for the user to apply the algorithm on a graph with unknown structure, since it tries to maximize the size of the feature space and node subsets and not the density and cohesion of nodes. A user who

wants to discover certain communities needs to know about their density and cohesion beforehand.

2.2.2. SA-Cluster

While the approach proposed in this thesis relies on the addition of edges to the existing graph, the *SA-Cluster* algorithm [21] by Yang Zhou et al. represents attribute similarity by node vicinity. A given attributed graph $G = (V, E, \wedge)$, where $\wedge = \{a_1, a_2, ..., a_d\}$ is a set of functions, which provide the attributes of a node in a way that $a_j(v)$ denotes the value of v on the attribute a_j, is augmented by the addition of so called attribute vertices to the existing graph. An attribute vertex (a_j, a_{jk}) represents a value of a_{jk} on attribute a_j. After adding the attribute vertices to the graph, the nodes are connected to them in a way that a node that has the value a_{jk} on attribute a_j is connected to the respective attribute vertex (a_j, a_{jk}). This approach – like the ILSCD method – makes it possible to use any structural partitioning algorithm – including a local one – on the augmented graph. However, the entire method does not work locally, since the addition of the attribute vertices requires to iterate over the entire graph. The use of attribute vertices also allows only discrete attribute dimensions and allows attribute similarity only between nodes that share exactly the same value on a certain attribute.

2.2.3. Attribute-aware modularity (MAM)

Maximization of Attribute-aware Modularity (MAM) [14] extends the well known structural clustering quality function *modularity* to an attributed variant, that also respects the attribute similarity between nodes within the clusters. The adjacency matrix of an unweighted graph $G = (V, E)$ is defined as

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

One of the definitions for modularity using the above definition of the adjacency matrix is

$$Q = \sum_{i,j \in V} \left(\frac{A_{ij}}{2 \cdot m} - \frac{k_i \cdot k_j}{(2 \cdot m)^2} \right) \cdot \delta(c_i, c_j),$$

where k_i is the degree of node i, m the total number of edges in the graph and c_i the community a node i belongs to. The measurement is extended using the definition of attribute compactness AC, which is the sum of the attribute relevances for every available attribute. This attribute compactness reflects the homogeneity of all attributes within a single cluster of a given graph clustering. The *Attribute-aware modularity* of a clustering $C = \{C_1, C_2, ..., C_k\}$ is then defined as

$$AQ(C) = \sum_{C_i \in C} AC(C_i) \cdot Q(C_i).$$

method/ algorithm	local algorithm	respecting attributes	arbitrary attribute domain	attribute weighting	respecting edge weights
CoPaM	×	✓	✓	✓	×
SA-Cluster	×	✓	×	×	✓
MAM	×	✓	×	×	✓
CODICIL	×	✓	✓	×	✓
PageRank-Nibble	✓	×	×	×	✓
GCE	✓	×	×	×	✓
ILSCD	✓	✓	✓	✓	✓

Table 2.1.: Comparative overview of related methods

This definition of modularity can be used with any modularity-driven clustering algorithm including even multilevel algorithms, for which their work proposes a generalized method to merge nodes with respect to their attributes. It is designed for global algorithms though, since modularity is a measurement for a global clustering of an entire graph. Further disadvantages may be the restriction to numeric attributes and the lack of a possibility for the user to provide a weighting to the attribute vectors.

2.2.4. CODICIL

The abbreviation *CODICIL* [12] stands for *COmunity Discovery Inferred from Content Information and Link-structure*. In their work Yiye Ruan et al. suggest an algorithm that combines content information and links from web graphs by adding edges representing content similarity to the graph. This is achieved by computing the content similarity using term vectors for every pair of nodes in the graph. The resulting union of edges is then sampled, retaining only the edges to the most relevant neighbours of a node. The amount k of neighbours to retain during the sampling process for each node must be provided by the user. Finally an arbitrary clustering algorithm can be used to detect communities on the sampled graph. While this approach allows arbitrary attribute domains, the time complexity of the preprocessing step lies within $O\left(n^2 \cdot \log(n)\right)$, since the algorithm has to iterate over all pairs of nodes, making it unusable as a local approach. However, we have used a local variant of this method in order to represent attribute similarity in the ILSCD algorithm.

3. Model

In this chapter we will introduce terminology and notions used throughout the thesis and we will establish a model of attributed graphs on the basis of which the algorithms and experiments have been developed.

3.1. Preliminaries

An *undirected attributed graph* is a tuple of three sets and a function $G = (V, E, A, \text{weight})$ where V with $|V| = n$ is the set of nodes, E with $|E| = m$ is the set of edges and $A = \{a_0, a_1, ..., a_{n-1}\}$ where $|a_0| = |a_1| = ... = |a_{n-1}| = d$ is the set of node attributes with dimension d. The nodes are identified by an integer value from the interval $[0, n)$. The elements of E are denoted as tuples (u, v) where $u \in V$ is the source node and $v \in V$ is the target node. On an undirected graph the order of the nodes in an edge tuple (u, v) is irrelevant, which means $\forall (u, v) \in E : (v, u) \in E$. The weight function $\text{weight} : E \to \mathbb{R}$ is a function that assigns a weight to each edge of the graph (see next paragraph).

The *weight* of an edge is denoted as $\text{weight}(u, v)$, where $(u, v) \in E$. On an unweighted graph the weight is $\text{weight}(u, v) = 1$ for all edges $(u, v) \in E$ and for an undirected graph, the equation $\text{weight}(u, v) = \text{weight}(v, u)$ holds for all edges.

The *degree* of a node u is $\deg(u) = \sum_{(u,v)\in E} \text{weight}(u, v)$ for undirected graphs. Note that this definition of the degree is equivalent to the number of incident edges of the node on an unweighted and undirected graph.

The *volume* of a subset $S \subseteq V$ is the sum of the node degrees: $\text{vol}(S) = \sum_{u \in S} \deg(u)$. The volume of the entire graph is $\text{vol}(V)$, which is $2 \cdot m$ on an unweighted graph.

3.2. Graph structure

In this thesis we will consider only undirected, simple graphs without loops and multi-edges and we will assume $\forall (u, v) \in E : weight(u, v) > 0$. No further restriction needs to be put on the structure of the input graphs.

Since the ILSCD method can be used with arbitrary local detection algorithms and therefore, with various objective functions a *community* is simply considered a subset of a graph's nodes $C \subset V$. We will not introduce any definition based on the quality of the subsets.

During evaluation we have used several objective functions and quality measures:

PageRank-Nibble uses *conductance* as an objective function to find a local structural community. This measurement has been adapted for usage on weighted graphs in this thesis. We define the weighted conductance of a community $C \subset V, C \neq \emptyset$ as follows:

$$\Phi(C) = \frac{\sum_{(u,v)\in E, u\in C, v\notin C} \text{weight}(u,v)}{\min(\text{vol}(C), \text{vol}(V\backslash C))}$$

If the weighted sum of edges connecting nodes inside of C to nodes outside of C is large compared to the volume of C, then the subset C is not a dense subset of V. Thus the PageRank-Nibble algorithm seeks to minimize Φ for the detection of a community around a seed node. The values of conductance lie within $[0, 1]$. A subset C with no edges connecting the nodes of the subset to exterior nodes has a conductance of $\Phi(C) = 0$, while a set of nodes that are not connected to each other at all has a conductance of $\Phi(C) = 1$.

We have also adapted the objective function "M", which is locally maximized by Greedy Community Expansion, to be able to use it on weighted graphs:

$$M(C) = \frac{\sum_{(u,v)\in E, u\in C, v\in C} \text{weight}(u,v)}{\sum_{(u,v)\in E, u\in C, v\notin C} \text{weight}(u,v)}$$

To measure the quality of community detection on synthetic data, we consider the *ground truth memberships*, which are the members of each community defined as $M = \{S_0, S_1, \ldots, S_{c-1}\}$, where c is the number of communities and $S_i \subset V$ for $i = 0, \ldots, c-1$. Note that this definition allows overlapping communities, since the communities $S_0, S_1, \ldots, S_{c-1}$ do not need to be disjoint. To compare the result produced by an algorithm with the ground truth community membership of the nodes, the *Jaccard-index* has been used, which estimates the equality of two sets A and B using the fraction of cut and union:

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

The *Jaccard-quality* of a community C with respect to the given ground truth is then defined as

$$JQ(C,M) = \max(J(C,S_0), J(C,S_1), \ldots, J(C,S_{c-1})).$$

3.3. Attribute structure

To simplify the computation of similarities, we assume the node attributes to be numeric, but not necessarily integer vectors: $a_i \in \mathbb{R}^d$. This makes it easier to use well known similarity functions on the attribute vectors. However, the similarity function for use with the ILSCD method is completely arbitrary and may be adapted by the users to their applications. Allowing a weight vector as an additional input to the similarity function makes it possible for the user to identify various alternative and even overlapping communities by weighting the attributes respectively. In general a *similarity function* has the following signature:

$$\text{similarity} : \mathbb{R}^d \times \mathbb{R}^d \times [0,1]^d \to [0,1]$$

The function takes two attribute and a weight vector and outputs a positive real number reflecting the attribute similarity between the two attribute vectors. Entries of the weight

vector must be within a range of $[0, 1]$ where a value of 0 causes the similarity function to not take the respective attribute into account at all, and a value of 1 includes the attribute fully into the similarity computation.

For the ILSCD method the cosine-similarity and the Jaccard-similarity have been used to calculate attribute similarity (definition follows). Both definitions are taken from [12] and extended with the possibility of adding a weight vector to the input. Additionally, to use the method on Facebook graph data, we will present another similarity function, that simply counts the number of attributes which are equal for a pair of attribute vectors. This has led to better results during evaluation, since an attribute like the home town of two persons can not be easily compared using a continuous similarity function. Those attributes are either equal (similarity value of 1) or non-equal (zero similarity).

The *weighted cosine-similarity* is defined as the cosine of the angle between two attribute vectors, where the direction of the vectors along a specific axis (the relevance of a specific attribute to the angle) can be controlled by the weight vector:

$$\text{cossim}(a, b, w) = \frac{\sum_{i=0}^{d-1} w[i] \cdot a[i] \cdot w[i] \cdot b[i]}{\sqrt{\sum_{i=0}^{d-1} (w[i] \cdot a[i])^2} \cdot \sqrt{\sum_{i=0}^{d-1} (w[i] \cdot a[i])^2}}$$

Likewise the *weighted Jaccard-similarity* is defined as follows:

$$\text{jaccardsim}(a, b, w) = \frac{\sum_{i=0}^{d-1} \min(a[i], b[i]) \cdot w[i]}{\sum_{i=0}^{d-1} \max(a[i], b[i]) \cdot w[i]}$$

The *weighted counting-similarity* function, which has been used on Facebook data, applies the weight vector on the binary equality of each attribute:

$$\text{countsim}(a, b, w) = \frac{\sum_{i=0}^{d-1} w[i] \cdot \delta(a[i], b[i])}{\sum_{i=0}^{d-1} w[i]}, \text{ where } \delta(x, y) = \begin{cases} 1 & x = y; x \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

By enforcing $x \neq 0$ for binary equality, the value of 0 can be used for unspecified attributes, defining two unknown values in an attribute as unequal.

We will present evaluation results for all similarity functions.

4. Method

In this chapter we will present the main algorithm of the ILSCD method and elaborate it in detail with the aid of pseudo code. At first we will give a short introduction to the two structural local algorithms PageRank-Nibble and Greedy Community Expansion, since they have been used in combination with the ILSCD method. Both have been described in Chapter 2 already, but we will elaborate them in detail and point out the differences between the original algorithms and the variants used for this thesis. The time and space complexity will be discussed and compared to other local clustering algorithms at the end of the chapter. Yet the comparison is difficult, as it depends on the underlying local structural community detection algorithm and the ILSCD method is an iterative one and might require the user to perform several steps until the result is satisfying.

4.1. PageRank-Nibble

The PageRank-Nibble algorithm is based on the approximation of a local PageRank [11] vector using the principle of random walks. It is composed of two steps: The local PageRank approximation and the identification of a set with minimum conductance. PageRank was originally developed as a method to rank web pages by their importance and relevance within the web graph. This ranking value is determined by analysing the graph structure, based on the assumption that nodes with a greater importance are more likely to receive links than nodes with small importance. For the approximation of the PageRank-values on a local subset in the first step of PageRank-Nibble, the algorithm has two input parameters α and ϵ, where α is the loop probability for the random walk and ϵ is the remaining error value. We have used a variant that needs only those two input parameters and a seed node provided by the user. The PageRank approximation assigns two values to all nodes, that have been discovered locally: The approximated PageRank-value and the remaining error probability. All PageRank-values and all remaining error probability can be added up to a value of exactly 1 at any state of execution. Upon initialization the remaining error probability of the seed node is set to be 1 and its PageRank-value is set to 0. A node queue is maintained throughout the process, which contains all nodes with an error probability greater than ϵ. Until this queue is empty, the algorithm performs push operations on a node from the queue, distributing the probability into the network. A push operation on a node transfers probability from the error probability to the node's PageRank-value, controlled by the parameter α. After finishing the queue, our PageRank-Nibble variant performs a so called *sweep* on the support set supp(pr), which contains all nodes with a PageRank-value greater than 0. The nodes within the support set are sorted by the fraction of their PageRank-values and their weighted degrees: $\mathrm{pr}(n)/\deg(n)$. From the sorted sequence of nodes $S = \{v_1, v_2, \ldots, v_{|\mathrm{supp(pr)}|}\}$ the subset $S_i = \{v_j \in S \mid j \leq i\}$ with the

smallest conductance is returned as the detected community. This differs slightly from the original approach, where the user may declare a desired maximum conductance and a minimum size of the community, which should be detected by PageRank-Nibble. The original algorithm returns an empty set, if no such community can be found.

4.2. Greedy Community Expansion

To provide evaluation results proving the flexibility of the ILSCD method, we have combined it with the GCE implementation by [16] as well. We have left the algorithm as described in Section 2.1. Starting from a set containing only the seed node $C = \{s\}$, the method iterates over the shell of C, which is the set of nodes, that are connected to a node within C. If a shell node is found that would improve the objective function of the community C if included, it is added to C and the shell is updated. However, we have adapted the objective function "M" for usage on weighted graphs:

$$M(C) = \frac{\sum_{(u,v)\in E, u\in C, v\in C} \text{weight}(u,v)}{\sum_{(u,v)\in E, u\in C, v\notin C} \text{weight}(u,v)}.$$

4.3. Algorithmic details

As previously stated, the ILSCD method is based on the iterative addition of edges to represent attribute similarity by structural cohesion. The algorithm can be used with any arbitrary local community detection algorithm. Whenever the local structural algorithm requests information about a node in the graph from the data source, the ILSCD method computes attribute similarities between the newly discovered node and all previously discovered nodes. If the similarity between two nodes exceeds the user given threshold τ, the method stores all required information about the new edge or the weight to be added to an existing edge. If the similarity is below the threshold, the edge is marked for removal. When the local structural algorithm finishes and returns a community, the graph is modified according to this information. A user may then change parameters such as the similarity function and run the algorithm again. The result is then affected by the modifications from the previous step and the changes to the parameters lead to yet another result in the next iteration.

Since the ILSCD method uses another arbitrary local algorithm, the easiest way of implementing it is to present the local algorithm not a graph but an interface to the graph, where the methods for accessing the nodes in any way are modified to call another method, that computes the node similarities to all nodes, that have been discovered so far, and decides about edges to be added or removed. Therefore, we add a new function *onNodeTouched* to the graph $G = (V, E, A, \text{weight})$, which is called, whenever the local structural algorithm calls any function on G to retrieve node information like degree or neighbours of a specific node. Additionally we introduce another weight function *attribweight* to G which represents the attribute weight of an edge, while the original weight function is renamed to *structureweight*, which only represents the structural weight. The weight function which is to be called by the local structural clustering algorithm

is *weight'*, which combines original structural weight and attribute weight of an edge. Furthermore the graph is extended by parameters τ and σ, of which the first is the above mentioned threshold to control whether an edge representing attribute similarity of two nodes is to be added or removed. The second parameter σ determines the mixing of structural and attribute similarity weight on every edge.

All in all the ILSCD algorithm takes the following parameters for initialization:

- An undirected attributed graph $G = (V, E, A, \text{weight})$.
- A local structural community detection algorithm $\text{scd}(G, u)$, which takes a graph G and a seed node u and returns a set of nodes $C \subset V$.
- A similarity function $\text{similarity}(a, b, w) : \mathbb{R}^d \times \mathbb{R}^d \times [0, 1]^d \to [0, 1]$, as defined in the previous chapter.
- An attribute weight vector $w \in [0, 1]^d$.
- The threshold for adding attribute similarity edges $\tau \in [0, 1]$.
- The mixing parameter for structural edge weight and attribute similarity edge weight $\sigma \in [0, 1]$.

Using the given parameters, the algorithm creates a modified graph G' as described:

```
1 Function initILSCD(G = (V, E, A, weight), scd, similarity, w, τ, σ) is
2 |   D ⟵ ∅;
3 |   E_add ⟵ ∅;
4 |   E_remove ⟵ ∅;
5 |   structureweight ⟵ weight;
6 |   attribweight ⟵ ∀(u, v) ∈ E : attribweight(u, v) = 0;
7 |   G' ⟵ (V, E, A, weight', onNodeTouched);
8 end
```

After initialization the user may perform as many iterative steps as necessary. Following each step he may change parameters (mostly the attribute weight vector) to discover alternative communities. Note that during initialization three sets D, E_{add} and E_{remove} have been declared of which D contains the nodes, that have been discovered so far. The set E_{add} contains tuples $(u, v, aw) \in V \times V \times [0, 1]$ of new edges to add after an iterative step. Likewise E_{remove} contains edges to remove after a step. Note that 'remove' means the edge is only entirely removed, if it is an edge existing solely from attribute similarity. Structural edges and weights of the graph are never altered (see the actual *run* function later). All variables and functions initialized remain the same, unless the user wishes to change them (using functions like *setAttributeWeights*, which are not further specified).

The function *onNodeTouched* is called by the extended graph whenever the local community detection algorithm scd requests information about a node (neighbours, degree, ...):

```
1  Function onNodeTouched(u) is
2      if u ∉ D then
3          for v ∈ D do
4              sim ⟵ similarity(A[u], A[v], w);
5              if sim ≥ τ then
6                  E_add ⟵ E_add ∪ {(u, v, sim)};
7              else
8                  if (u, v) ∈ E and attribweight(u, v) > 0 then
9                      E_remove ⟵ E_remove ∪ {(u, v)};
10                 end
11             end
12         end
13         D ⟵ D ∪ {u};
14     end
15 end
```

For a new, undiscovered node u, the function computes all attribute similarities between u and the already discovered nodes in D. Whenever the attribute similarity between u and another node $v \in D$ exceeds the threshold τ, a new tuple is added to the E_{add} set. If the similarity is below the threshold and the graph contains an edge between u and v, that has an attribute weight, this edge is marked for removal. Those two sets will be processed in the *run* function only after the local community detection algorithm has completed, since the graph modification should not interfere with it.

The replacement *weight'* method combines the structural and attribute weight using the mixing parameter σ:

```
1  Function weight'((u, v)) is
2      return σ · attribweight(u, v) + (1.0 − σ) · structureweight(u, v);
3  end
```

On unweighted graphs σ could be set to 0.5 to achieve an equal balance between attribute similarity and structure weights. However if the similarity values tend to be very small or the graph is weighted with edge weights exceeding one, it might be necessary to increase σ to get better results.

Using all the above definitions and functions, the *run* function can now be defined. The usage is just like one would expect from a local community detection algorithm: It takes

a seed node as the input parameter and returns a set of nodes, which is the discovered community:

```
1  Function run(u) is
2      C ⟵ scd(G′, u);
3      for (u, v, aw) ∈ E_add do
4          if (u, v) ∈ E then
5              attribweight(u, v) ⟵ aw;
6          else
7              E ⟵ E ∪ {(u, v)};
8              structureweight(u, v) ⟵ 0;
9              attribweight(u, v) ⟵ aw;
10         end
11     end
12     for (u, v) ∈ E_remove do
13         attribweight(u, v) ⟵ 0;
14         if weight'(u, v) == 0 then
15             E ⟵ E\{(u, v)};
16         end
17     end
18     E_add ⟵ ∅;
19     E_remove ⟵ ∅;
20     D ⟵ ∅;
21     return C;
22 end
```

The first line in the method causes the structural clustering algorithm to run on the modified graph with the given seed node. For the first execution of the run method the result is equivalent to what the scd algorithm would deliver on its own, after applying it on the unmodified graph. However, the ILSCD-modified graph has computed attribute similarities of the nodes that have been discovered by scd using the onNodeTouched method. The sets E_{add} and E_{remove} have now been filled with edges to add and edges to remove and after the local structural algorithm has finished, the run method modifies the graph according to the content of those sets. If an edge to add exists already as a structural edge, the algorithm only sets the attribweight of that edge appropriately. If no edge exists at all yet, it adds a pure attribute edge with a structureweight of zero in lines $7 - 9$. The edge removal loop simply sets the attribweight of the edge to zero and removes it, if it has no structural weight. Lastly the method empties the three sets, which have been filled by the onNodeTouched method.

4.4. Complexity

Since the ILSCD method depends on the underlying local structural clustering algorithm, the discussion of time and space complexity is difficult. In general only one assumption can be made: If the local algorithm's running time is not dependent on n or m, then the

running time of the ILSCD method in combination with that algorithm does not depend on them either, since the local algorithm can not consider the entire graph then and neither can ILSCD. We have evaluated the running time of the iterative approach proposed in this thesis for two steps of the algorithm (two calls of the run method described in the previous chapter).

If the number of nodes touched by the local algorithm is c, then the ILSCD method adds an overhead within $O(c^2 \cdot d)$ to the first iteration, assuming the similarity between two nodes can be computed in time $O(d)$ which is the case for the similarity functions used in this thesis. For the second step, the overhead is exactly the same, however, c could be different in some cases. If and how exactly c changes depends on the underlying local algorithm.

To provide an example, we will discuss the complexity of ILSCD in combination with the PageRank-Nibble algorithm [1], which was the main method we have used for evaluation. As stated at the beginning of this chapter, PageRank-Nibble consists of two steps: The approximation of a local PageRank value pr and the computation of a minimum conductance set within the support set of the PageRank approximation. The running time of the PageRank-Approximation $\mathit{ApproximatePageRank}(u, \alpha, \epsilon)$, where u is the seed node, α is the loop probability of the random walk and ϵ is the remaining error value, is within $O\left(\frac{1}{\alpha \cdot \epsilon}\right)$. The volume of the support set, on which the PageRank value is approximated, can be estimated by $\text{vol}(\text{supp}(\text{pr})) \in O\left(\frac{1}{\alpha \cdot \epsilon}\right)$ as well. With no further information the number of nodes, which have been touched by the approximation algorithm, could still at most be the size of the volume. Therefore, the overhead of the ILSCD method using PageRank-Nibble is within $O\left(\left(\frac{1}{\alpha \cdot \epsilon}\right)^2 \cdot d\right)$ for the first iteration step. If the user does not change any of the parameters α and ϵ (which is the case for the evaluation results in this thesis), the overhead does not change for the second step as well.

For the total running time of two steps of ILSCD in combination with PageRank-Nibble, the second step of PageRank-Nibble, which does not add to the amount of nodes touched, must be considered as well. It consists only of the previously explained sweep on the support set $\text{supp}(\text{pr})$. Such a sweep can be performed in time within $O\left(|\text{supp}(pr)| \cdot \log(|\text{supp}(\text{pr})|)\right) = O\left(\frac{1}{\alpha \cdot \epsilon} \cdot \log\left(\frac{1}{\alpha \cdot \epsilon}\right)\right)$.

Combining the above results, the total running time for two iterative steps of the ILSCD method using PageRank-Nibble with parameters α and ϵ is within $O\left(\left(\frac{1}{\alpha \cdot \epsilon}\right)^2 \cdot d\right)$, since $O\left(\frac{1}{\alpha \cdot \epsilon} \cdot \log\left(\frac{1}{\alpha \cdot \epsilon}\right)\right) \subset O\left(\left(\frac{1}{\alpha \cdot \epsilon}\right)^2\right)$.

In terms of space complexity, the ILSCD method adds the additional attribweight function, which means for every edge, that is added by the algorithm, the computed similarity must be stored. If the local algorithm touches c nodes, the overhead in storage space is within $O(c^2)$, where the maximum c^2 is reached, if the threshold τ is smaller than all computed attribute similarities between the c nodes. Considering the combination of ILSCD and PageRank-Nibble again, the overhead in storage space is within $O\left(\left(\frac{1}{\alpha \cdot \epsilon}\right)^2\right)$.

5. Evaluation

The ILSCD method has been evaluated on synthetic data as well as on real world graph data. For testing on synthetic data, we have combined an overlapping variant of the LFR generator [6] with a method to generate attribute vectors to correspond with the community structure. Real graph data has been extracted from the Facebook friendship network and the Amazon co-purchase network.

5.1. Synthetic data

To generate random graphs with a power law degree sequence, we have used the the LFR benchmark [6], which uses the configuration model [8]. The configuration model can be used to generate random graphs with a given degree sequence. Essentially, all nodes are inserted into a set as many times as defined by their degree. On the resulting set a random matching is generated by drawing pairs of nodes from it using a uniform distribution. Given the number of nodes n of the desired graph, the LFR generator first draws n node degrees from a power law distribution, which is defined by an exponent τ_1 and the limits *mindeg* and *maxdeg*. Node degrees are then split into internal and external degrees with respect to a parameter μ such that the internal degree, which is the number of edges to nodes within a common community, of a node v is $(1.0-\mu)\cdot\deg(v)$. From a second power law distribution the generator draws community sizes using a second exponent τ_2 and the limits *minsize* and *maxsize*. Community sizes are drawn until all desired memberships of the nodes can be satisfied. For the overlapping variant, the nodes are allowed to have multiple community memberships. The input parameter *on* controls the amount of overlapping nodes, while *om* specifies the number of communities an overlapping node belongs to. Altogether, the community sizes must add up to $n + on \cdot (om - 1)$ in order to fulfil all desired memberships. After the node degrees and community sizes have been determined, the LFR generator assigns nodes to communities using a variant of the configuration model for bipartite graphs. The assignment is generated as a bipartite graph, where one partition consists of the nodes and the other partition consists of the communities. When all nodes have been assigned to communities, the configuration model is used twice more: To generate the community subgraphs using the internal degree sequence and to generate the rest of the graph using the external degree sequence. Since the internal degrees are altered during the generation of the external graph structure, a rewiring is performed as the last step of the algorithm to reassess the internal node degrees. Note that our implementation of the LFR generator uses the minimum and maximum degree as well as the minimum size and maximum size of communities as its input parameters, whereas the limits of the degree distribution are determined using a given maximum degree and average degree

and the community distribution limits are inferred from the degree limits in the original implementation.

The LFR benchmark produces a random, unweighted and undirected graph with a power law degree sequence. The sharpness of the generated graph's community structure is affected by the node degree mixing parameter μ. Theoretical analysis of the edge distribution [7] shows, that communities are well defined for $\mu < \frac{n-|C_{\max}|}{n}$, where $C_{\max}$ is the largest community in the graph.

To test the ILSCD method, we have combined the graphs generated by the LFR benchmark with a set of generated attribute vectors. Those attribute vectors can be generated using the memberships provided by the LFR benchmark. For a given number of communities c the dimension of attribute space is chosen as $d = c$. The communities are identified each by an integer value within the interval $[0, c)$. Let $M[u]$ be the set of communities a node u is a member of, then the synthetic attribute vector to be assigned to u is defined as $A[u] = \sum_{m \in M[u]} \mathbf{e_m} + \mathbf{s_m}$, where $\mathbf{e_m}$ is the unit vector according to the integer identifier m of the community and $\mathbf{s_m}$ is a random scatter vector with $\mathbf{s_c} \cdot \mathbf{e_c} = 0$ and $\mathbf{s_c} \leq \mathit{scatter}$. The maximum length parameter *scatter* of the scatter vectors is given by the user. This method can be visualized for a three-dimensional attribute space (which means three existing communities) with the 3D-figure 5.1. For three communities and no overlapping nodes, the attribute vectors are generated within cones around the three unit vectors in three-dimensional space. An overlapping node u, which is a member of the green and red communities (x and y-axis in $\mathbb{R}^3$), would receive the sum of the two unit vectors and two random scatter vectors as its attribute vector: $A[u] = \mathbf{e_x} + \mathbf{e_y} + \mathbf{s_x} + \mathbf{s_y}$.

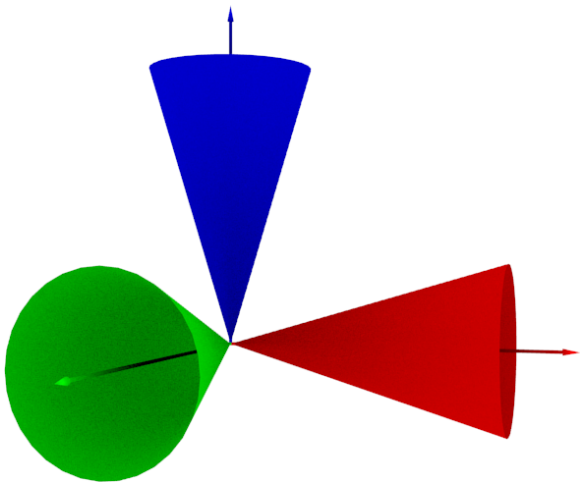

Figure 5.1.: Visualization of synthetic attribute vectors in 3D attribute space

The approach is intuitively compatible with the use of cosine-similarity, however, experiments with the ILSCD method and Jaccard-similarity have resulted in similar quality improvements.

We have measured community detection quality using the JQ quality function, defined in Chapter 4. For every experimental result we have generated random graphs using the LFR generator and selected a random seed node for each graph. Those random nodes remained

fixed throughout the experiment (i.e. for all different algorithms). Using multiple random seed nodes for every graph to get more samples and eliminate random quality spikes was impossible due to the long duration of the execution. However, we will provide some diagrams such as Figure 5.3, where the quality of most plotted algorithms is independent of the selected type of x-values. In compliance with the original LFR implementation, we have set $\tau_1 = -2.0$ and $\tau_2 = -1.0$ for graph generation. Since the attribute space dimension is dependent on the number of communities, we have also selected the community size limits dependent on the graph size as minsize = $\frac{n}{30}$ and maxsize = $\frac{n}{10}$, in order to achieve a constant amount of communities for every generated graph. This causes the evaluation of running times to be unaffected by the similarity computation.

Whenever we have used PageRank-Nibble on its own or in combination with ILSCD (which will be denoted as ILSCD-PRN), its parameters have been chosen as $\alpha = 0.5$ and $\epsilon = 0.001$. In that case the choice of parameters is not conform with the recommendations in the original publication, but prevents PageRank-Nibble from working on the entire graph and thus acting as a global algorithm especially in terms of running time (at the expense of quality). Although the speed of many local algorithms like Greedy Community Expansion (GCE) depends on the size of the communities (which means a dependence on n in our experiments), PageRank-Nibble only depends on its parameters α and ϵ, which makes the ILSCD-PRN method truly independent of the total number of nodes.

To compare the ILSCD method with another algorithm that combines graph structure and attributes, we have gathered evaluation data from the CODICIL method as well. Since the CODICIL method is a global algorithm, it usually delivers communities with better quality than the ILSCD method, yet it is much slower and in some cases, which will be presented later, the local algorithm is even able to find better quality communities. The CODICIL algorithm has been combined with a parallel implementation of the Louvain method [**Louvain**] from [17]. This global partitioning algorithm, which is named *PLM*, has been given the graphs modified by CODICIL and we have compared the community in the resulting clustering, that contained the selected random node, with the results of the local algorithms. To achieve equal balancing between structural and attribute edges, the mixing parameter α of CODICIL has been set to 0.5. As described in Chapter 2, during the sampling step of CODICIL, only the k most relevant neighbours of every node are retained. We have set the parameter k to the average degree of the original graph, which is recommended in the original publication: $k = \lfloor \frac{2 \cdot m}{n} \rfloor$.

The local Greedy Community Expansion (GCE) algorithm has been used for speed comparison and some results of the ILSCD method in comparison to GCE and in combination with it will be available in the appendix. Whenever GCE has been used, the objective function has been the function "M" from [16] extended with edge weight as defined in Chapter 4:

$$M(C) = \frac{\sum_{(u,v)\in E, u\in C, v\in C} \text{weight}(u,v)}{\sum_{(u,v)\in E, u\in C, v\notin C} \text{weight}(u,v)}$$

Since the ILSCD method is an iterative algorithm, it has always been applied twice and we have compared the results after the second step. Likewise for speed comparison, two steps of the method have been taken into account. To achieve an equal mixing between

attribute similarity and the unweighted graph structure, we have set the mixing parameter of ILSCD to $\sigma = 0.5$. The threshold for addition of edges was set to $\tau = 0.5$, preventing most of the irrelevant edge additions, and the weight vector was simply set to a vector of dimension d where every entry of the vector is 1, to take all attributes into account: $\forall i = 0, 1, ..., d : w[i] = 1.0$.

All algorithms have been implemented using C++11 and Python within the open-source network analysis tool kit *NetworKit* [18] and have been executed on a server with a 16-core AMD Opteron CPU at 2.6GHz per core and 126GB of RAM.

The first experiment was required to determine an appropriate value for the degree mixing parameter μ of the LFR algorithm. On graphs with a constant amount of nodes n and with constant node degree limits $mindeg = 40$ and $maxdeg = 100$, we have chosen to let μ grow from 0.2 to 0.9:

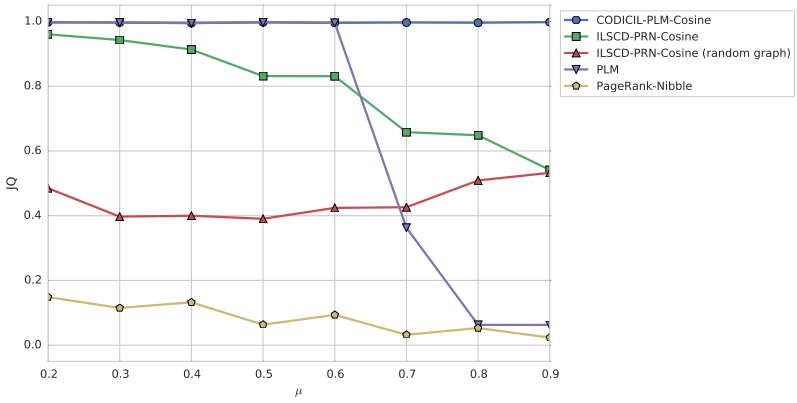

Figure 5.2.: Community detection quality on random graphs with community structure, without overlap, constant node degree limits and growing degree mixing parameter using cosine-similarity.

Due to this result, we have set $\mu = 0.65$ in the following experiments, since that seems to be the threshold for PLM to detect communities adequately. According to the theoretical analysis of the μ parameter in [7], communities are well defined as long as $\mu < 0.9$. The above plot shows the effectiveness of attribute based methods on graphs, where communities have only a few more density than arbitrary subsets.

In the next experiment, the algorithms have been used with cosine-similarity on random generated graphs with sizes from 1.000 to 5.000 nodes and constant node degree limits $mindeg = 40$ and $maxdeg = 100$:

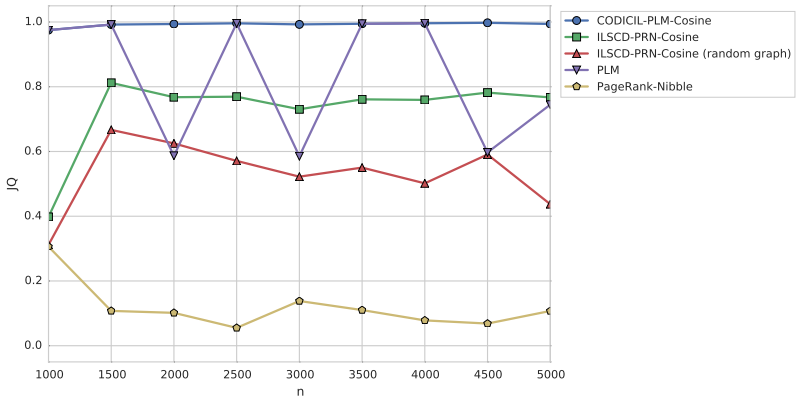

Figure 5.3.: Community detection quality on random graphs with community structure, without overlap and constant node degree limits using cosine-similarity.

As can be seen, the ILSCD method is able to detect the ground truth communities better than PageRank-Nibble alone by using attribute and structure information. Also from the result of applying ILSCD on a random graph without community structure but with the same attribute vectors, the importance of both, attribute similarity and structural data, is visible. However, the global CODICIL algorithm is able to detect the ground truth communities of the random seed nodes without any error at all. A very similar result has been gathered from the same setup using the Jaccard-similarity. The related Figure A.1 can be found in the appendix.

The running times of the various algorithms for the computation of the previous result has been plotted in the following figure:

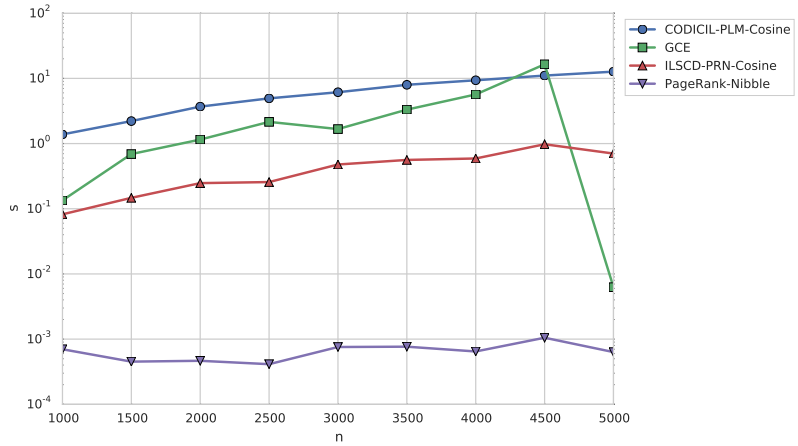

Figure 5.4.: Running times on random graphs with community structure, without overlap and constant node degree limits using cosine-similarity.

As previously mentioned, PageRank-Nibble and also the ILSCD method are independent of the number of nodes, whereas the global CODICIL method needs to compute similarities between every pair of nodes in the entire graph, resulting in a quadratic dependence on

the amount of nodes. The running time of GCE depends on the size of the local community, which makes it dependent on the total graph size as well, since we have chosen to let community sizes grow with the graph size.

For the next experiment, the node degree limits have been chosen to grow with the number of nodes in the graph. This is often the case in real world data as well. For example data from the Facebook friendship network, which will be presented in the next section, is available from 2005 and from 2015. The extracts from 2015 are not only bigger in terms of the number of nodes, but also in terms of node degrees. More persons joining the Facebook network (a growth in nodes) have caused more friendships to be established (growth in node degrees), since members have more of their friends available in the network. Node degree limits have been chosen dependent on the graph size as $mindeg = n/30$ and $maxdeg = n/10$. As in the previous experiments, the generated graphs did not contain overlapping community structure and the graph sizes have been set to a range from 1.000 to 5.000:

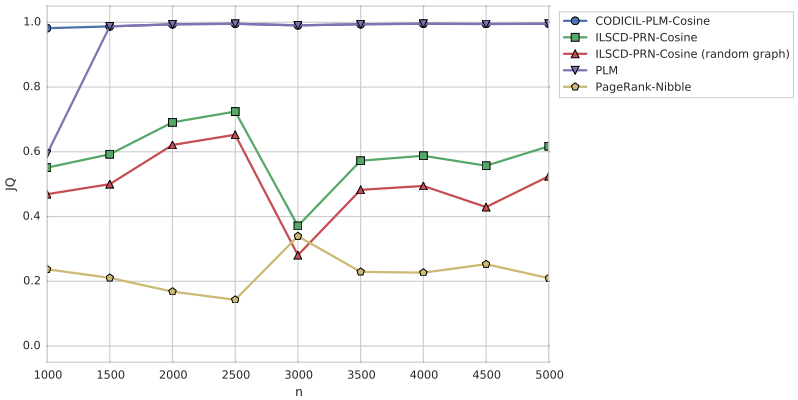

Figure 5.5.: Community detection quality on random graphs with community structure, without overlap and growing node degree limits using cosine-similarity.

The quality of detected communities is similar to the previous experiment. However, there has been a change in running times:

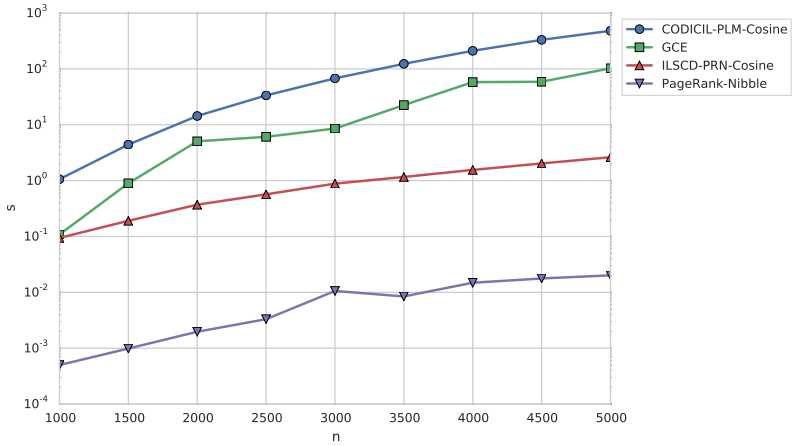

Figure 5.6.: Running times on random graphs with community structure, without overlap and growing node degree limits using cosine-similarity.

Since the node degree or volume can not be determined in constant time within the NetworKit implementation, PageRank-Nibble and the ILSCD method are dependent on the node degrees in this experiment as well. Yet the ILSCD method is still able to outperform CODICIL and GCE by far and the dependence is only linear in node degree.

The quality of communities detected by ILSCD becomes even better than the quality of communities detected by CODICIL-PLM, when it comes to overlapping community structures. In the next experiment, all generated graphs shared the same amount of nodes, which is 5.000, and the same node degree limits, which have been set to $mindeg = 40$ and $maxdeg = 100$. We have also set the degree mixing parameter μ to 0.4 for this experiment, to ensure the communities would have been detected by the algorithms without attribute information, if there had been no overlapping structures. To be able to satisfy a lot of memberships, the community size limits have also been set to $minsize = 40$ and $maxsize = 100$. The LFR benchmark has been set to produce overlapping structures, where the number of overlapping nodes grows for each graph by 1.000 and the amount of memberships per overlapping node grows by one ($om = n/1000$):

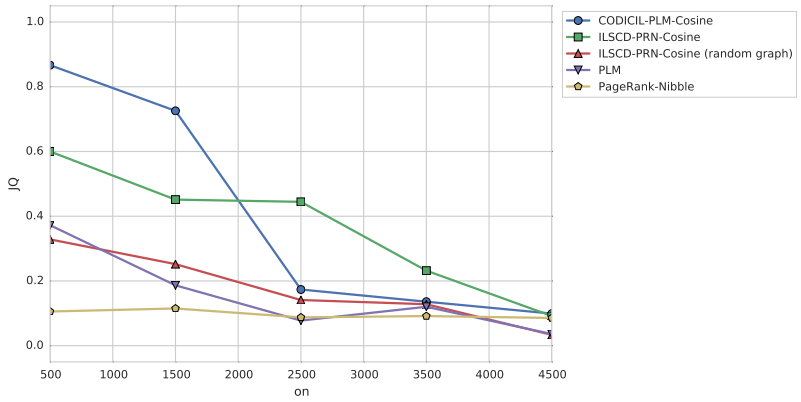

Figure 5.7.: Community detection quality on random graphs of 5.000 nodes with community structure, with growing number of overlapping nodes and constant node degree limits using cosine-similarity.

We have chosen the weight vector for the ILSCD method and the random nodes as follows to get the above result: Random nodes have always been drawn from the subset of overlapping nodes only. A random community has then been selected from the communities the random node belonged to. The weight vector has been set to zero at those indices that corresponded to the other two communities, causing the ILSCD method would focus on a single one of the three available communities. This shows the effectiveness of appropriate attribute weighting and the use of ILSCD on graphs with overlapping community structure.

The last experiment has been set up to verify the scalability of the ILSCD method combined with PageRank-Nibble, using bigger graphs, on which the execution of CODICIL takes too long to complete. Therefore, random graphs with a constant node degree between 100 and 500 and 10.000 to 50.000 nodes have been generated. The running times of ILSCD-PRN-Cosine and PageRank-Nibble are shown in the following figure:

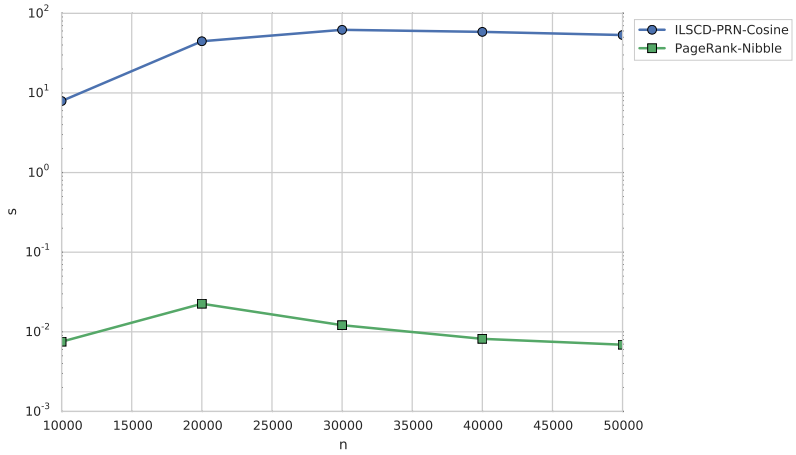

Figure 5.8.: Running times on random graphs of 10.000 to 50.000 nodes with community structure, without overlap and constant node degree limits using cosine-similarity.

While the PageRank-Nibble algorithm performs much faster, the running time of the ILSCD method is still independent of the graph size. The quality improvement for the above experiment is similar to the previous results and can be found in the appendix of this thesis. Furthermore, the appendix contains evaluation results of the ILSCD method combined with PageRank-Nibble and the Jaccard-similarity and also results of the combination with GCE instead of PageRank-Nibble.

5.2. Facebook friendship graph data

For evaluation on real networks, data has been collected from the Facebook friendship network. Nodes represent persons and edges reflect the friendship relations between them. We will present results for a dataset from 2005, used in [20, 19], as well as for current data from 2015, that has been extracted from the network. The dataset from 2005 contains the complete Facebook subsets of 100 different colleges and universities. For each node (person) in a graph, the following attributes are available: Faculty status flag, gender, major, second major/minor, dormitory, year and high school. All data has been anonymized and the names of the institutions have been obfuscated. As an example we present the graph of the “Caltech36” institution with 769 nodes and 16.656 edges:

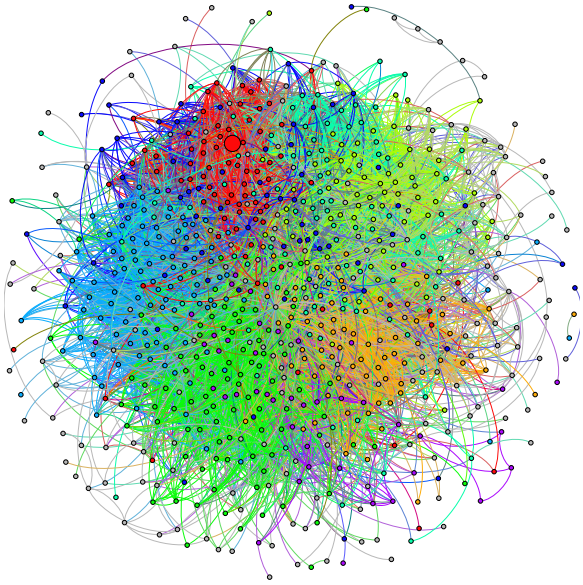

Figure 5.9.: Facebook friendship graph of the “Caltech36” institution, coloured by student’s dormitory.

In the above figure the friendship graph has been drawn using the open-source tool *Gephi* [2] together with the *Fruchterman-Reingold* force-based layout [5]. Colouring the nodes by the student’s dormitories, reveals a certain relation between community structure in the network and the dormitories. However, using PageRank-Nibble alone on a seed node, which is emphasized by enlargement in the visualizations, has not led to detection of the visually discovered communities:

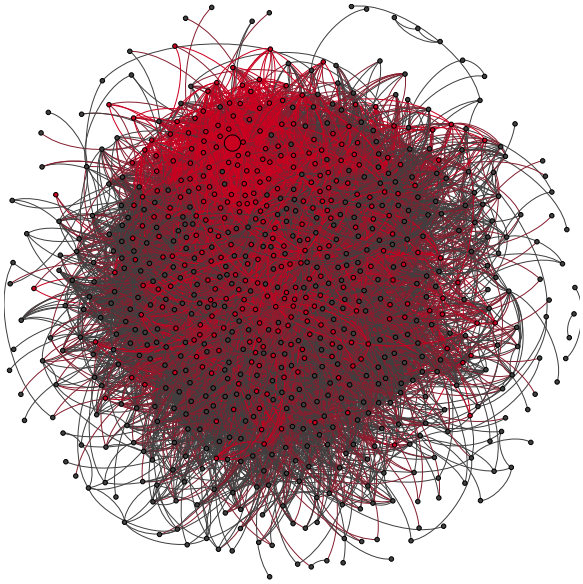

Figure 5.10.: Facebook friendship graph of the “Caltech36” institution with the community detected by PageRank-Nibble marked red.

Setting the attribute weight vector of the ILSCD method to 0, except for the dormitory attribute, has resulted in the detection of a much smaller community, which reflects the group of persons around the seed node that resides in the same dormitory:

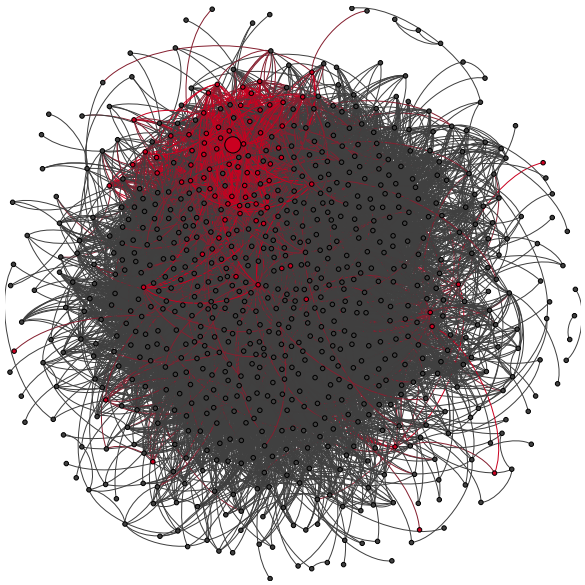

Figure 5.11.: Facebook friendship graph of the "Caltech36" institution with the community detected by ILSCD-PRN-Counting (weighted for dormitory attribute) marked red.

The "Caltech36" example evaluates the use of the ILSCD method to find communities more precisely. On Facebook data extracted from February 2015 we will demonstrate the use of ILSCD to find alternative communities in an overlapping graph structure will be demonstrated. As the network has grown immensely since 2005, a breadth-first search from a seed person node with depth two already leads to a graph with more than 20.000 nodes. Therefore, to provide graphs that can be drawn in reasonable time only the direct neighbourhoods of selected persons have been collected. The following attributes have been gathered from the public person profiles as far as they were available: Gender, age, home town, current place of residence and the current and first educational institution. Coverage ranges from about 20% (age) to about 80% (gender). Persons with no available data in an attribute have been assigned the value 0 and have been coloured grey. The following figure shows the neighbourhood of a selected person (who will be named Alice for the sake of anonymity), consisting of 323 nodes and 6.481 edges, aligned using the Fruchterman-Reingold layout and coloured by the first educational institute of persons (if specified):

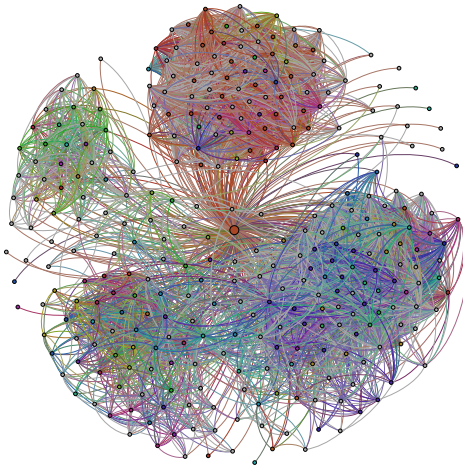

Figure 5.12.: Facebook friendship neighbourhood graph of a single person, coloured by first educational institute of the persons.

Using PageRank-Nibble on the seed node Alice has led to the detection of the lower three visible communities:

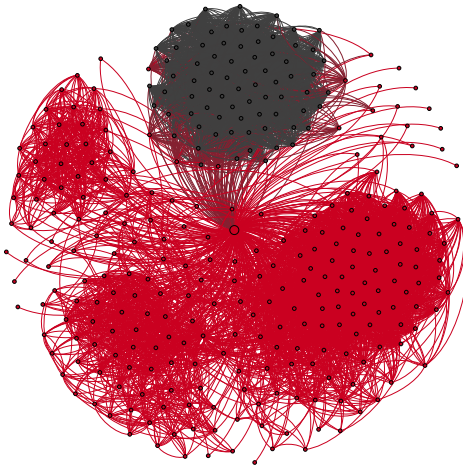

Figure 5.13.: Facebook friendship neighbourhood graph of a single person with the community detected by PageRank-Nibble marked red.

The very dense, upper community has not been detected, although the previous graph, coloured by first place of education, suggests a strong connection between Alice and the nodes of the upper community and also in between nodes of that community in terms of this attribute. By weighting the attribute similarity solely on the first educational institute attribute, the ILSCD method has been able to detect the following community around Alice:

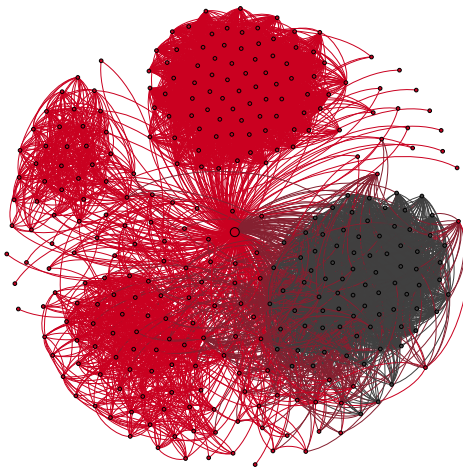

Figure 5.14.: Facebook friendship neighbourhood graph of a single person with the community detected by ILSCD-PRN-Counting (weighted for first educational institute attribute) marked red.

A third alternative community has been detected by setting a weight vector, that takes the home town and the first place of education into account. In that case the ILSCD method excludes the upper community and lower right community, because their internal attribute similarity is stronger than the similarity to Alice, who has not specified a home town:

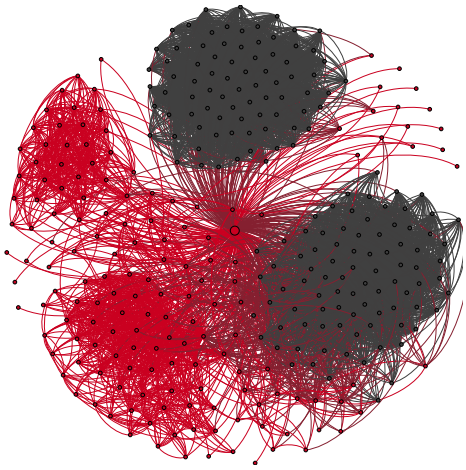

Figure 5.15.: Facebook friendship neighbourhood graph of a single person with the community detected by ILSCD-PRN-Counting (weighted for first educational institute attribute and home town attribute) marked red.

5.3. Amazon co-purchase graph data

As a second example of applying the ILSCD method on real graph data, we will present an extract from the Amazon co-purchase network. The nodes of this network represent Amazon products while an edge between two products implies, that the two products

have been purchased together. For each node attributes such as the rating, the Amazon price and the number of reviews are available. Contained in this *Disney* extract from [10] are only Disney movie DVDs, a small subset of 124 nodes and 335 edges. In the following figure, nodes have been coloured according to their average rating:

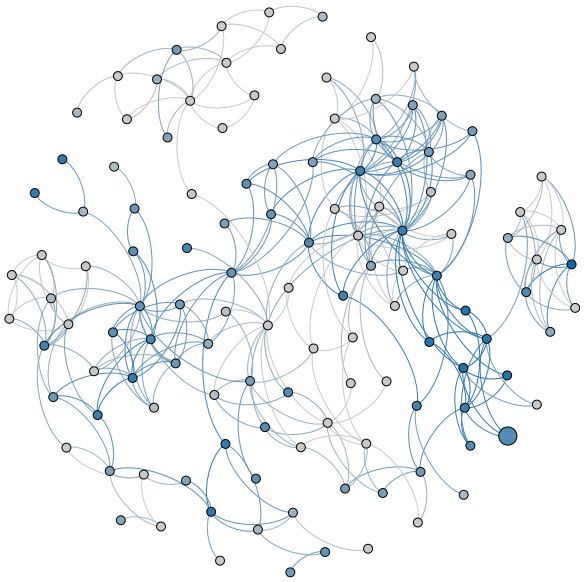

Figure 5.16.: Amazon co-purchase subset of Disney movies, coloured by average rating.

We have selected one of the products as the seed node, which is enlarged in the graph visualization, for the following evaluation. It is the movie “A Bug’s Life” on DVD as a collector’s edition. Together with the surrounding nodes, the highlighted node seems to form a structural cluster, consisting of Disney-Pixar movies such as “Toy Story”, “Toy Story 2” and “Monsters, Inc.”. Using PageRank-Nibble alone on the given seed node has led to the detection of a large community, which has no interpretation:

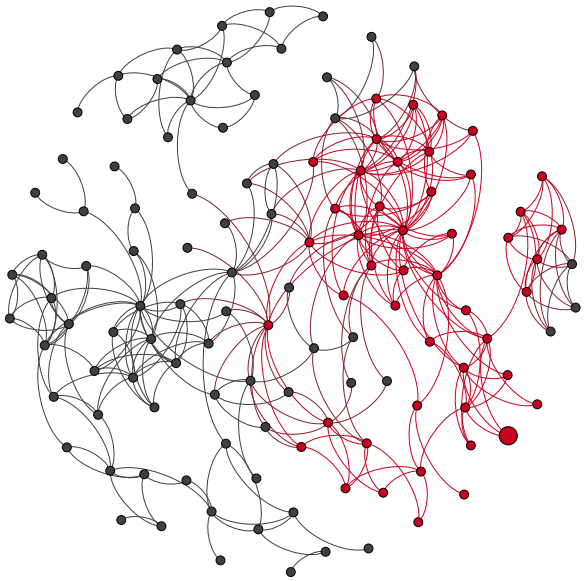

Figure 5.17.: Amazon co-purchase subset of Disney movies with the community around a given seed node detected by PageRank-Nibble marked red.

From the figure coloured by average rating we can deduce an elevated attribute similarity within the Disney-Pixar cluster as well. Using the ILSCD-PRN algorithm without any specific weighting on certain attributes and the Jaccard-similarity has resulted in the following community:

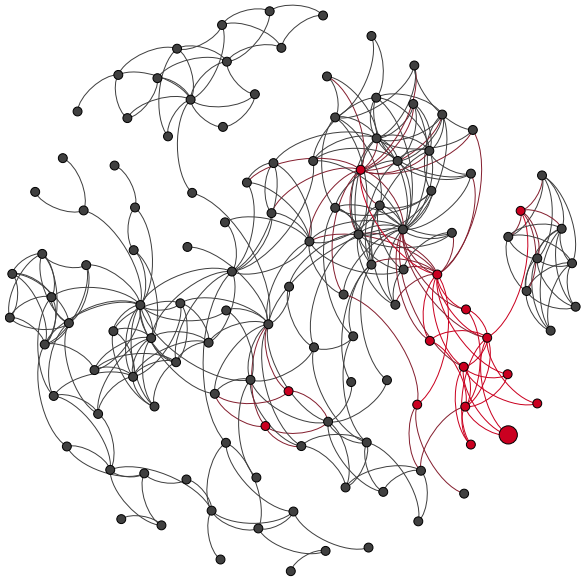

Figure 5.18.: Amazon co-purchase subset of Disney movies with the community around a given seed node detected by ILSCD-PRN-Jaccard marked red.

Except for a few additional nodes, the ILSCD method has been able to find the Disney-Pixar community almost perfectly.

6. Conclusion

Iterative Local Selective Community Detection is probably the first approach to local graph clustering combining structure and attribute data. While the evaluation proves, that it can lead to improved detection of communities and allows to handle overlapping structure, the algorithm has potential for various extending experiments and improvements. In this chapter, we will discuss the results of the previous one and outline further ideas for developing the method.

Experiments on synthetic data – generated with the LFR generator – have shown, that the combination of structural and attribute information leads to significant quality improvements, if structural communities and attribute clusters are correlated. Overlapping communities can be detected by weighting the similarity functions accordingly, if their members have a higher similarity in different attribute subspaces. The iterative approach allows to compute various alternatives for a single given node easily. Since an arbitrary similarity function as well as an arbitrary clustering algorithm may be used with ILSCD, the method is very flexible and can easily be adapted to the user's application. Using it on large graphs proves the theoretical result from Section 4.4: The running time of ILSCD depends on the underlying structural algorithm but is in general independent of the entire graph size, if a true local algorithm like PageRank-Nibble is used. However, if the structural algorithm touches all nodes of a given input graph, the running time method is dependent on the number of nodes.

On the provided real world datasets from the Facebook friendship network and the Amazon co-purchase network the method has successfully been used to detect structural communities, visualized by using the Fruchterman-Reingold layout, more precisely and find alternative solutions for a single given seed node even on graphs with a low attribute coverage, such as the Facebook graphs from 2015.

Yet the experiments with real world data have also pointed out flaws and parts of the algorithm, which could use further improvement and development. The ILSCD method as described in this thesis takes two parameters to control the addition of edges during the clustering process. σ determines the mixing between structural edge weight and attribute similarity and τ allows to provide a threshold on similarity for the addition of attribute weights or edges. Upon using the method on real world data, those parameters have proven to be critical to the successful determination of alternative or better quality communities. It has been difficult to chose them appropriately for the presented results and it is certainly even more difficult with less information about the data. Furthermore, all graphs used for evaluation have been unweighted, simplifying the choice of σ. A great improvement would be, to have the algorithm determine the parameters automatically:

The parameter σ should be selected dependent on the maximum structural edge weight and the maximum attribute similarity within the discovered subgraph. Let C be the set of nodes discovered in a step of ILSCD. A possible solution could be to chose σ as the solution

of the following equation, which would lead to an equal weighting of the maximum values of strucural edge weight and attribute similarity:

$$\begin{aligned} \sigma \cdot \max_{u,v \in C} (\text{attribweight}(u,v)) &= (1.0 - \sigma) \cdot \max_{u,v \in C} (\text{structureweight}(u,v)) \\ \Leftrightarrow \quad \sigma &= \frac{\max_{u,v \in C} (\text{structureweight}(u,v))}{\max_{u,v \in C} (\text{attribweight}(u,v)) + \max_{u,v \in C} (\text{structureweight}(u,v))} \end{aligned}$$

The same could be done with the averages and not the maxima of weights. Another possibility is not to use a single mixing parameter but rather to different weightings for attribute similarity and edge weight so as to scale both ranges of values into the $[0, 1]$ interval. Determining the parameter fully by the algorithm might be also disadvantageous, since the attribute coverage might be low, or the graph might have weak structural community structure only, which might make the user want the algorithm to use more weight on one of the two information sources.

While we will not propose a method for the automated selection of the parameter τ, it might be more convenient not to specify a similarity threshold but rather a limit on the edges to be added either with a total number or a relative percentage of the existing amount of edges. A user does not have to know about the range and distribution of similarity between nodes of the graph then, but can rather input for example a relative value of 1.0 to declare that only as many attribute similarity edges as there are structural edges should be added to the graph.

For an existing threshold τ – whether it is user given or automatically inferred – better results might be delivered, if the remaining similarity range $[\tau, 1]$ is transformed into the original range $[0, 1]$ again:

$$\text{scaled\_attribweight}(u,v) = \frac{\text{similarity}(u,v) - \tau}{1.0 - \tau}$$

It is unknown if this would lead to different/better/worse community detection results. On a network where the similarity values are very high in general, this might allow the algorithm to distinguish them better. For other value distributions, it might as well lead to worse clustering quality, if the threshold is set very high and nodes with a high similarity suddenly have a very low one after the transformation. Further evaluation of this variant would be necessary.

A user has a third very important possibility to control the ILSCD method's clustering process, which is the attribute weighting vector. While it would not be useful to determine the attribute weight vector automatically, since it would keep the user from using it to discover alternative communities, the algorithm could suggest congruent subspaces (subspaces of the attribute space that correspond with the graph structure) for the user to select in the next iterative step of ILSCD. Methods for detection of congruent subspaces like [13] might be adaptable to a local variant.

Bibliography

[1] Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. "Local Graph Partitioning using PageRank Vectors". In: *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*. 2006, pp. 475–486.

[2] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. "Gephi: An Open Source Software for Exploring and Manipulating Networks". In: *Proceedings of the Third International Conference on Weblogs and Social Media, ICWSM 2009, San Jose, California, USA, May 17-20, 2009*. 2009.

[3] Facebook. *Company Info | Facebook Newsroom*. Dec. 2014. URL: `http://newsroom.fb.com/company-info/`.

[4] Santo Fortunato. "Community detection in graphs". In: *CoRR* abs/0906.0612 (2009).

[5] Thomas M. J. Fruchterman and Edward M. Reingold. "Graph Drawing by Force-directed Placement". In: *Softw., Pract. Exper.* 21.11 (1991), pp. 1129–1164.

[6] Andrea Lancichinetti and Santo Fortunato. "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities". In: *Phys. Rev. E* 80 (2009), p. 016118.

[7] Andrea Lancichinetti and Santo Fortunato. "Community detection algorithms: A comparative analysis". In: *Phys. Rev. E* 80 (2009).

[8] Michael Molloy and Bruce A. Reed. "A Critical Point for Random Graphs with a Given Degree Sequence". In: *Random Struct. Algorithms* 6.2/3 (1995), pp. 161–180.

[9] Flavia Moser et al. "Mining Cohesive Patterns from Graphs with Feature Vectors". In: *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*. 2009, pp. 593–604.

[10] Emmanuel Müller et al. "Ranking outlier nodes in subspaces of attributed graphs". In: *Workshops Proceedings of the 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*. 2013, pp. 216–222.

[11] Lawrence Page et al. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Nov. 1999.

[12] Yiye Ruan, David Fuhry, and Srinivasan Parthasarathy. "Efficient community detection in large networks using content and links". In: *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*. 2013, pp. 1089–1098.

[13] Patricia Iglesias Sanchez et al. “Statistical Selection of Congruent Subspaces for Mining Attributed Graphs”. In: *2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013*. 2013, pp. 647–656.

[14] Patricia Iglesias Sánchez et al. “Efficient Algorithms for a Robust Modularity-Driven Clustering of Attributed Graphs”. In: *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM. May 2015.

[15] Daniel A. Spielman and Shang-Hua Teng. “Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems”. In: *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*. 2004, pp. 81–90.

[16] Christian Staudt, Yassine Marrakchi, and Henning Meyerhenke. “Detecting communities around seed nodes in complex networks”. In: *2014 IEEE International Conference on Big Data, Big Data 2014, Washington, DC, USA, October 27-30, 2014*. 2014, pp. 62–69.

[17] Christian Staudt and Henning Meyerhenke. “Engineering High-Performance Community Detection Heuristics for Massive Graphs”. In: *42nd International Conference on Parallel Processing, ICPP 2013, Lyon, France, October 1-4, 2013*. 2013, pp. 180–189.

[18] Christian Staudt, Aleksejs Sazonovs, and Henning Meyerhenke. “NetworKit: An Interactive Tool Suite for High-Performance Network Analysis”. In: *CoRR* abs/1403.3005 (2014).

[19] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. “Social Structure of Facebook Networks”. In: *CoRR* abs/1102.2166 (2011).

[20] Amanda L. Traud et al. “Comparing Community Structure to Characteristics in Online Collegiate Social Networks”. In: *SIAM Review* 53.3 (2011), pp. 526–543.

[21] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. “Graph Clustering Based on Structural/Attribute Similarities”. In: *PVLDB* 2.1 (2009), pp. 718–729.

A. Appendix

A.1. Further evaluation results on synthetic data with Jaccard-similarity

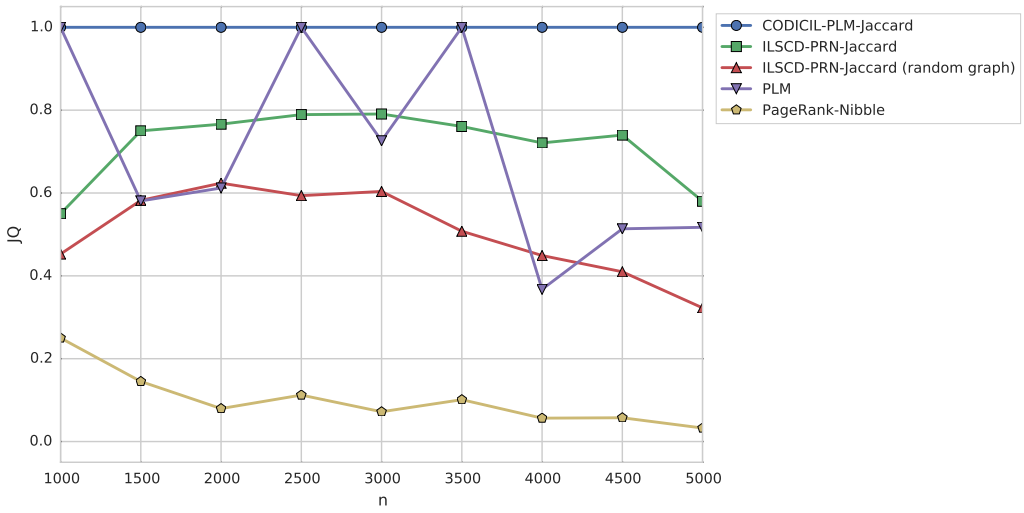

Figure A.1.: Community detection quality on random graphs with community structure, without overlap and constant node degree limits using Jaccard-similarity.

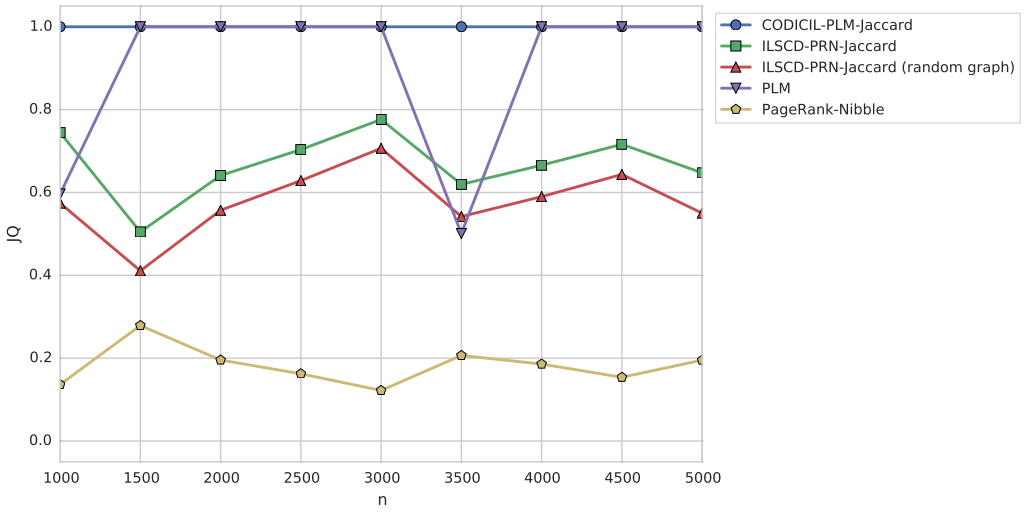

Figure A.2.: Community detection quality on random graphs with community structure, without overlap and growing node degree limits using Jaccard-similarity.

A.2. Further evaluation results on synthetic data with large graphs

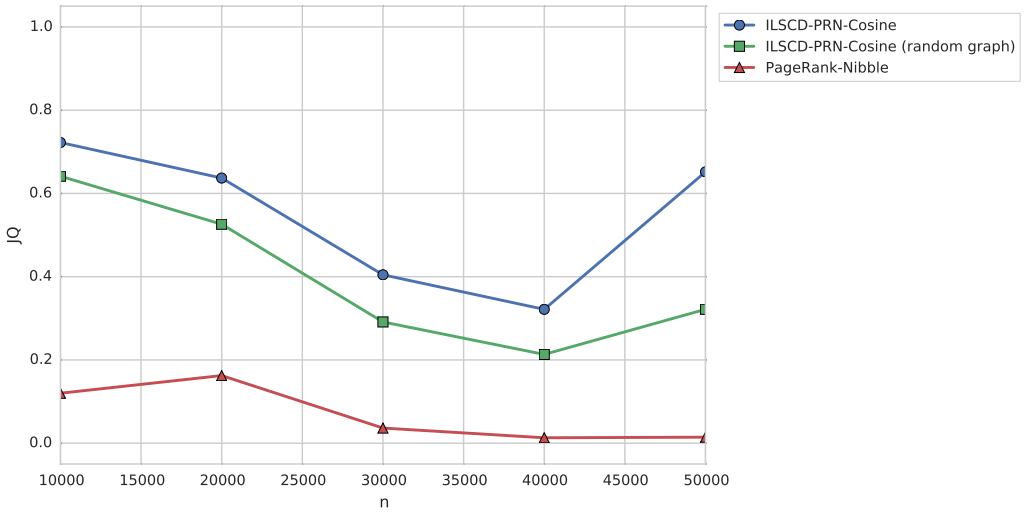

Figure A.3.: Community detection quality on large random graphs with community structure, without overlap and constant node degree limits using cosine-similarity.

A.3. Further evaluation results on synthetic data with Greedy Community Expansion (GCE)

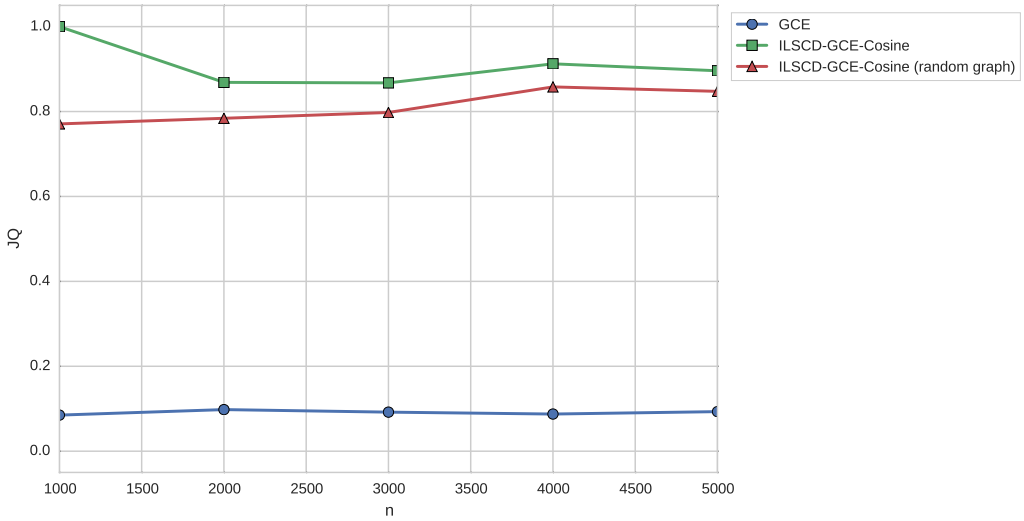

Figure A.4.: Community detection quality on random graphs without overlap and constant node degree limits using cosine-similarity and GCE.

Note that for the generation of the above figure, the parameter μ, controlling the internal degrees generated by LFR, and the randomness of generated attribute vectors have been raised to $\mu = 0.8$ and $scatter = 2.0$ in order to weaken the community structure. Otherwise GCE would have already detected a community with perfect quality using the graph structure only, just like ILSCD-GCE-Cosine would have detected communities perfectly using the attribute similarities only.