Algorithmic Aspects of Clustering

Theory, Experimental Evaluation, and Applications in Network Analysis and Visualization

zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften

der Fakultät für Informatik
der Universität Fridericiana zu Karlsruhe (TH)

genehmigte

Dissertation

von

Marco Gaertler

aus Schauffhausen, Schweiz

Tag der mündlichen Prüfung:	14. Februar 2007
Erste Gutachterin:	Frau Prof. Dr. Dorothea Wagner
Zweiter Gutachter:	Herr Prof. Dr. Ulrik Brandes

Acknowledgment

The various aspects of graph clustering have filled the better part of my last five years. I was introduced to the topic by Ulrik Brandes when I worked as a student helper in the working group of Dorothea Wagner. I was fairly fascinated by the simple and informal description, yet a formal description was missing. Although, many ideas were discarded and many dead ends occurred on the way, the original magic has survived and still motivates me to explore its intrinsic obstacles.

I would like to acknowledge my deep gratitude to my supervisor Prof. Dr. Dorothea Wagner for all her guidance and support. She taught me many different things along the way. In addition, I would like to express my gratefulness to my second reviewer Prof. Dr. Ulrik Brandes for introducing me to the problem, listening to sometimes crazy ideas, and giving useful advises.

My thanks also go to my colleagues in the Institute for Theoretical Informatics – ITI Wagner for providing support and a friendly and stimulating environment. Special thanks goes to Daniel Delling, Robert Görke, Bastian Katz, Marcus Krug, and Steffen Mecke for proofreading and providing implementations and visualizations. As a note to all of you who are astonished about my desktop, be advised that there is probably a wicked quality measure, which someone finds fairly intuitive, such that the chaos on my desk makes perfect sense; after five years I know the only remaining task is to explicitly find this measure. In addition, I would like to thank Prof. Dr. Stefano Leonardi, Prof. Dr. Giuseppe Di Battista, and Dr. Maurizio Patrignani for the very friendly welcome to Rome and the possibility to be a part of their working groups.

Beside the daily study, my friends kept my spirit high and motivated me to continue my way. The list of names is way too long and probably I would forget most. Nevertheless a special thanks goes to Katja and Jens Lohner, Kirsten and Martin Oberhofer, Peggy and Roland Martin, Iris and Stephan Kolassa, Steffen Mecke, Nicole Kröger and Dieter Gluche, Zoran Nikoloski, Leon Peeters, Jan-Willem Goossens, Silke Wagner, and Sonja Ziniel.

Last but not least, I wish to thank my family, Beatrice, Robert, Verena, and Carina Honz and Käthe Gaertler for giving me all the love and support I needed in my life. – Thanks!

Contents

Chapter 1

Motivation

The year 2006 is declared as the year of computer and information science (*Jahr der Informatik* [64]). It is part of the series 'year of science' that focuses on the dialog between science and society. The selection of the current year reflects the importance of computer and information technology for our society as well as the global economy. Furthermore, it is long suspected that we already live in or are heading towards an *information society*, in which the creation, distribution and manipulation of information is a significant economic and cultural activity ([105]). Approval facts are the expanding industrial sector of services and consulting or the availability and use of digital and Internet services such as online banking or shopping. Although an information society is not necessarily related to the Internet or a digital world, the corresponding aspects contribute heavily.

Especially the availability of digitized data or Internet services influences our personal activities. Public wireless networks and broadband Internet access like Digital Subscriber Lines (DSLs) have become a common means of communication. In Germany, for example, every fourth household possesses such a connection [60] and the current market situations indicate that this number is going to increase. Many providers cut their prices or offer combinations of telephone and Internet access. Although this availability of information provides an improved foundation for our decision-making processes, the relation between the associated data become more complex. As an example, consider the problem of finding a new book to read. Usually, one considers the top-seller list or recommendations of friends and colleagues. If this fails, due to the fact that all suggested books are already read, then a quick search in one of the (online) book shops could help. However, the sheer quantity of books make it impossible to manually scan them. In such a case automatic processing and reduction of the complexity can help to find a suitable answer. For our book example, at least one of the online shops (Amazon [5]) implemented such a recommendation system that provides for each offered product a short list of other products that could match the buyer's interest. Overall, the advantage of this technique is that the construction process for the recommendations includes a large quantity of information, while the output presents only relevant parts in a condensed form.

Many applications in the field such as project planning, transportation management, text analysis, bio-informatics, communication networks, epidemiology, and social network analysis require the analysis and evaluation of networks or network-like data which usually consist of a set of objects and an associated binary relation. Most of these fields independently developed their methodology and share only a small formal foundation. In addition, many methods are specifically designed for applications or have been re-invented several times. Unfortunately, a unified framework has not yet emerged and the construction of such a one is clearly beyond the scope of this thesis. Instead, we investigate various algorithmic aspects of clusterings. More precisely, we seek a methodical approach to formulate, analyze, and apply density-based clustering techniques for graphs. In the theoretical and experimental part of the thesis, we present several concepts for quality measuring, comparing clusterings and generating graphs with significant clustering structure as well as techniques for finding clusterings. The second part contains a case study of applying such technique to a network taken from the real world. In more details, this dissertation is structured as follows.

Structure

Chapter 2: *Introduction to Clustering*

A basic overview of clustering is presented. Beside the informal description of the underlying paradigm, i. e., intra-cluster density versus inter-cluster sparsity, the notation and some preliminaries are stated.

Chapter 3: *Quality Measures*

We introduce a general framework for expressing quality measures based on the paradigm of intra-cluster density versus inter-cluster sparsity. The measures coverage, performance, intra-cluster conductance, inter-cluster conductance, and modularity are presented and investigated in details.

Chapter 4: *Comparison of Clusterings*

A brief overview of comparison techniques for clusterings is presented. In the first part of the chapter, we focus on lattice-based concepts that have been established in literature, while in the second part, extensions and new techniques incorporating the graph structure more deeply are introduced.

Chapter 5: *Clustering Methods*

Concepts of clustering methods and their realization is the target of this chapter. In the first part, we state the abstract definition of various concepts such as greedy and shifting techniques as well as ideas from general optimization. The second part contains several realizations of these concepts, including general remarks about their applicability. In the final part, we present a selection of several implemented algorithms that were considered in our study of quality assessment of algorithms.

Chapter 6: *Experimental Evaluation*

We describe our benchmark suit for systematically and methodically evaluating various clustering concepts such as quality measures, comparators, generators, and algorithms based experimental observation. Concepts for generating graphs with a known (significant) clustering structure are introduced and potential pit falls are discussed as well as interdependencies with the techniques to test. The main part of the chapter contains the experiments, results, and gained insights of our performed experiments.

Chapter 7: *Variations of Clustering*

Partition-based clustering covers a large fraction of cluster analysis. In this chapter, we give a brief overview of several extensions such as dynamic problems and alternative data structures. More precisely, we model the update problem, where a given clustering has to be updated when the underlying graph changes as well as a sequential problem, where a series of graphs is given and we are interested in the stable groups over time. As structural extensions we state fuzzy and hierarchical extensions.

Chapter 8: *Case Study: Autonomous Systems*

We perform a case study of the network of the Autonomous Systems which is an abstract view of the physical Internet. The aspects of data reduction and filtering irrelevant information are initially considered. Furthermore, clustering is used to investigate short- and long-term behavior of the network as well as to extract information about baselines. Clustering also contributed significantly to the development of the presented visualization technique that highlights structural properties in a unique way. The final part of the case study consists of the analysis of an embedded applications, i. e., the file-sharing application Gnutella which creates a network on top of the Internet; although communication takes place in this created network, the traffic flows through the Internet. Thus, the corresponding analysis must consider the interplay of both networks. We introduce a novel technique that utilizes the prior presented visualization technique.

Chapter 9: *Landscape Like Visualization Techniques*

The visualization technique that we developed during the case study of the network of Autonomous Systems is further refined in this chapter. We discuss several potential extensions to derive a general visualization technique for arbitrary nested decompositions.

Chapter 10: *Conclusion*

The thesis is concluded with a summary of the presented results and a brief outlook.

Parts of this work has been published in [24, 13, 49, 3, 14, 47, 50, 4, 20, 21, 32, 33, 48, 34, 23].

Part I.

Theory and Experimental Evaluation

Chapter 2

Preliminaries

Clustering is a synonym for the decomposition of a set of entities into 'natural groups'. There are two major aspects to this task: the first involves algorithmic issues on the identification of such decompositions such as tractability, while the second concerns the evaluation, i. e., the quality of a clustering or the comparison between clusterings. Owing to the informal notion of natural groups, many different disciplines have developed their view of clustering independently. Originally, clustering was introduced to the data mining research as the unsupervised classification of patterns into groups [65]. Since that time a comprehensive framework has slowly started to evolve. In the following, the simple yet fundamental paradigm of intra-cluster density versus inter-cluster sparsity will be discussed exclusively. This restriction is necessary in order to provide some insight into clustering theory, and to keep the scope of the thesis. However, many other interpretations of natural decomposition extend this framework, or are relatively similar. Another specialty is that the input data is represented as networks that are not complete in general. In the classic clustering theory, entities were embedded in metric spaces, and the distance between them was related to their similarity. Thus, all pairwise similarities were known at the beginning. In standard network analysis, the input networks are usually sparse. Even if they are not, it is very unlikely that they are complete. This will be the motivation for studying clustering methods that deal with network input data.

Clustering, based either on the simple paradigm of intra-cluster density versus inter-cluster sparsity or on other more sophisticated formulations, focuses on disjoint cliques as the ideal situation. In some instances the desired cases are totally different. An overview of models, where clusters and their connection between each other can have more complex roles, or traditional blockmodels can be found in [22, Ch. 9,10].

The popularity of density-based clustering is due to its similarity to the human perception. Most things in our daily life are naturally grouped into categories. For example books are classified with respect to their content, e. g., scientific, fiction, guidebooks, law, etc. Each topic can be refined, e. g., scientific publications can be grouped according to their scientific discipline. The relationship of elements within the same group is strong, whereas elements in different groups typically have a weak relation. Most approaches that deal with information processing are based upon this fact. For example finding a book with a certain content. First, related topics are selected and only those books that belong to these groups are examined closer. The recursive structure of topics and subtopics suggests a repeated application of this technique. Using the clustering information on the data set, one can design methods that explore and navigate within the data with a minimum of human interaction. Therefore, it is a fundamental aspect of automatic information processing

2.1. Notation

We will use the notation of [47], however restrict ourselves to the undirected cases, more precisely: Let $G = (V, E)$ be an undirected graph, then a *clustering* $\mathcal{C} = \{C_1, \dots, C_k\}$ of G is a partition of the node set V into non-empty subsets C_i. The set $E(C_i, C_j)$ is the set of all edges that have their origin in C_i and their destination in C_j; $E(C_i)$ is a short-hand for $E(C_i, C_i)$. Then $E(\mathcal{C}) := \bigcup_{i=1}^{k} E(C_i)$ is the set of *intra-cluster edges* and $\overline{E(\mathcal{C})} := E \setminus E(\mathcal{C})$ the set of *inter-cluster edges*. The number of intra-cluster edges is denoted by $m\,(\mathcal{C})$ and the number of inter-cluster edges by $\overline{m}\,(\mathcal{C})$. In the following, we often identify a cluster C_i with the induced subgraph of G, i. e., the graph $G[C_i] := (C_i, E(C_i))$. A clustering is called *trivial* if either $k = 1$ (*1-clustering*) or $k = |V|$ (*singletons*). A clustering with $k = 2$ is also called a *cut*.

The set of all possible clusterings is denoted by $\mathcal{A}\,(G)$. The set $\mathcal{A}\,(G)$ is partially ordered with respect to inclusion. Given two clusterings $\mathcal{C}_1 := \{C_1, \dots, C_k\}$ and $\mathcal{C}_2 := \{C'_1, \dots, C'_\ell\}$, the partial ordering is defined in Equation (2.1).

$$\mathcal{C}_1 \leq \mathcal{C}_2 \colon \iff \forall\, 1 \leq i \leq k \colon \exists\, j \in \{1, \dots, \ell\} \colon C_i \subseteq C'_j \tag{2.1}$$

Clustering $\mathcal{C}_1$ is called a *refinement* of $\mathcal{C}_2$, and $\mathcal{C}_2$ is called a *coarsening* of $\mathcal{C}_1$. The *common refinement* of $\mathcal{C}_1$ and $\mathcal{C}_2$ is defined as

$$\mathcal{C}_1 \wedge \mathcal{C}_2 := \left\{ C \mid \exists\, 1 \leq i \leq k, 1 \leq j \leq \ell \colon C = C_i \cap C'_j \neq \emptyset \right\} \tag{2.2}$$

The associated $(k \times \ell)$-confusion matrix $M = (m_{ij})$ is defined by $m_{ij} := \left|C_i \cap C'_j\right|$. A chain of clusterings, i. e., a subset of clusterings such that every pair is comparable, is also called a *hierarchy*. The hierarchy is called *total* if both trivial clusterings are contained. A hierarchy that contains exactly one clustering of k clusters for every $k \in \{1, \dots, n\}$ is called *complete*. It is easy to see that such a hierarchy has n clusterings and that no two of these clusterings have the same number of clusters.

Given a weighted graph $G = (V, E, \omega)$ and a clustering $\mathcal{C} = \{C_1, \dots, C_k\}$, we define an abstracted version of G as weighted graph

$$G_{\mathcal{C}} := (\{1, \dots, k\}, \{\{i, j\} \mid \exists\, \{u, v\} \in E \colon u \in C_i \wedge v \in C_j\}, \omega_{\mathcal{C}}),$$

where $\omega_{\mathcal{C}}(\{i, j\}) := \sum_{e \in E(C_i, C_j)} \omega(e)$. This graph is obtained from the input graph G by shrinking each cluster into a single node and replacing parallel edges.

Besides viewing a clustering as a partition, it can also be seen as an equivalence relation $\sim_{\mathcal{C}}$ on $V \times V$, where $u \sim_{\mathcal{C}} v$ if u and v belong to the same cluster in $\mathcal{C}$. Note that the edge set E is also a relation over $V \times V$, and it is an equivalence relation if and only if the graph consists of the union of disjoint cliques.

The power set of a set X is the set of all possible subsets, and is denoted by $\mathcal{P}(X)$. A *cut function* $S\colon \mathcal{P}(V) \to \mathcal{P}(V)$ maps a set of nodes to a subset of itself, i. e.,

$$\forall V' \subseteq V\colon S(V') \subseteq V' \ . \tag{2.3}$$

Cut functions formalize the idea of cutting a node-induced subgraph into two parts. For a given node subset V' of V the cut function S defines a cut by $(S(V'), V' \backslash S(V'))$. In order to exclude trivial functions, we require a cut function to assign a non-empty proper subset whenever possible. *Proper* cut functions in addition fulfill the condition (2.4).

$$\forall V' \subseteq V\colon |V'| > 1 \Longrightarrow \emptyset \neq S(V') \subset V' \ . \tag{2.4}$$

These functions are important for clustering techniques that are based on recursive cutting.

Chapter 3

Quality Measures

As was pointed out in the introduction (Chapter 2), clustering techniques are used to find groups that are internally dense and that are sparsely connected with each other. Although this paradigm of intra-cluster density versus inter-cluster sparsity is more precise than the term 'natural groups', it is still based on our intuition and not on a formal quantification. One way to mathematically express it is by structural indices. These are mappings that assign a non-negative real number to each clustering. Often their range is normalized to the unit interval, where one means best possible structural behavior and zero means worst possible structural behavior. Thus, these indices provide a quantitative means to formalize the notion of clustering and to qualitatively evaluate them. More detail on the evaluation of clusterings are given in Chapter 6. In the following a general framework for indices is presented along the lines of [47]. Most of the existing measures can be expressed within it.

Let $G = (V, E, \omega)$ be a simple, weighted and undirected graph, where $\omega \colon E \to \mathbb{R}_0^+$ represents the strength of the similarity relation modeled by the edges, and let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a clustering of G. Although some applications require negative similarity scores, e. g., for simultaneously expressing similarities and dissimilarities, we will only consider non-negative similarity for the framework. Some extensions are given in Chapter 7. For the unweighted case, the weighting function ω is assumed to be constantly one. In many cases ω will be a mapping to $\mathbb{R}^+$, however, in some cases it is useful to distinguish between edges with weight zero and those node pairs that are not connected by an edge. We will also use the following short-cut for summing up the weight of an edge subset:

$$\omega(E') := \sum_{e \in E'} \omega(e) \qquad \text{for } E' \subseteq E \ .$$

For simplicity, we assume that $\omega(E) \neq 0$.

3.1. General Framework

Before defining the actual indices, their framework is presented. The indices will be composed of two independent functions $f, g \colon \mathcal{A}(G) \to \mathbb{R}_0^+$, where f measures the density inside the clusters and g the sparsity between clusters. The functions are combined in the following way:

$$\mathsf{index}\left(\mathcal{C}\right) := \frac{f\left(\mathcal{C}\right) + g\left(\mathcal{C}\right)}{\max\{f\left(\mathcal{C}'\right) + g\left(\mathcal{C}'\right) : \mathcal{C}' \in \mathcal{A}\left(G\right)\}} \tag{3.1}$$

In order to guarantee the well-definition of Equation (3.1), we assume that there is at least one clustering $\mathcal{C}'$ such that $f\left(\mathcal{C}'\right) + g\left(\mathcal{C}'\right)$ is not zero. If not then $\mathsf{index}\left(\mathcal{C}\right)$ is defined as zero. For some indices either f or g is constantly zero. These indices examine only the (internal) density or the (external) sparsity.

Indices serve two different purposes simultaneously: first and foremost, they rate a partition with respect to clustering paradigms, and, second, they compare clusterings regarding quality. Before we explain both aspects in detail, please note that, while our indices have these properties, there exist other measures that realize only one aspect.

The quality of a partition as a clustering is expressed in quantitative terms, i. e., an index (discretely) counts certain substructures like intra-cluster edges, triangles, or cliques. These structural elements are related to clustering properties. Many intra-cluster edges, triangles, or cliques inside the clusters indicate a large intra-cluster density. In an analogous way the lack of these elements between clusters imply the inter-cluster sparsity. Thus, the quality of clustering is reduced to a quantitative aspect. Intuitively, there will always be a maximum number of these indicators. The number of intra-cluster edges, triangles, or cliques is limited by the size of clusters. In the ideal case the bounds are met. Thus, if an index counts only half of these witnesses, then the clustering is only half as good as possible. As we will see, these bounds are not tight for all indices and all input graphs. Thus, most of them are 'absolute' in the sense that they do not depend on the input graph, but rather on the clustering paradigms. In conclusion, the actual range of indices provides useful information.

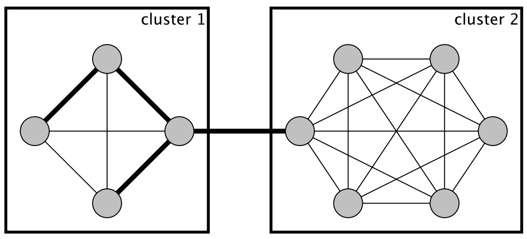
Because of this absolute, quantitative structure of our indices, they can be used to compare clusterings regardless of whether the underlying graph is the same or not. Different graphs can have different bounds for the number of substructures, but the indices rate the quality relative to the individual bounds. For example, in a graph with 10 nodes and 15 edges a clustering could have 12 intra-cluster edges, and in another graph with 18 nodes and 30 edges another clustering could have 27 intra-cluster edges. Although we have three inter-cluster edges in both cases, the second clustering would be better since $27/30 = 9/10 > 4/5 = 12/15$. This property is required when algorithms are evaluated with random instances, or the data of the input network is not reliable, i. e., different data collections result into different networks. The clustering methods could be applied to all networks, and the clustering with the best score would be chosen. However, if the index uses characteristics of the input graph, like the number of edges, maximum weight, etc., then this comparison can only be done when the underlying graph is the same for all clusterings. Therefore, indices that depend on the input graph are not appropriate for all applications, such as benchmarks. Although this dependency will seldom occur, one has to consider these facts when designing new indices.

3.2. Coverage

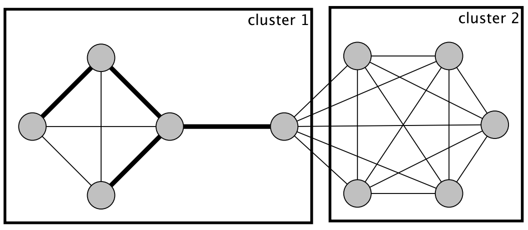
The *coverage* $\operatorname{cov}(\mathcal{C})$ measures the weight of intra-cluster edges, compared to the weight of all edges. Thus $f(\mathcal{C}) = \omega(E(\mathcal{C}))$ and $g \equiv 0$. The maximum value is achieved for the 1-clustering. Equation (3.2) shows the complete formula.

$$\operatorname{cov}(\mathcal{C}) := \frac{\omega(E(\mathcal{C}))}{\omega(E)} = \frac{\sum_{e \in E(\mathcal{C})} \omega(e)}{\sum_{e \in E} \omega(e)} \tag{3.2}$$

Coverage measures only the accumulated density within the clusters. Therefore, an individual cluster can be sparse or the number of inter-cluster edges can be large. This is illustrated in Figure 3.1. Coverage is also the probability of randomly select-



(a) intuitive clustering



(b) non-trivial clustering with best coverage

Figure 3.1.: A situation where coverage splits an intuitive cluster. The thickness of an edge corresponds to its weight. If normal edges have weight one and bold edges weight 100, then the intuitive clustering has cov $= 159/209 \approx 0.76$ while the optimal value for coverage is $413/418 \approx 0.99$

ing an intra-cluster edge (where the probability of selection an edge is proportional to its weight, i. e., $\Pr[e] \sim \omega(e)$). The structure of clusterings with optimal coverage value is related to the connectivity structure of the graph. A clustering is *compatible* with the connectivity structure if clusters consist only of unions of connected components of the graph. Proposition 3.1 gives a characterization of clusterings with optimal coverage value.

Proposition 3.1. *A clustering has* cov $= 1$ *if and only if either the set of inter-cluster edges is empty or all inter-cluster edges have weight zero. Especially, clusterings that are compatible with the connectivity structure have coverage value 1.*

Sketch of Proof. The edge set is disjointly partitioned into intra-cluster edges and inter-cluster edges, therefore Equation (3.3) holds for any clustering $\mathcal{C}$.

$$\omega(E) = \omega(E(\mathcal{C})) + \omega(\overline{E(\mathcal{C})}) \tag{3.3}$$

Thus, coverage $\operatorname{cov}(\mathcal{C}) = 1$ holds if and only if $\omega(\overline{E(\mathcal{C})}) = 0$.

Because clusterings that are compatible with the connectivity structure of the graph have no inter-cluster edge, one can use the equivalence to prove the proposition. □

A conclusion of Proposition 3.1 is that the 1-clustering always has a coverage value 1. The case of disconnected input graphs is rather special and most techniques even assume that it is connected. Proposition 3.1 can be extended to characterize non-trivial clusterings that are optimal with respect to coverage. A further characterization of non-trivial clusterings that are optimal when trivial clusterings are excluded is possible and given in Proposition 3.2.

Proposition 3.2. *Let $G = (V, E)$ be a connected graph where every cut has positive weight. Then the clusterings that have more than one cluster and have optimal coverage value are those that are induced by a minimum cut.*

Proof. First, it is obvious that every clustering $\mathcal{C}$ with $k > 1$ can be transformed into a clustering $\mathcal{C}'$ with less than k clusters, with $\operatorname{cov}(\mathcal{C}) \leq \operatorname{cov}(\mathcal{C}')$. This is achieved by merging any two clusters. Therefore, a non-trivial clustering with optimal coverage has two clusters, and thus is a cut. Second, maximizing coverage is equivalent to minimizing the weight of the inter-cluster edges, and the edges that are contained in the cut have minimum weight. That completes the proof. $\square$

Because of the properties described in Propositions 3.1 and 3.2, coverage is rarely used as the only quality measurement of a clustering. Minimum cuts often cannot catch the intuition, and separate only a very small portion of the graph. However, there are a few exceptions [58], where both, the input graph and good clusterings, have a very special structure. In the next section, we will investigate an alternative cut measure that considers cut size as well as the balance of the two components.

3.3. Conductance Cut Measures

In contrast to coverage, which measures only the accumulated edge weight within clusters, one can consider further structural properties like connectivity. Intuitively, a cluster should be well connected, i. e., many edges need to be removed to bisect it. Two clusters should also have a small degree of connectivity between each other. In the ideal case, they are already disconnected. Cuts are a useful method to measure connectivity. The standard minimum cut has certain disadvantages (Proposition 3.2), therefore an alternative cut measure will be considered: *conductance*. It compares the weight of the cut with the edge weight in either of the two induced subgraphs. Informally speaking, the conductance is a measure for bottlenecks. A cut is a bottleneck, if it separates two parts of roughly the same size with relatively few edges.

Definition 3.3. *Let $\mathcal{C}' = (C_1', C_2')$ be a cut, i. e., $(C_2' = V \setminus C_1')$ then the* conductance-weight $a(C_1')$ *of a cut side and the* conductance $\phi(\mathcal{C}')$ *are defined in Equations* (3.4) *and* (3.5).

$$a(C_1') := \sum_{(u,v)\in E(C_1',V)} \omega((u,v)) \tag{3.4}$$

$$\phi(\mathcal{C}') := \begin{cases} 1, & \textit{if } C_1' \in \{\emptyset, V\} \\ 0, & \textit{if } C_1' \notin \{\emptyset, V\}, \omega(\overline{E(\mathcal{C})}) = 0 \\ \dfrac{\omega(\overline{E(\mathcal{C})})}{\min(a(C_1'), a(C_2'))}, & \textit{otherwise} \end{cases} \tag{3.5}$$

The conductance *of the graph G is defined by*

$$\phi(G) = \min_{C_1 \subseteq V} \phi((C_1, V \setminus C_1)) \ . \tag{3.6}$$

Note that the case differentiation in Equation (3.5) is only necessary in order to prevent divisions by zero. Before presenting further general information about conductance, graphs with maximum conductance are characterized.

Lemma 3.4. *Let $G = (V, E, \omega)$ be an undirected and positively weighted graph. Then G has maximum conductance, i. e., $\phi(G) = 1$ if and only if G is connected and has at most three nodes, or is a star.*

Proof. Before the equivalence is shown, two short observations are stated:

1. All disconnected graphs have conductance 0 because there is a non-trivial cut that has zero weight and the second condition of the Formula (3.5) holds.

2. For a non-trivial cut $\mathcal{C}' = (C_1', V \setminus C_1')$ the conductance-weight $a(C_1')$ can be rewritten as

$$a(C_1') = \sum_{e \in E(C_1',V)} \omega(e) = \omega(E(C_1')) + \omega(\overline{E(\mathcal{C})})$$

in undirected graphs. Thus, the third condition in Formula (3.5) can be simplified to

$$\frac{\omega(\overline{E(\mathcal{C})})}{\min\Big(a(C_1'), a(V \setminus C_1')\Big)} = \frac{\omega(\overline{E(\mathcal{C})})}{\omega(\overline{E(\mathcal{C})}) + \min\Big(\omega(E(C_1')), \omega(E(V \setminus C_1'))\Big)} \ . \tag{3.7}$$

'$\Longleftarrow$': If G has one node, then the first condition of Formula (3.5) holds and thus $\phi(G) = 1$.

If G has two or three nodes or is a star, then every non-trivial cut $\mathcal{C}' = (C_1', V \setminus C_1')$ isolates an independent set, i. e., $E(C_1') = \emptyset$. This is achieved by setting C_1' to the smaller cut set if G has at most three nodes and to the cut set that does not contain the center node if G is a star. Therefore $\omega(E(C_1')) = 0$ and Equation (3.7) implies $\phi(\mathcal{C}') = 1$. Because all non-trivial cuts have conductance 1, the graph G has conductance 1 as well.

'$\Longrightarrow$': If G has conductance one, then G is connected (observation 1) and for every non-trivial cut $\mathcal{C}' = (C'_1, V \setminus C'_1)$ at least one edge set $E(C'_1)$ or $E(V \setminus C'_1)$ has 0 weight (observation 2). Because ω has only positive weight, at least one of these sets has to be empty.

It is obvious that connected graphs with at most three nodes fulfill these requirements, therefore assume that G has at least four nodes. The graph has a diameter of at most two because otherwise there is a path of length three with four pairwise distinct nodes $v_1, \dots, v_4$, where $e_i := \{v_i, v_{i+1}\} \in E$ for $1 \leq i \leq 3$. Then the non-trivial cut $\mathcal{C}' = (\{v_1, v_2\}, V \setminus \{v_1, v_2\})$ cannot have conductance 1, because first the inequality $\omega(\overline{E(\mathcal{C}')}) \geq \omega(e_2) \geq 0$ implies the third condition of Formula (3.5) and second both cut sides are non-empty ($e_1 \in E(\{v_1, v_2\})$ and $e_3 \in E(V \setminus \{v_1, v_2\})$). By the same argument, G cannot contain a simple cycle of length four or greater. It also cannot have a simple cycle of length three. Assume G has such a cycle v_1, v_2, v_3. Then there is another node v_4 that is not contained in the cycle but in the neighborhood of at least one v_i. Without loss of generality $i = 1$. Thus, the non-trivial cut $(\{v_1, v_4\}, V \setminus \{v_1, v_4\})$ is a counterexample. Thus G cannot contain any cycle and is therefore a tree. The only trees with at least four nodes, and a diameter of at most two, are stars. □

It is $\mathcal{NP}$-hard to calculate the conductance of a graph [7]. Fortunately, it can be approximated with a guarantee of $\mathcal{O}(\log n)$ [101] and $\mathcal{O}\left(\sqrt{\log n}\right)$ [6]. For some special graph classes, these algorithms have constant approximation factors. Several of the involved ideas are found in the theory of Markov chains and random walks. There, conductance models the probability that a random walk gets 'stuck' inside a non-empty part. It is also used to estimate bounds on the rate of convergence. This notion of 'getting stuck' is an alternative description of bottlenecks. One of these approximation ideas is related to spectral properties. Lemma 3.5 indicates the use of eigenvalues as bounds.

Lemma 3.5 ([95, Lemma 2.6])**.** *For an ergodic*[1] *reversible Markov chain with underlying graph G, the second (largest) eigenvalue λ_2 of the transition matrix satisfies:*

$$\lambda_2 \geq 1 - 2 \cdot \phi(G) \ . \tag{3.8}$$

A proof of that can be found in [95, p. 53]. Conductance is also related to isoperimetric problems as well as expanders, which are both related to similar spectral properties themselves.

For unweighted graphs the conductance of the complete graph is often a useful boundary. It is possible to calculate its exact conductance value. Proposition 3.6 states the result. Although the formula is different for even and odd number of nodes, it shows that the conductance of complete graphs is asymptotically 1/2.

[1]Aperiodic and every state can be reached an arbitrary number of times from all initial states.

Proposition 3.6. *Let n be an integer, then equation* (3.9) *holds.*

$$\phi(K_n) = \begin{cases} \frac{1}{2} \cdot \frac{n}{n-1} & \text{, if } n \text{ is even} \\ \frac{1}{2} + \frac{1}{n-1} & \text{, if } n \text{ is odd} \end{cases} \tag{3.9}$$

Proof. Evaluating Equation (3.5) in Definition 3.3 with $G = K_n$ leads to

$$\phi(K_n) = \min_{C \subset V, 1 \leq |C| < n} \frac{|C| \cdot (n - |C|)}{\min(|C|(|C| - 1), (n - |C|)(n - |C| - 1))} \ . \tag{3.10}$$

Node subsets of size k of a complete graph are pairwise isomorphic, therefore only the size of the subset C matters. Thus, Equation (3.10) can be simplified to

$$\phi(K_n) = \min_{1 \leq k < n} \frac{k(n-k)}{\min(k^2 - k, n^2 - 2nk - n - k)} \ . \tag{3.11}$$

The fraction in Equation (3.11) is symmetric, thus it is sufficient if k varies in the range from 1 to $\lceil n/2 \rceil$. Using the fact that the fraction is also monotonic decreasing with increasing k, the minimum is assumed for $k = \lfloor n/2 \rfloor$. A simple case differentiation for even and odd ks leads to the final Equation (3.9). □

In the following, two clustering indices are derived with the help of conductance. These will be intra-cluster conductance and inter-cluster conductance that have been introduced in [101] and [24], respectively. Both focus on one property only. The first one measures internal density, while the second rates the connection between clusters.

The *intra-cluster conductance* α is defined as the minimum conductance occurring in the cluster-induced subgraphs $G[C_i]$, i. e.,

$$f(\mathcal{C}) = \min_{1 \leq i \leq k} \phi(G[C_i]) \qquad \text{and} \qquad g \equiv 0 \ . \tag{3.12}$$

Note that $G[C_i]$ in $\phi(G[C_i])$ denotes a subgraph, and therefore is independent of the rest of the original graph G. The conductance of a (sub)graph is small if it can naturally be bisected, and great otherwise. Thus, in a clustering with small intra-cluster conductance there is supposed to be at least one cluster containing a bottleneck, i. e., the clustering is possibly too coarse in this case. The minimum conductance cut itself can also be used as a guideline to split the cluster further. The *inter-cluster conductance* δ considers the cuts induced by clusters, i. e.,

$$f \equiv 0 \qquad \text{and} \qquad g = \begin{cases} 1, & \text{if } \mathcal{C} = \{V\} \\ 1 - \max_{1 \leq i \leq k} \phi((C_i, V \setminus C_i)), & \text{otherwise} \end{cases} \ . \tag{3.13}$$

Note that $(C_i, V \setminus C_i)$ in $\phi((C_i, V \setminus C_i))$ denotes a cut within the graph G. A clustering with small inter-cluster conductance is supposed to contain at least one cluster that has relatively strong connections outside, i. e., the clustering is possibly too fine. In contrast to the intra-cluster conductance, one cannot directly use the induced cut information to merge two clusters.

For both indices the maximum of $f + g$ is one, which leads to the final formula:

$$\alpha(\mathcal{C}) := \min_{1 \le i \le k} \phi(G[C_i]) \tag{3.14}$$

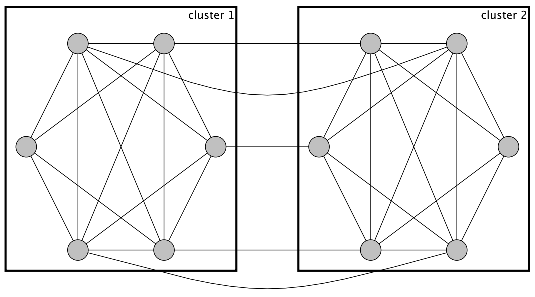
$$\delta(\mathcal{C}) := \begin{cases} 1, & \text{if } \mathcal{C} = \{V\} \\ 1 - \max\limits_{1 \le i \le k} \phi((C_i, V \setminus C_i)), & \text{otherwise} \end{cases} \tag{3.15}$$

Again a characterization of optimum clusterings is possible.

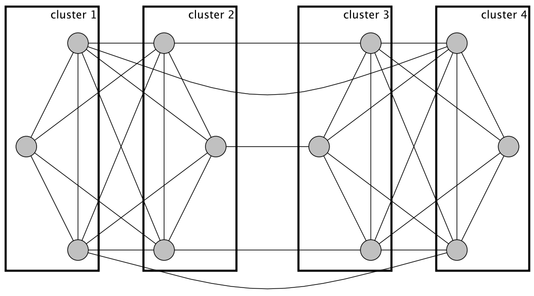
Proposition 3.7. *Only clusterings where the clusters consist of connected subgraphs that are stars or have size of at most three, have maximum intra-cluster conductance of* 1.

Proposition 3.8. *Only clusterings that have an inter-cluster edge weight of zero, including the* 1*-clustering, have maximum inter-cluster conductance of* 1.

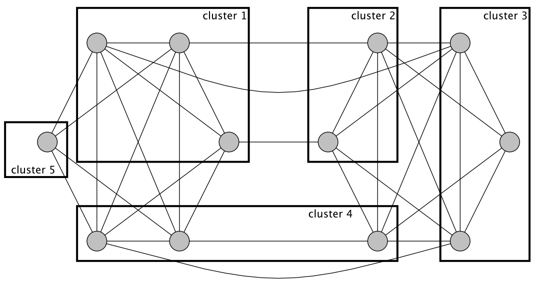
Both Proposition 3.7 and 3.8 are immediate consequences of the Definition 3.3 and Lemma 3.4. Both measures, intra-cluster conductance and inter-cluster conduc-



(a) intuitive clustering



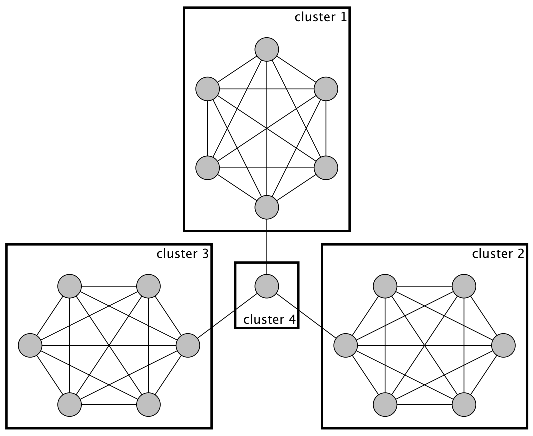
(b) non-trivial clustering with best intra-cluster conductance



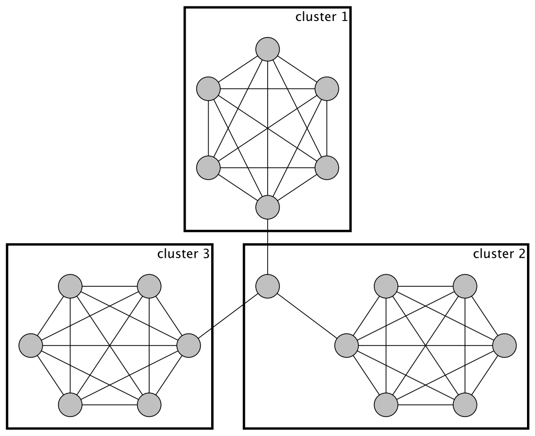
(c) non-trivial clustering with best intra-cluster conductance

Figure 3.2.: A situation where intra-cluster conductance splits intuitive clusters. The intuitive clustering has $\alpha = 3/4$, while the other two clusterings have $\alpha = 1$. The split in Figure 3.2(b) is only a refinement of the intuitive clustering, while Figure 3.2(c) shows a clusterings with same intra-cluster conductance value that is skew to the intuitive clustering

tance, have certain disadvantages. Two examples are shown in Figures 3.2 and 3.3.



(a) intuitive clustering



(b) non-trivial clustering with best inter-cluster conductance

Figure 3.3.: A situation where two very similar clusterings have very different inter-cluster conductance values. The intuitive clustering has $\delta = 0$, while the other has the optimum value of 8/9

Both examples explore the 'artificial' handling of small graphs considering their conductance. In practical instances, intra-cluster conductance values are usually below 1/2. Small clusters with few connections have relatively small inter-cluster conductance values.

3.4. Performance

The next index combines two non-trivial functions for the density measure f and the sparsity measure g. It simply counts certain pairs of nodes and was introduced in [100]. According to the general intuition of intra-cluster density versus inter-cluster sparsity, we define for a given clustering a *'correct' classified* pair of nodes as two nodes either belonging to the same cluster and being connected by an edge, or belonging to different clusters and being not connected by an edge. The resulting index is called *performance*. Its density function f counts the number of edges within all clusters while its sparsity function g counts the number of nonexistent edges between clusters, i. e.,

$$\begin{aligned} f(\mathcal{C}) &:= \sum_{i=1}^{k} |E(C_i)| \quad \text{and} \\ g(\mathcal{C}) &:= \sum_{u,v \in V} [(u,v) \notin E] \cdot [u \in C_i, v \in C_j, i \neq j] \ . \end{aligned} \tag{3.16}$$

The definition is given in Iverson Notation, first described in [63], and adapted by Knuth in [73]. The term inside the parentheses can be any logical statement. If the statement is true the term evaluates to 1, otherwise the term is 0. The maximum of $f+g$ has $n \cdot (n-1)/2$ as upper bound because this is the total number of different pairs of nodes. Please recall that loops are not present and each pair contributes with either zero or one. Calculating the maximum of $f+g$ is $\mathcal{NP}$-hard (see [93]), therefore this bound is used instead of the real maximum. By using some duality aspects, such as the number of intra-cluster edges and the number of inter-cluster edges sum up to the whole number of edges, the formula of performance can be simplified as shown in Equation (3.18).

$$\begin{aligned}
\mathsf{perf}(\mathcal{C}) &= \frac{m(\mathcal{C}) + \left(\frac{1}{2}n(n-1) - \frac{1}{2}\sum_{i=1}^{k}|C_i|(|C_i|-1) - \overline{m}(\mathcal{C})\right)}{\frac{1}{2}n(n-1)} \\
&= \frac{\frac{1}{2}n(n-1) - m + 2m(\mathcal{C}) - \frac{1}{2}\sum_{i=1}^{k}|C_i|(|C_i|-1)}{\frac{1}{2}n(n-1)} \qquad (3.17) \\
&= 1 - \frac{m(1-2\frac{m(\mathcal{C})}{m}) + \frac{1}{2}\sum_{i=1}^{k}|C_i|(|C_i|-1)}{\frac{1}{2}n(n-1)} \quad . \qquad (3.18)
\end{aligned}$$

Note that the derivation from Equation (3.17) to (3.18) applies the equality $m = m(\mathcal{C}) + \overline{m}(\mathcal{C})$, and that $m(\mathcal{C})/m$ is just the coverage $\operatorname{cov}(\mathcal{C})$ in the unweighted case. Similarly to the other indices, performance has some disadvantages. Its main drawback is the handling of very sparse graphs. Graphs of this type do not contain subgraphs of arbitrary size and density. Thus, the gap between the number of feasible edges (with respect to the structure) and the maximum number of edges (regardless of the structure) is also huge. For example, a planar graph cannot contain any complete graph with five or more nodes, and the maximum number of edges such that the graph is planar is linear in the number of nodes, while in general it is quadratic. In conclusion, clusterings with good performance tend to have many small clusters. Such an example is given in Figure 3.4.

Before introducing a weighted version of performance, we present several optimality results. In contrast to the previous indices, we are not aware of a general characterization for clusterings with maximum performance. On the other hand, for some graph families such results exists.

Lemma 3.9. *Let G be an undirected and unweighted graph. Every clustering $\mathcal{C}$ which has maximum performance consists of connected clusters.*

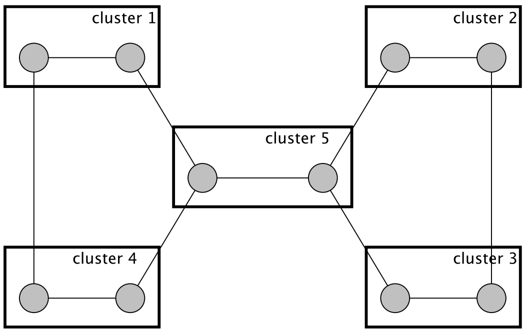
Proof. Assume otherwise and let $\mathcal{C}$ be a counter-example. More precisely, let $C \in \mathcal{C}$ be a cluster consisting of $k > 1$ connected components C_i. We define the clustering $\mathcal{C}'$ as follows:

$$\mathcal{C}' := (\mathcal{C} \setminus \{C\}) \uplus \{\{C_i\} \mid 1 \leq i \leq k\} \quad .$$

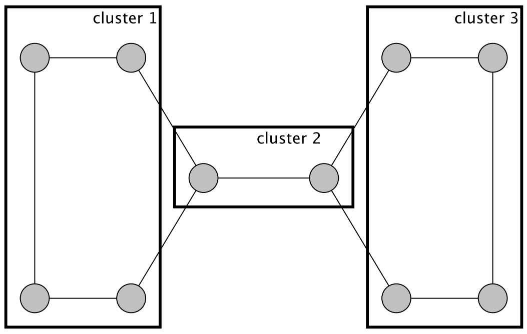
Since $E(C_i, C_j) = \emptyset$ for $i \neq j$, we obtain $f(\mathcal{C}) = f(\mathcal{C}')$ and

$$g(\mathcal{C}') = g(\mathcal{C}) + \sum_{i<j} |C_i|\,|C_j| \quad .$$

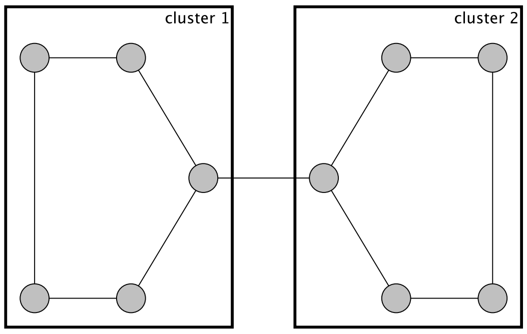
Since $k > 1$ the sum is positive and thus $\mathsf{perf}(\mathcal{C}) < \mathsf{perf}(\mathcal{C}')$ which contradicts the optimality of $\mathcal{C}$. □



(a) clustering with best performance



(b) intuitive clustering



(c) another intuitive clustering

Figure 3.4.: A situation where the clustering with optimal performance is a refinement (Figure 3.4(b)) of an intuitive clustering and is skew (Figure 3.4(c)) to another intuitive clustering

Theorem 3.10. *Let G be an the disjoint union of k complete graphs with nodeset V_i. Then the clustering $\mathcal{C} := \{\{V_i\} \mid 1 \leq i \leq k\}$ is the only clustering with maximum performance.*

Proof. Due to Lemma 3.9, each cluster in a clustering with maximum performance is connected, thus each cluster is contained within one of the cliques. Assume, there is another clustering $\mathcal{C}'$ that has maximum performance and at least one clique is partitioned into ℓ (non-empty) clusters $\{C_1, \ldots, C_\ell\}$. We define the clustering $\mathcal{C}''$ as

$$\mathcal{C}'' := (\mathcal{C}' \setminus \{C_1, \ldots, C_\ell\}) \uplus \left\{ \biguplus_{i=1}^{\ell} C_i \right\} .$$

We obtain $g(\mathcal{C}'') = g(\mathcal{C}')$ and $f(\mathcal{C}'') = f(\mathcal{C}') + \prod_{i<j} |C_i| \, |C_j|$, which contradicts the optimality of $\mathcal{C}'$. Thus $\mathcal{C}$ is the unique clustering having maximum performance. $\square$

Lemma 3.11. *Let T be an undirected and unweighted tree. Every clustering $\mathcal{C}$ which has maximum performance has clusters of size at most three.*

Proof. Assume otherwise and let $\mathcal{C}$ be a counter-example. More precisely, let $C \in \mathcal{C}$ with at least four nodes. According to Lemma 3.9, the cluster C is connected and thus $T' := T[C]$ is a subtree of T. Let $u \in C$ be a leaf in T', the node $v \in C$ its parent. We define the clustering $\mathcal{C}'$ as follows:

$$\mathcal{C}' := (\mathcal{C} \setminus C) \uplus \{\{u, v\}\} \uplus \{\{C'\} \mid C' \text{ is a connected component of } C \setminus \{u, v\}\}$$

In the following, we use the term *inter-cluster node pair* to denote a pair of nodes contained in different clusters that are not connected with an edge. The tree without the nodes $\{u, v\}$ has $\deg(v) - 1$ connected components. If $\deg(v) = 2$, then the clustering $\mathcal{C}'$ has one intra-cluster edge less and at least three inter-cluster node pairs more than $\mathcal{C}$. Similarly, if $\deg(v) > 2$, then $\mathcal{C}'$ has $\deg(v) - 1$ intra-cluster edges less and at least $\deg(v) - 1 + 1$ inter-cluster node pairs more than $\mathcal{C}$. In both cases $\mathsf{perf}(\mathcal{C}) < \mathsf{perf}(\mathcal{C}')$ holds, which contradicts the optimality of $\mathcal{C}$. □

Theorem 3.12. *Let T be an undirected and unweighted tree. Every clustering $\mathcal{C}$ that consists of a maximum matching has maximum performance.*

Proof. Assume otherwise and let $\mathcal{C}'$ be a clustering such that $\mathsf{perf}(\mathcal{C}) < \mathsf{perf}(\mathcal{C}')$. According to Lemma 3.11 all clusters in $\mathcal{C}'$ have at most three nodes. Further note, that for a clustering $\mathcal{C}''$ where each cluster has at most three nodes the following equation holds:

$$\begin{aligned} f(\mathcal{C}'') + g(\mathcal{C}'') &= f(\mathcal{C}'') + \binom{n}{2} - (n-1) + f(\mathcal{C}'') - n_2 - 3n_3 \\ &= \binom{n}{2} - n + 1 + n_3 + n_2 \ , \end{aligned}$$

where n is the number of nodes, n_i is the number of clusters having i nodes. Thus the clustering $\mathcal{C}'$ has the maximum number of cluster of size two and three. If $\mathcal{C}'$ has no clusters of size three, then $\mathcal{C}'$ cannot contain more cluster of size two than $\mathcal{C}$, thus $\mathsf{perf}(\mathcal{C}) = \mathsf{perf}(\mathcal{C}')$. Therefore $\mathcal{C}'$ has at least one cluster with three elements. We replace all clusters of size three in $\mathcal{C}'$ by a connected cluster of size two and the single node. The number of clusters of size two and three remains constant which implies the same performance score. Since the newly build clustering cannot have more clusters of size two than $\mathcal{C}$ its performance value is less or equal than $\mathsf{perf}(\mathcal{C})$ which contracts the assumption. □

Corollary 3.13. *Let $P_n := (\{1, \ldots, n\}, \{\{i, i+1\} \mid 1 \le i < n\})$ be the undirected, unweighted path with n nodes. Then every clustering $\mathcal{C}$ with maximum performance has at most one cluster with an odd number of nodes.*

Proof. According to proof of Theorem 3.12, the clustering $\mathcal{C}$ maximizes $n_3 + n_2$, where n_i is the number of clusters of size i. By transposing neighboring clusters, we achieve that the first $3 \cdot n_3$ nodes belong to clusters of size three, the next n_1 nodes are singletons, and the remaining $2 \cdot n_2$ are grouped in clusters of size two. Assume that $n_3 + n_1 > 1$ for $\mathcal{C}$, thus there are two neighboring clusters C_i and C_{i+1} both of odd size. Since $s := |C_i \cup C_{i+1}|$ is even, we can regroup them in groups of two. If $s = 6$ then, we can replace the two clusters each of size three by three clusters each of size two, if $s = 4$ then, we can obtain two clusters of size two, and if $s = 2$ then a cluster of size two is the replacement. In each case the sum $n_3 + n_2$ increases, which contradicts the assumption of $n_3 + n_1 > 1$. □

Note that, due to the similar structure of simple cycles and paths, Corollary 3.13 can be stated analogously:

Lemma 3.14. *Let $C_n := (\{1, \dots, n\}, \{\{i, i+1 \mod n\} \mid 1 \leq i \leq n\})$ be the undirected, unweighted cycle with n nodes. Then every clustering $\mathcal{C}$ with maximum performance has at most one cluster with an odd number of nodes and consists of cluster with size at most three.*

We omit the formal proof due to the high degree of similarity. Informally speaking, a cycle is just a path with an additional edge. This concluded the characterizations of clusterings with maximum performance. In the following, we consider different extensions of performance for weighted graphs.

Weighted Version There exist miscellaneous variations of performance that use more complex models for classification. However, many modifications highly depend on their application-specific background. Instead of presenting them, some variations to include edge weights are given. As pointed out in Section 3.1, indices serve two different tasks. In order to preserve the comparability aspect, we assume that all the considered edge weights have a meaningful maximum M. It is not sufficient to replace M with the maximum occurring edge weight because this value depends on the input graph. Also, choosing an extremely large value of M is not suitable because it disrupts the range aspects of the index. An example of such weightings with a meaningful maximum are probabilities where $M = 1$. The weighting represents the probability that an edge can be observed in a random draw. Using the same counting scheme of performance, one has to solve the problem of assigning a real value for node pairs that are not connected. This problem will be overcome with the help of the meaningful maximum M.

The first variation is straightforward and leads to the measure functions given in Equation (3.19):

$$\begin{aligned} f(\mathcal{C}) &:= \sum_{i=1}^{k} \omega(E(C_i)) \quad \text{and} \\ g(\mathcal{C}) &:= \sum_{u,v \in V} M \cdot [(u,v) \notin E] \cdot [u \in C_i, v \in C_j, i \neq j] \ . \end{aligned} \tag{3.19}$$

Please note the similarity to the unweighted definition in Formula (3.16). However, the weight of the inter-cluster edges is neglected. This can be integrated by modifying g:

$$g'(\mathcal{C}) := g(\mathcal{C}) \underbrace{+ M \cdot \left|\overline{E(\mathcal{C})}\right| - \omega\left(\overline{E(\mathcal{C})}\right)}_{=:g_w(\mathcal{C})} \ . \tag{3.20}$$

The additional term $g_w(\mathcal{C})$ corresponds to the difference of weight that would be counted if no inter-cluster edges were present and the weight that is assigned to the actual inter-cluster edges. In both cases the maximum is bounded by $M \cdot n(n-1)$, and the combined formula would be:

$$\mathsf{perf}_w(\mathcal{C}) = \frac{f(\mathcal{C}) + g(\mathcal{C}) + \theta \cdot g_w(\mathcal{C})}{n(n-1)M} \ , \tag{3.21}$$

where $\theta \in [0,1]$ is a scaling parameter that rates the importance of the weight of the inter-cluster edges (with respect to the weight of the intra-cluster edges). In this way there is a whole family of weighted performance indices.

An alternative variation is based on the duality. Instead of counting 'correct' classified node pairs the number/weight of the errors is measured. Equation (3.18) will be the foundation:

$$\begin{aligned}\tilde{f}\,(\mathcal{C}) &= \sum_{i=1}^{k} \Big(M|C_i|(|C_i|-1) - \vartheta \cdot \omega(E(C_i)\Big) \quad \text{and} \\ \tilde{g}\,(\mathcal{C}) &= \omega\left(\overline{E(\mathcal{C})}\right) \quad ,\end{aligned} \tag{3.22}$$

where ϑ is a scaling parameter that rates the importance of the weight of the intra-cluster edges (with respect to the weight of the inter-cluster edges). The different symbols for density $\tilde{f}$ and sparsity $\tilde{g}$ functions are used to clarify that these functions perform inversely to the standard functions f and g with respect to their range: small values indicate better structural behavior instead of large values. Both can be combined to a standard index via

$$\mathsf{perf}_m\,(\mathcal{C}) = 1 - \frac{\tilde{f}\,(\mathcal{C}) + \tilde{g}\,(\mathcal{C})}{n(n-1)M} \quad . \tag{3.23}$$

Note that the versions are the same for $\theta = \vartheta = 1$. In general, this is not true for other choices of θ and ϑ. Both families have their advantages and disadvantages. The first version (Equation (3.21)) should be used, if the clusters are expected to be heavy, while the other version (Equation (3.23)) handles clusters with inhomogeneous weights better. Note, that these extensions for $\vartheta > 0$ are not compatible extensions with respect to the unweighted case, since $\mathsf{perf}_w\,(\mathcal{C})$ and $\mathsf{perf}\,(\mathcal{C})$ have different values when evaluated on the same clustering on a undirected and unweighted graph.

3.5. Modularity and Significance

Another index that has recently gained a lot of attention is *modularity* [28]. The founding idea is to evaluate a clustering based on the fraction of intra-cluster edges and on an approximation of this fraction considering a certain random model. Although, modularity does not directly realize the intra-cluster density versus inter-cluster sparsity paradigm, an experimental evaluation confirms its relatedness. The standard definition is given in Equation (3.24).

$$\mathsf{q}\,(\mathcal{C}) = \sum_{C \in \mathcal{C}} \left(\frac{\omega(E(C))}{\omega(E)} - \frac{1}{4\omega(E)^2} \left(\sum_{v \in C} \sum_{\{v,w\} \in E} \omega(\{v,w\}) \right)^2 \right) \tag{3.24}$$

Extending the range of g to $\mathbb{R}$, the index can be expressed using the function f and g as given in Equations (3.25), where the maximum $f + g$ is trivially bounded by $\omega(E)$.

$$f\left(\mathcal{C}\right) := \omega(E(\mathcal{C})) \qquad \text{and} \qquad g\left(\mathcal{C}\right) := -\frac{1}{4\omega(E)}\left(\sum_{v\in C}\sum_{\{v,w\}\in E}\omega(\{v,w\})\right)^2 \tag{3.25}$$

Unfortunately, modularity can also attain negative values, which violates the original range of the images of $[0;1]$. Before showing in Lemma 3.15 that $-1/2$ is a lower bound for modularity, note that the following equality:

$$\mathsf{q}\left(\mathcal{C}\right) = \sum_{C\in\mathcal{C}}\left(\frac{\omega(E(C))}{\omega(E)} - \frac{1}{4\cdot\omega(E)^2}\left(2\cdot\omega(E(C)) + \sum_{C\neq C'\in\mathcal{C}}\omega(E(C,C'))\right)^2\right)$$

Lemma 3.15. *Let G be an undirected and unweighted graph and $\mathcal{C}\in\mathcal{A}\left(G\right)$. Then the following inequalities hold:*

$$-\frac{1}{2} \le q\left(\mathcal{C}\right) \le 1 \ . \tag{3.26}$$

Proof. Let $C\in\mathcal{C}$, m be the number of edges, $m_i = |E(C)|$ be the number of edges inside cluster C and $m_e = \sum_{C\neq C'\in\mathcal{C}}|E(C,C')|$ be the number of edges having exactly one end-node in C. Then the contribution of C is:

$$\frac{m_i}{m} - \left(\frac{m_i}{m} + \frac{m_e}{2m}\right)^2 \ .$$

This expression is strictly decreasing in m_e and, when varying m_i, the only maximum point is at $m_i = (m - m_e)/2$. Hence, the contribution of a cluster is minimized when m_i is 0 and m_e is as large as possible. Using the inequality $(a+b)^2 \ge a^2 + b^2$ for all non-negative numbers a and b, modularity has a minimum score for two clusters where all edges are inter-cluster edges. This proves the lower bound, the upper bound trivially given by the upper bound on $f + g$. □

Before considering special graph families such as cliques or cycles, we state two features that cannot occur in clusterings with maximum modularity, namely a node of degree one forming its own cluster and disconnected clusters.

Lemma 3.16. *A clustering with maximum modularity has no cluster that consists of a single node with degree one.*

Proof. Suppose for contradiction that there is a clustering $\mathcal{C}$ with a cluster $C_v = \{v\}$ and $\deg(v) = 1$. Consider a cluster C_u that contains the neighbor node u. Suppose there are a number of m_i intra-cluster edges in C_u and m_e inter-cluster edges connecting C_u to other clusters. Together these clusters add

$$\frac{m_i}{m} - \frac{(2m_i + m_e)^2 + 1}{4m^2}$$

to $\mathsf{q}\left(\mathcal{C}\right)$. Merging C_v with C_u results in a new contribution of

$$\frac{m_i + 1}{m} - \frac{(2m_i + m_e + 1)^2}{4m^2}$$

The merge yields an increase of

$$\frac{1}{m} - \frac{2m_i + m_e}{2m^2} > 0$$

in the modularity, because $m_i + m_e \leq m$ and $m_e \geq 1$. This proves the statement. □

Lemma 3.17. *There is always a clustering with maximum modularity, in which any cluster consists of a connected subgraph. If the graph has no nodes of degree 0, than every clustering with maximum modularity consists of connected clusters.*

Proof. Consider for contradiction a clustering $\mathcal{C}$ with a cluster C of m_i intra- and m_e inter-cluster edges that consists of a set of more than one connected subgraph. All components that consists of nodes of degree 0 can be put in individual clusters without changing the modularity, thus we assume that all components have nodes of degree at least 1. The subgraphs in C do not have to be disconnected in G, they are only disconnected when we consider the edges $E(C)$. Cluster C adds

$$\frac{m_i}{m} - \frac{(2m_i + m_e)^2}{4m^2}$$

to $\mathsf{q}(\mathcal{C})$. Now suppose we create a new clustering $\mathcal{C}'$ by splitting C into two new clusters. One cluster C_v consists of the component of one component, i.e. all vertices, which can be reached from a vertex v with a path running only through vertices of C, i.e. $C_v = \bigcup_{i=0}^{\infty} C_v^i$, where $C_v^i = \{w \mid \exists (w, w_i) \in E(C) \text{ with } w_i \in C_v^{i-1}\}$ and $C_v^0 = \{v\}$. The other nonempty cluster is given by $C - C_v$. Let C_v have m_i^v intra- and m_e^v inter-cluster edges. Together the new clusters add

$$\frac{m_i}{m} - \frac{(2m_i^v + m_e^v)^2 + (2(m - m_i^v) + m - m_e^v)^2}{4m^2} \tag{3.27}$$

to $\mathsf{q}(\mathcal{C}')$. For $a, b \geq 0$ obviously $a^2 + b^2 \leq (a+b)^2$. Thus modularity of clustering $\mathcal{C}'$ exceeds $\mathsf{q}(\mathcal{C})$.

If G has no isolated nodes, both terms $(2m_i^v + m_e^v)$ and $(2(m - m_i^v) + m - m_e^v)$ are strictly positive, thus $\mathsf{q}(\mathcal{C}') > \mathsf{q}(\mathcal{C})$. □

3.5.1. $\mathcal{NP}$-Completeness

Before characterizing the clusterings with maximum modularity for certain regular graph families, we briefly show that optimizing modularity is $\mathcal{NP}$-complete. The whole proof can be found in [20, 21].

We consider the decision problem associated with optimizing modularity: Given a graph G and a number K, is there a clustering $\mathcal{C}$ of G with $\mathsf{q}(\mathcal{C}) \geq K$. Although, the number K could be real, note that $4m^2 \cdot \mathsf{q}(\mathcal{C})$ is an integer for every clustering $\mathcal{C}$ and is polynomially bounded in the size of G, where m denotes the number of edges in the graph. The proof uses a transformation to the problem 3–Partition which is defined as follows: Given $3k$ positive integers $a_1, \ldots, a_{3k}$ such that the sum $\sum_i a_i = kb$ and $b/4 < a_i < b/2$ for an integer b and for all $1 \leq i \leq 3k$, is there a partition of these numbers into k sets, such that the numbers in each set sum up to b. Note that 3–Partition is strongly $\mathcal{NP}$-complete [52], i. e., the problem remains $\mathcal{NP}$-complete even if the input is represented in unary coding.

In order to show the hardness of optimizing modularity, we show that given an instance of 3–Partition, we can create an instance of the decision problem of maximizing modularity, such that the instance of 3–Partition is solvable if and only if the decision problem is solvable. More precisely, let $A := \{a_1, \dots, a_{3k}\}$ be an instance of 3–Partition, then we define a graph $G(A) := (V(A), E(A))$ in the following way:

$$\begin{aligned} V(A) &:= \{1, \dots, 3k\} \uplus \biguplus_{i=1}^{k} \left\{ c_j^{(i)} \,\middle|\, 1 \le j \le \sum_{\ell=1}^{3k} a_\ell \right\} \\ E(A) &:= \left\{ \left\{ j, c_{j'}^{(i)} \right\} \,\middle|\, 1 \le i \le k; 1 \le j \le 3k; \sum_{\ell=1}^{j-1} a_\ell < j' \le \sum_{\ell=1}^{j} a_\ell \right\} \\ &\qquad \uplus \left\{ \left\{ c_j^{(i)}, c_{j'}^{(i)} \right\} \,\middle|\, 1 \le i \le k; 1 \le j \ne j' \le \sum_{\ell=1}^{3k} a_\ell \right\} \end{aligned}$$

An illustrating example for the instance $A = \{2, 2, 2, 2, 3, 3\}$ is given in Figure 3.5. Informally speaking, the graph $G(A)$ consists of a node for each integer a_i and k

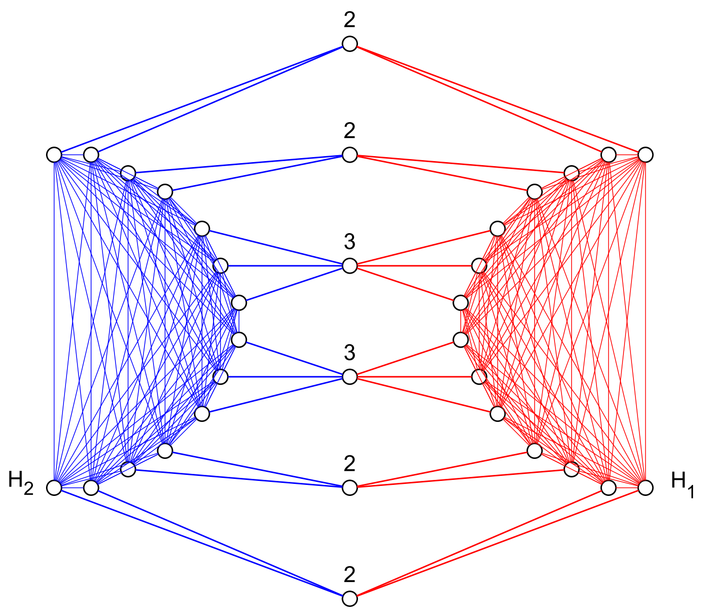


Figure 3.5.: An example graph $G(A)$ for the instance $A = \{2, 2, 2, 2, 3, 3\}$ of 3–Partition. Node labels indicate the corresponding value of $a_i \in A$.

complete subgraphs which individual size is defined by the total sum of the a_i. Furthermore, each representative node of an a_i is connected to a_i (unique) nodes in each clique. Before defining the necessary bound $K(A)$ for the decision problem of modularity, we briefly summarize characteristics of the clustering with maximum modularity in such a graph $G(A)$.

The first two important issues are that none of the complete subgraphs are split over several clusters and no pair is of them is contain in the same cluster. In other words, each clustering with maximum modularity has at least k clusters and k of them contain each one of the complete subgraphs completely. The proof is straightforward, but very technical; basically, one can improve the modularity of a clustering violating the above requirements by either shifting nodes such that the cliques are no long split or by splitting the cluster containing more than one complete subgraph. The final an most interesting characteristic is that each node corresponding to one of the a_i is contained in a cluster along with one of the k big cliques. This feature ensures that we can read of the required partition for 3–Partition directly from the clustering with maximum modularity. Furthermore by setting an appropriate threshold $K(A)$, we can decide if the clusters are balanced and thus if the instance of 3–Partition is solvable or not. The proof of the characteristic uses the fact, that clusterings with maximum modularity have no disconnected clusters (Lemma 3.17); more precisely, a node j has clique nodes $c_{j'}^{(i)}$ and thus, one can improve the modularity, if it is not contained in a cluster containing a clique.

Summarizing, given an instance $A := \{a_1, \ldots, a_k\}$ of 3–Partition, we can decide if A is solvable or not, by deciding the corresponding instance of the modularity problem using $(G(A), K(A) := (k-1)(a-1)/(k(a+1)))$, where $a := \sum_i a_i$.

3.5.2. Optimality Results for Cliques and Cycles

In the following, we present some optimality results for cliques and cycles. Please note, that modularity can be simplified as given in Corollary 3.18 for general d-regular graphs.

Corollary 3.18. *Let $G = (V, E)$ be an unweighted d-regular graph and $\mathcal{C} \in \mathcal{A}(G)$ a clustering with $\mathcal{C} = \{C_1, \ldots, C_k\}$. Then the following equality holds:*

$$q(\mathcal{C}) = \operatorname{cov}(\mathcal{C}) - \frac{1}{|V|^2} \sum_{i=1}^{k} |C_i|^2 \quad .$$

The correctness of the corollary can be read off the definition given in Equation (3.24) and the fact that $|E| = d|V|/2$. Thus modularity only depends on cluster size and coverage.

Cliques

We first deal with the case of the complete graphs. The Corollary 3.19 provides a simplified formulation for modularity. From this rewriting, one directly reads of the clustering with maximum modularity.

Corollary 3.19. *Let K_n be a complete graph on n nodes and $\mathcal{C} := \{C_1, \ldots, C_k\} \in$*

$\mathcal{A}(K_n)$. Then the following equality holds:

$$q(\mathcal{C}) = -\frac{1}{n-1} + \frac{1}{n^2(n-1)} \sum_{i=1}^{k} |C_i|^2 \quad . \tag{3.28}$$

Proof. Coverage of $\mathcal{C}$ can be expressed in terms of cluster sizes as follows:

$$\begin{aligned}
|E(\mathcal{C})| &= \binom{n}{2} - \sum_{i=1}^{k} \prod_{j>i} |C_i| \cdot |C_j| = \binom{n}{2} - \frac{1}{2} \sum_{i=1}^{k} \prod_{j \neq i} |C_i| \cdot |C_j| \\
&= \binom{n}{2} - \frac{1}{2} \sum_{i=1}^{k} |C_i| \cdot \sum_{j \neq i} |C_j| = \binom{n}{2} - \frac{1}{2} \sum_{i=1}^{k} |C_i| \cdot (n - |C_i|) \\
&= \binom{n}{2} - \frac{1}{2} \left(n^2 - \sum_{i=1}^{k} |C_i|^2 \right) = -\frac{n}{2} + \frac{1}{2} \sum_{i=1}^{k} |C_i|^2 \quad .
\end{aligned}$$

Thus, we obtain

$$\begin{aligned}
\mathsf{q}(\mathcal{C}) &= -\frac{1}{n-1} + \frac{1}{n(n-1)} \sum_{i=1}^{k} |C_i|^2 - \frac{1}{n^2} \sum_{i=1}^{k} |C_i|^2 \\
&= -\frac{1}{n-1} + \frac{1}{n^2 \cdot (n-1)} \sum_{i=1}^{k} |C_i|^2 \quad ,
\end{aligned}$$

which proves given equation. □

Thus maximizing modularity is equivalent to maximizing the squares of cluster sizes. Using the general inequality $(a+b)^2 \geq a^2 + b^2$ for non-negative real numbers, the clustering with maximum modularity is the 1–clustering. More precisely:

Theorem 3.20. *Let ℓ and n be integers, $K_{\ell n}$ be the complete graph on $\ell \cdot n$ nodes and $\mathcal{C}$ a clustering such that each clusters contains exactly n elements. Then the following equality holds:*

$$q(\mathcal{C}) = \left(-1 + \frac{1}{\ell} \right) \cdot \frac{1}{\ell n - 1} \quad .$$

For fixed $\ell > 1$ and as n tends to infinity, modularity is always strictly negative, but tends to zero. Only for $\ell = 1$ modularity is zero and thus is the global maximum.

Simple Cycles

A simple cycle is a connected 2-regular graph. In the following, we prove that clusterings with maximum modularity are balanced with respect to number and size of clusters.

Corollary 3.21. *Let C_n be a simple cycle with n nodes and $\mathcal{C} = \{C_1, \ldots, C_k\}$ a clustering such that all clusters are connected. Then the following equality holds:*

$$q(\mathcal{C}) = \frac{n-k}{n} - \frac{1}{n^2}\sum_{i=1}^{k} |C_i|^2 \quad . \tag{3.29}$$

As proven in Lemma 3.17, the clustering with maximum modularity have only connected clusters, thus, we can safely use Equation (3.29). As preparation, we first study the case of fixed number k of clusters and characterize the cluster sizes that have maximum modularity.

Proposition 3.22. *Let k and n be integers, the set $D^{(k)} \subset \mathbb{N}^k$ defined as*

$$D^{(k)} := \left\{ (x_1, \ldots, x_k) \in \mathbb{N}^k \,\middle|\, \sum_{i=1}^{k} x_i = n \right\} \quad ,$$

and the function $F\colon D^{(k)} \to \mathbb{R}$ defined as

$$F(x) := \frac{k}{n} + \frac{1}{n^2}\sum_{i=1}^{k} x_i^2 \qquad \text{for } x \in D^{(k)} \quad .$$

Then F has a global minimum at

$$x_{\text{opt}} = \Big(\underbrace{\left\lfloor \frac{n}{k} \right\rfloor, \ldots, \left\lfloor \frac{n}{k} \right\rfloor}_{k-r \text{ times}}, \underbrace{\left\lceil \frac{n}{k} \right\rceil, \ldots, \left\lceil \frac{n}{k} \right\rceil}_{r \text{ times}} \Big) \quad ,$$

where $0 \le r < k$ and $r \equiv n \mod k$.

Proof. Since k and n are given, minimizing F is equivalent to minimizing $\sum_i x_i^2$. Thus let us rewrite this term:

$$\begin{aligned}
\sum_{i=1}^{k} \left(x_i - \frac{n}{k}\right)^2 &= \sum_{i=1}^{k} x_i^2 - 2\frac{n}{k}\sum_{i=1}^{k} x_i + k \cdot \left(\frac{n}{k}\right)^2 \\
&= \sum_{i=1}^{k} x_i^2 - 2\frac{n^2}{k} + \frac{n^2}{k} \\
\iff \qquad \sum_{i=1}^{k} x_i^2 &= \underbrace{\sum_{i=1}^{k} \left(x_i - \frac{n}{k}\right)^2}_{=:h(x)} + \frac{n^2}{k}
\end{aligned}$$

Thus minimizing F is equivalent to minimizing h. If r is 0, then $h(x_{\text{opt}}) = 0$. For every other vector y the function h is strict positive, since at least one summand is positive. Thus x_{opt} is a global optimum.

Let $r > 0$. First, we show that every vector $x \in D^{(k)}$ that is close to $(\frac{n}{k}, \ldots, \frac{n}{k})$ has (in principle) the form of x_{opt}. Let $x \in D \cap [\lfloor \frac{n}{k} \rfloor, \lceil \frac{n}{k} \rceil]^k$, then there are $k - r$ entries that have value $\lfloor \frac{n}{k} \rfloor$ and the remaining r entries have value $\lceil \frac{n}{k} \rceil$.

Subproof: Since $x_i \geq \left\lfloor \frac{n}{k} \right\rfloor$, we obtain

$$n = \sum_i x_i \geq k \cdot \left\lfloor \frac{n}{k} \right\rfloor = k \cdot \frac{n-r}{k} = n - r \enspace ,$$

thus r entries have to have value $\left\lceil \frac{n}{k} \right\rceil$. ■

Any 'shift of one unit' between two variables having the same value, increases the corresponding cost: Let $\varepsilon := \left\lceil \frac{n}{k} \right\rceil - \frac{n}{k}$ and $x_i = x_j = \left\lceil \frac{n}{k} \right\rceil$. Replacing x_i with $\left\lfloor \frac{n}{k} \right\rfloor$ and x_j with $\left\lceil \frac{n}{k} \right\rceil + 1$, causes an increase of h by $5 + 2\varepsilon > 0$. Similarly, in the case of $x_i = x_j = \left\lfloor \frac{n}{k} \right\rfloor$ and the reassignment $x_i = \left\lceil \frac{n}{k} \right\rceil$ and $x_j = \left\lfloor \frac{n}{k} \right\rfloor - 1$, causes an increase of h by $2 > 0$.

Finally, we show that any vector of $D^{(k)}$ can be reach from x_{opt} by 'shifting one unit' between variables. Let $x \in D^{(k)}$ and with loss of generality, we assume that $x_i \leq x_{i+1}$ for all i. We define a sequence of elements in $D^{(k)}$ as follows:

1. $x^{(0)} := x_{\text{opt}}$
2. if $x^{(i)} \neq x$, define $x^{(i+1)}$ as follows

$$x_j^{(i+1)} := \begin{cases} x_j^{(i)} - 1 & \text{, if } j = \min\left\{\ell \mid x_\ell^{(i)} > x_\ell\right\} =: L \\ x_j^{(i)} + 1 & \text{, if } j = \max\left\{\ell \mid x_\ell^{(i)} < x_\ell\right\} =: L' \\ x_j^{(i)} & \text{, otherwise} \end{cases}$$

Note that all obtained vectors $x^{(i)}$ are elements of $D^{(k)}$ and meet the condition of $x_j^{(i)} \leq x_{j+1}^{(i)}$. Furthermore, we gain the following formula for the cost:

$$\sum_j \left(x_j^{(i+1)}\right)^2 = \sum_j \left(x_j^{(i)}\right)^2 + 2\left(x_{L'}^{(i)} - x_L^{(i)} + 1\right) \enspace .$$

Since $L < L'$, one obtains $x_{L'}^{(i)} \geq x_L^{(i)}$. Thus x_{opt} is a global optimum in $D^{(k)}$. □

Due to the special structure of simple cycles, we can swap neighboring clusters without changing the modularity. Thus, we can safely assume that clusters are sorted according to their sizes starting with the smallest element. Thus x_{opt} is the only optimum. Evaluating F at x_{opt} leads to a term that depends only on k and n. Thus, we can characterize the clusterings with maximum modularity only with respect to the number of clusters. A summarizing lemma is Lemma 3.23.

Lemma 3.23. *Let C_n be a simple cycle with n nodes, $h\colon [1, \ldots, n] \to \mathbb{R}$ a function defined as*

$$h(x) := x \cdot n + n + \left\lfloor \frac{n}{x} \right\rfloor \left(2n - x \cdot \left(1 + \left\lfloor \frac{n}{x} \right\rfloor\right)\right) \enspace ,$$

and k_{opt} be the global optimum of h. Then every clustering of C_n with maximum modularity has k_{opt} clusters.

Proof. Note, that $h(k) = n^2 \cdot F(x_{\text{opt}})$, where F is the function of Proposition 3.22 with the given k. Consider first the following equations:

$$\begin{aligned}
\sum_{i=1}^{k} (x_{\text{opt}})_i^2 &= (k-r) \cdot \left\lfloor \frac{n}{k} \right\rfloor^2 + r \cdot \left\lceil \frac{n}{k} \right\rceil^2 \\
&= (k-r)\frac{(n-r)^2}{k^2} + r\left(\frac{(n-r)}{k} + 1\right)^2 \\
&= \frac{(n-r)^2}{k} - r\frac{(n-r)^2}{k^2} + r\frac{(n-r)^2}{k^2} + 2 \cdot r\frac{(n-r)}{k} + r \\
&= \frac{n-r}{k}\left((n-r) + 2r\right) + r = \frac{n^2 - r^2}{k} + r \\
&= \frac{1}{k}\left(n^2 - \left(n - \left\lfloor \frac{n}{k} \right\rfloor k\right)^2\right) + n - \left\lfloor \frac{n}{k} \right\rfloor k \\
&= \frac{1}{k}\left(n^2 - n^2 + 2nk\left\lfloor \frac{n}{k} \right\rfloor - k^2 \left\lfloor \frac{n}{k} \right\rfloor^2\right) + n - \left\lfloor \frac{n}{k} \right\rfloor k \\
&= 2n\left\lfloor \frac{n}{k} \right\rfloor - k\left\lfloor \frac{n}{k} \right\rfloor^2 + n - \left\lfloor \frac{n}{k} \right\rfloor k \\
&= n + \left\lfloor \frac{n}{k} \right\rfloor \left(2n - k\left(\left\lfloor \frac{n}{k} \right\rfloor + 1\right)\right)
\end{aligned}$$

Since maximizing the modularity is equivalent to minimize the expression $k/n + 1/n^2 \sum_i x_i^2$ for $(x_i) \in \bigcup_{j=1}^{n} D^{(j)}$. Note that every vector (x_i) can be realized as clustering with connected clusters. Since we have characterized the global minima for fixed k, it is sufficient to find the global minima by varying k. □

In the following, we show some properties of h regarding monotonicity which will help us to restrict the definition range containing global minima. But first note the following continuous bounds given in

Lemma 3.24. *Given the function h of Lemma 3.23, then h is bounded by*

$$kn + \frac{n^2}{k} \leq h(k) \leq kn + \frac{n^2}{k} + \frac{k}{4} \quad . \tag{3.30}$$

Proof. Let ε_k be defined as $n/k - \lfloor n/k \rfloor$ (≥ 0). The formula of h can be rewritten as follows:

$$\begin{aligned}
h(k) &= kn + n + \left\lfloor \frac{n}{k} \right\rfloor \left(2n - \left(1 + \left\lfloor \frac{n}{k} \right\rfloor\right) k\right) \\
&= kn + n + \left(\frac{n}{k} - \varepsilon_k\right)\left(2n - \left(1 + \frac{n}{k} - \varepsilon_k\right) k\right) \\
&= kn + n + \frac{2n^2}{k} - (1-\varepsilon_k)n - \frac{n^2}{k} - 2n\varepsilon_k + (1-\varepsilon_k)k\varepsilon_k + n\varepsilon_k \\
&= kn + \frac{n^2}{k} + (1-\varepsilon_k)\varepsilon_k k \quad .
\end{aligned}$$

Replacing the term $(1-\varepsilon_k)\varepsilon_k k$ by a lower (upper) bound of 0 $(k/4)$ proves the given statements. □

Lemma 3.25. *Given the function h of Lemma 3.23, then h is monotonically increasing for $k \geq 1/2 + \sqrt{1/4+n}$ and monotonically decreasing for $k \leq n/\sqrt{n+\sqrt{n}} - 1$.*

Proof. For the first part, it is sufficient to show that $h(k) \leq h(k+1)$ for every suitable k.

$$\begin{aligned} h(k+1) - h(k) &= (k+1)n + n + \left\lfloor \frac{n}{k+1} \right\rfloor \left(2n - \left(1 + \left\lfloor \frac{n}{k+1} \right\rfloor\right)(k+1)\right) \\ &\quad -kn - n - \left\lfloor \frac{n}{k} \right\rfloor \left(2n - \left(1 + \left\lfloor \frac{n}{k} \right\rfloor\right) k\right) \\ &= n + 2n\left(\left\lfloor \frac{n}{k+1} \right\rfloor - \left\lfloor \frac{n}{k} \right\rfloor\right) - \left(1 + \left\lfloor \frac{n}{k+1} \right\rfloor\right)\left\lfloor \frac{n}{k+1} \right\rfloor \\ &\quad +k\left(\left(1 + \left\lfloor \frac{n}{k} \right\rfloor\right)\left\lfloor \frac{n}{k} \right\rfloor - \left(1 + \left\lfloor \frac{n}{k+1} \right\rfloor\right)\left\lfloor \frac{n}{k+1} \right\rfloor\right) \end{aligned}$$

Since $\lfloor\cdot\rfloor$ is discrete and $|\lfloor x \rfloor - \lfloor x-1 \rfloor| \leq 1$, one obtains:

$$h(k+1) - h(k) = \begin{cases} n - \left\lfloor \frac{n}{k} \right\rfloor^2 - \left\lfloor \frac{n}{k} \right\rfloor & \text{, if } \left\lfloor \frac{n}{k} \right\rfloor = \left\lfloor \frac{n}{k-1} \right\rfloor \\ 3n - \left\lfloor \frac{n}{k} \right\rfloor^2 - \left\lfloor \frac{n}{k} \right\rfloor + 2k\left\lfloor \frac{n}{k} \right\rfloor & \text{, otherwise} \end{cases} \tag{3.31}$$

Since $3n - \lfloor n/k \rfloor^2 - \lfloor n/k \rfloor + 2k\lfloor n/k \rfloor > n - \lfloor n/k \rfloor^2 - \lfloor n/k \rfloor$, it is sufficient to show that $n - \lfloor n/k \rfloor^2 - \lfloor n/k \rfloor \geq 0$. This inequality is fulfilled if $n - (n/k)^2 - n/k \geq 0$. Solving the quadratic equations leads to $k \geq 1/2 + \sqrt{1/4+n}$.

For the second part, it is sufficient to show that

$$kn + \frac{n^2}{k} - (k+1)n - \frac{n^2}{k+1} - \frac{k+1}{4} \geq 0 \ , \tag{3.32}$$

since this implies that the upper bound of $h(k+1)$ is smaller than (the lower bound of) $h(k)$. One can rewrite the left side of Inequality (3.32) as:

$$kn + \frac{n^2}{k} - (k+1)n - \frac{n^2}{k+1} - \frac{k+1}{4} = -n + \frac{n^2}{k(k+1)} - \frac{k+1}{4} \ .$$

Since $h(k) - h(k+1)$ is monotonically decreasing for $0 \leq k \leq \sqrt{n}$, it is sufficient to show that $h(k) - h(k+1)$ is non-negative for the maximum value of k. We show that the lower bound $h_-(k) := -n + n^2/(k+1)^2 - (k+1)/4$ is non-negative.

$$\begin{aligned} h_-\left(\frac{n}{\sqrt{n+\sqrt{n}}} - 1\right) &= -n - \frac{n}{4\sqrt{n+\sqrt{n}}} + \frac{n^2(n+\sqrt{n})}{n^2} \\ &= \sqrt{n} - \underbrace{\frac{n}{4\sqrt{n+\sqrt{n}}}}_{\leq \frac{1}{4}\sqrt{n}} \geq 0 \qquad \square \end{aligned}$$

By combining these partial results we obtain the final characterization for clusterings with maximum modularity for simple cycles.

Theorem 3.26. *Let n be an integer and C_n a simple cycle with n nodes. Then a clustering $\mathcal{C}$ with maximum modularity has k cluster of almost equal size, where*

$$k \in \left[\frac{n}{\sqrt{n+\sqrt{n}}} - 1, \frac{1}{2} + \sqrt{\frac{1}{4} + n}\right] .$$

Furthermore, there are only 3 possible values for k for sufficiently large n.

Proof. As a result of Lemma 3.25, the number of clusters k can only be contained in the given interval, since outside the function h (of Lemma 3.23) is either monotonically increasing or decreasing.The length of the interval is less than

$$\frac{1}{2} + \underbrace{\sqrt{\frac{1}{4} + n} - \frac{n}{\sqrt{n+\sqrt{n}}}}_{=:\ell(n)} + 1 .$$

The function $\ell(n)$ can be rewritten as follows:

$$\begin{aligned} \ell(n) &= \frac{\sqrt{\left(\frac{1}{4}+n\right)\left(\sqrt{n+\sqrt{n}}\right)} - n}{\sqrt{n+\sqrt{n}}} \\ &\leq \frac{\left(n + \frac{1+\varepsilon}{2}\sqrt{n}\right) - n}{\sqrt{n+\sqrt{n}}} \qquad (3.33) \\ &\leq \frac{1+\varepsilon}{2}\sqrt{\frac{n}{n+\sqrt{n}}} , \end{aligned}$$

for every positive ε. Inequality (3.33) is due to the fact that

$$\begin{aligned} \left(\frac{1}{4}+n\right)\left(\sqrt{n+\sqrt{n}}\right) &\leq n^2 + n\sqrt{n} + \frac{1}{4}\left(n+\sqrt{n}\right) \\ &\leq n^2 + 2\frac{1+\varepsilon}{2}n\sqrt{n} + \frac{(1+\varepsilon)^2}{4}n \\ &= \left(n + \frac{1+\varepsilon}{2}\sqrt{n}\right)^2 , \end{aligned}$$

for sufficiently large n. □

Figure 3.6 gives a brief impression of the tightness of the bounds used in Theorem 3.26.

3.6. Other Indices

Clearly these indices are only a small fraction of the whole spectrum used to formalize and evaluate clusterings. However, they clarify different concepts very well. This part covers some historical indices that have been used for clustering.

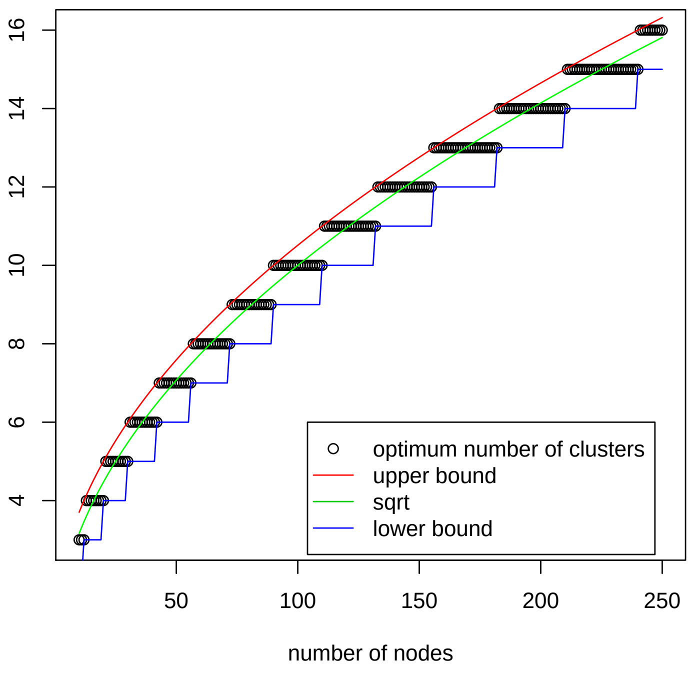


Figure 3.6.: The diagram shows for each C_n the size of one clustering that has maximum modularity. The green line indicates the square-root function, while the other two are lower and upper bound.

Clustering was originally applied to entities embedded in metric spaces or vector spaces with a norm function. Many indices have been developed that use essential structural properties of these spaces, e. g., the possibility to calculate a barycenter or geometric coverings. Over time, some indices have been transferred to graph clustering as well. Because these spaces usually provide information for all pair, the first problem is to estimate the similarity of nodes that are not connected with an edge. This is often solved using shortest path techniques that respect all information, like weight or direction, or only partial information, or none. The most common measures resulting are: diameter, edge weight variance, and average distance within the clusters. In contrast to the previous indices, these measures do not primarily focus on the intra-cluster density versus inter-cluster sparsity paradigm. Most of them even ignore the inter-cluster structure completely. Another difference is that these indices usually rate each cluster individually, regardless of its position within the graph. The resulting distribution is then rated with respect to the average or the worst case. Thus, a density measure[2] π can be transformed into an index by applying π on all cluster-induced subgraphs and rating the resulting distribution of

[2]greater values imply larger density

values, e. g., via minimum, maximum, average, or mean:

$$\begin{aligned} \text{worst case:} \quad & \min_i \{\pi(G[C_1]), \ldots, \pi(G[C_k])\} \\ \text{average case:} \quad & \frac{1}{k} \sum_i \pi(G[C_i]) \\ \text{best case:} \quad & \max_i \{\pi(G[C_1]), \ldots, \pi(G[C_k])\} \end{aligned}$$

Their popularity is partially based on the easy computation in metric or normed vector spaces, where the distance (inverse similarity) of (all) pairs is defined by their distance within the given space. The other reason is their use for some greedy approaches.

Chapter 4

Comparison of Clusterings

Several clustering techniques and applications are based on the formalization of 'natural groups' given by indices. Due to intuitive concept of clusterings and the large variety of existing quality measures, many different algorithms have been developed. Although most behave very well with respect to certain indices, all have their weaknesses and drawbacks. Several tasks for designing and engineering new algorithms require the comparison of clusterings. Such issues are robustness, i. e., the sensitivity with respect to small perturbations, comparison of exact techniques and heuristics, and benchmarks where an 'optimal' clustering is known in advanced.

In the following, we present a short overview of comparison methods based on the summary [103, 31]. The experimental evaluation is given in Chapter 6. Standard comparators are founded on the lattice structure of the set of all clusterings (for a fixed graph), thus they are independent of the quality measures. However, counter-intuitive situations can arise; an example is given in Figure 4.1. The exam-

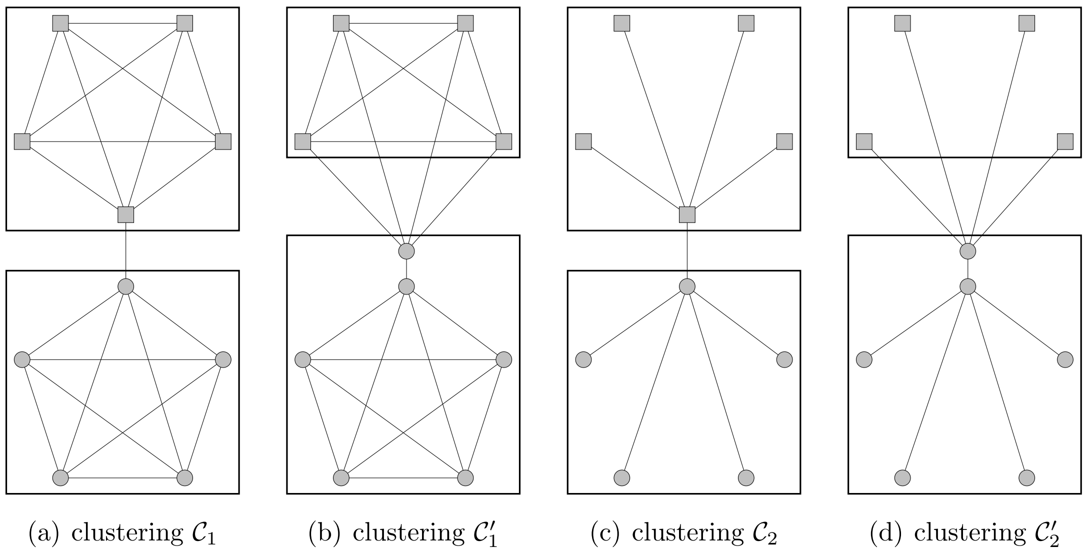

(a) clustering $\mathcal{C}_1$ (b) clustering $\mathcal{C}'_1$ (c) clustering $\mathcal{C}_2$ (d) clustering $\mathcal{C}'_2$

Figure 4.1.: Counter-intuitive example where a lattice-based comparator cannot distinguish the two cases.

ple consists of two cliques of size five connected by an edge and two stars of with four leaves where the two center nodes are connected. In both cases the clustering $\mathcal{C}_i$ groups the cliques (stars) and corresponds to the favored partition, while $\mathcal{C}'_i$ results from $\mathcal{C}_i$ by moving one incident node of the inter-cluster edge to the other cluster. Although, clustering $\mathcal{C}'_1$ is suboptimal, it is still interpretable as clustering. Since the upper cluster of $\mathcal{C}'_2$ is not only disconnected by does not contain any edges at all, clustering $\mathcal{C}_2$ is meaningless. As lattice-based comparators are independent of the edge set, the distance between clustering $\mathcal{C}_1$ and $\mathcal{C}'_1$ is the same as the distance between $\mathcal{C}_2$ and $\mathcal{C}'_2$.

Although independent concepts for quality and comparison are beneficial for engineering and evaluating, counter-intuitive situations easily out weight such advantages. Thus, we present some new approaches, introduced in [31, 33], that incorporate quality aspects as well. In contrast to the general framework for quality measures, no comprehensive framework has emerged yet; still the comparators can be roughly classified according to their origins. Since our goal is to obtain a uniform framework that can be easily integrated into a benchmark environment, we will present all comparators as distance measures, although they might have been introduced as similarity measure in the literature. Note that the given overview is a first and preliminary step towards the design and engineering of comparators for graph clusterings. As the performed evaluation reveals, all graph-structural comparators suffer certain drawbacks similar to the quality indices.

4.1. Lattice-Based Approach

Most comparators defined in literature are founded on the lattice structure for partitions. The set of all clusterings (for a fixed graph) forms a lattice-ordered set, where $\inf(\mathcal{C}_1, \mathcal{C}_2) := \mathcal{C}_1 \wedge \mathcal{C}_2$ and $\sup(\mathcal{C}_1, \mathcal{C}_2) := \mathcal{C}$ where $\mathcal{C}$ is the smallest coarsening of $\mathcal{C}_1$ and $\mathcal{C}_2$. Thus $\mathcal{A}(G)$ of a graph G forms a lattice. Many lattice-based comparators are solely based on counting pairs or matchings. In the following, we give a short summary of existing measures; for this presentation let $G = (V, E)$ be an undirected and unweighted graphs with two clusterings $\mathcal{C}_1 := \{C_1, \ldots, C_k\}$ and $\mathcal{C}_2 := \{C'_1, \ldots, C'_\ell\}$.

4.1.1. Pair Comparators

Similar to the quality index performance the set of all pairs of nodes is partitioned into four groups:

$$S_{ab} := \left\{ \{u, v\} \in \binom{V}{2} \middle| [u \sim_{\mathcal{C}_1} v] = a \wedge [u \sim_{\mathcal{C}_2} v] = b \right\} \quad , \tag{4.1}$$

for $a, b \in \{0, 1\}$. As short-cut for the sizes of the sets, we define $n_{ab} := |S_{ab}|$. Lemma 4.1 shows the relation between the sizes and the confusion matrix.

Lemma 4.1. *Let $M = (m_{ij})$ be the confusion matrix of $\mathcal{C}_1$ and $\mathcal{C}_2$. Then the following equations holds:*

$$n_{11} = \frac{1}{2}\left(\sum_{i=1}^{k}\sum_{j=1}^{\ell} m_{ij}^2 - |V|\right) \tag{4.2}$$

$$n_{10} = \frac{1}{2}\left(\sum_{i=1}^{k}|C_i|^2 - \sum_{i=1}^{k}\sum_{j=1}^{\ell} m_{ij}^2\right) \tag{4.3}$$

$$n_{01} = \frac{1}{2}\left(\sum_{j=1}^{\ell}|C'_j|^2 - \sum_{i=1}^{k}\sum_{j=1}^{\ell} m_{ij}^2\right) \quad . \tag{4.4}$$

Proof. All pairs of nodes that are in some clusters (with respect to both clusterings) are those located in the same cluster with respect to $\mathcal{C}_1 \wedge \mathcal{C}_2$. Thus, we obtain the following equation:

$$n_{11} = \sum_{i,j}\binom{m_{ij}}{2} = \frac{1}{2}\sum_{ij}\left(m_{ij}^2 - m_{ij}\right) \quad .$$

Since $\mathcal{C}_1 \wedge \mathcal{C}_2$ is a clustering of G, it has $|V|$ nodes and thus $\sum_{ij} m_{ij} = |V|$. For the other equations, note the symmetry of n_{10} and n_{01} and that $n_1 := \sum_{i=1}^{k}\binom{|C_i|}{2}$ is the number of pairs within clusters (with respect to $\mathcal{C}_1$). Thus equation $n_{10} = n_1 - n_{11}$ holds. Using the Equation (4.2) and substituting n_{11} results in Equation (4.3). $\square$

The comparators that corresponds to performance are *Rand* R and *Adjusted Rand* R_{adj} which are defined as:

$$\mathsf{R}(\mathcal{C}_1, \mathcal{C}_2) := 1 - \frac{n_{11} + n_{00}}{\binom{|V|}{2}}$$

$$\mathsf{R}_{\text{adj}}(\mathcal{C}_1, \mathcal{C}_2) := 1 - \frac{n_{11} - t_3}{\frac{1}{2}(t_1 + t_2) - t_3} \quad , \text{ with}$$

$$t_1 := \sum_{i=1}^{k}\binom{|C_i|}{2}, t_2 := \sum_{k=1}^{\ell}\binom{|C'_j|}{2}, \text{ and } t_3 := \frac{t_1 t_2}{\binom{|V|}{2}} \quad .$$

The founding idea of the Rand comparator is to count the number of pairs classified in the same way with respect to both clusterings. Rand has some disadvantages, if number of clusters is large. An attempt to counter-act this drawback is given in Adjusted Rand comparator, which normalizes Rand by the expected value of Rand under the null hypothesis of a hypergeometric distribution. A scaled version of Rand is also known as *Equivalence Mismatch Distance* or *Mirkin Metric*, its defined as in Formula (4.5).

$$\sum_{i=1}^{k}|C_i|^2 + \sum_{j=1}^{\ell}|C'_j|^2 - 2\sum_{i=1}^{k}\sum_{j=1}^{\ell} m_{ij}^2 \tag{4.5}$$

It corresponds to the Hamming distance when clusterings are interpreted as binary function on the set of all pairs of nodes. Two more variations arise from different applications: Information Retrieval, geology, and ecology. The first one is *Fowlkes-Mallows* which is defined as the geometric mean of precision and recall, defined in Equation (4.6).

$$\mathsf{FM}(\mathcal{C}_1, \mathcal{C}_2) := \begin{cases} 1 - \dfrac{n_{11}}{\sqrt{(n_{11}+n_{10})(n_{11}+n_{01})}} & \text{, if } |\mathcal{C}_1| \neq |V| \wedge |\mathcal{C}_2| \neq |V| \\ \dfrac{\big|\, |\mathcal{C}_1| - |\mathcal{C}_2| \,\big|}{|V|-1} & \text{, otherwise} \end{cases} \tag{4.6}$$

In Information Retrieval, *precision* is defined as the ratio of the number of retrieved relevant documents to the total number of retrieved documents whereas *recall* is the ratio of the number of retrieved documents to the total number of relevant documents. Similar to the Adjusted Rand, Fowlkes-Mallows corresponds to the deviation from the expected value under the null hypothesis of independent clusterings with fixed sizes. A similar version which uses a different normalization factor is the *Jaccard* comparator J defined in Equation (4.7).

$$\mathsf{J}(\mathcal{C}_1, \mathcal{C}_2) := \begin{cases} 1 - \dfrac{n_{11}}{n_{11}+n_{10}+n_{01}} & \text{, if } |\mathcal{C}_1| \neq |V| \wedge |\mathcal{C}_2| \neq |V| \\ 0 & \text{, otherwise} \end{cases} \tag{4.7}$$

4.1.2. Matching Comparators

Another group of lattice-based comparators employ matchings, i. e., they consider the relation between the common refinement $\mathcal{C}_1 \wedge \mathcal{C}_2$ and the two clusterings $\mathcal{C}_1$ and $\mathcal{C}_2$. A very intuitive comparator is the maximum matching of the two clusterings. Let $|\mathcal{C}_1| \geq |\mathcal{C}_2|$ then one considers surjective mapping $m\colon \mathcal{C}_1 \to \mathcal{C}_2$ that meet the restriction that each cluster of $\mathcal{C}_1$ has a non-empty intersection with its image. The set of all these function defines the set $\mathrm{match}(\mathcal{C}_1, \mathcal{C}_2)$. The formula of the comparator is given in (4.8).

$$1 - \frac{1}{|V|} \cdot \max_{m \in \mathrm{match}(\mathcal{C}_1, \mathcal{C}_2)} \sum_{i=1}^{k} |C_i \cap m(C_i)| \tag{4.8}$$

Since the *Maximum Matching Comparator* can potentially ignore many clusters, it is almost never used. Instead the versions introduced by Meila and Heckerman or van Dongen are considered. The *Meila-Heckerman* comparator considers the individual best match for a cluster; its definition is given in Equation (4.9).

$$\mathsf{MH}(\mathcal{C}_1, \mathcal{C}_2) := 1 - \frac{1}{|V|} \sum_{i=1}^{k} \max_{j \in \{1,\dots,\ell\}} m_{ij} \tag{4.9}$$

Similar to the maximum matching, it is asymmetric. Its symmetric counterpart, introduced by van Dongen, is defined as in Equation (4.10).

$$\mathsf{D}(\mathcal{C}_1, \mathcal{C}_2) := 1 - \frac{1}{2\,|V|} \left(\sum_{i=1}^{k} \max_{j \in \{1,\dots,\ell\}} m_{ij} + \sum_{j=1}^{\ell} \max_{i \in \{1,\dots,k\}} m_{ij} \right) \tag{4.10}$$

Note the following equality $\mathsf{D}(\mathcal{C}_1, \mathcal{C}_2) = (\mathsf{MH}(\mathcal{C}_1, \mathcal{C}_2) + \mathsf{MH}(\mathcal{C}_2, \mathcal{C}_1))/2$. The corresponding version of Fowlkes-Mallows is the *F-Measure* given in Equation (4.11). Note that the term $2m_{ij}/(|C_i|+|C'_j|)$ is the harmonic mean of $m_{ij}/|C_i|$ and $m_{ij}/|C'_j|$. These corresponds to recall and precision under the assumption that clustering $\mathcal{C}_1$ is topic-related and $\mathcal{C}_2$ is an arbitrary clustering.

$$\mathsf{F}(\mathcal{C}_1, \mathcal{C}_2) := 1 - \frac{1}{|V|} \sum_{i=1}^{k} |C_i| \max_{j \in \{1,\dots,\ell\}} \frac{2m_{ij}}{|C_i| + |C'_j|} \tag{4.11}$$

4.1.3. Information-Theoretic Comparators

The last group of comparators is based on information-theoretic concepts, namely entropy and mutual information. The *entropy* of the clustering $\mathcal{C}_1$ measures the uncertainty of the corresponding cluster to which a uniformly at random picked node belongs; it is given in Equation (4.12).

$$H(\mathcal{C}_1) := -\sum_{i=1}^{k} \frac{|C_i|}{|V|} \cdot \log_2 \frac{|C_i|}{|V|} \tag{4.12}$$

This can be extended to the *mutual information* of two clusterings shown in Equation (4.13). Note that $0 \cdot \log 0$ is set to zero by convention.

$$I(\mathcal{C}_1, \mathcal{C}_2) := \sum_{j=1}^{\ell} \sum_{i=1}^{k} \frac{m_{ij}}{|V|} \cdot \log_2 \frac{m_{ij} \cdot |V|}{|C_i| \cdot |C'_j|} \tag{4.13}$$

Mutual information describes the average reduction of the entropy of $\mathcal{C}_1$ when knowing clustering $\mathcal{C}_2$.

Two comparators directly use mutual information with different normalization, i. e., *Strehl-Ghosh* uses the geometric mean of the entropies of the individual clusterings, while *Fred-Jain* considers the arithmetic mean. The formulas are given in Equations (4.14) and (4.15).

$$\mathsf{SG}(\mathcal{C}_1, \mathcal{C}_2) := \begin{cases} 1 - \dfrac{I(\mathcal{C}_1, \mathcal{C}_2)}{\sqrt{H(\mathcal{C}_1) \cdot H(\mathcal{C}_2)}} & \text{, if } |\mathcal{C}_1| \neq |V| \wedge |\mathcal{C}_2| \neq |V| \\ \dfrac{\big| |\mathcal{C}_1| - |\mathcal{C}_2| \big|}{|V| - 1} & \text{, otherwise} \end{cases} \tag{4.14}$$

$$\mathsf{FJ}(\mathcal{C}_1, \mathcal{C}_2) := \begin{cases} 0 & \text{, if } |\mathcal{C}_1| = |\mathcal{C}_2| = |V| \\ 1 - \dfrac{2 \cdot I(\mathcal{C}_1, \mathcal{C}_2)}{H(\mathcal{C}_1) + H(\mathcal{C}_2)} & \text{, otherwise} \end{cases} \tag{4.15}$$

A related comparator is *Variation of Information* given in Equation (4.16); informally, it measures the loss of information with respect to $\mathcal{C}_1$ and the gain of information with respect to $\mathcal{C}_2$ when switching from $\mathcal{C}_1$ to $\mathcal{C}_2$.

$$\mathsf{VI}(\mathcal{C}_1, \mathcal{C}_2) := \frac{H(\mathcal{C}_1) + H(\mathcal{C}_2) - 2 \cdot I(\mathcal{C}_1, \mathcal{C}_2)}{\log_2 |V|} \tag{4.16}$$

4.2. Graph-Structural Approaches

As indicated by the example in Figure 4.1, lattice-driven approaches suffer certain drawbacks. A self-evident way is to combine a comparator with a quality index. However this may introduce other artefacts. In the following, we present some extensions for pair-counting comparators as well as new ideas to incorporate the graph structure. More details can be found in [31].

4.2.1. Extended Pair Comparators

The presented extensions are true extensions, i. e., if the underlying graph is complete then both—the extended and the original—versions of the comparator yield the same value. Pair comparators presented in Section 4.1.1 are founded on the classification of pairs of nodes. The basis of our extensions is to define a weighed matrix that assign to each pair of nodes a weight representing the relevance of that pair for the comparator, as given in Definition 4.2.

Definition 4.2. *Let $G = (V, E)$ be an undirected and unweighted graph. A* pair weighting *M is a symmetric $|V| \times |V|$–matrix over $[0, 1]$ such that*

$$\forall\, u, v \in V \colon \{u, v\} \in E \Longrightarrow M[u, v] = 1 \ .$$

The accumulated weights of the sets S_{ab} for $a, b \in \{0, 1\}$ are denoted as n^M_{ab}, i. e.,

$$n^M_{ab} := \sum_{\{u,v\} \in S_{ab}} M[u, v] \ .$$

Note that in the case of complete graphs, off-diagonal elements in M have to be one, thus $n_{ab} = n^M_{ab}$ for $a, b \in \{0, 1\}$ and arbitrary pair weightings. The shortcut $M(V')$ for $V' \subseteq V$ denotes the sum of weights over all pairs of nodes in V'. Replacing n_{ab} with n^M_{ab} in the formula given in Section 4.1.1, we obtain the Equations (4.17)–(4.20).

$$\mathsf{R}^M(\mathcal{C}_1, \mathcal{C}_2) := 1 - \frac{n^M_{11} + n^M_{00}}{M(V)} \tag{4.17}$$

$$\mathsf{R}^M_{\text{adj}}(\mathcal{C}_1, \mathcal{C}_2) := 1 - \frac{n^M_{11} - t_3}{\frac{1}{2}(t_1 + t_2) - t_3} \ , \text{ with} \tag{4.18}$$

$$t_1 := \sum_{i=1}^{k} M(C_i), t_2 := \sum_{k=1}^{\ell} M(C'_j), \text{ and } t_3 := \frac{t_1 t_2}{M(V)}$$

$$\mathsf{FM}^M(\mathcal{C}_1, \mathcal{C}_2) := \begin{cases} 1 - \dfrac{n^M_{11}}{\sqrt{(n^M_{11} + n^M_{10})(n^M_{11} + n^M_{01})}} & \text{, if } |\mathcal{C}_1| \neq |V| \wedge |\mathcal{C}_2| \neq |V| \\ \dfrac{\big|\, |\mathcal{C}_1| - |\mathcal{C}_2| \,\big|}{|V| - 1} & \text{, otherwise} \end{cases} \tag{4.19}$$

$$\mathsf{J}^M(\mathcal{C}_1, \mathcal{C}_2) := \begin{cases} 1 - \dfrac{n^M_{11}}{n^M_{11} + n^M_{10} + n^M_{01}} & \text{, if } |\mathcal{C}_1| \neq |V| \wedge |\mathcal{C}_2| \neq |V| \\ 0 & \text{, otherwise} \end{cases} \tag{4.20}$$

The adjacency matrix A, i. e., the $|V| \times |V|$-matrix over $\{0, 1\}$ where $A[u, v] = 1$ if and only if $\{u, v\} \in E$, is a pair weighting. The resulting values n_{ab}^{A} are then just the sizes of $S_{ab} \cap E$ for $a, b \in \{0, 1\}$. Another pair weighting is the truncated inverse distance matrix D_k with $k \in \mathbb{N}$ defined as

$$D_k[u, v] := \begin{cases} \frac{1}{\operatorname{dist}(u,v)} & \text{, if } \operatorname{dist}(u, v) \leq k \\ 0 & \text{, otherwise} \end{cases} , \tag{4.21}$$

where $\operatorname{dist}(u, v)$ denotes the graph-theoretical distance between the nodes u and v. Note for $k = 1$ we obtain the adjacency matrix, i. e., $A = D_1$.

4.2.2. Editing-Set Comparators

Although the above presented extended pair comparators are true extensions, their current definition does not allow the comparison of two clusterings of two graphs with the same nodeset but different edgeset. The original pair comparators could handle such cases, since they solely depended on the nodeset. In the following, we present a first approach for a graph-structural measure that is based on editing sets and can deal with different edgesets. Further comparators and a more complete discussion is given in [31, 33].

The editing set $F_{\mathcal{C}}$ of a clustering $\mathcal{C}$ is the set of pairs of nodes that have either to be added or delete in order to obtained disjoint and complete clusters. Informally, this set describes the errors of $\mathcal{C}$ with respect to the ideal case of disjoint cliques. Note, that the editing set and performance are high related as can be seen in Equation (4.22):

$$\mathsf{perf}\,(\mathcal{C}) = 1 - \frac{|F_{\mathcal{C}}|}{\frac{1}{2}\,|V|\,(|V| - 1)} . \tag{4.22}$$

The *Editing Set Difference* (Equation (4.23)) is defined as the normalized cardinality of the geometric difference of the two editing sets, i. e.,

$$\mathsf{ESD}(\mathcal{C}_1, \mathcal{C}_2) := \frac{|F_{\mathcal{C}} \cup F_{\mathcal{C}'}| - |F_{\mathcal{C}} \cap F_{\mathcal{C}'}|}{|F_{\mathcal{C}} \cup F_{\mathcal{C}'}|} = 1 - \frac{|F_{\mathcal{C}} \cap F_{\mathcal{C}'}|}{|F_{\mathcal{C}} \cup F_{\mathcal{C}'}|} . \tag{4.23}$$

A potential drawback is its sensitivity to small perturbations. Consider the cases where $|F_{\mathcal{C}} \cup F_{\mathcal{C}'}| = 2$ and $|F_{\mathcal{C}} \cap F_{\mathcal{C}'}| = 1$, then Editing Set Difference measures always a distance of 0.5. Thus, we defined also a node-normalized version of Editing Set Difference as given in Equation (4.24).

$$\mathsf{ESD}_{\mathsf{n}}(\mathcal{C}_1, \mathcal{C}_2) := \frac{|F_{\mathcal{C}} \cup F_{\mathcal{C}'}| - |F_{\mathcal{C}} \cap F_{\mathcal{C}'}|}{\binom{|V|}{2}} . \tag{4.24}$$

Chapter 5

Clustering Methods

In this chapter, we present an overview of algorithmic paradigms and approaches used to calculate clusterings. It is split in three parts: First, some fundamental techniques are explained. Second, instances that embody these principles are described. Third, a detail description of some selected algorithms that have been used in our experimental evaluation is presented. The first two parts have been published in [47].

5.1. Generic Paradigms

The description of generic methods is structured into three groups: Greedy and shifting approaches as well as general optimizing techniques utilized to find near-optimal clusterings. Commonly, the concepts and algorithms apply a certain number of reductions to obtain an instance where a solution can easily be computed, and then extend it to the original input. This is very similar to the standard divide and conquer techniques, where instances are recursively divided until the problem is trivially solvable. In contrast to this, the reduction step is more general and can modify the instance completely. Thus, the reduced instance does not need to be a subinstance of the original problem. The reduction step is usually composed of two parts itself. First, significant substructures are identified. These can be bridges that are likely to separate clusters, or dense groups that indicate parts of clusters. Such an identification is often formulated as a hypothesis, and the recognized substructures are considered as evidence for its correctness. After the recognition phase a proper modification is applied to the current input graph. Such transformations can be arbitrary graph operations, however, the usual modifications are: addition and deletion of edges, as well as collapsing subgraphs, i. e., representing a subgraph by a new meta-node. A sketch of this idea is given in Figure 5.1. The 'shapes' of the (sub)instances indicate the knowledge of clustering, i. e., smooth shapes point to fuzzy information while the rectangles indicate exact knowledge. In the initial instance (left figure, upper row), no clustering information is present. During the reduction phases parts of the graph became more and more separated, which is indicated by the disjoint objects. The right instance in the upper row can be easily solved, and the fuzzy information is transformed into exact information with respect to the given instance. Based upon it, the expansion phase transfers this knowledge to the original instance (left figure, lower row). An additional difference

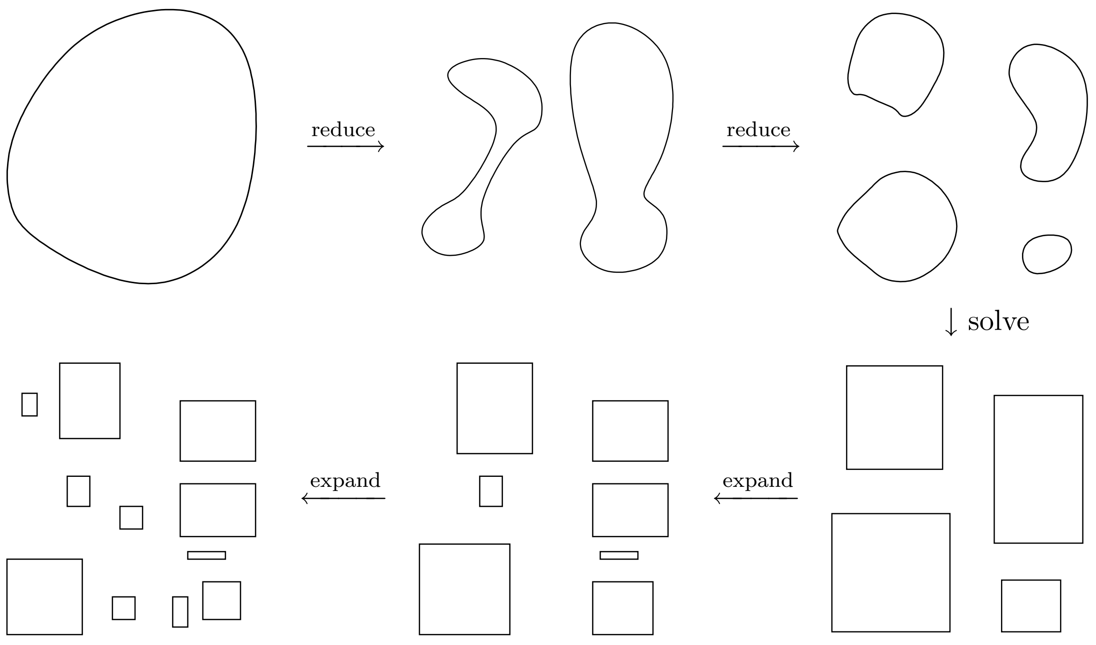


Figure 5.1.: Example of successive reductions. The 'shapes' of the (sub)instances indicate the knowledge of clustering, i. e., smooth shapes point to fuzzy information while the rectangles indicate exact knowledge. The first row shows the application of modifications, while the second indicates the expansion phases

to divide and conquer methods is that the size of the considered graphs can increase during the reduction phases.

5.1.1. Greedy Concepts

Most greedy methods fit into the following framework: start with a trivial and feasible solution and use update operations to lower its costs recursively until no further optimization is possible. This scheme for a greedy approach is shown in Algorithm 1, where $c(L)$ denotes the cost of solution L and $N_{\mathrm{g}}(L)$ is the set of all solutions that can be obtained via an update operation starting with solution L. This iterative scheme can also be expressed for clusterings via hierarchies. A hierarchy represents an iterative refinement (or coarsening) process. Greedy methods that use either merge or split operations as updates define a hierarchy in a natural way. The restriction to one of these operations guarantees the comparability of clusterings, and thus leads to a hierarchy. These two concepts will be formalized shortly, before that some facts of hierarchies are briefly mentioned.

Hierarchies provide an additional degree of freedom over clusterings: the number of clusters is not fixed. Thus, they represent the group structure independently of its

Algorithm 1: Scheme for greedy methods

```
let $L_0$ be a feasible solution
$i \leftarrow 0$
while $\{L : L \in N_g(L_i), c(L) < c(L_i)\} \neq \emptyset$ do
    $L_{i+1} \leftarrow \operatorname{argmin}_{L \in N_g(L_i)} c(L)$
    $i \leftarrow i + 1$
return $L_i$
```

granularity. However, this feature usually increases the space requirement, although a hierarchy can be implicitly represented by a tree, also called a dendrogram, that represents the merge/split operations. Algorithms tend to construct it explicitly. Therefore, their space consumption is often quadratic or larger. For a few special cases these costs can be reduced with the help of data structures [40].

The *Linkage process* (Agglomeration) iteratively coarses a given clustering by merging two clusters until the 1-clustering is reached. The formal description is shown in Definition 5.1.

Definition 5.1 (Linkage)**.** *Given a graph $G = (V, E, \omega)$, an initial clustering $\mathcal{C}_1$ and either a global cost function $c_{global} \colon \mathcal{A}(G) \to \mathbb{R}_0^+$ or a cost function $c_{local} \colon \mathcal{P}(V) \times \mathcal{P}(V) \to \mathbb{R}_0^+$ for merging operations, the* Linkage *process merges two clusters in the current clustering $\mathcal{C}_i := \{C_1, \ldots, C_k\}$ while possible in the following recursive way:*

global: *let P be the set of all possible clusterings resulting from $\mathcal{C}_i$ by merging two clusters, i. e.,*

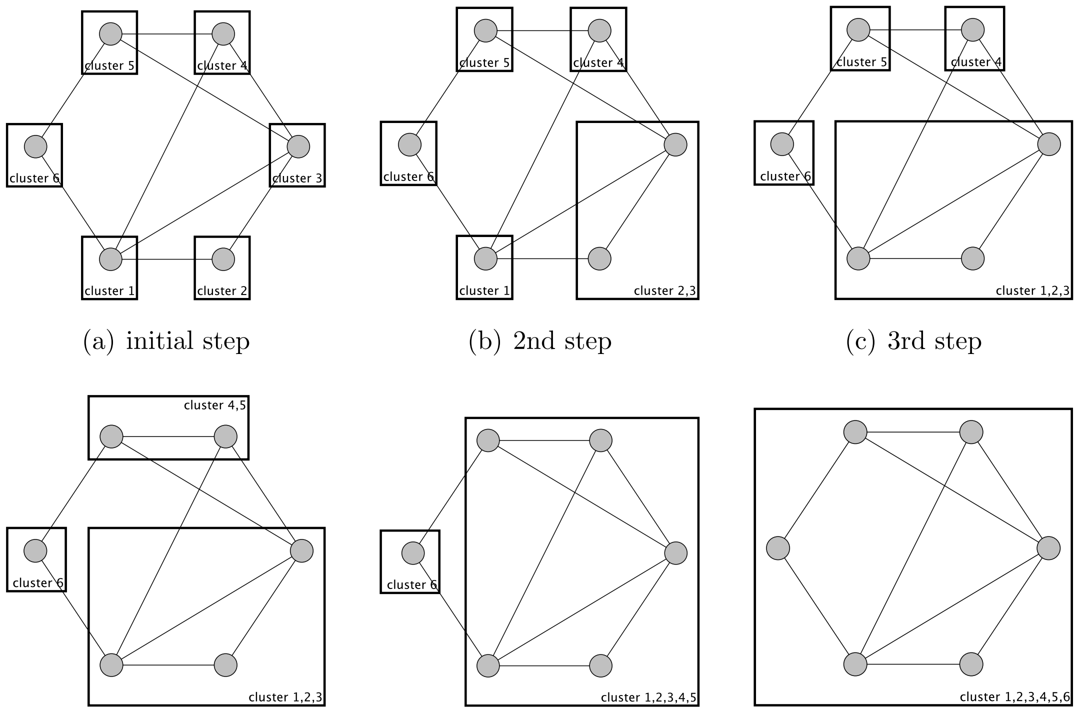
$$\mathsf{P} := \Big\{\{C_1, \ldots, C_k\} \setminus \{C_\mu, C_\nu\} \cup \{C_\mu \cup C_\nu\} \mid \mu \neq \nu\Big\} \ ,$$

then the new clustering $\mathcal{C}_{i+1}$ is defined as an element in P with minimum cost with respect to c_{global}.

local: *let μ, ν be those two distinct indices such that c_{local} has one global minimum in the pair (C_μ, C_ν), then the new clustering $\mathcal{C}_{i+1}$ is defined by merging C_μ and C_ν, i. e.,*

$$\mathcal{C}_{i+1} := \{C_1, \ldots, C_k\} \setminus \{C_\mu, C_\nu\} \cup \{C_\mu \cup C_\nu\} \ .$$

Although the definition is very formal, the basic idea is to perform a cheapest merge operation. The cost of such an operation can be evaluated using two different view points. A local version charges only the merge itself, which depends only on the two involved clusters. The opposite view is a global version that considers the impact of the merge operation. These two concepts imply also the used cost functions, i. e., a global cost function has the set of clusterings as domain while the local cost function uses a pair of node subsets as arguments. An example of linkage is given in Figure 5.2. The process of linkage can be reversed, and, instead of merging two clusters, one cluster is split into two parts. This dual process is called *Splitting* (Diversion). The formal description is given in Definition 5.2.



(a) initial step (b) 2nd step (c) 3rd step

(d) 4th step (e) 5th step (f) final step

Figure 5.2.: Example of linkage

Definition 5.2 (Splitting)**.** *Let a graph $G = (V, E, \omega)$, an initial clustering $\mathcal{C}_1$, and one of the following function sets be given:*

global: *a cost function $c_{global} \colon \mathcal{A}(G) \to \mathbb{R}_0^+$*

semi-global: *a cost function $c_{global} \colon \mathcal{A}(G) \to \mathbb{R}_0^+$ and a proper cut function $S_{local} \colon \mathcal{P}(V) \to \mathcal{P}(V)$*

semi-local: *a cost function $c_{local} \colon \mathcal{P}(V) \times \mathcal{P}(V) \to \mathbb{R}_0^+$ and a proper cut function $S_{local} \colon \mathcal{P}(V) \to \mathcal{P}(V)$*

local: *a cost function $c_{local} \colon \mathcal{P}(V) \times \mathcal{P}(V) \to \mathbb{R}_0^+$*

The Splitting *process splits one cluster in the current clustering $\mathcal{C}_i := \{C_1, \ldots, C_k\}$ into two parts. The process ends when no further splitting is possible. The cluster that is going to be split is chosen in the following way:*

global: *let P be the set of all possible clusterings resulting from $\mathcal{C}_i$ by splitting one cluster into two non-empty parts, i. e.,*

$$P := \Big\{ \{C_1, \ldots, C_k\} \setminus \{C_\mu\} \cup \{C'_\mu, C_\mu \setminus C'_\mu\} \mid \emptyset \neq C'_\mu \subsetneq C_\mu \Big\} \ ,$$

then the new clustering $\mathcal{C}_{i+1}$ is defined as an element in P with minimum cost with respect to c_{global}.

semi-global: let P be the set of all possible clusterings resulting from $\mathcal{C}_i$ by splitting one cluster into two non-empty parts according to S_{local}, i. e.,

$$\mathsf{P} := \Big\{ \{C_1, \ldots, C_k\} \setminus \{C_\mu\} \cup \{S_{local}(C_\mu)\,, C_\mu \setminus S_{local}(C_\mu)\} \Big\} \ ,$$

then the new clustering $\mathcal{C}_{i+1}$ is defined as an element in P with minimum cost with respect to c_{global}.

semi-local: let μ be an index such that c_{local} has one global minimum in the pair $(S_{local}(C_\mu)\,, C_\mu \setminus S_{local}(C_\mu))$ then the new clustering $\mathcal{C}_{i+1}$ is defined by splitting cluster C_μ according to S_{local}, i. e.,

$$\mathcal{C}_{i+1} := \{C_1, \ldots, C_k\} \setminus \{C_\mu\} \cup \{S_{local}(C_\mu)\,, C_\mu \setminus S_{local}(C_\mu)\} \ .$$

local: let μ be an index and C_ν be a proper subset of cluster C_μ such that c_{local} has one global minimum in the pair $(C_\nu, C_\mu \setminus C_\nu)$, then the new clustering $\mathcal{C}_{i+1}$ is defined by splitting cluster C_μ according to S_{local}, i. e.,

$$\mathcal{C}_{i+1} := \{C_1, \ldots, C_k\} \setminus \{C_\mu\} \cup \{C_\mu, C_\mu \setminus C_\nu\} \ .$$

Similar to the Linkage process, the definition is rather technical but the basic idea is to perform the cheapest split operation. In contrast to the Linkage method, the cost model has an additional degree of freedom because clusters can be cut in several ways. Again, there is a global and a local version that charge the impact of the split and the split itself, respectively. Both correspond to the views in the Linkage process. However, rating every possible non-trivial cut of the clusters is very time consuming and usually requires sophisticated knowledge of the involved cost functions. One way to reduce the set of possible splittings is to introduce an additional cut function S_{local}. It serves as an 'oracle' to produce useful candidates for splitting operations. The semi-global and semi-local versions have the same principles as the global and the local version, however, their candidate set is dramatically reduced. Therefore, they are often quite efficiently computable, and no sophisticated knowledge about the cost function is required. However, the choice of the cut function has usually a large impact on the quality.

Both, the Linkage and the Splitting process, are considered to be greedy for several reasons. One is the construction of the successive clusterings, i. e., an update operation chooses always the cheapest clustering. These can produce total or complete hierarchies quite easily. Total hierarchies can be achieved by simply adding the trivial clusterings to the resulting hierarchy. They are comparable to all other clusterings, therefore preserving the hierarchy property. Recall that complete hierarchies are hierarchies such that a clustering with k clusters is included for every integer $k \in [1, n]$. Both processes lead to a complete hierarchy when initialized with the trivial clusterings, i. e., singletons for Linkage and the 1-clustering for Splitting. Note that in the case of the Splitting process, it is essential that the cut functions are proper. Therefore, it is guaranteed that every cluster will be split until each cluster contains only one node. Although the cost can be measured with respect to the result or the operation itself, clairvoyance or projection of information into the future, i. e., accepting momentarily higher cost for a later benefit, is never possible.

Because of their simple structure, especially in the local versions, both concepts are frequently used and are also the foundation of clustering algorithms in general. The general local versions can be very efficiently implemented. For the Linkage process, a matrix containing all cluster pairs and their merging cost is stored. When an update operation takes place, only the cost of merging the new resulting cluster with another is recalculated. For certain cost functions this scheme can even be implemented with less than quadratic space and runtime consumption [40]. In the case of the Splitting process, only the cut information needs to be stored for each cluster. Whenever a cluster gets split, one has to recompute the cut information only for the two new parts. This is not true for any global version in general.

5.1.2. Shifting Concept

In contrast to the global action of the previous greedy strategies, the shifting approaches work more locally. They choose an initial clustering and iteratively modify it until a local optimum is found. Usually three operations are permitted: first, a node moves from one cluster to another existing cluster, second, a node moves from one cluster to create a new cluster, and, third, two nodes exchange their cluster assignments. Sometimes more complex operations, like instant removal of one cluster and reassigning nodes to already existing clusters, are allowed. However, these complex operations are usually only used for speed-up, or to avoid artifical situations, therefore they will not be discussed here. Note that one can typically simulate them by a sequence of simple operations. Regarding algorithmic aspects, shifting concepts are relatively close to the greedy ones. Algorithm 2 shows the general procedure where $N_{\mathrm{s}}(L)$ denotes the set of all clusterings that can be obtained by applying the modifications to the clustering L. Step 2 can be based on

Algorithm 2: Scheme for shifting methods

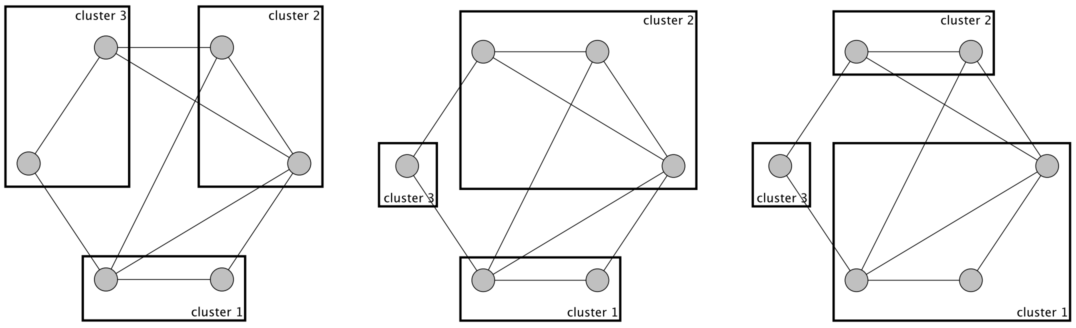
```
  let $L_0$ be an initial solution
  $i \leftarrow 0$
  while $N_s(L_i) \neq \emptyset$ do
1     choose $L_{i+1} \in N_{\mathrm{s}}(L_i)$
      $i \leftarrow i + 1$
  return $L_i$
```

cost or potential functions, random selecting, or on the applicational background. The technical definition of the shifting concept is given in Definition 5.3, and uses a potential function as selection criteria.

Definition 5.3. *Given a graph $G = (V, E, \omega)$, an initial clustering $\mathcal{C}_1$ and a potential function $\Phi\colon \mathcal{A}(G) \times \mathcal{A}(G) \to \mathbb{R}$, the* Shifting *process is defined as performing any operation on the current clustering $\mathcal{C}_i$ that results in a new clustering $\mathcal{C}_{i+1}$ such that $\Phi(\mathcal{C}_i, \mathcal{C}_{i+1}) > 0$.*

Shifting concepts have many degrees of freedom. The choice of potential functions is critical. There are two common subtypes of potentials: type-based and compressed. *Type-based potentials* are functions that heavily depend on the type of operations. They are often used to preserve certain properties. In the case that creating a new cluster via a node movement is very expensive in contrast to the other node movements, it is very likely that the number of clusters in the final clustering is roughly the same as in the initial clustering. *Compressed* or *sequenced shifts* collapse a series of operations into one meta-operation, and rate only the output of this operation with its argument. Thus, a certain number of operations are free of charge. These functions are often used in combination with a standard potential function that has many local optima. By ignoring a number of intermediate steps, it may be easier to reach a global optimum.

Owing to their contingent iterative nature, shifting approaches are rarely used on their own. There can be sequences of shifting operations where the initial and final clustering are the same, so-called loops. Also, bounds on the runtime are more difficult to establish than for greedy approaches. Nonetheless, they are a common postprocessing step for local improvements. An example of shifting is given in Figure 5.3.



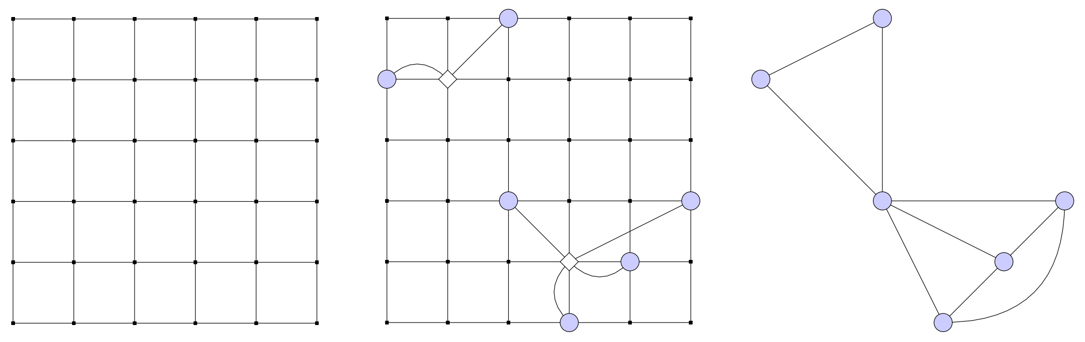
(a) initial step (b) 2. step (c) 3. step

Figure 5.3.: Example of shifting

5.1.3. General Optimization Concepts for Clustering

The two previous concepts, the greedy and the shifting framework, were fairly adapted. Both defined precisely the permitted operations and constraints, and the conditions of their application. The following concepts can be used for arbitrary optimization approaches. They are based on the idea that clusterings can be formulated as the result of a generic optimization process. The input data may be generated in a certain way with an implicit clustering structure. The optimization problem is to extract a clustering that is relatively close to the hidden one. Alternatively, the contained clustering is the result of an unknown optimization process. It is only known that this process respects certain paradigms, like intra-cluster density, inter-cluster sparsity, or both. The related problem is again to extract a clustering that is relatively close to the hidden one. The variety of techniques to solve optimization problems is gigantic, therefore only the following are considered here: parameter estimation, evolutionary and search-based approaches.

Parameter estimation is based on the assumption that the input data was created by a (random) sampling process: There is a hidden graph with a certain clustering, then a sample of the (whole) graph is drawn that will be the input graph. The approach then tries to estimate the parameters of this sampling process. These parameters will be used to reconstruct the clustering of the hidden graph. Originally, these methods were introduced to find clusterings for data embedded in metric spaces [66, Section 5.3]. There are clusterings that may be represented by a union of distributions, and the goal is to estimate the number of distributions and their parameters (mean, deviation, etc.). The *Expectation Maximization* (EM) is the most commonly used method. In general, it is only applied to data that is embedded in a metric space. Although graphs are typically not embedded, one can think of many processes that involve an implicit (metric) topology. An example is the following: let a finite space with a topology, a set of points, referred to as cluster centers, and a probability distribution for each cluster center be given; then n nodes/points are introduced by choosing one cluster center and a free spot around it that respects both the topology and its distribution. Two nodes will be connected by edges if their distance (with respect to the topology) is smaller than or equal to a given parameter. An example of this process is shown in Figure 5.4. Thus, the graph



(a) initial topology (b) 2 cluster centers (diamonds) and 6 nodes (circles) (c) resulting graph with distance threshold 3

Figure 5.4.: Example of generating a graph and its clustering using distributions.

(Figure 5.4(c)) would be the input graph, and the estimation approach would try to estimate the number of cluster centers as well as the assignment of each node to a cluster center. In the EM case, the resulting clustering should have the largest expectation to be the original hidden clustering, i. e., the same number of cluster points and the correct node cluster-point assignment. A similar technique as given in the example is used to generate embedded graphs with a known clustering which will be introduced in Section 6.2.2.

Evolutionary approaches such as genetic algorithms (GA), evolution strategies (ES) and evolutionary programming (EP) iteratively modify a population of solution candidates by applying certain operations. ‘Crossover’ and ‘mutation’ are the most common ones. The first creates a new candidate by recombining two existing ones, whilst the second modifies one candidate. To each candidate a fitness value is associated, usually the optimization function evaluated on the candidate. After a number of basic operations, a new population is generated based on the existing one, where candidates are selected according to their fitness value. A common problem is to guarantee the feasibility of modified solutions. Usually this is accomplished by the model specification. In the context of clustering, the model can either use partitions or equivalence relations.

Search-based approaches use a given (implicit) topology of the candidate space and perform a random walk starting at an arbitrary candidate. Similar to evolutionary approaches, the neighborhood of a candidate can be defined by the result of simple operations like the mutations. The neighborhood of a clustering usually is the set of clusterings that result from node shifting, cluster merging, or cluster splitting. The selection of a neighborhood is also based on some fitness value, usually the optimization function evaluated on the candidate. The search usually stops after a certain number of iterations, after finding a local optimum, or a combination of both.

5.2. Implementation of the Paradigms

Clustering methods have been developed in many different fields. They were usually very adapted, either for specific tasks or under certain conditions. The reduction of algorithms to their fundamental ideas, and constructing a framework on top, started not that long ago. Thus, this part can only give a short synopsis about commonly used methods.

Instances of Linkage

The different instances of the Linkage framework were originally designed for distance edge weights. Distances are the 'dual' version of similarities. Historically, the input data for clusterings algorithms was metrically embedded and complete (the similarity/dissimilarity of every pair is known). In these scenarios, it is possible to find clusterings using distance functions instead of similarities, i. e., one has to search for spatially dense groups that are well-separated from each other. If the distance function is only partially known, it is no longer possible to derive information about the similarity of two objects from their distance to other objects.

However, the use of distance functions had certain advantages that can be carried over to similarity weights. One reason was that distances can be easily combined to estimate the distance of a path. The most common way is the summation of the edge weights along the path. The standard local cost functions are defined as:

$$c_{local}(C_i, C_j) := \bigodot \{d(u,v) \colon u \in C_i, v \in C_j\} \ , \tag{5.1}$$

where $d(u,v)$ is the length of the shortest path connecting u and v, and $\bigodot$ is a function that evaluates sets, like minimum, average, or maximum. Indeed these three versions are called *Single Linkage*, *Average Linkage*, and *Complete Linkage*. A possible explanation of the name Single Linkage is that the cheapest, shortest path will be just an edge with minimum weight inside of $E(C_i, C_j)$. Note that the cost function can be asymmetric and have infinity as its value. Also, note that it is necessary to use the length of the shortest paths because not every node pair $(u, v) \in C_i \times C_j$ will be connected by an edge. In fact, the set $E(C_i, C_j)$ will be empty in the ideal case.

When dealing with similarities instead of distances one has to define a meaningful path 'length'. A simple way is to ignore it totally and define the cost function as

$$c_{local}(C_i, C_j) := \bigodot\{M - \omega(e) \colon e \in E(C_i, C_j)\} \ , \tag{5.2}$$

where M is the maximum edge weight in the graph. Alternatively, one can define the similarity of a path $P \colon v_1, \ldots, v_\ell$ by:

$$\omega(P) := \left(\sum_{i=1}^{\ell-1} \frac{1}{\omega(v_i, v_{i+1})} \right)^{-1} \ . \tag{5.3}$$

Although this definition is compatible with the triangle inequality, the meaning of the original range can be lost along with other properties. Similar to cost definition in Equation (5.2), the distance value (in Equation(5.1)) would be replaced by $(n-1)M - \omega(P)$. These 'inversions' are necessary to be compatible with our interpretation of range of cost functions. Another definition that is often used in the context of probabilities is

$$\omega(P) := \prod_{i=1}^{\ell-1} \omega(v_i, v_{i+1}) \ . \tag{5.4}$$

If $\omega(v_i, v_{i+1})$ is the probability that the edge (v_i, v_{i+1}) is present and these probabilities are independent of each other, then $\omega(P)$ is the probability that the whole path exists.

Lemma 5.4. *Given a undirected, weighted graph $G = (V, E, \omega)$, where ω measures the distance or dissimilarity, then the dendrogram of Single Linkage corresponds to a Minimum Spanning Tree (MST) of G.*

Only a sketch of the proof will be given. A complete proof can be found in [65]. The idea is the following: consider the algorithm of Kruskal where edges are inserted in non-decreasing order, and only those that do not create a cycle. From the clustering perspective of Single Linkage, an edge that would create a cycle connects two nodes belonging to the same cluster, thus that edge cannot be an inter-cluster edge, and thus would have never been selected.

The Linkage framework is often applied in the context of sparse networks and networks where the expected number of inter-cluster edges is rather low. This is based on the observation that many Linkage versions tend to produce chains of clusters. In the case where either few total edges or few inter-cluster edges are present, these effects occur less often.

$$\begin{aligned}
S(V) &:= \min_{\emptyset \neq V' \subset V} \omega(E(V', V \setminus V')) && (5.5)\\
S_{\text{ratio}}(V) &:= \min_{\emptyset \neq V' \subset V} \frac{\omega(E(V', V \setminus V'))}{|V'| \cdot (|V| - |V'|)} && (5.6)\\
S_{\text{balanced}}(V) &:= \min_{\emptyset \neq V' \subset V} \frac{\omega(E(V', V \setminus V'))}{\min(|V'|, (|V| - |V'|))} && (5.7)\\
S_{\text{conductance}}(V) &:= \min_{\emptyset \neq V' \subset V} \delta(V') && (5.8)\\
S_{2\text{-sector}}(V) &:= \min_{\substack{V' \subset V,\\ \lfloor |V|/2 \rfloor \leq |V'| \leq \lceil |V|/2 \rceil}} \omega(E(V', V \setminus V')) && (5.9)
\end{aligned}$$

Table 5.1.: Definition of various cut functions

Instances of Splitting

Although arbitrary cut functions are permitted for Splitting, the idea of sparse cuts that separate different clusters from each other has been the most common one. Among these are: standard cuts (Equation (5.5)), Ratio Cuts (Equation (5.6)), balanced cuts (Equation (5.7)), Conductance Cuts (Equation (5.8)), and Bisectors (Equation (5.9)). Table 5.2 contains all the requisite formulae.

Ratio Cuts, balanced cuts, and Bisectors (and their generalization, the k–Sectors) are usually applied when the uniformity of cluster size is an important constraint. These measures are $\mathcal{NP}$-hard to compute in general, expect for the standard cut. Therefore, approximation algorithms or heuristics are used as replacement. Note that balanced cuts and Conductance Cuts are based on the same fundamental ideas: rating the size/weight of the cut in relation to the size/weight of the smaller induced cut side. Both are related to node and edge expanders as well as isoperimetric problems. These problems focus on the intuitive notion of bottlenecks and their formalizations (see Section 3.3 for more information about bottlenecks). Some relation to spectral properties are covered in [27]. Beside these problems, the two cut measures have more in common. There are algorithms ([101]) that can be used to simultaneously approximate both cuts. However, the resulting approximation factor differs.

Splitting is often applied to dense networks or networks where the expected number of intra-cluster edges is extremely high. An example for dense graphs are networks that model gene expressions [58]. A common observation is that Splitting methods tend to produce small and very dense clusters.

Non-standard Instances of Linkage and Splitting

There are several algorithms that perform similar operations to 'linkage' or 'splitting', but do not fit into the above framework. In order to avoid application-specific details, only some general ideas will be given without the claim of completeness.

The *Identification of Bridge Elements* is a common Splitting variant, where cuts are replaced by edge or node subsets that should help to uncover individual clusters. One removal step can lead to further connected components, however it is not required. Figure 5.5 shows such an example. Most of the techniques that are used

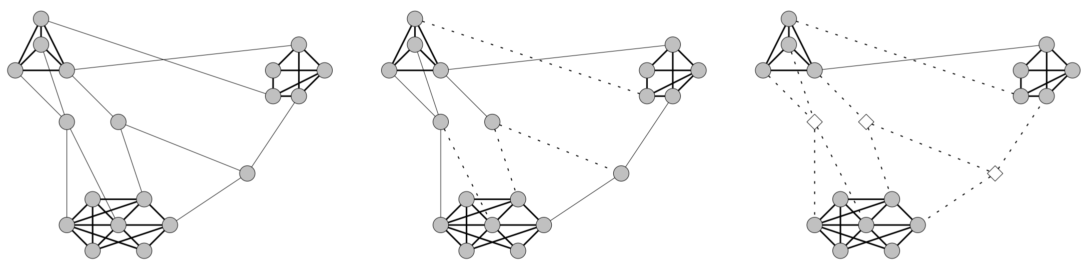


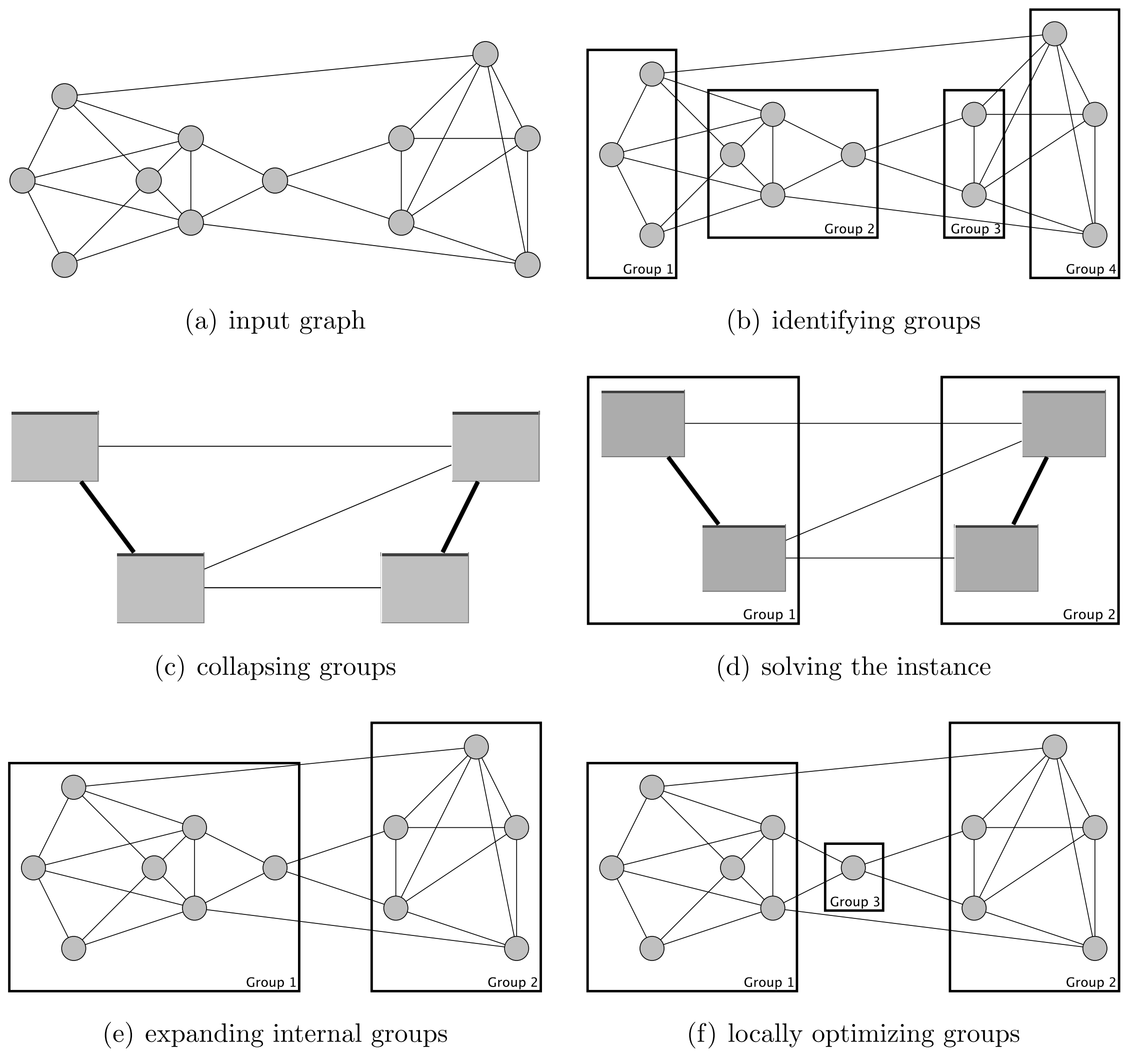
Figure 5.5.: Example for the removal of bridge elements. Removed elements are drawn differently: edges are dotted and nodes are reduced to their outline

to identify bridge elements are based on structural indices, or properties derived from shortest path or flow computations. Also centralities can be utilized for the identification [78].

Multi-Level Approaches are generalizations of the Linkage framework, where subsets of nodes are collapsed into a single element until the instance becomes solvable. Afterwards, the solution has to be transformed into a solution for the original input graph. During these steps the previously formed groups need not be preserved, i. e., a group can be split and each part can be assigned to individual clusters. In contrast to the original Linkage framework, here it is possible to tear an already formed cluster apart. Furthermore, the identified subsets of nodes that are collapsed are relative small. For example maximum matchings or cliques of constant sizes. Multi-level approaches are more often used in the context of equi-partitioning, where k groups of roughly the same size should be found that have very few edges connecting them. In this scenario, they have been successfully applied in combination with shiftings. Figure 5.6 shows an example. See also [69, 70] for further examples.

5.3. Algorithms for Closer Examinations

In the following, we present some algorithms in detail that are part of our experimental evaluation given in Section 6. The selection includes both greedy and shifting algorithms. Some algorithms employ the *normalized adjacency matrix* of G, i. e., $M(G) := D(G)^{-1}A(G)$ where $A(G)$ is the weighted adjacency matrix and $D(G)$ the diagonal matrix of the weighted node degrees. In order to define $D(G)^{-1}$, we require that G contains no isolated nodes.



(a) input graph

(b) identifying groups

(c) collapsing groups

(d) solving the instance

(e) expanding internal groups

(f) locally optimizing groups

Figure 5.6.: Example of a multi-level approach

5.3.1. Markov Clustering (MCL)

The key intuition behind *Markov Clustering* (MCL) [100, p. 6] is that a "random walk that visits a dense cluster will likely not leave the cluster until many of its vertices have been visited." Rather than actually simulating random walks, MCL iteratively modifies a matrix of transition probabilities. Starting from $M = M(G)$ (which corresponds to random walks of a length of at most one), the following two operations are iteratively applied:

- *expansion*, in which M is taken to the power $e \in \mathbb{N}_{>1}$ thus simulating e steps of a random walk with the current transition matrix (Algorithm 3, Step 1)
- *inflation*, in which M is re-normalized after taking every entry to its rth power, $r \in \mathbb{R}^+$. (Algorithm 3, Steps 2–4)

Note that for $r > 1$, inflation emphasizes the heterogeneity of probabilities within a row, while for $r < 1$, homogeneity is emphasized. The iteration is halted upon reaching a recurrent state or a fixed point. A recurrent state of period $k \in \mathbb{N}$ is a matrix that is invariant under k expansions and inflations, and a fixed point is a recurrent state of period 1. It is argued that MCL is most likely to end up in a fixed point [100]. The clustering is induced by connected components of the graph underlying the final matrix. Pseudo-code for MCL is given in Algorithm 3. Except for the stop criteria, MCL is deterministic, and its complexity is dominated by the expansion operation which essentially consists of matrix multiplication. Note that

Algorithm 3: Markov Clustering (MCL)

```
Input: $G = (V, E, \omega)$, expansion parameter $e$, inflation parameter $r$
$M \leftarrow M(G)$
while $M$ is not fixed point do
    // expansion
1   $M \leftarrow M^e$
    // inflation
2   forall $u \in V$ do
3       forall $v \in V$ do $M_{uv} \leftarrow M_{uv}^r$
4       forall $v \in V$ do $M_{uv} \leftarrow M_{uv} / \sum_{w \in V} M_{uw}$
$H \leftarrow$ graph induced by non-zero entries of $M$
$\mathcal{C} \leftarrow$ clustering induced by connected components of $H$
```

MCL demonstrates the reduction phase initially described in Section 5.1 quite well. The updated matrix M can be interpreted as weighted graph, i. e., every non-zero entry corresponds to an appropriated weighted edge. During the process, clusters become more and more isolated and also sparser; in the graph corresponding to a fixed point, the cluster are almost always stars. However, the number of non-zero entries during the first few iterations increases drastically. In a straight forward implementation, one expansion step requires cubic runtime (with respect to the number of nodes) and one inflation step needs quadratic runtime. As suggested by the author, one can reduce the runtime and the space consumption drastically by applying a heuristic which keeps only the k largest value per row. In this way the matrix M is kept sparse.

5.3.2. Iterative Conductance Cutting (ICC)

The basis of *Iterative Conductance Cutting* (ICC) [101] is to iteratively split clusters using minimum conductance cuts. Finding a cut with minimum conductance is $\mathcal{NP}$–hard, therefore the following poly-logarithmic approximation algorithm is used. Consider the node ordering implied by an eigenvector to the second largest eigenvalue of $M(G)$. Among all cuts that split this ordering into two parts, one of minimum conductance is chosen. Splitting of a cluster ends when the approximation value of the conductance exceeds an input threshold α^* first. Pseudo-code for ICC is given in Algorithm 4. Except for the eigenvector computations, ICC is deterministic. While the overall running time depends on the number of iterations, the running time of the conductance cut approximation is dominated by the eigenvector computation which needs to be performed in each iteration. The runtime of such an eigenvector computation heavily depends on the structure of subgraph and can take time $\mathcal{O}((|V| + |E|) \log |V|)$ in the worst case.

Algorithm 4: Iterative Conductance Cutting (ICC)

```
Input: $G = (V, E, \omega)$, conductance threshold $0 < \alpha^* < 1$
$\mathcal{C} \leftarrow \{V\}$
while there is a $C \in \mathcal{C}$ with $\phi(G[C]) < \alpha^*$ do
    $x \leftarrow$ eigenvector of $M(G[C])$ associated with second largest eigenvalue
    $\mathcal{S} \leftarrow \left\{ S \subset C \mid \max_{v \in S}\{x_v\} < \min_{w \in C \setminus S}\{x_w\} \right\}$
    $C' \leftarrow \arg\min_{S \in \mathcal{S}}\{\phi(S)\}$
    $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C\}) \cup \{C', C \setminus C'\}$
```

5.3.3. Geometric MST Clustering (GMC)

Geometric MST Clustering (GMC) is a graph clustering algorithm combining spectral partitioning with a geometric clustering technique introduced by us in [46, 24]. A geometric embedding of G is constructed from d distinct eigenvectors $x_1, \ldots, x_d$ of $M(G)$ associated with the largest eigenvalues less than 1. The edges of G are then weighted by a distance function induced by the embedding and a minimum spanning tree (MST) of the weighted graph is determined. A MST T implies a sequence of clusterings as follows: For a threshold value τ let $F(T, \tau)$ be the forest induced by all edges of T with weight at most τ. For each threshold τ, the connected components of $F(T, \tau)$ induce a clustering. Note that there are at most $|V| - 1$ thresholds resulting in different forests. The resulting clustering of $F(T, \tau)$ does not depend on the actual MST T (see Lemma 5.8), therefore we denote it with $\mathcal{C}(\tau)$. In order to verify this statement, we prove the following three lemmas, which handle locality in the connected components (Lemma 5.5), very similar MSTs (Lemma 5.6), and sequences of MSTs (Lemma 5.7).

Lemma 5.5. *Let $G = (V, E, \omega)$ be an undirected weighted graph with $\omega \colon E \to \mathbb{R}^+$. Let $T = (V, E')$ be a spanning tree and V' the node set of a connected subtree T' in T. Then the following equation holds for every threshold τ:*

$$F(T, \tau) \upharpoonright V' = F(T', \tau) \ . \tag{5.10}$$

Proof. The clustering $F(T, \tau) \upharpoonright V'$ of V' can be rewritten as

$$F(T, \tau) \upharpoonright V' = \{C \cap V' \mid C \in F(T, \tau) \wedge C \cap V' \neq \emptyset\} \ .$$

We prove equation 5.10 by using mutual inclusion. First, we show that the left side is included in the right one. Let $C' \in F(T, \tau) \upharpoonright V'$ and $C \in F(T, \tau)$ such that $\emptyset \neq C' = C \cap V'$. Then for every pair of nodes contained in C' there exists an unique path p in T such that each edge has a weight less than τ. Since T' is connected and spans V', every path in T connecting two nodes in V' is totally contained in T'. Thus, there exists a $C'' \in F(T', \tau)$ such that $C' \subseteq C''$. For every node pair in C'' there exists an unique path in T' such that each edge has a weight less than τ. This path is also a path in T with the same property, therefore $C' = C''$.

Second, we show that the right side is included in the left one. Let $C \in F(T', \tau)$, then there exists a unique path in T' between every pair in C such that each edge has weight less than τ. This path is also a path in T with the same property, thus there exists a cluster $C' \in F(T, \tau)$ with $C \subseteq C'$. Moreover the following inclusion holds:

$$C = C \cap V' \subseteq C' \cap V' \ .$$

Thus, it is sufficient to show that $C' \cap V' = C$. Suppose otherwise and let u be a node in C and v a node in $(C' \cap V') \setminus C$. Then there exists a unique path p connecting u and v in T such that every edge in p has weight less than τ. Since T' is connected and $u, v \in V'$, the path p has to be contained in T' as well. However, every path connecting u and v in T' contains an edge weight greater or equal to τ, otherwise v would be in C. This contradicts $C \subsetneq C' \cap V'$. □

Lemma 5.6. *Let $G = (V, E, \omega)$ be an undirected weighted graph with $\omega \colon E \to \mathbb{R}^+$. Let $T = (V, E')$ and $T' = (V, E'')$ be two MSTs such that $E'' = E' \setminus \{e'\} \cup \{e''\}$. Then the clusterings $F(T, \tau)$ and $F(T', \tau)$ are the same.*

Proof. Since both trees T and T' are minimum spanning trees, both edges e' and e'' have the same weight. Furthermore, let $C = (V', E_C)$ denote the cycle formed by e'' and the path (in T) connecting its end-nodes. This cycle also contains e'. The subgraph $(V', E_C \setminus \{e''\})$ is the unique path in T connecting the two end-nodes of e''. Suppose this path does not contain e', then it is also a path in T'. Thus C is contained in T' which contracts T' being a tree.

Using Lemma 5.5, it is sufficient to show the following equality

$$F(T, \tau) \restriction C = F(T', \tau) \restriction C \ .$$

In the case that $\omega(e') < \tau$, both clusterings equal $\{V'\}$ and are thus the same. Therefore, let us assume that $\omega(e') \geq \tau$. We divide the cycle into subpaths p_i such that each edge in the paths has weight less than τ. Since this division is independent of e' and e'', we obtain the following equation:

$$F(T, \tau) \restriction C = \{V_i \mid V_i \text{ is the node set of path } p_i\} = F(T', \tau) \restriction C \ ,$$

which concludes the lemma. □

Lemma 5.7. *Let $G = (V, E, \omega)$ be an undirected weighted graph with $\omega \colon E \to \mathbb{R}^+$, $T = (V, E')$ and $T' = (V, E'')$ be two different MSTs. Then there exists an MST $\widetilde{T} = (V, \widetilde{E})$ such that*

$$\exists\, e'' \in E'' \setminus E', e' \in E' \colon \widetilde{E} = E' \setminus \{e'\} \cup \{e''\} \ .$$

Proof. Let $\Delta E' := E'' \setminus E'$ be the set of tree-edges (in T') that are not contained in T. If $|\Delta E'| = 1$ then $\widetilde{T} = T'$ suffices. Otherwise, let $e' \in \Delta E'$. This edge splits T' and thus partitions V into two non-empty parts V_1 and V_2. Since T is a spanning tree that does not contain e', there exists an edge e that connects V_1 and V_2. Both edges e' and e have the same weight, otherwise not both trees T and T' could have minimum weight. We define $\widetilde{T} := (V, E' \setminus \{e\} \cup \{e'\})$, it is still spanning and has the same weight as T, therefore it is an MST. □

Lemma 5.8. *The clustering induced by the connected components of $F(T, \tau)$ is independent of the particular MST T.*

Proof. Let T and T' be two different MSTs. By Lemma 5.7, we can construct a sequence of MSTs such that every two consecutive MSTs differ in exactly one edge. Using Lemma 5.6, the clusterings induced by such a pair are the same, therefore the clustering of T and T' are the same. □

Among the $\mathcal{C}(\tau)$ we choose one optimizing some measure of quality. Potential measures of quality are, e. g., the indices defined in Section 3 or combinations thereof. This universality allows to target different properties of a clustering. Pseudo-code for GMC is given in Algorithm 5. Except for the eigenvector computations, GMC is deterministic. Note that, opposite to ICC, they form a preprocessing step with their number bounded by a (typically small) input parameter. Assuming that the quality measure can be computed fast, the asymptotic time and space complexity of the main algorithm is dominated by the MST computation. GMC combines two proven concepts from geometric clustering and graph partitioning. The idea of using a MST that way has been considered before [107]. However, to our knowledge the MST decomposition was only used for geometric data before, not for graphs. In our case, general graphs without additional geometric information are considered. Instead, spectral graph theory [27] is used to obtain a geometric embedding that already incorporates insight about dense subgraphs. This induces a canonical distance on the edges which is taken for the MST computation.

Algorithm 5: Geometric MST Clustering (GMC)

```
Input: $G = (V, E, \omega)$, embedding dimension $d$, clustering valuation $quality$
  $(1, \lambda_1, \dots, \lambda_d) \leftarrow d + 1$ largest eigenvalues of $M(G)$
  $d' \leftarrow \max\{i \mid 1 \le i \le d, \lambda_i > 0\}$
  $x^{(1)}, \dots, x^{(d')} \leftarrow$ eigenvectors of $M(G)$ associated with $\lambda_1, \dots, \lambda_{d'}$
  forall $e = (u, v) \in E$ do  $w(e) \leftarrow \sum_{i=1}^{d'} |x_u^{(i)} - x_v^{(i)}|$
  $T \leftarrow$ MST of $G$ with respect to $w$
  $\mathcal{C} \leftarrow \mathcal{C}(\tau)$ for which $quality(\mathcal{C}(\tau))$ is maximum over all $\tau \in \{w(e) \mid e \in T\}$
```

5.3.4. Greedy Algorithm for Modularity

Modularity was originally introduced as a selection criteria for a divisive hierarchical clustering algorithm [79] and later used to define a greedy, agglomerative clustering algorithm [28]. The greedy algorithm starts with the singleton clustering and itera-

Algorithm 6: GREEDY ALGORITHM FOR MAXIMIZING MODULARITY

```
Input: graph $G = (V, E, \omega)$
Output: clustering $\mathcal{C}$ of $G$
$\mathcal{C} \leftarrow$ singletons
initialize matrix $\Delta$
while $|\mathcal{C}| > 1$ do
    find $\{i, j\}$ with $\Delta_{i,j}$ is the maximum entry in the matrix $\Delta$
    merge clusters $i$ and $j$
    update $\Delta$
return clustering with highest modularity
```

tively merges those two clusters that yield a clustering with the best modularity, i. e., the largest increase or the smallest decrease is chosen. After $n-1$ merges the clustering that achieved the highest modularity is returned. The algorithm maintains a symmetric matrix Δ with entries $\Delta_{i,j} := \mathsf{q}\left(\mathcal{C}_{i,j}\right) - \mathsf{q}\left(\mathcal{C}\right)$, where $\mathcal{C}$ is the current clustering and $\mathcal{C}_{i,j}$ is obtained from $\mathcal{C}$ by merging clusters C_i and C_j. The pseudo-code for the greedy algorithm is given in Algorithm 6. An efficient implementation using sophisticated data-structures requires $\mathcal{O}\left(n^2 \log n\right)$ runtime. Note that, $n-1$ iterations is an upper bound and one can terminate the algorithms when the matrix Δ contains only non-positive entries. This property is called uni-modularity and is proven in [28]. As shown in Section 3.5.1, it is $\mathcal{NP}$-hard to maximize modularity in general graphs. Thus the greedy approach cannot be optimal. More precisely, we show that its approximation factor is greater or equal to 2. In order to prove this statement, we introduce a general construction scheme given in Definition 5.9.

Definition 5.9. *Let $G = (V, E)$ and $H = (V', E')$ be two non-empty, simple, undirected, and unweighted graphs and let $u \in V'$ be a node. The* product $G \star_u H$*is defined as the graph (V'', E'') with*

- *the nodeset $V'' := V \cup V \times V'$ and*
- *the edgeset $E'' := E \cup E''_c \cup E''_H$ where*

$$\begin{aligned} E''_c &:= \{\, \{v, (v, u)\} \mid v \in V\} \qquad \textit{and} \\ E''_H &:= \{\, \{(v, v'), (v, w')\} \mid v \in V, v', w' \in V'', \{v', w'\} \in E\} \ . \end{aligned}$$

An example is given in Figure 5.7. Informally speaking, the product $G \star_u H$ consists of G and for each node in G a copy of H which is connected by one edge. In the following, we consider only a special case: Let $n \geq 2$ be an integer, $H = (V', E')$ be an undirected and connected graph with at least two nodes, and $u \in V'$ an arbitrary but fixed node. Furthermore let $\mathcal{C}_k^g$ be the clustering obtained with the

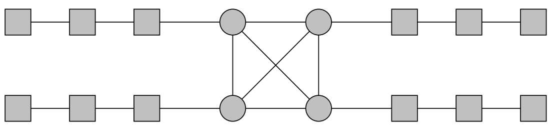


Figure 5.7.: Example of the product $K_4 \star_u P_3$, where u is one of the endnode of the P_3.

greedy algorithm applied to $K_n \star_u H$ starting from singletons and performing at most k steps that all have a positive increase in modularity and m the number of edges in $K_n \star_u H$. Next, we present three lemmas that state possible and impossible merges of the greedy algorithm.

Lemma 5.10. *If $2 \cdot |E'| < n$ and $\mathcal{C}_k^g$ has two clusters C_i and C_j such that both belong to the same copy of H and $E(C_i, C_j) \neq \emptyset$, then merging C_i and C_j increases the modularity.*

Proof. The difference of modularity before and after the merge is

$$\Delta\mathsf{q}\,(C_i, C_j) := \frac{|E(C_i, C_j)|}{m} - \frac{\sum_{u \in C_i} \deg u \cdot \sum_{u' \in C_j} \deg u'}{2m^2} \quad .$$

Since $|E(C_i, C_j)| \geq 1$ and $\sum_{u \in C_i \cup C_j} \deg u \leq 2|E'| + 1$, we obtain the following inequalities:

$$\begin{aligned} \Delta\mathsf{q}\,(C_i, C_j) &\geq \frac{1}{m} - \frac{\sum_{u \in C_i} \deg u \cdot \sum_{u' \in C_j} \deg u'}{2m^2} \\ &\geq \frac{1}{m} - \frac{1}{m^2}\left(\frac{2|E'|+1}{2}\right)^2 \quad . \end{aligned}$$

Due to the fact that $2|E'| < n$, we can conclude

$$m \cdot \Delta\mathsf{q}\,(C_i, C_j) \geq 1 - \frac{n^2 + 2n + 1}{2n + 2\binom{n}{2} + 2n} \geq 0 \quad .$$

□

Lemma 5.11. *If $2 \cdot |E'| + 1 < n$ and $\mathcal{C}_k^g$ has the cluster $C := \{v\}$ and a cluster C_i containing only nodes of the v-copy of H and $u \in C_i$, then merging C_i and C increases the modularity. Furthermore such a merge increases the modularity more than a merge of two clusters $\{w\}, \{w'\}$ for $w, w' \in V$.*

Proof. First note that $\sum_{w \in C_i} \deg w \leq 2|E'| + 1$. Thus the increase of modularity is

$$m \cdot \Delta\mathsf{q}\,(C_i, C) \geq 1 - \frac{n \cdot (n-1)}{n^2 + 3n} \geq 0$$

Since $n(n-1)/m < n^2/m$, we conclude $\Delta\mathsf{q}\,(C_i, C) \geq \Delta\mathsf{q}\,(\{v\}, \{w\}) \geq 0$. □

Lemma 5.12. *If $2 \cdot |E'| + 1 < n$ and $\mathcal{C}_k^g$ has two clusters C_i and C_j each containing at least two nodes where (exactly) one belongs to V, then the merge of C_i and C_j is never executed.*

Proof. Let $v \in V$ be the node of C_i. First, if $\{v\} \times V'$ is not completely contained in C_i, then let C be a cluster of $\mathcal{C}_k^g$ such that $C \cap \{v\} \times V' \neq \emptyset$ and $C \neq C_i$. Since the greedy algorithm only merges connected clusters $C \subset \{v\} \times V'$. Then merging C_i and C increases the modularity more than the merge of C_i and C_j:

$$\begin{aligned} m \cdot \Delta\mathsf{q}\,(C_i, C) &\geq 1 - \frac{(n+d) \cdot d'}{2m} \\ m \cdot \Delta\mathsf{q}\,(C_i, C_j) &= 1 - \frac{(n+d) \cdot (n+\widetilde{d})}{2m} \quad , \end{aligned}$$

where d is the sum of degrees of nodes in C_i without v, d' is the sum of degrees of nodes in C, and $\widetilde{d}$ is the sum of degrees of nodes in C_j without those belonging to V. Since $d' \leq 2|E'| \leq n$ and $\widetilde{d} \geq 1$ the merge of C_i and C is performed before the merge of C_i and C_j.

Second, if $C_i = \{v\} \cup \{v\} \times V'$ and $C_i = \{w\} \cup \{w\} \times V'$, then the merge of the two clusters decreases the modularity:

$$m \cdot \Delta\mathsf{q}\,(C_i, C_j) = 1 - \frac{(n + 2|E'| + 1)^2}{n^2 + (3 + |E'|)n} \quad .$$

Since $|E'| \geq 1$, the change of modularity Δq is negative. Thus the merge will not be executed. $\square$

Thus, we can now characterize the final clustering $\mathcal{C}_n^g$.

Theorem 5.13. *Let $n \geq 2$ be an integer and $H = (V', E')$ be a undirected and connected graph with at least two nodes. If $2|E'| + 1 < n$ then the greedy algorithm returns the clustering $\mathcal{C}^g$ with*

$$\mathcal{C}^g := \{\{v\} \cup \{v\} \times V' \mid v \in V\}$$

for $K_n \star_u H$ (for any fixed $u \in H$). This clustering has a modularity score of

$$4m^2 \cdot q\,(\mathcal{C}^g) = 4m\,((|E'| + 1) \cdot n) - n\,(2|E'| + 1 + n)^2 \quad .$$

Proof. Since the greedy algorithm only merges two clusters, if they are connected, the above Lemmas 5.10, 5.11 and 5.12 ensure that $\mathcal{C}^g = \mathcal{C}_k^g$ for some sufficient large k. According to Lemma 5.12, no further merge can occur. $\square$

Next we show that the clustering where G and each copy of H form individual clusters has a greater modularity score. First note, the explicit expression for modularity.

Corollary 5.14. *The clustering $\mathcal{C}^s$ defined as*

$$\mathcal{C}^s := \{V\} \cup \{\{v\} \times V' \mid v \in V\}$$

and its modularity is

$$4m^2 \cdot \mathsf{q}\left(\mathcal{C}^s\right) = 4m\left(|E'|n + \binom{n}{2}\right) - n\left(2|E'|+1\right)^2 - \left(n \cdot (n-1+1)\right)^2 \quad .$$

If $n \geq 2$ and $2|E'|+1 < n$, then clustering $\mathcal{C}^s$ has higher modularity than $\mathcal{C}^g$.

Theorem 5.15. *The approximation factor of the greedy algorithm for finding clusterings with maximum modularity is at least 2.*

Proof. We prove this statement by showing that the quotient $\mathsf{q}\left(\mathcal{C}^s\right)/\mathsf{q}\left(\mathcal{C}^g\right)$ is asymptotically two for a certain graph family. Therefore, we simplify the modularity scores with respect to their dominant terms. Note that $4m = 2n^2 + 4n\,|E'| + o\left(n^{1.5}\right)$,

$$4m\left(\left(|E'|+1\right) \cdot n\right) - n\left(2|E'|+1+n\right)^2 = 4m \cdot n \cdot |E'| + o\left(n^{3.5}\right) \tag{5.11}$$

and

$$\begin{aligned} 4m\left(|E'|n + \binom{n}{2}\right) & - n\left(2|E'|+1\right)^2 - \left(n \cdot (n-1+1)\right)^2 \\ &= 4m \cdot n \cdot |E'| + 2mn^2 - n^4 + o\left(n^{3.5}\right) \\ &= 4m \cdot n \cdot |E'| + 2n^3\,|E'| + o\left(n^{3.5}\right) \end{aligned} \tag{5.12}$$

Thus, we obtain the following equation:

$$\mathcal{R}^g := \frac{\mathsf{q}\left(\mathcal{C}^s\right)}{\mathsf{q}\left(\mathcal{C}^g\right)} = 1 + \frac{2n^3\,|E'| + o\left(n^{3.5}\right)}{4mn\,|E'| + o\left(n^{3.5}\right)}$$

and for sufficiently large n we can omit the additional terms which are contain in $o\left(n^{3.5}\right)$:

$$\frac{2n^2}{2n^2 + 4n\,|E'| + o\left(n^{1.5}\right)} = \frac{1}{1 + \frac{2|E'|}{n} + o\left(1/\sqrt{n}\right)}$$

By selecting paths with $1/2\sqrt{n}$ edges as graphs H, we obtain that $\mathcal{R}^g \geq 2 - \varepsilon$ for every positive ε. □

Chapter 6

Experimental Evaluation

The designing and engineering of systematical benchmarks and evaluations suits are discussed in this chapter. The first two sections give an overview of requirements, goals and potential pit-falls. The following sections introduce our experimental study as wells as gained insights.

6.1. General Description and Goals

All quality indices and comparators that have been introduced and studied in the literature suffer from drawbacks or artifical behavior, nevertheless they are used as optimization functions for finding good clusterings or for quality assessment in general. A frequently given justification is that only very specific circumstances cause these drawbacks and artefacts and that such cases seldomly occur in real-world applications.

Our goal is to provide an experimental setup for benchmarking and evaluating clustering algorithms, quality indices, comparators, and other clustering-related techniques. More precisely, the evaluation framework consists of (simple) *unit tests* that function as building blocks. The concept of unit tests was originally introduced in the field of software engineering and programming as an independent code module that ensures the correct functionality of a component. For example, such a test ensures that the associated methods of a data structure operate properly. They are frequently used when the implementation of a component is changed due to optimization, yet the functionality should remain. In our case, the provided experiments indicate the usability of a clustering technique. Similar to the tests in software engineering, our tests are only indicators, i. e., a meaningless technique can still successfully pass all test, while a failed test reveals its impracticality. In addition, the results of the tests themselves can be used to compare techniques and deepen the understanding.

Since we focus on an experimental evaluation which requires certain degrees of (statistical) significance, an essential part is the large availability of pre-clustered graphs, i. e., graphs with a significant clustering. In the following section, we introduce some generators for pre-clustered graphs and discuss potential drawbacks and circular dependencies.

6.2. Generators For Pre-Clustered Graphs

Most research has been focused on finding the decomposition of a given graph into natural clusters and on evaluating the corresponding quality. In the following, the dual problem is considered, namely generating a graph such that a given partition is significantly represented by the graph structure. Although, we restricted ourselves to the basic paradigm of intra-cluster density versus inter-cluster sparsity, the quantification of the significance essentially depends on the formalization and implementation. Most realizations of the paradigm favor different ideal situations (see Chapter 3) and generally agree only when the graph consists of disjoint and complete subgraphs.

Before introducing several implemented generators that have been heavily used in our experimental setup, a formal yet abstract description of the above duality is given. Consider the simplified view where clustering algorithms, quality indices, and pre-clustered graph generators are all interpreted as mappings: a clustering algorithm assigns to the each graph G and significance threshold τ a clustering $\mathcal{C}$ which has a significance score larger or equal to τ; a quality index maps a pair consisting of a graph G and a clustering $\mathcal{C}$ to a significance score τ; a pre-clustered graph generators assigns to each clustering $\mathcal{C}$ and significance score τ a graph G such that $\mathcal{C}$ has at least significance τ with respect to G. A pictorial overview is given in Figure 6.1. Although comparators are not explicitly itemized, they are part of the threefold interdependency, e. g., as structural quality measure or an optimization criteria in algorithms. As mentioned this view is simplified, since algorithms and

$$\begin{tikzcd}
G \arrow[dr, bend left] & & \tau \arrow[dl, bend right] \\
& \text{clustering algorithm} \arrow[d] & \\
& \mathcal{C} &
\end{tikzcd}
\qquad
\begin{tikzcd}
G \arrow[dr, bend left] & & \mathcal{C} \arrow[dl, bend right] \\
& \text{quality index} \arrow[d] & \\
& \tau &
\end{tikzcd}$$

$$\begin{tikzcd}
\mathcal{C} \arrow[dr, bend left] & & \tau \arrow[dl, bend right] \\
& \text{generator} \arrow[d] & \\
& G &
\end{tikzcd}$$

Figure 6.1.: Pictorial overview of input and output for clustering algorithms, quality indices, and generators.

generators usually do not use an explicit significance threshold as input. However, the input parameter for algorithms or generators can usually be tune in order to obtain various significance levels. For example, one can restrict the number of clusters or the number of edges to generate. The duality is reflected by the fact, that an quality index or clustering algorithm can be used to define a generator and vice versa. For example, let index be an arbitrary quality index. Then a the corresponding generator maps a partition $\mathcal{C}$ of the (node-)set V and quality threshold to a graph, where the edgeset is chosen from $\left\{E \subseteq \binom{V}{2} \mid \mathsf{index}((V,E),\mathcal{C}) \geq \tau\right\}$. Note, that a necessary condition is that for every partition of V there is an edgeset that has significance 1, since otherwise the set of possible edgesets can be empty for some τs. Similar, a generator $\mathsf{generator}$ defines a index by mapping the pair $(G,\mathcal{C})$ to that τ such that G has the maximum probability to be generated by $\mathsf{generator}(\mathcal{C},\tau)$. A necessary condition is that every graph is generated with positive probability for every clustering. Similar correspondings can be made for clustering techniques, however we omit them since this is only an illustrating example and other more suitable realizations may exists.

We introduce two simple generator types: first, an extension of the Erdős–Rényi random graph model $G(n,p)$ and, second, a geometric generators based on Voronoi Diagrams. In contrast to the above simplified view, they will not have one parameter defining the significance that has to be achieved, but several parameters controlling the perturbation of the ideal case of disjoint cliques.

6.2.1. Random Generators

The *random pre-clustered graph generator* uses an integer array representing the cluster sizes and two probabilities p_{in} and p_{out} for the existence of intra-cluster edges and inter-cluster edges as input. The graph is created by first assigning nodes randomly into clusters respecting the given clusters sizes and then inserting an edge between pairs of nodes in the same cluster with probability p_{in} and between other pairs with probability p_{out}. The generator extends the random graph model $\mathcal{G}(n,p)$, introduced in [41], with a group structure.

The *Gaussian generator* is a restriction of the random pre-clustered graph generator using a Gaussian distribution for the cluster sizes. More precisely, let n be the number of nodes the graph should contain. Then the number of clusters k is uniformly at random selected from the interval $[\log_{10}(n),\sqrt{n}]$. The cluster sizes are chosen from the Gaussian distribution defined by the mean $\lfloor n/k \rfloor$ and the deviation $\lfloor n/(4k) \rfloor$. Due to the random selection of cluster sizes, the number of nodes is only approximated. One can insert additional nodes or delete nodes in order to obtain exactly n nodes, however this can introduce artifacts which affect quality indices. This phenomena was observed by us for the measure inter-cluster conductance in [24].

For increasing n and a fixed pair $(p_{\text{in}}, p_{\text{out}})$ the growth of inter-cluster edges exceeds the growth of intra-cluster edges. Thus, we refine the Gaussian generator to the *significant Gaussian generator* that substitutes the parameter p_{out} by the parameter ρ defined as the ratio of expected inter-cluster edges and expected intra-cluster edges.

Note that ρ is highly dependent on the number of clusters. The generator initially creates a Gaussian partition as described for the Gaussian generator, dynamically calculates p_{out} according to Equation (6.1) and calls the same procedure as the Gaussian generator for building the edgeset.

$$\rho = \frac{p_{\text{out}}(n - n/k)}{p_{\text{in}}(n/k - 1)} \tag{6.1}$$

6.2.2. Geometric Generators

The *attractor generator* uses geometric properties to generate a significant clustering based on the following idea: k cluster centers, so called attractor nodes, are placed uniformly at random with a certain minimum distance t in the plane. The remaining $n - k$ satellite nodes are assigned to the attractors and their corresponding clusters using the following policy. At a random position a satellite node u is inserted with probability $d(u, a)/t$, where $d(u, a)$ is the Euclidean distance from u to the nearest attractor node a. If u is inserted, the edge $\{u, a\}$ is inserted as well. The parameter ρ sets the threshold for connecting nodes with a certain distance. Note, that the attractor nodes represent the point sites of a Voronoi Diagram and the satellite nodes are assigned to the clusters according to the Voronoi cells.

Our implementation uses the unit square for modeling the plane and $t = \sqrt{2/(\pi k)}$ as threshold for the minimum distance between attractor nodes and $\rho \cdot \sqrt{\pi k/2}$ as maximum distance between two satellite nodes in order to insert an edge. Global connectivity of the graph is ensured by connecting two attractor nodes if their distance is below $2.5 \cdot t$.

6.2.3. Drawbacks and Circular Dependencies

The presented generators are defined to be mostly independent from the quality indices, since they are solely based on perturbation of disjoint cliques. Thus they can be used to evaluate quality indices. However, hidden dependencies may exist and, even more severely, their perturbation parameters may be counterintuitive for humans. As mentioned, for increasing values of n and a fixed pair of parameters $(p_{\text{in}}, p_{\text{out}})$ the growth of the share of the inter-cluster edges exceed that of the intra-cluster edges when using the Gaussian generator. Thus, larger instances require a rescaling of p_{out} in order to obtain graphs with similar degrees of perturbation. Some of these drawbacks can be fixed by intelligent adjustments of parameters, e. g., Gaussian generator versus significant Gaussian generator. On the other hand one can exploit them to reveal drawbacks in other techniques such as quality indices or clustering algorithms.

As briefly mentioned at the beginning of Section 6.2, generators, quality measures, comparators, and clustering algorithms are different aspects of the one and the same problem, i. e., formalizing the term natural groups. Thus, there is a threefold interdependence between these concepts that implies certain pitfalls for meaningful benchmarking. Since generators, algorithms, comparators, and quality indices appear at different stages of the benchmarks, they have to be properly selected. As a simple example, consider an algorithm based on minimum cuts. Its evaluation may not only rely on coverage in order to maintain comparability with other techniques. Similarly, clustering algorithms that produce balanced clustering with respect to cluster sizes, should not be solely evaluated on graphs with screw cluster size distribution. More general, a too strongly correlated selection of generator, algorithms, comparators, and quality indices will only imply the high usability of the technique, while in the opposite case of un- or anti-correlated concepts, the obtained results are more or less random. Thus the design of meaningful benchmarks does not only require a deepened comprehension of each individual component such as generators, quality measures, comparators, or clustering techniques, but also of their mutual interdependencies. In principle, there is a theoretical and an experimental approach to improve this understanding. Although, theoretical examinations provide general characterizations, they are often hard to perform or tend to be highly specialized. On the other hand, the feasibility of experimental evaluations mainly depends on efficient implementations. Moreover, they often reveal additional intrinsic patterns and provide guidances for theoretical analyses. Our experimental benchmarking is founded on several basic and intuitive unit tests that are described in the next section. These tests provide a rough guideline for the behavior of clustering techniques.

6.3. Engineering Experiments

In general, our evaluation framework is based on the repeated execution of experiments with fixed parameters. Each experiment consists of the following three stages: (1) generation of preclustered graphs, (2) execution of clustering algorithms, and, finally, (3) evaluation of obtained clusterings using quality indices and comparators. Due to the randomness inherent in the generators for preclustered graphs, each experiment has to be executed until (statistical) significance has been achieved.

Regardless of the concept to test, a general strategy would be to start with very basic and intuitive instances and then gradually increase the complexity of the testing methods. Our tests are accordingly divided into simple and advanced ones. Note, that the complexity level is mainly defined by the needed requirements, i. e., an advanced unit test relies on already tested components. In the following, we present our unit tests in their general form. Although, we speak of succeeded and failed unit tests, this always referees to the result on a large number of experiments and not to a single instance. Note that adapted tests for specific concepts are not included, but could be derived from the given ones.

6.3.1. Simple Strategies

The most basic quality measure is coverage. Although it has drawbacks due to its simplicity, we can derive an initial unit test for generators and clustering algorithms based on it.

Unit-Test 1 (Basic Generators/Algorithms)**.** *For fixed generator and number of* *nodes, an increase (decrease) in the perturbation must not cause an increase (decrease) in coverage of the clustering used by the generator or obtained with an algorithm.*

Suitable generators have to fulfill Unit Test 1 as a necessary condition. However, for algorithms the situation is more complex. First note, that an algorithm may produce clusterings with a significantly different number of clusters as a result of varying the perturbation of the generator. On the other hand, coverage highly depends on number of clusters, for example the 1–clustering always has maximum coverage. Thus a failed Unit Test 1 does not necessarily imply a defect in the algorithm. We illustrate such a case in Figure 6.12(b), where an increase in perturbation also lead to an increase in coverage. The case is discussed in Section 6.4.3. Unit Test 1 can be reengineered in order to test quality indices as given in Unit Test 2.

Unit-Test 2 (Basic Quality Index)**.** *For a fixed quality measure and fixed cluster sizes, an increase (decrease) in the perturbation of a random pre-clustered graph generator must not cause an increase (decrease) in quality of the reference clustering.*

Unlike the previous unit test, we are not aware of a violation of Unit Test 2 for any suitable index. However, special attention has to be paid to worst-case indices that can have big jump discontinuities. For example, consider intra-cluster conductance, which rates star graphs with a maximum score of 1, but inserting an arbitrary edge in such a graph drastically decreases the scores.

The quality index coverage and random pre-clustered graph generators are reliably simple and thus suffice as a first indicator. In order to reveal additional drawbacks of the concepts to be tested and in order to reduce any potential correlations between the considered concepts in the experiments, each can be replaced by a related or more sophisticated one, such as modularity or geometric generators. On the one hand, this provides more meaningful insights, on the other hand, more complex artifacts can arise (see Section 6.4.2). An adaptation of Unit Test 2 for structural similarity is given in Unit Test 3.

Unit-Test 3 (Basic Comparators)**.** *For a random pre-clustered graph generator, the distance between the reference clustering and a clustering that is obtained from the reference clustering by randomly exchanging nodes, may not increase with increasing perturbation. Additionally, it has to be greater than or equal to the distance between two clusterings that were both obtained by randomly exchanging nodes of the reference clustering.*

The founding intuition is that first the distance between clusterings with similar structural properties should depend on the significance of the clusterings. More precisely, comparing a high-quality clustering with a random one should yield a larger distance than comparing two random clusterings, although their size distributions are similar. Second, the test also ensures, that if the quality score decreases and thus the high-quality clustering becomes random as well, then the distance does not increase. Obviously, a comparator that considers clusterings only as partitions passes Unit Test 3, since the partitions have similar structural properties, such as (almost) matching sizes of clusters, and the comparator will be more or less constant and be independent of the significance of the clusterings.

Besides replacing fundamental concepts in the basic unit tests, each can be refined in order to focus on individual properties of generators, algorithms, indices, or comparators.

6.3.2. Advanced Strategies

Recalling our clustering paradigm of intra-cluster density and inter-cluster sparsity, an extension of the Unit Tests 1 and 2 distinguishes between the two aspects of the paradigm. More precisely, quality measures should exhibit appropriate behavior with respect to both aspects.

Unit-Test 4 (Comprehensive Sensitivity of the Paradigm's Aspects)**.**
Keeping inter-cluster sparsity roughly constant, for a fixed quality index and fixed cluster sizes, an increase (decrease) in the intra-cluster density of the random pre-clustered graph generator must not cause a decrease (increase) in quality of the reference clustering. Analogously, keeping intra-cluster density roughly constant, an increase (decrease) in the inter-cluster sparsity must not cause a decrease (increase) in quality of the reference clustering.

The founding motivation is that quality indices should react accordingly to each parameter of the perturbation, ruling out one-sided dependencies. On the other hand, a failure of this unit test can either imply such a defect or suggest a reduction of the parameter set. The evaluation of quality indices is a necessary foundation for benchmarking algorithms. One corresponding unit test is given in Unit Test 5.

Unit-Test 5 (Algorithm)**.** *Let $\mathcal{G}$ be a generator passing Unit-Test 1 and* index *be an index passing Unit-Test 2. For small perturbation, the quality of the clustering calculated by an algorithm has to be close to the quality of the clustering used by the generator. Analogously, large perturbation must result in greater quality than the reference clustering of the generator.*

For small perturbation, the initial clustering used by the generator should be highly significant and thus it is very unlikely that a much better clustering exists. Therefore, a decent clustering algorithm has to find a clustering with similar quality. In the case of large perturbation, the algorithm should find a better clustering due to the fact that the generated graph structure does not significantly represent the initial clustering. A failure of Unit Test 5 for small perturbations indicates a potential defect in the algorithm. However, detecting the reason for a failed test for large perturbations is quite challenging. Potentially each of the three components or yet unknown interdependencies may cause the failure. For example, if the perturbation exceeds a certain level, the generated graph likely contains no significant clustering and thus the algorithm cannot extract one. We detail a more specific example in Section 6.4.3. Similar to the duality of Unit Test 1 and 2, Unit Test 5 can be reengineered in order to obtain a stronger test for generators.

Unit-Test 6 (Generator)**.** *Let $\mathcal{A}$ be an algorithm passing Unit Test 5 and* index *be the corresponding index used in that unit test. The expected behavior of generators should be the following: The initial clustering used by a generator has to be at least as significant as the clustering calculated by $\mathcal{A}$.*

Note, that Unit Test 5 and 6 can be alternately executed in order to find and evaluate improvements of algorithms and generators. The adaption for comparators is given in Unit Test 7.

Unit-Test 7 (Quality versus comparators)**.** *Let $\mathcal{G}$ be a generator passing Unit Test 1 and* index *be an index passing Unit Test 2. The distance between the reference clustering and a clustering obtained from the reference clustering by moving nodes in order to decrease the quality with respect to* index*, may not increase with increasing perturbation.*

Unit Test 7 tests the dependency between distance and quality as the structural similarity of the clusterings decreases. If many nodes are moved in order to decrease the quality, the distance should increase for fixed perturbation. On the other hand, if the number of moved nodes is fixed and the perturbation is increased, the distance should decrease or at least not increase due to the loss of significance in the clusterings.

Since tests have been now defined for generators, algorithm, quality indices, and comparators, one might be tempted to combine all previous unit tests in one such as: For a fixed generator, quality index, algorithm, and comparator, all passing corresponding unit tests, the clustering calculated by the algorithm and the reference clustering of the generator should exhibit similar quality with respect to the given index and a small distance with respect to the given comparator at least for small perturbations. Indeed such a test sounds very intuitive and probably will hold for many combination. However, consider the following case, where the relation between two different quality measures is evaluated. For example, performance and modularity. The generator selects only cycles where the reference clustering has maximum performance (see Lemma 3.14), the algorithms computes a clustering with maximum modularity (see Theorem 3.26), the quality index is coverage, and the comparator is an arbitrary one. Since the reference clustering contains only clusters of size up to three and the calculated clustering has only clusters of size roughly square root of the number of nodes, the coverage scores will be fairly different and the distance will be large. Thus the test fails, however, this is not due to a failure in one of the component but due to the unrelatedness of the two quality measures at least on the tested graphs. The given example is very artifical and we strongly believe that the stated test would properly extend our current unit tests. However, we explicitly do not include it due to large degree of complexity and interdependency which it would introduce. In general, all above mentioned unit tests constitute our benchmark foundation and can be combined in order to develop a sound test suite. Theoretical and experimental insight should be incorporated in the unit tests to further deepen the understanding. We present a collection of performed unit tests including their results and interpretations in the next section.

6.4. Results and Re-Engineering

We present several evaluations of algorithms, generators, comparators, and quality indices according to our introduced framework. Many of the results have been previously published in [24, 34, 33, 23, 32].

Statistical significance is important for evaluating clustering techniques. In our case, we consider the average of the selected quality indices and repeated each experiment at least 50 times and at most 1000 times, but stopped when the length of the confidence interval fell below 0.1 with a probability of 0.95.

We selected the Markov Clustering (MCL) and the greedy algorithm for optimizing modularity, both described in Section 5.3, as illustrating examples for the unit tests. All four presented algorithms are used for a mutual comparison.

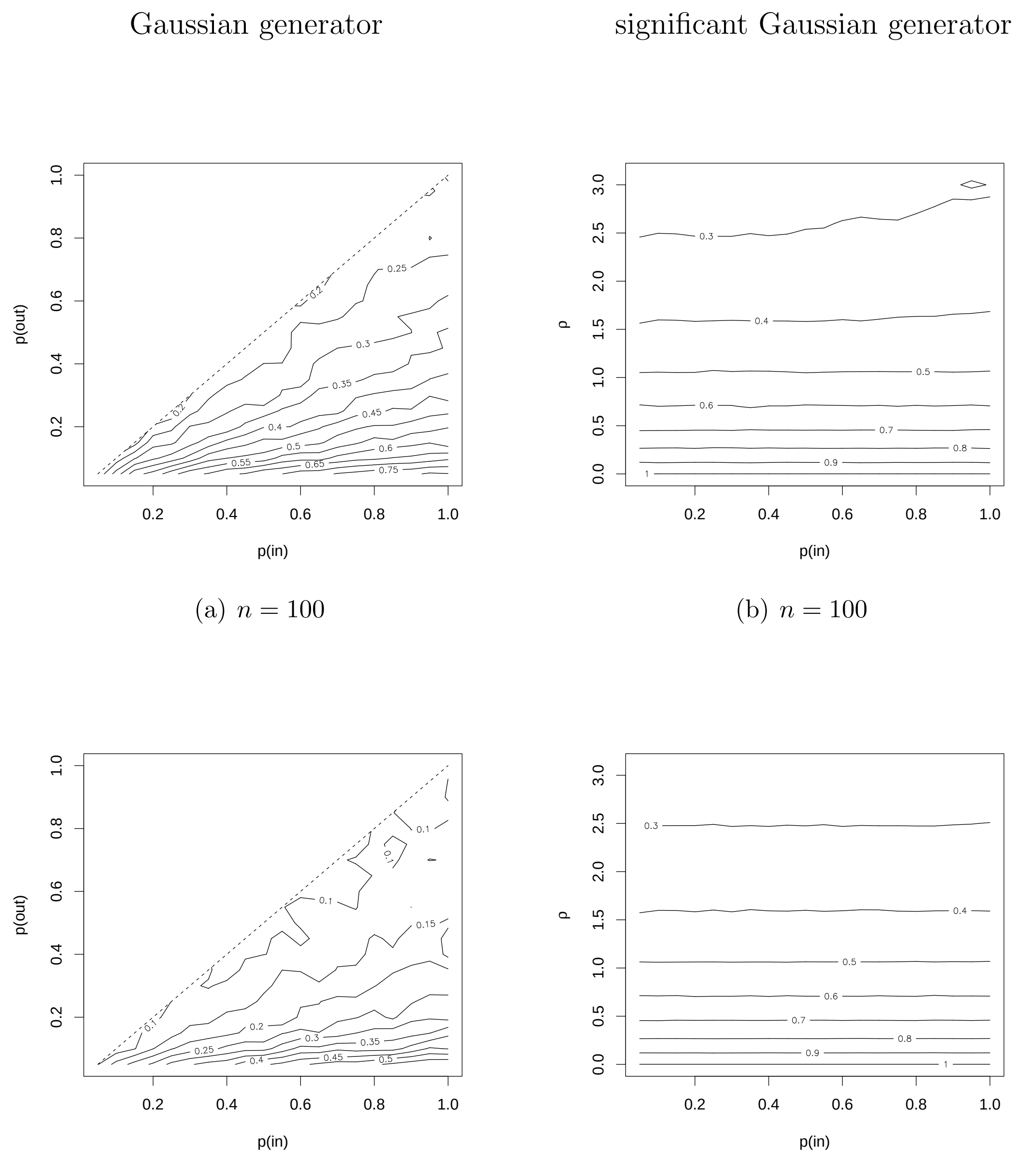
In addition to the quality indices introduced in Section 3, we consider the arithmetic mean of the average density inside the clusters and the sparsity between clusters. The formal definition is given in Equation (6.2); we use the convention, that the fraction in the term intra-cluster density for a singleton is defined to be 1, and that the inter-cluster sparsity of the 1-clustering is also 1.

$$\mathsf{density}(\mathcal{C}) := \frac{1}{2}\left(\underbrace{\frac{1}{|\mathcal{C}|}\sum_{C\in\mathcal{C}} \frac{|E(C)|}{\binom{|C|}{2}}}_{\text{intra-cluster density}} + \underbrace{1 - \frac{\left|\overline{E(\mathcal{C})}\right|}{\binom{n}{2} - \sum_{C\in\mathcal{C}}\binom{|C|}{2}}}_{\text{inter-cluster sparsity}} \right) \tag{6.2}$$

6.4.1. Generators

Random Generators As a first result, we verify the observation which lead to the definition of the significant Gaussian generator originally, namely, the fact that for increasing number of nodes n and a fixed pair $(p_{\text{in}}, p_{\text{out}})$, the growth of the inter-cluster edges exceeds the growth of the intra-cluster edges. Recall that coverage is just the normalized number of intra-cluster edges (with respect to the total number of edges). Figure 6.2 exemplarily illustrates this fact for two different numbers of nodes. By comparing Figures 6.2(a) with 6.2(c), the decrease is roughly 20% of the edges; while for small perturbation roughly 80% of the edges are intra-cluster edges for $n = 100$, only 60% remain in the case of $n = 1000$. Similarly, in the "random" case of $p_{\text{in}} \approx p_{\text{out}}$, 20% of the edges are still intra-cluster edges in the case of $n = 100$, in contrast to the 10% for $n = 1000$. Beside this artifical behavior of Gaussian generator, both generators successfully pass unit test 1.

By replacing coverage with inter-cluster conductance, modularity, or density, the observation still holds as can be seen in Figure 6.3. First of all, all three indices could be used in stead of coverage and both generators would successfully pass unit test 1. Second, modularity and inter-cluster conductance are (almost) independent of the parameter p_{in} for the significant Gaussian generator. This indicates, that the significance of the reference clustering is not correlated with the density of the clusters. Note that such 'feature' can be quite useful, when evaluating algorithms that are specifically designed for sparse graphs. In contrast, density cannot exhibit the same behavior, since it is explicitly defined by the local densities.



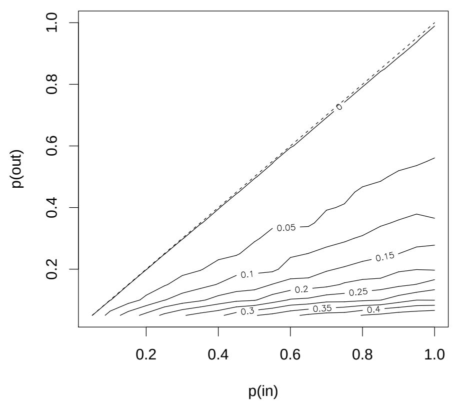
(a) $n = 100$ (b) $n = 100$

(c) $n = 1000$ (d) $n = 1000$

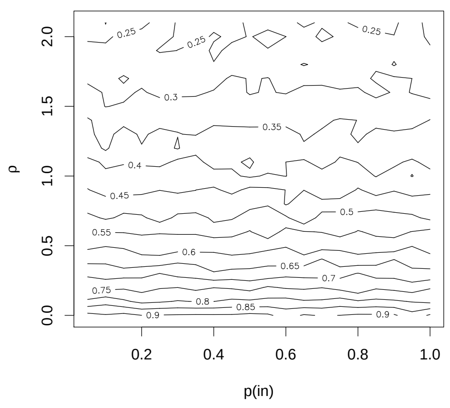
Figure 6.2.: Comparison of Gaussian generator and significant Gaussian generator using the quality index coverage.

Gaussian generator

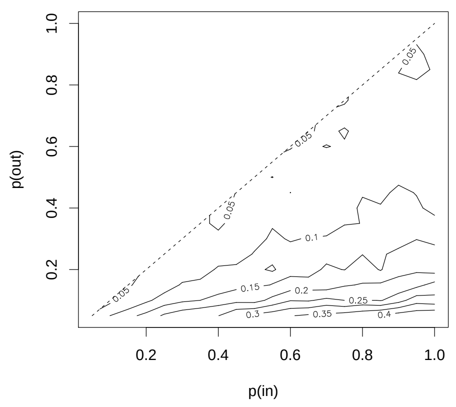
significant Gaussian generator



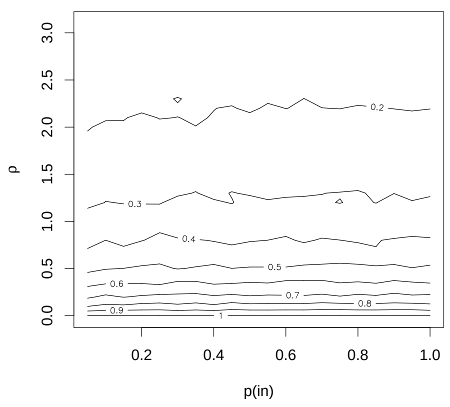
(a) modularity



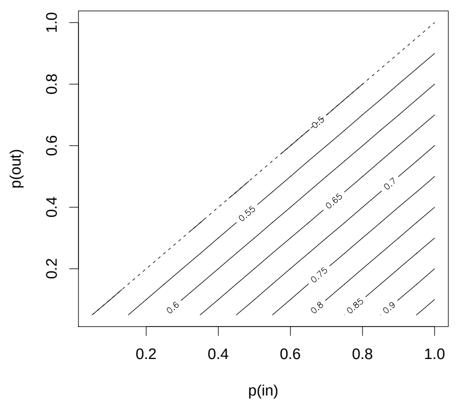
(b) modularity



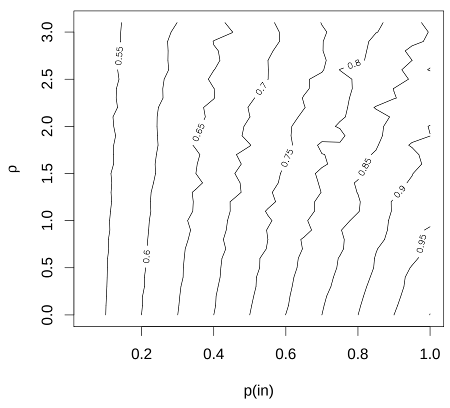
(c) inter-cluster conductance



(d) inter-cluster conductance



(e) density



(f) density

Figure 6.3.: Evaluation of the quality of the reference clustering for Gaussian generator and significant Gaussian generator and number of nodes $n = 1000$.

More about the behavior of the indices and its implications is given in Section 6.4.2.

Geometric Generators The validation of the attractor generator is presented in Figure 6.4. As can be observed, the quality of the reference clusterings does

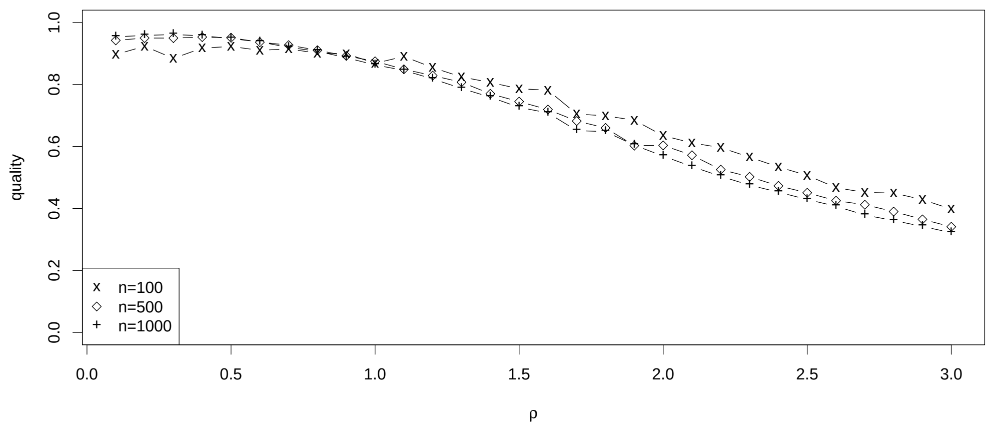
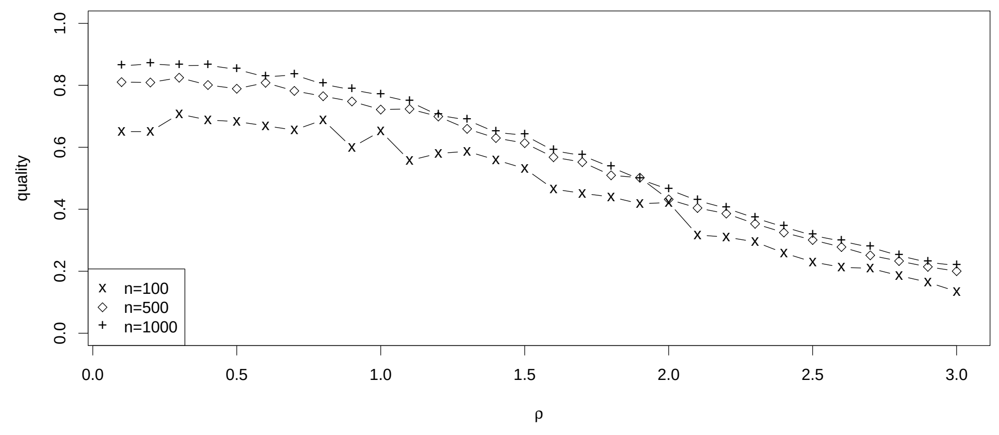


Figure 6.4.: Verification of Unit Test 1 for attractor generator.

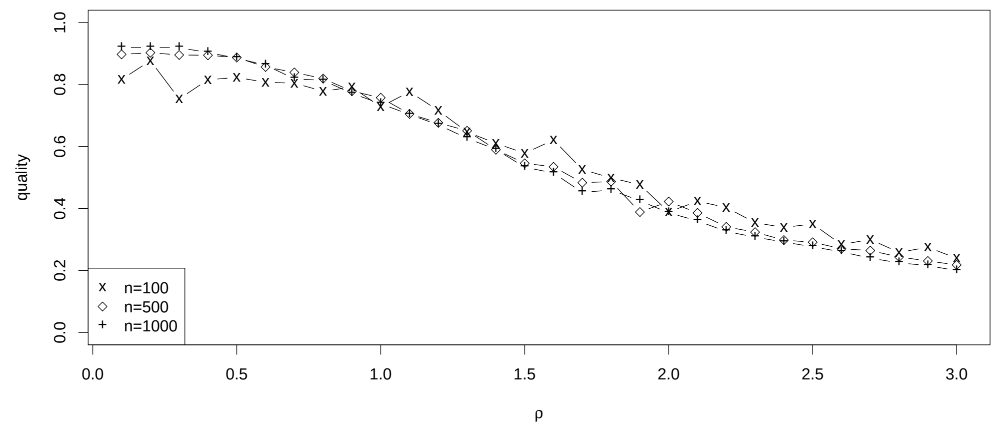
not heavily depend on the number of nodes. The generator successfully passes Unit Test 1, since coverage decreases for increasing $\rho \geq 0.5$. In the short interval of $[0, 0.5]$ coverage slightly increases, but not significantly. This phenomena is due to the fact that with increasing density value ρ, the distance threshold for connecting to two nodes decreases and thus the growth of intra-cluster edges briefly exceeds that of inter-cluster edges for very sparse graphs. Similar to the two random generators, the attractor generator still passes Unit Test 1, if coverage is replace by modularity or inter-cluster conductance as can be seed in Figure 6.5.

6.4.2. Indices

As has been already observed in the last section (Figure 6.3), modularity, inter-cluster conductance, and density pass Unit Test 2 when using Gaussian generator or significant Gaussian generator as generator. Furthermore all three pass Unit Test 4 for the Gaussian generator, while only density is sensitive to both parameters of the significant Gaussian generator. The other two indices do not fail the unit test, since the quality remains the same for varying the probability of the existence of intra-cluster edges. As previously mentioned, this need neither be a defect in the generator nor the quality indices. It also does not indicate a reduction of the parameter set of significant Gaussian generator, since intra-cluster conductance is sensitive to p_{in} (Figure 6.6). However, this dependency is rooted in fact, that intra-cluster conductance considers only the induced subgraph (of clusters) and its density. Thus, intra-cluster conductance has to change for varying probability p_{in}. As has been shown in Lemma 3.4, intra-cluster conductance has a maximum, if all clusters

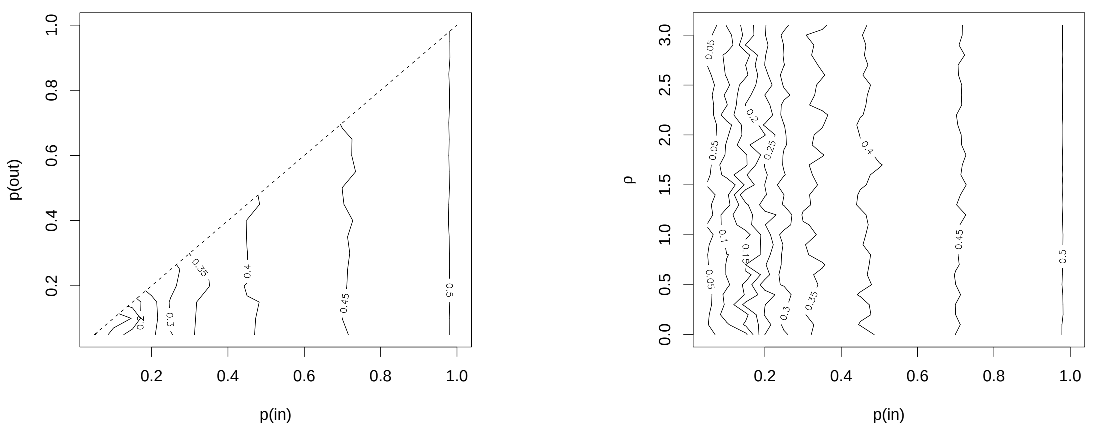


(a) modularity



(b) inter-cluster conductance

Figure 6.5.: Quality of the reference clustering of attractor generator evaluated with modularity and inter-cluster conductance.



(a) Gaussian generator (b) significant Gaussian generator

Figure 6.6.: Evaluating intra-cluster conductance for Gaussian generator and significant Gaussian generator using $n = 1000$ nodes.

are stars. Since the attractor generator creates local stars and then depending on the density score inserts "local" edges, intra-cluster conductance fails Unit Test 2 when using this generator. On the other hand, for suitable density score, all clusters are cliques, thus intra-cluster conductance will be constant for a sufficiently large threshold (see Figure 6.7).

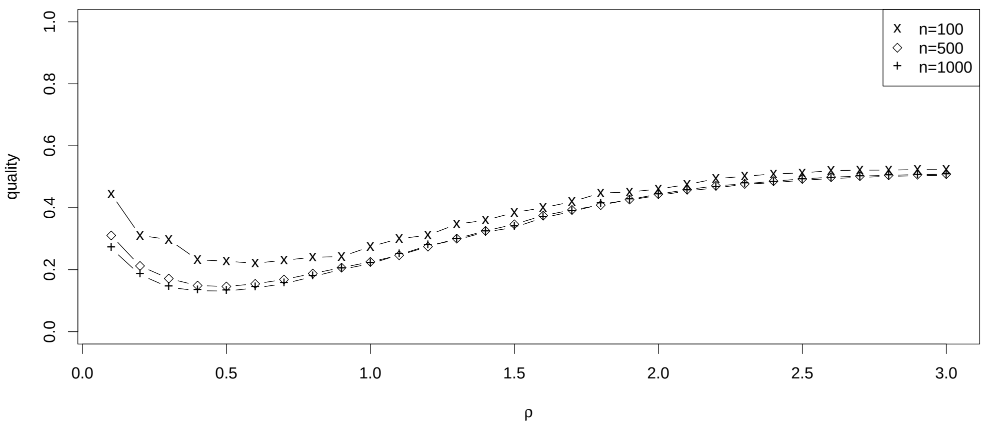
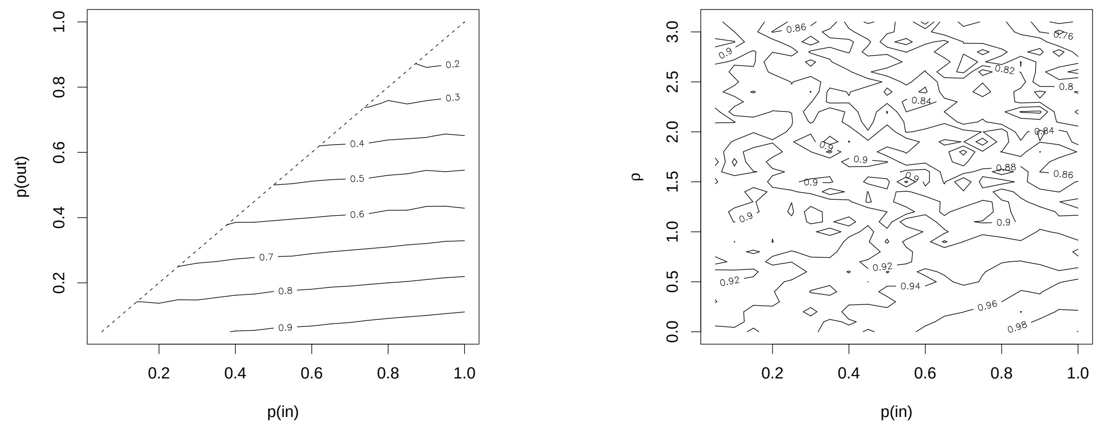


Figure 6.7.: Evaluating intra-cluster conductance for the attractor generator using $n = 1000$ nodes.

Another artifact arises for the quality index performance. While it successfully passes Unit Test 2 and 4 for the Gaussian generator (Figure 6.8(a)), it has very



(a) Gaussian generator (b) significant Gaussian generator

Figure 6.8.: Evaluating performance for Gaussian generator and significant Gaussian generator using $n = 1000$ nodes.

homogenous scores for the significant Gaussian generator. Still the cases of slightly perturbed disjoint cliques are rated highest. On the other hand, the generated graphs with $\rho \approx 2$ have only a coverage score of 0.35 and still performance rates the clustering with a score in the interval $[0.84, 0.9]$. This 'defect' is due to the fact, that the generated graphs are quite sparse (Figure 6.9) and contain at most 30% of all possible edges. Summarizing, performance should not be used, when the considered

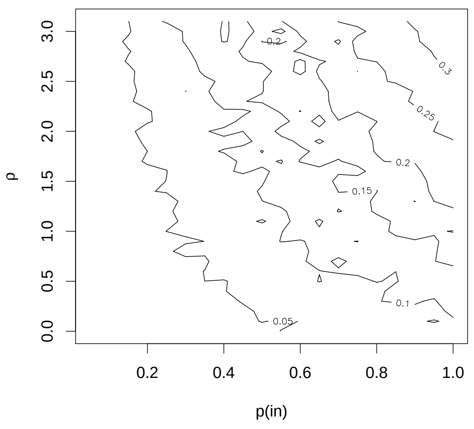


Figure 6.9.: (Average) Normalized number of edges of the graphs generated by significant Gaussian generator with $n = 1000$ nodes.

graph is very sparse due to its insensitivity.

During the evaluation of the generators in the last section and our first observations in this section, density turned out to be a fairly good index. However, an evaluation using the attractor generator reveals a pit fall. As can be observed in

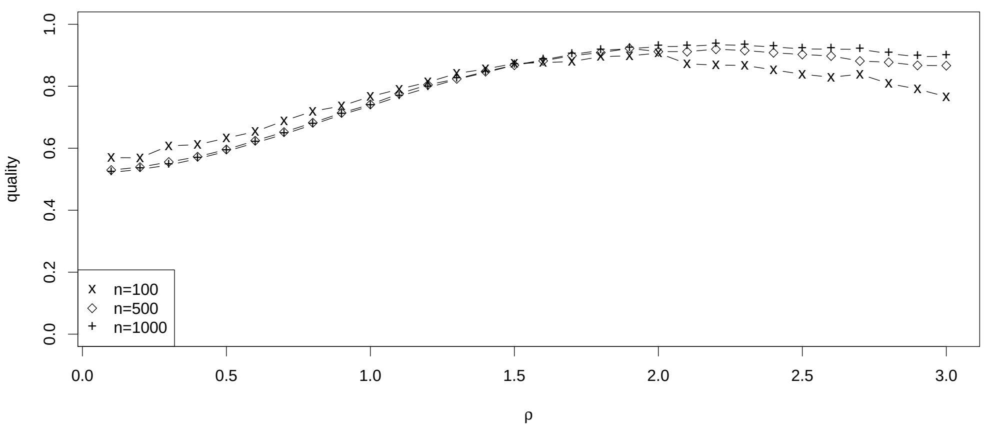


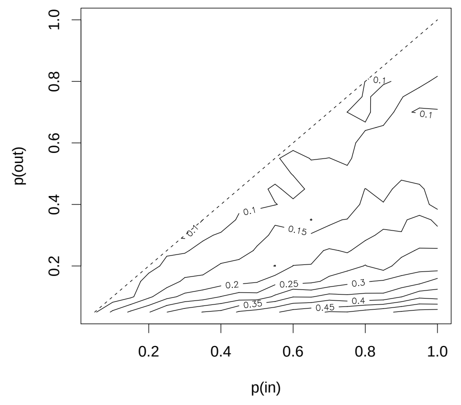
Figure 6.10.: Evaluating density for the attractor generator using $n = 1000$ nodes.

Figure 6.10, the quality index density increases with increasing density parameter ρ. Since the attractor generator inserts edges according to the density parameter, the intra-cluster density increases. Furthermore, the overall density, i. e., the ratio of realized edges to the total possible number of edges, is still relative small, thus, analogously to performance, density is dominated by the intra-cluster density. Summarizing, density should not be used, when the perturbation of generator is correlated with the overall density.

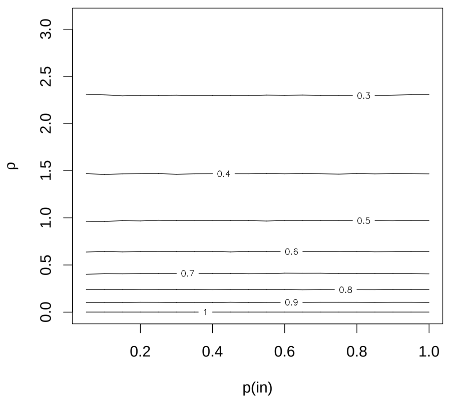
The final example target the worst-case index inter-cluster conductance. The corresponding average-case index is defined as given in Equation (6.3) and the evaluation for the Gaussian generator and significant Gaussian generator is give in Figure 6.3.

$$\delta_{\text{avg}}(\mathcal{C}) := \begin{cases} 1, & \text{if } \mathcal{C} = \{V\} \\ 1 - \dfrac{1}{k} \displaystyle\sum_{1 \le i \le k} \phi\left((C_i, V \setminus C_i)\right), & \text{otherwise} \end{cases} . \qquad (6.3)$$

Comparing theses figures with the ones for inter-cluster conductance (Figure 6.3(c) and 6.3(d)), we observe that the average-case version exhibits more or less the same behavior; the two scales are slightly shifted, which is to be expected. The high degree of similarity is due to the fact, that all clusters are almost equal in size, their density, and their connectivity to the remaining graph. As we observed in [24], (worst-case) inter-cluster conductance is heavily affected, if the generators produce a graph with exactly n nodes. In these cases, the probability that a significant smaller cluster exists, is very large, and thus this cluster will cause a significant smaller score with respect to inter-cluster conductance. In the next section, we note a similar artifact for algorithms.



(a) Gaussian generator



(b) significant Gaussian generator

Figure 6.11.: Evaluation of an average-case inter-cluster conductance for Gaussian generator and significant Gaussian generator using $n = 1000$.

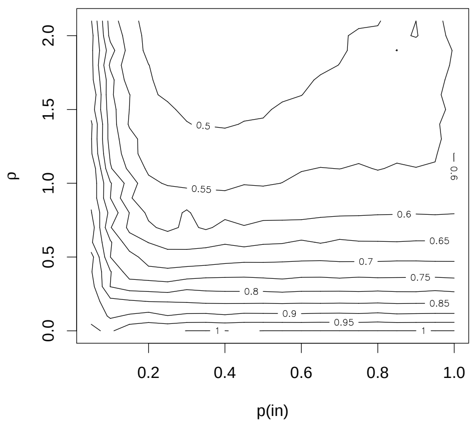
6.4.3. Algorithms

As already pointed out, the interpretation of unit test for algorithms is more complex, since potential defects may be intentional due to application-specific requirements. In the following, we consider two algorithms as illustrating examples.

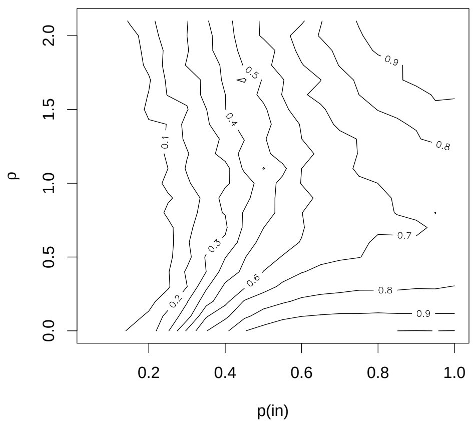
The first observation is that both algorithms successfully pass Unit Test 1 using the significant Gaussian generator, as can be seen in Figure 6.13. While the achieved coverage score of the greedy algorithms seems to be independent of p_{in}, MCL exhibits a threshold phenomena. For $p_{\text{in}} < 0.4$ the coverage of clustering found by MCL heavily depends on p_{in} and ρ. This potential defect is rooted in the realization of MCL: Since MCL iteratively squares the matrix of transition probabilities, applies a ‘rich-get-richer’ shifting, and a renormalization, the input graph may not be too sparse nor too dense. In the first case, the obtained clustering will surely contain trivial clusters, while in the later case, the clustering will likely contain on very large cluster. By tuning the parameters of MCL, one might counteract this feature, however we did not pursue this possibility.

By comparing these scores with the reference clustering (Figure 6.2(b)) with those of the calculated clusterings, it is evident that the greedy algorithm also passes Unit Test 5, while MCL only succeeds for small perturbation (large p_{in} and small ρ). This is again due density issues, mentioned above. Replacing coverage with density, both algorithms passes Unit Test 5; although, in the case of MCL and $p_{\text{in}} < 0.2$, the difference is not significant, which makes the decision a bit ambiguous. Similarly, the usage of modularity instead of coverage shows that both algorithms fail the corresponding unit test. Again, the differences for the greedy algorithms are hardly significant for large perturbations.

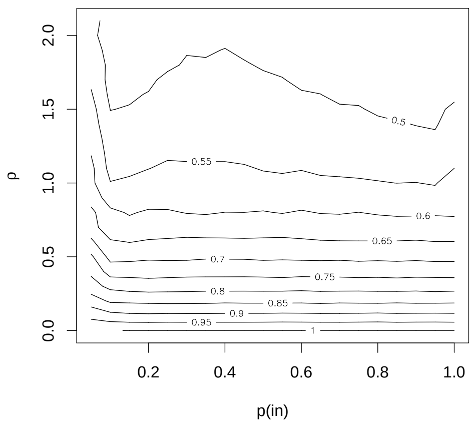
Summarizing these first observations, both algorithms have certain defects with respects to our unit test. Still they find very significant clusterings at least for



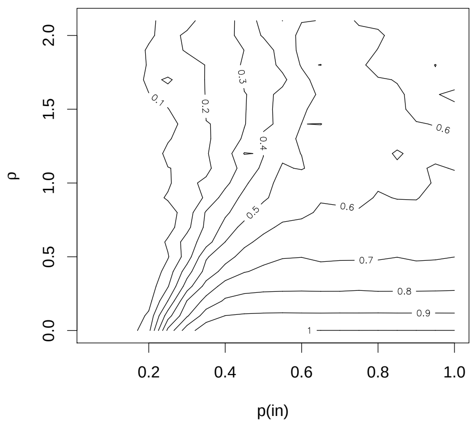
(a) greedy algorithm



(b) MCL

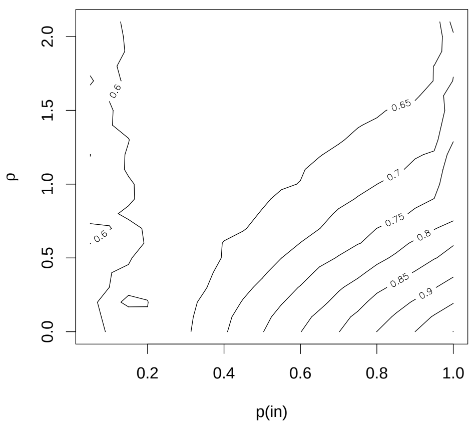


(c) greedy algorithm

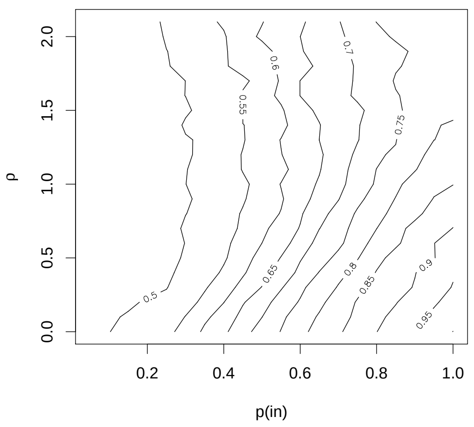


(d) MCL

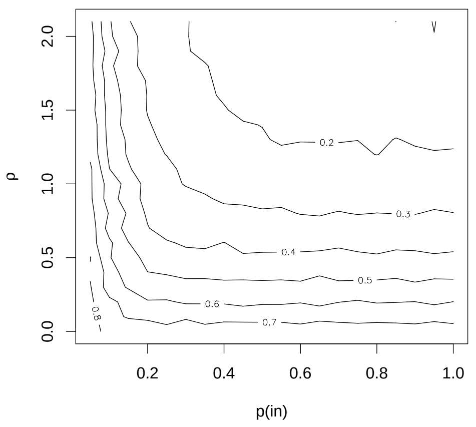
Figure 6.12.: Quality of the calculated clustering (with respect to coverage) using significant Gaussian generator with $n = 100$ (Figures 6.12(a) and 6.12(b)) and $n = 500$ (Figures 6.12(c) and 6.12(d)).



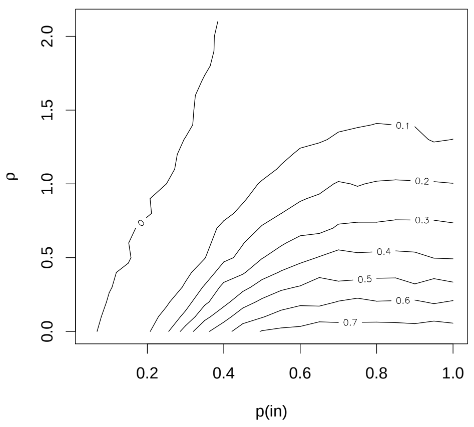
(a) greedy algorithm



(b) MCL



(c) greedy algorithm

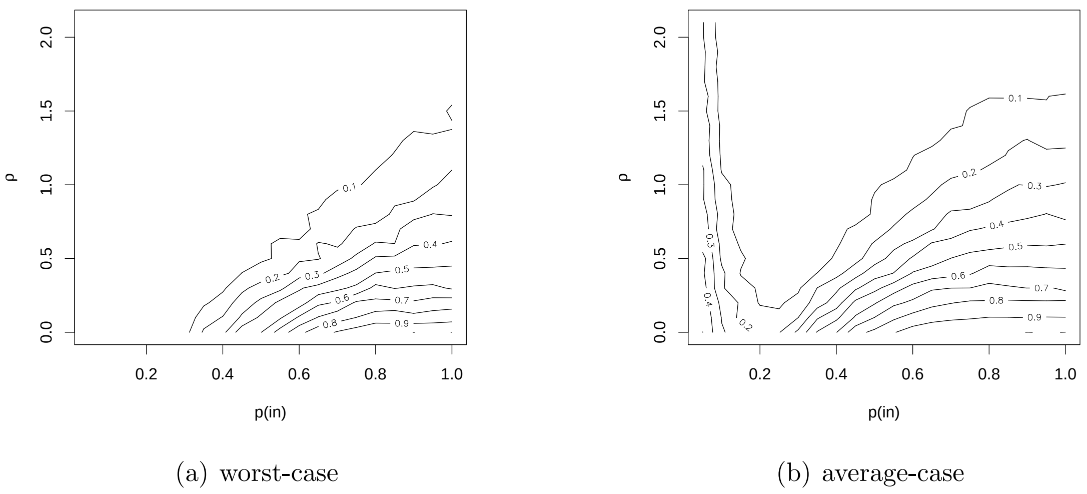


(d) MCL

Figure 6.13.: Quality of the calculated clustering with respect to density (Figures 6.13(a) and 6.13(b)) and modularity (Figures 6.13(c) and 6.13(d)) using significant Gaussian generator with $n = 100$.

graphs generated with small perturbation. Thus, the applicability of these algorithms remains an issue of the input data, parameter tuning, and desired clustering features.

As stated in the last section, the (worst-case) inter-cluster conductance is fairly sensitive to the (in)homogeneity of cluster sizes. The corresponding diagrams are given in Figure 6.14. In the case of small perturbations, both scores are fairly



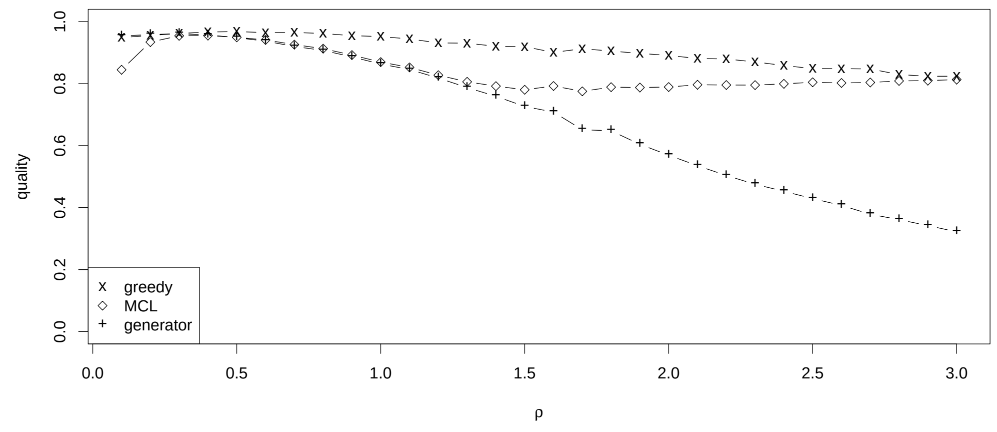
(a) worst-case (b) average-case

Figure 6.14.: Quality of the MCL clustering with respect to inter-cluster conductance using significant Gaussian generator with $n = 100$.

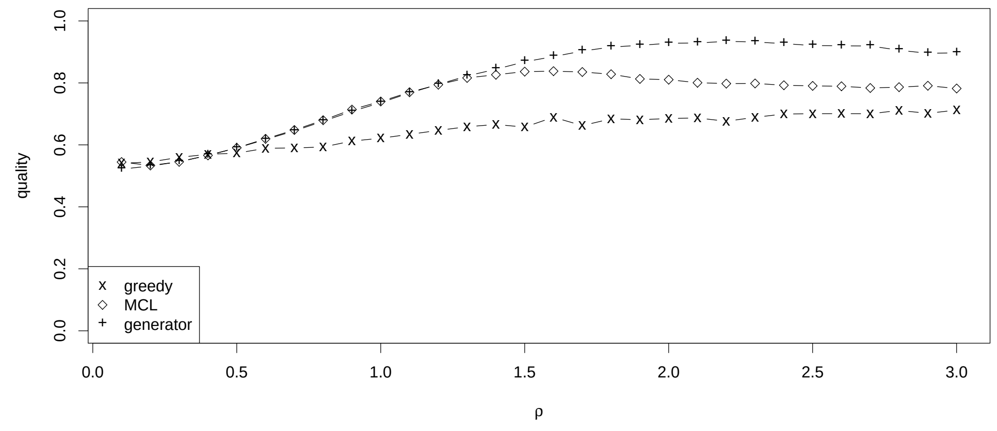
similar, while for large perturbations and sparse graphs ($p_{\text{in}} < 0.3$ and $\rho < 1$) the average-case inter-cluster conductance is significantly larger. This verifies that the calculated clustering has at least one very small cluster.

The final example, uses the greedy algorithm in combination with coverage and density in order to evaluate the attractor generator using Unit Test 6. As can be seen in Figure 6.15(a), the attractor generator partially meets the condition of Unit Test 6 with respect to coverage. More precisely, for small perturbations ($\rho < 0.75$), the clustering of the generator and the calculated clustering of the greedy algorithms are the same (in the statistical sense), while for large perturbations the coverage of reference clustering is significantly worse than the calculated one. In the case of density, the attractor generator successfully passes Unit Test 6. Although, this implies a certain validity of the generator, the result is ambiguous, due to the fact, that density fails the simple Unit Test 2 when using attractor generator.

Summarizing, the two advanced tests, Unit Test 5 and 6, examine certain intuitive patterns for the behavior of generators and algorithms. Both may easily fail to the strict formulations. For example the greedy algorithm would pass Unit Test 5, if the demanded relation between generated and calculated clustering was relaxed. Such reformulations could be introduced as general unit test, however, their validity heavily depends on the corresponding application.



(a) coverage



(b) density

Figure 6.15.: Quality of the reference clustering and the calculated clusterings (using greedy algorithm and MCL) of attractor generator evaluated with coverage and density.

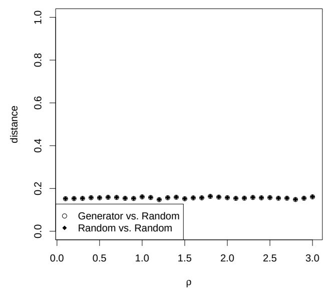
6.4.4. Comparators

In Section 4 comparators for graph clusterings have been summarized. Although the summary contained well-known measures such as Rand or Meila-Heckerman as well as our proposals which respect the graph structure in addition, the usability for almost all measures remains open. In the following first experimental validation using our unit tests, we show that lattice-based approaches are independent of the distribution of the edges and that both the extended pair comparators and the editing-set comparators still have drawbacks and artifical behavior. Parts of this evaluation can be found in [31, 33].

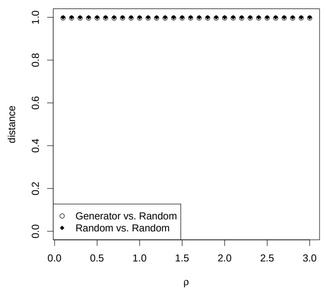
As a first result, we verify the already mentioned fact that lattice-based comparators are independent of the edgeset. We use the attractor generator and generate two clusterings that have the same size distribution as the reference clustering, but are otherwise random. The results are shown in Figure 6.16. As can be seen, all comparators are more or less constant and do not distinguish the two cases: reference clustering versus random clustering and random clustering versus random clustering. With the exception of Rand, all comparators yield a large distance between the two clusterings which is an appropriate behavior. The small distance measured by Rand is due to the fact that the size distributions are the same. The results for the same test using the graph-structural comparators are given in Figure 6.17. First note that Rand R^{A} and Editing Set Difference $\mathsf{ESD_n}$ successfully pass the unit test. For increasing perturbation ρ, the distance between the reference clustering and the random clustering decreases which indicates the loss of significance in the reference clustering. An interpretation of the behavior of Editing Set Difference for the comparison of two random clusterings is that as the perturbation increases the graphs become denser and thus the structural difference between the random clusterings with respect to the graph structure decreases. Interestingly, both Adjusted Rand $\mathsf{R}^{A}_{\mathrm{adj}}$ and Jaccard J^{A} do not distinguish the two cases. While in the case of Adjusted Rand, both distances are maximum and thus it passes the unit test, the situation for Jaccard is more complex. The means are not significantly different for most parameters ρ and thus one can argue that it also passes the unit test. On the other hand, the distances are not maximum and an experiment with less nodes revealed a significant gap between the two cases for small ρ. Thus the indifference for the experiment with 1000 nodes may be an artifact of the normalization and the sparsity of the graphs. Summarizing, the graph-structural comparators Rand, Adjusted Rand, and Editing Set Difference clearly passes Unit Test 3, while Jaccard remains open.

6.4.5. Quality Assessment of Algorithms

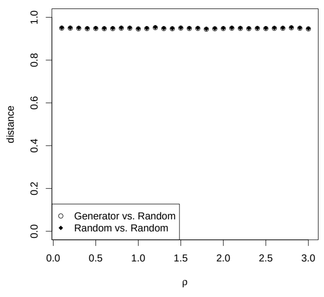
In the final part of the experimental evaluation, we compare the algorithms MCL, ICC, and GMC with each other. While the above unit tests present indicator, whether the quality of a clustering calculated with an algorithm can compete with the reference clustering or not, we focus now on their mutual comparison. Note that such task could formulated as unit tests as well and that comparators could provide additional insights. However, we choose not such an approach due to hidden



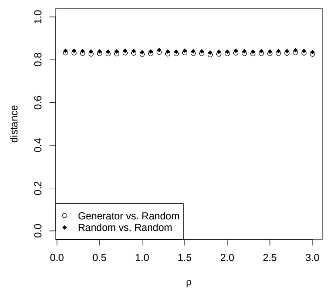
(a) Rand



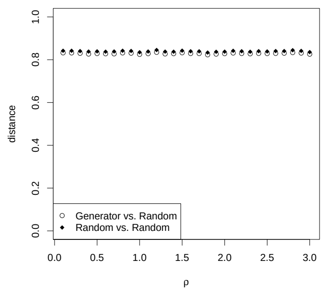
(b) Adjusted Rand



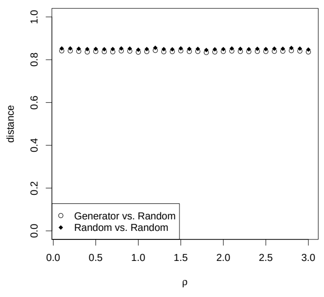
(c) Jaccard



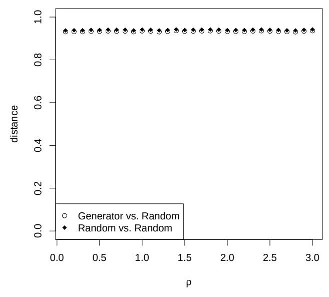
(d) van Dongen



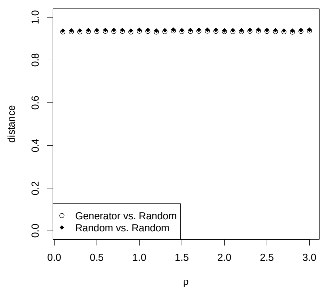
(e) Meila-Heckerman



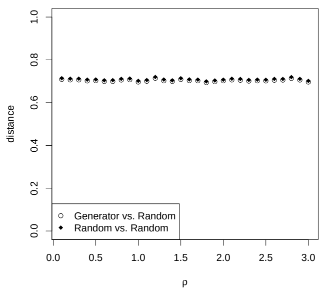
(f) F-Measure



(g) Strehl-Ghosh

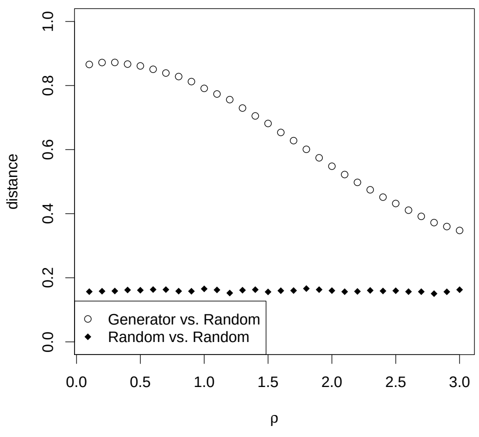


(h) Fred-Jain

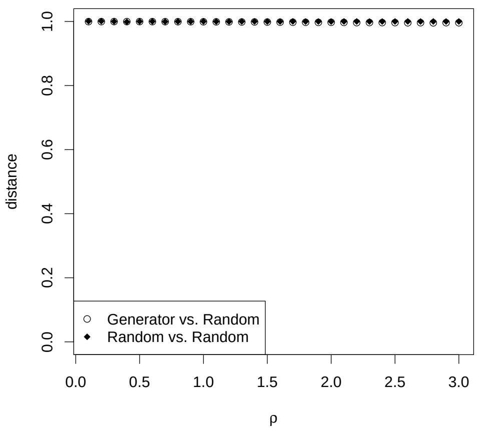


(i) Variation of Information

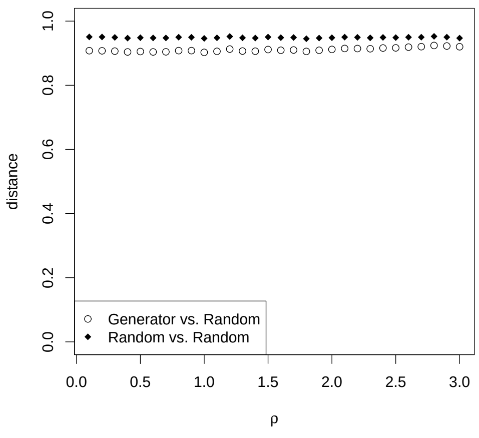
Figure 6.16.: Evaluation of the lattice-based comparators considering Unit Test 3 using the attractor generator with $n = 1000$ nodes.



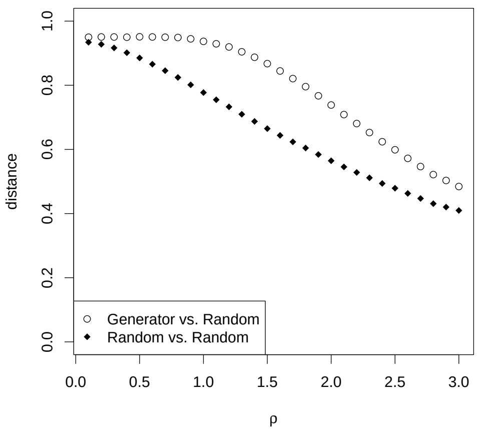
(a) Rand R^A



(b) Adjusted Rand $\mathsf{R}^A_{\text{adj}}$



(c) Jaccard J^A



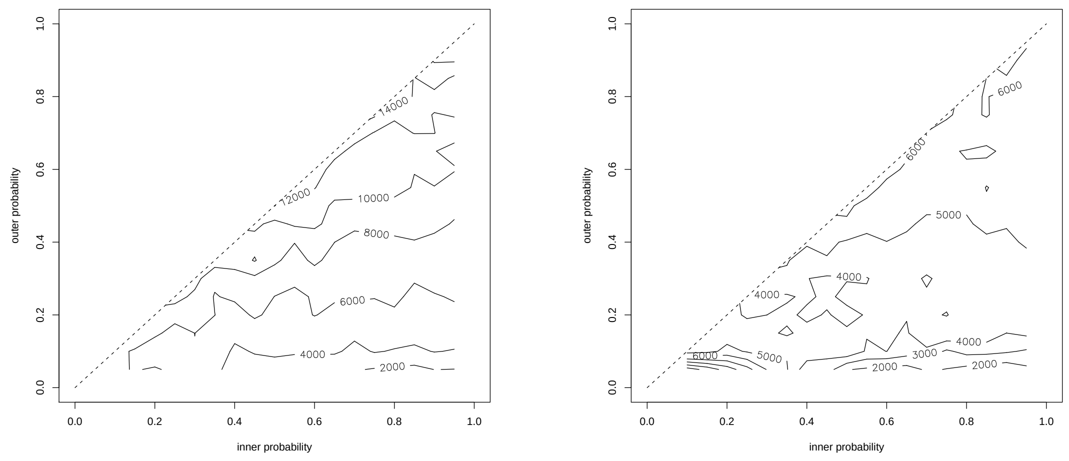
(d) Editing Set Difference $\mathsf{ESD_n}$

Figure 6.17.: Evaluation of the graph-structural comparators considering Unit Test 3 using the attractor generator with $n = 1000$ nodes.

dependencies. The experiments uses the Gaussian generator and the significant Gaussian generator with $n = 1000$. The free parameters of the algorithms are set to $e = 2$ and $r = 2$ in MCL, $\alpha^* = 0.4$ and $\alpha^* = 0.2$ in ICC, and dimension $d = 2$ in GMC. As objective function *quality* in GMC, coverage, performance, inter-cluster conductance, as well as their geometric mean are considered.

Running Time All presented clustering algorithms were implemented using sophisticated data structure and software engineering techniques. However, there are certain limitations, especially with respect to runtime measurements in Java which are very difficult. Since such measurements are rarely significant on small scales, none of the implementations were especially optimized with respect to running time. Nevertheless, the following results show certain tendencies.

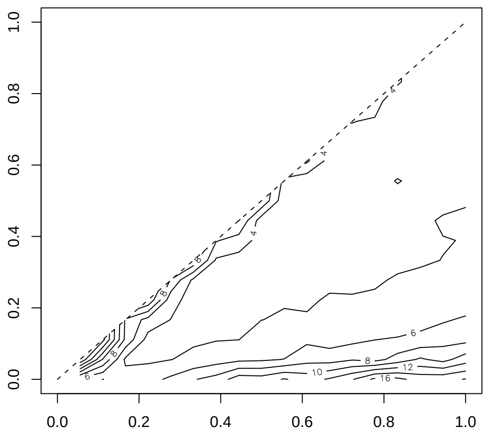
The experimental study confirms the theoretical statements in Section 5.3 about the asymptotic worst-case complexity of the algorithms. MCL is significantly slower than ICC and GMC. Not surprisingly as the running time of ICC depends on the number of splittings, ICC is faster for $\alpha^* = 0.2$ than for $\alpha^* = 0.4$. Note the coarseness of the clustering computed by ICC depends on the value of α^*. In contrast, all version of GMC were equally fast, except those that included intra-cluster conductance in their *quality* index. On sparse graphs GMC and ICC require roughly the



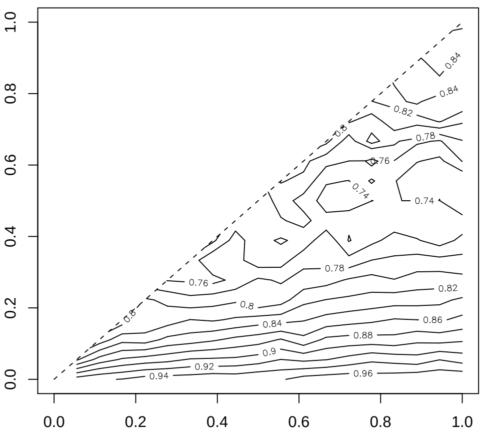
(a) GMC (b) ICC

Figure 6.18.: Running time of GMC and ICC where the x-axis represents the inner probability p_{in} and the y-axis shows the outer probability p_{out}. Time is measured in milliseconds.

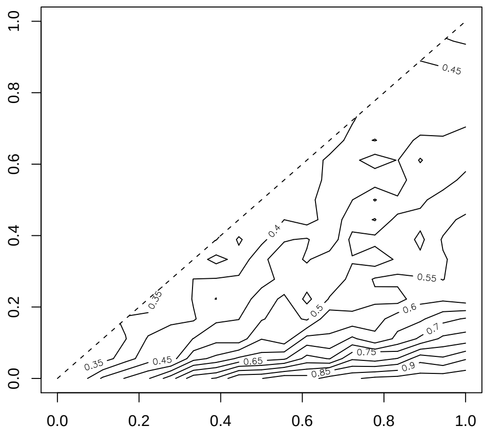
same amount of time, while ICC was up to two times faster on dense graphs. The complete results are given in Figure 6.18. ICC performs on dense graphs so well since the approximation of intra-cluster conductance yield large values and thus only a few number of cuts are calculated. In other words, the divisive structure of the ICC is more suitable for dense graphs than for sparse ones, while the agglomerative GMC benefits from a sparse edge set. Not very surprisingly the runtime depends much more on the outer probability p_{out} than on the inner probability p_{in} which is due to the fact that the number of potential inter-cluster edges is much larger than the number of potential intra-cluster edges (for most values of k).



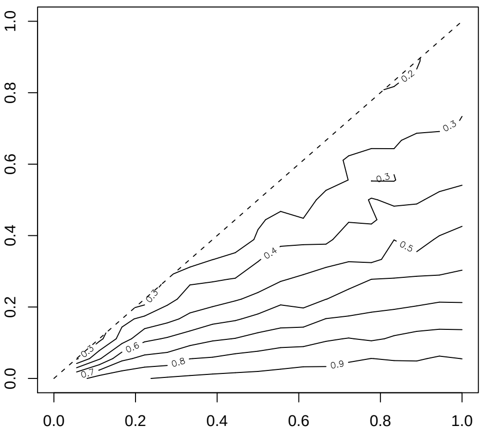
(a) number of clusters



(b) coverage



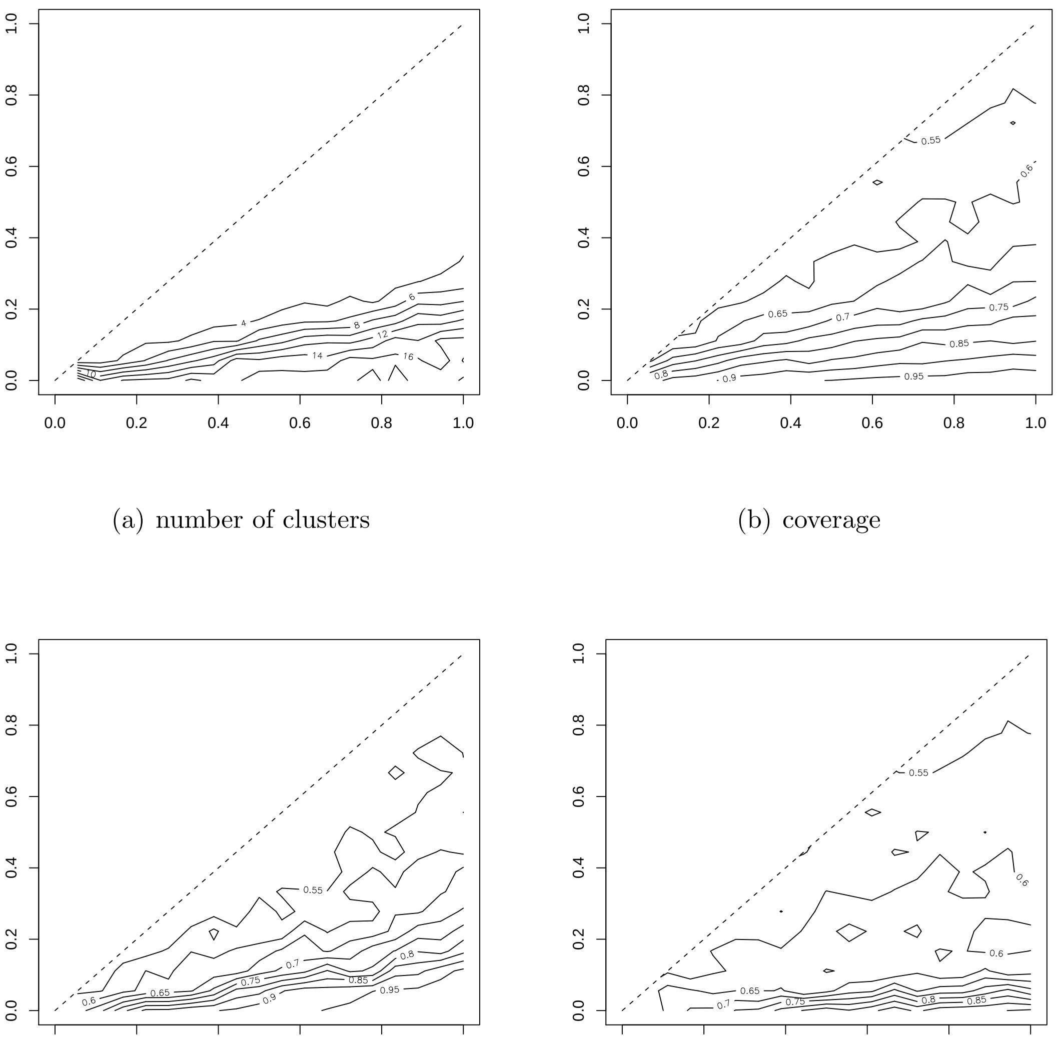
(c) performance



(d) inter-cluster conductance

Figure 6.19.: GMC using geometric mean of coverage, performance, and inter-cluster conductance

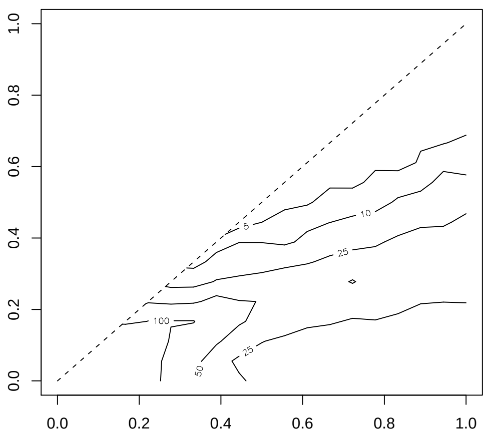
Mutual Comparison Figures 6.19–6.22 show the different quality indices for the different algorithms for the first group of experiments. All diagrams show the inner probability p_{in} as x-axis and the outer probability p_{out} as y-axis. A significant observation when comparing the three algorithms with respect to the quality indices regards their behavior for dense graphs. All algorithms (Figure 6.19(a), 6.21(a),



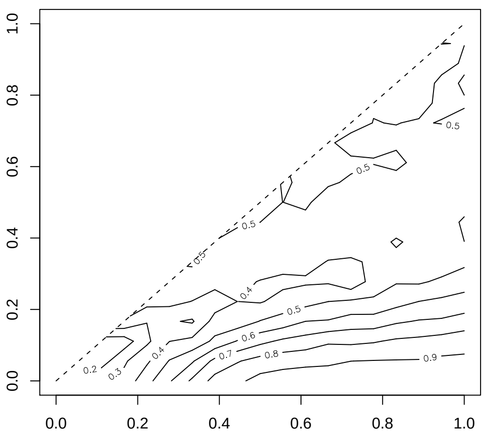
(a) number of clusters (b) coverage

(c) performance (d) inter-cluster conductance

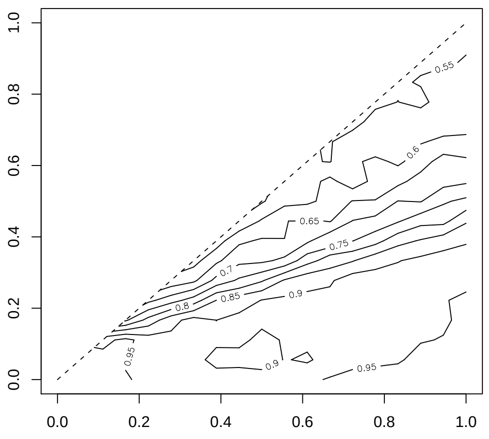
Figure 6.20.: ICC with $\alpha = 0.2$



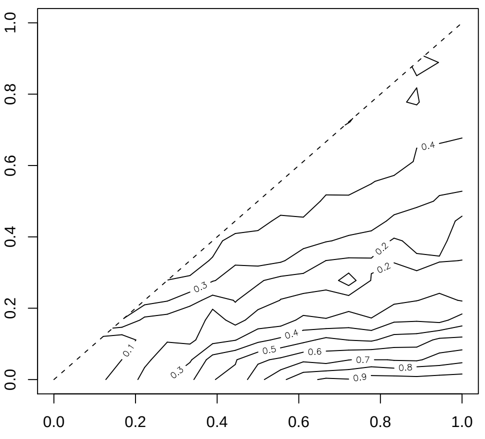
(a) number of clusters



(b) coverage

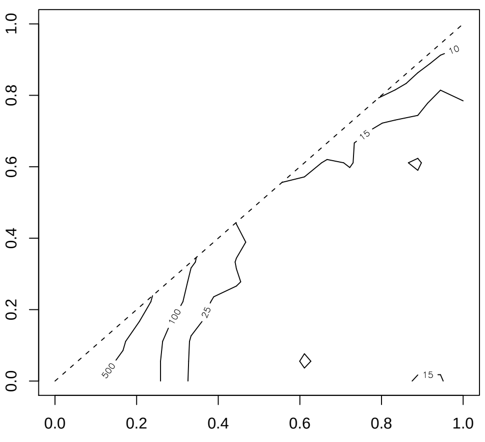


(c) performance

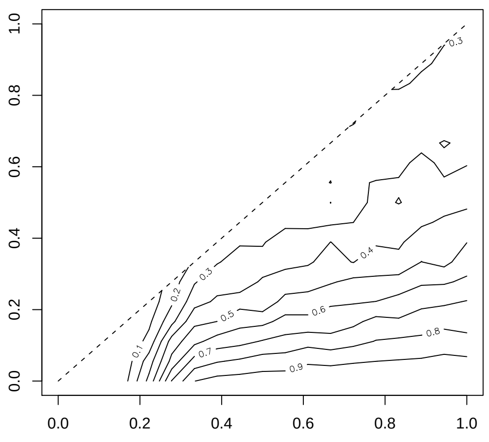


(d) inter-cluster conductance

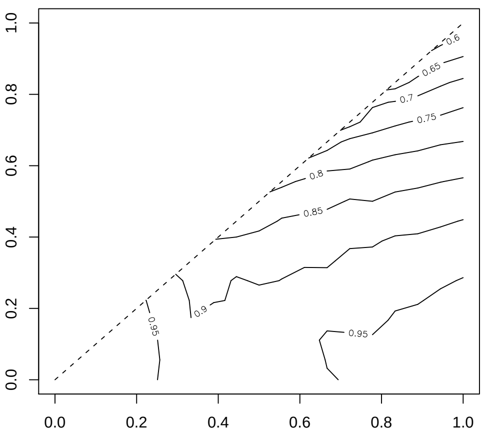
Figure 6.21.: ICC with $\alpha = 0.4$



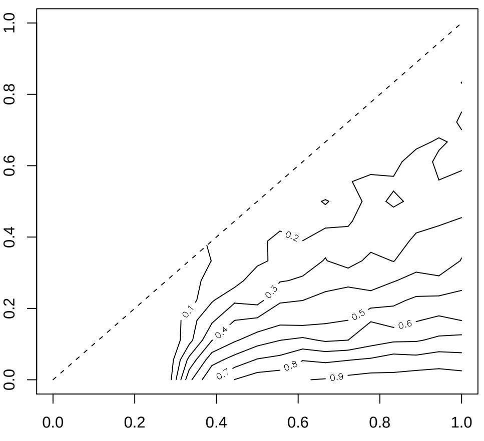
(a) number of clusters



(b) coverage



(c) performance



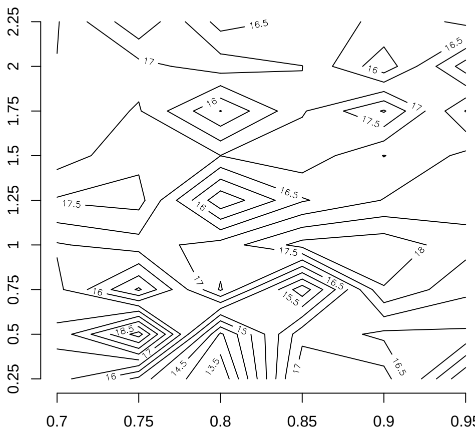
(d) inter-cluster conductance

Figure 6.22.: MCL

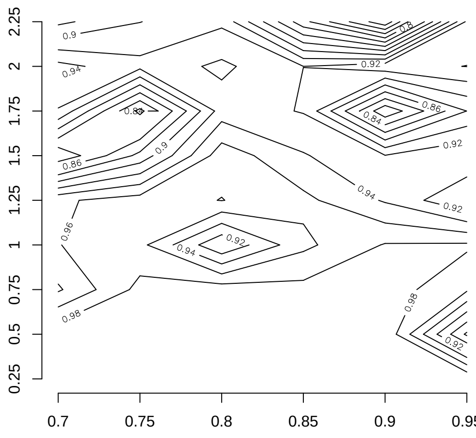
and 6.22(a)) have a tendency to return trivial or very coarse clusterings containing only few clusters. As mentioned previously, this is due to the fact that the number of potential inter-cluster edges is much larger than the number of potential intra-cluster edges. In contrast for sparse graphs, ICC and MCL only find clusterings with many clusters. This suggests modifications to at least incorporate bounds for the number of clusters in order to avoid too coarse clusterings. However, for ICC such a modification would be a significant deviation from its intended procedure. The consequences of forcing ICC to split even if the condition for splitting is violated are not clear at all. On the other hand, the approximation guarantee for intra-cluster conductance is no longer maintained if ICC is prevented from splitting even if the condition for splitting is satisfied. For MCL it is not even clear how to incorporate the restriction to non-trivial clusterings. In contrast, it is easy to modify GMC in such a way that only clusterings with bounded (from below, above, or both) numbers of clusters are computed. This is accomplished by limiting the search space of τ.

Both ICC and MCL are comparably good with respect to performance, although neither of them optimizes it explicitly. While GMC is not as good as them with respect to performance, it outperforms MCL with respect to inter-cluster conductance and ICC with respect to $\operatorname{cov}(\mathcal{C})$. Still, all three algorithms find clusterings with acceptable quality. Further more, the calculated clusterings similarly react to changes in the generation parameters, i. e., the quality drops when approaching random graphs (diagonal). More precisely, GMC (Figure 6.19(d)) and ICC (Figure 6.21(d)) are more sensitive (with respect to inter-cluster conductance) than the initial clustering. The results of the second group of experiments using the significant Gaussian generator are shown in Figures 6.23–6.25. All diagrams show the inner probability p_{in} as x-axis and the parameter ρ as y-axis. Recall that ρ roughly estimates the ratio of (expected) inter-cluster edges to (expected) intra-cluster edges. Intuitively speaking, the parameter ρ is inversely proportional to the significance of the initial clustering. The Figures 6.23-6.25 clearly illustrate that both GMC and ICC find a clustering that is very similar to the initial one with respect to quality.

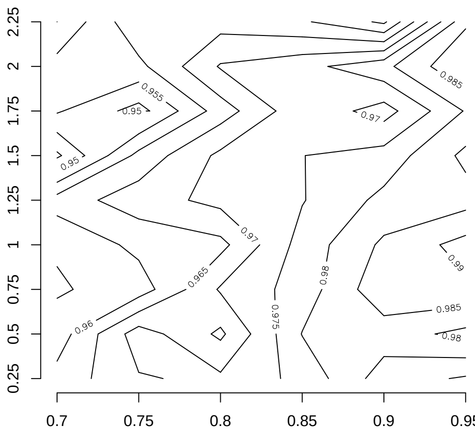
Summary The experimental evaluation of the algorithms provided several insights: asymptotic running time bounds are verified as well as generic applicability of certain paradigms, i. e., splitting performs better on dense graphs, while agglomeration is more suited for sparse graphs. Furthermore individual features and artifacts of the algorithms and choices of parameters are revealed. For example the relation between the cutting threshold of the ICC and the coarseness of the resulting clustering. Although the experiments were not founded on explicit unit test, one might derive specific tests in order to verify or falsify the applicability of improving heuristics with respect to the original algorithms. Thus such test provide a useful tool in the design and re-engineering of clustering algorithms.



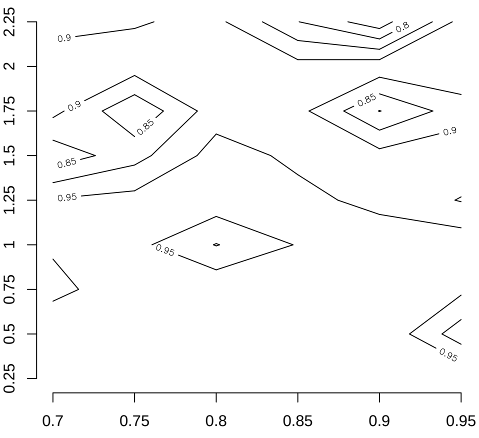
(a) number of clusters



(b) coverage

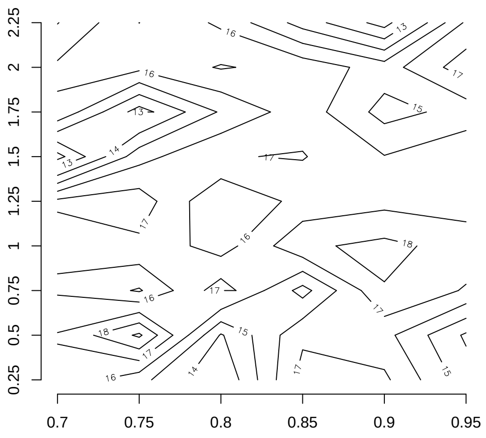


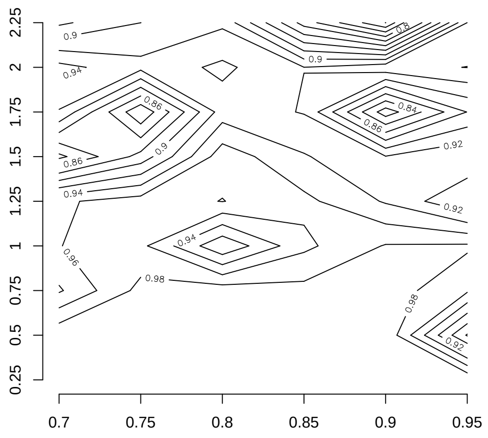
(c) performance

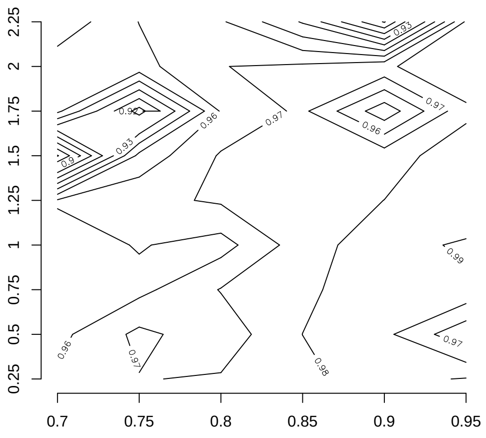


(d) inter-cluster conductance

Figure 6.23.: initial clustering


(a) number of clusters


(b) coverage


(c) performance

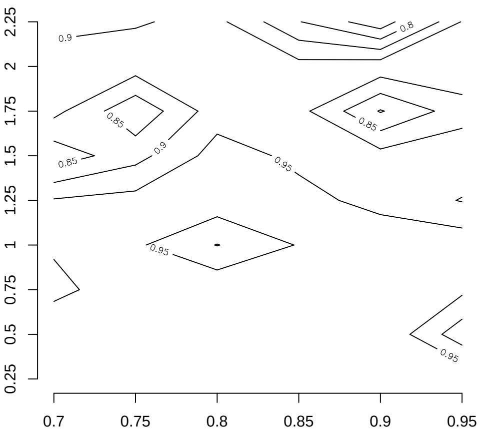
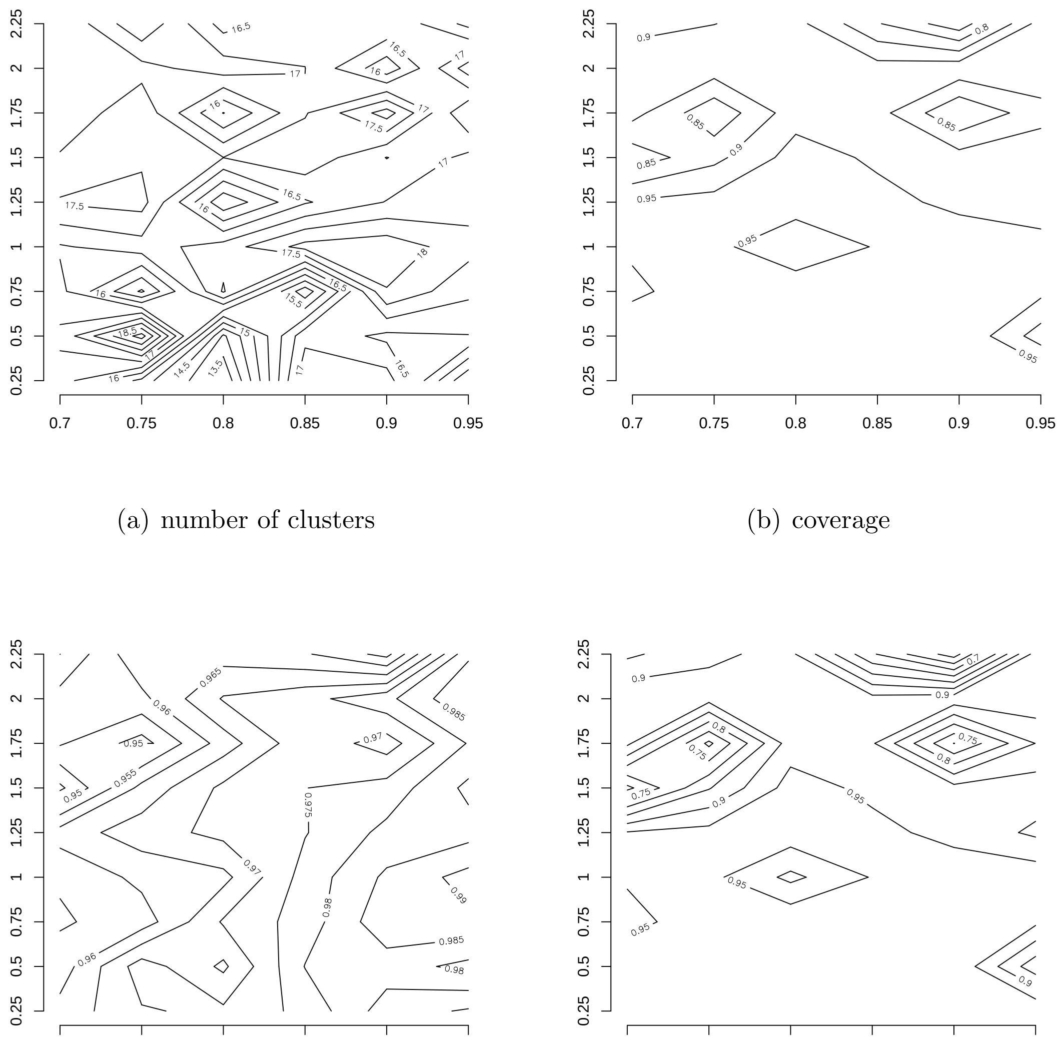

(d) inter-cluster conductance

Figure 6.24.: GMC using geometric mean of cov $(\mathcal{C})$, $\mathsf{perf}(\mathcal{C})$, and inter-cluster conductance



(a) number of clusters

(b) coverage

(c) performance

(d) inter-cluster conductance

Figure 6.25.: ICC with $\alpha = 0.4$

Chapter 7

Variations of Clustering

So far, clustering has been considered as a partition of the node set of the input graph. However, this view is insufficient when considering additional information such as temporal data or when the input graph has strong hierarchical characteristics. In this section, we present some extension of the classical partition model in order to include temporal aspects or to deal with inhomogeneous densities. In the first part different dynamic problems, like updating clusterings, are discussed and modeled. The second part gives an overview of structural variations such as fuzzy clusterings or hierarchical decompositions.

7.1. Dynamics

Clustering is a frequently used tool in the analysis and evaluation of large and complex networks, ranging from the Internet, the World Wide Web, networks of sexual contacts, scientific collaboration networks, to metabolic networks. Most such networks result from or model dynamic processes, thus clustering techniques need to be adapted. In the following, we introduce a model for clustering graphs that are subject to changes. More precisely, we address the update problem, i. e., maintaining the clustering while nodes and edges can be inserted and deleted, as well as the issue of clustering graph sequences. Furthermore, we give some illustrating examples and point out several pitfalls. This is only a first step towards a sound foundation for clustering on evolving graphs. The results have been published in [48].

7.1.1. Preliminaries

In order to deal with changes in the graph structure, we extend the notation given in Section 2.1. Let $\mathcal{G} := \{(V, E, \omega) \mid E \subseteq \binom{V}{2}, \omega\colon E \to \mathbb{R}^+\}$ be the set of all undirected weighted graphs. For a given graph $G = (V, E, \omega)$, five structural operations are defined:

$$\begin{aligned}
\mathit{deleteEdge}\,(G) &:= \left\{(V, E', \omega') \in \mathcal{G} \mid \exists\, e \in E \colon E' = E \setminus \{e\}, \omega' = \omega_{|E'}\right\} \\
\mathit{insertEdge}\,(G) &:= \{G' \in \mathcal{G} \mid G \in \text{deleteEdge}(G')\} \\
\mathit{deleteNode}\,(G) &:= \left\{(V', E', \omega') \in \mathcal{G} \;\middle|\; \begin{array}{r} \exists\, u \in V \colon V' = V \setminus \{u\}, \\ E' = E \cap \binom{V'}{2}, \omega' = \omega_{|E'} \end{array}\right\} \\
\mathit{insertNode}\,(G) &:= \{G' \in \mathcal{G} \mid G \in \text{deleteNode}(G')\} \\
\mathit{changeWeight}\,(G) &:= \left\{(V, E, \omega') \in \mathcal{G} \mid \exists\, e \in E \colon \omega'_{|E\setminus\{e\}} = \omega_{|E\setminus\{e\}}\right\}
\end{aligned}$$

A *graph modification* is a mapping $\Delta \colon \mathcal{G} \to \mathcal{G}$. It is called *simple*, if the modification can be expressed by one of the above structural operations or the identity function for all graphs.

Let $T \subseteq \mathbb{N}$ be a finite collection of nonnegative integers. For every $t \in T$, the immediate predecessor (successor) is naturally defined as the largest (smallest) integer t' that is contained in T and smaller (larger) than t, if such an element exists. Otherwise, we set it to t. A *graph sequence* is a mapping $s \colon T \to \mathcal{G}$. The sequence is *simple*, if there exists a simple graph modifications Δ_t for all $t \in T$ such that

$$\forall t \in T \colon t \neq \mathit{succ}(t) \Longrightarrow s(\mathit{succ}(t)) = \Delta_t(s(t)) \ .$$

Note that any graph modification can be composed by a sequence of simple graph

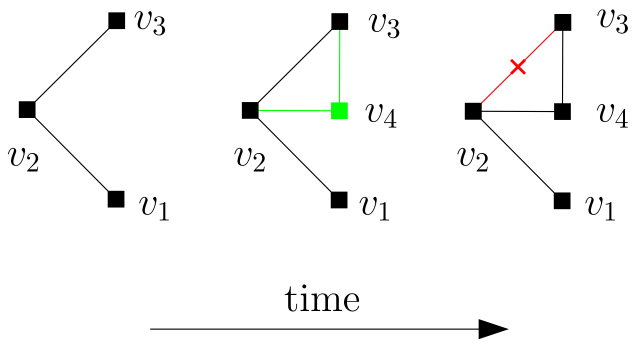


Figure 7.1.: An example of a simple graph sequence. In the second time step, a new node v_4 with neighborhood $\{v_2, v_3\}$ is inserted and, in the third time step, the edge between the nodes v_2 and v_3 is removed.

modifications. Thus, by replacing each graph modification of a non-simple graph sequence by an equivalent sequence of simple graph modifications, we can translate any graph sequence into a, possibly much longer, simple graph sequence. *Time-independent* simple graph sequences are simple graph sequences where the modification function Δ_t is the same for all $t \in T$. Note that the sequence (G, G, H) for two different graphs $G \neq H$ is a simple graph sequence but not time-independent. An example of a simple graph sequence is given in Figure 7.1.

Furthermore, we will distinguish between a *clustering technique* and a *clustering algorithm*. A clustering technique is a mapping that maps each graph to a set of clusterings. For example, the set of all clusterings having maximum score with respect to a quality index. In contrast, a clustering algorithm or an *implementation* maps each graph to exactly one clustering. A clustering algorithm $\mathcal{A}$ is associated with a clustering technique $\mathcal{T}$, if for every graph G the clustering $\mathcal{A}(G)$ is contained in $\mathcal{T}(G)$. Note that the crucial point is that a specific implementation generally does not find all optimal solutions, while the image of the corresponding technique consists of all optimal solutions. For example, an implementation resolves degrees of freedom always in the same manner, thus excluding equivalent solutions.

7.1.2. Dynamic Clustering

The *dynamic clustering* problem is to update a given clustering with respect to a clustering technique when the associated graph changes. Since every graph modification can be modeled as a sequence of simple graph modifications, we restrict ourselves to Problem 7.1 as follows.

Problem 7.1. *Given a graph G, a clustering technique $\mathcal{T}$, a clustering $\mathcal{C} \in \mathcal{T}(G)$ and a simple graph modification Δ. Calculate a clustering $\mathcal{C}' \in \mathcal{T}(\Delta(G))$.*

Naturally, Problem 7.1 can be solved by applying an implementation of the technique $\mathcal{T}$ to the modified graph $\Delta(G)$. However, implementations may be demanding with respect to running time or space consumption; in addition, this approach does not take any advantage of the given clustering $\mathcal{C}$.

On the other hand, an update strategy that performs better than every algorithm for the clustering technique $\mathcal{T}$, cannot improve the time complexity by more than a factor of n, where n is the number of nodes. Otherwise, we could use the update strategy to define a faster algorithm which incrementally builds the graph with *insertNode*–operations.

In some cases, the exact update of a clustering technique may produce undesirable clusterings. For example, when the clustering technique consists of optimizing a quality measure, the global optima in G and $\Delta(G)$ might differ substantially. Many applications that are based on dynamic clusterings not only rely on the induced quality of the clustering but also on structural properties. Thus, it is desirable to preserve structural properties already obtained. As a consequence, the formulation of Problem 7.1 needs to be subjected to further constraints. Therefore, Problem 7.1 becomes a bicriterial optimization problem, where one criterion is the quality aspect (induced by the technique) and the other criterion is the similarity to the given clustering $\mathcal{C}$. This additional restriction is also called the preservation of the *mental map* [75]. Most of the techniques for measuring the similarity of two clusterings are based on distance functions on partitions. However, these approaches neglect the edge set entirely, since they have their origin in the data mining community where pairwise information is available. For current approaches including a generalization for graph clusterings see [33].

Heuristic Approaches to Dynamic Clustering

Although the above model reflects all theoretical aspects of dynamic clustering an implementation is highly non-trivial. Among the problems are that most clustering algorithms are either heuristics since the associated optimization problem is $\mathcal{NP}$-hard or defined as an iterative process without optimizing a global quality function. Examples are the algorithm Iterative Conductance Cutting [101, 24], which is a heuristic approach to optimize the intra-cluster conductance, or the betweenness clustering algorithm [79], which iteratively removes an edge with maximum betweenness. On the other hand, there is no obvious way to formalize the mental map property by distance measures.

Feasible approaches to dynamic clustering in general are local updates or shifting strategies, see [47] for an overview. For several optimization criteria, the update can be determined with a low computational cost. By limiting the number of executed updates, one naturally obtains a clustering close to the initial clustering, thus preserving the mental map. In addition the search range for the updates can be scaled to achieve a trade-off between the obtained quality and the preservation of the mental map.

Counter-Examples

While the above models for clustering on changing graphs incorporate several important theoretical aspects, there clearly are situations where attention has to be paid to counterintuitive behavior. In the following, we give some basic examples consisting of the insertion of an intra-cluster edge (Figure 7.2(a)) and the deletion of an inter-cluster edge (Figure 7.2(b)).

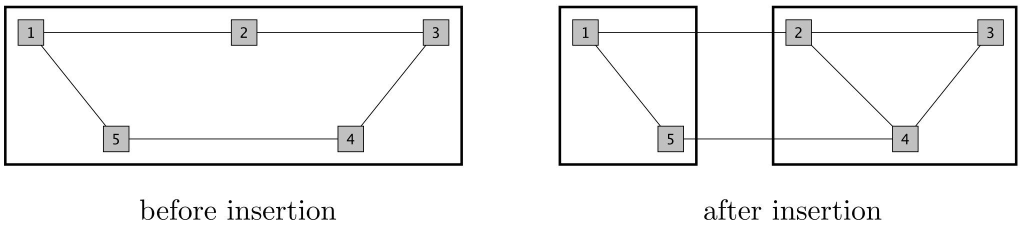
7.1.3. Time-Dependent Clustering

The *time-dependent clustering* problem is to identify structural groups within a given graph sequence. As a simple example consider the temporal evolution of a recommendation system for books as used for example by `Amazon.com`. In particular, the book 'The Lord of the Rings' by J.R.R. Tolkien. Before the major success of the film adaptation, the book belonged to the community of role playing and fantasy literature, afterwards it belonged to a much broader community including other popular bestsellers. Another typical example, which we will illustrate in Figure 7.4 below, are the collaboration dynamics in science. Researchers usually start out working in a narrow field, then, by interdisciplinary commitment, they enter other communities, possibly migrating to another field entirely.

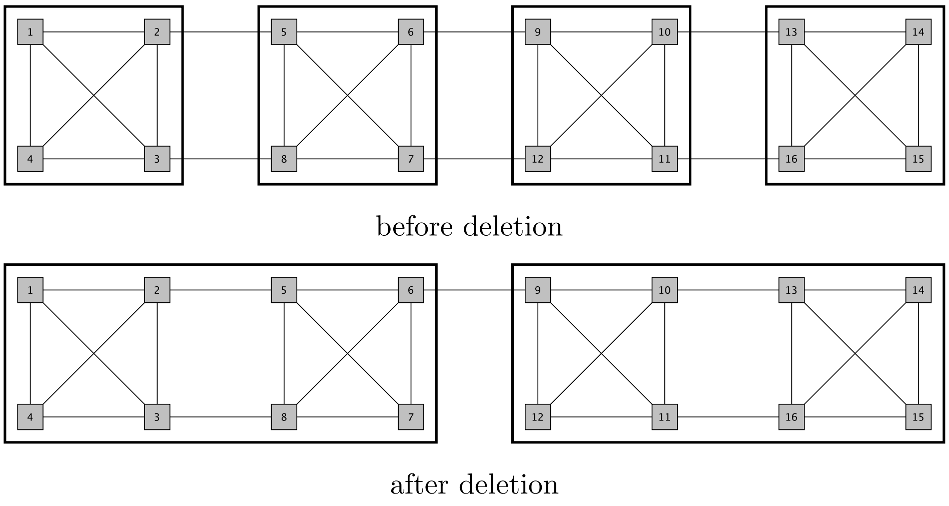
A formal definition of the term time-dependent clustering is given in Definition 7.2.

Definition 7.2. *Given a graph sequence $s\colon T \to \mathcal{G}$, where T is a finite subset of $\mathbb{N}$. The node set of graph $s(t)$ is denoted by $V(t)$. A* time-dependent clustering $\mathcal{C}(s)$ *is a partition of the* time-dependent node set $\mathcal{V}$ *which is defined as*

$$\mathcal{V} := \{(v,t) \mid t \in T, v \in V(t)\} \quad . \tag{7.1}$$



(a) insertion of an intra-cluster edge



(b) deletion of an inter-cluster edge

Figure 7.2.: Two examples of counterintuitive splitting and merging resulting from the insertion and the deletion of a single edge, respectively.

Clustering each graph of the sequence independently yields a sequence of clusterings that naturally form a time-dependent clustering. Since temporal relations are not taken into account, the clustering cannot be used to identify temporal trends. However, the sequence of clusterings may serve as a starting point and can be transformed into a 'real' time-dependent clustering. One possible transformation could be to merge temporally neighbored clusters that share a large fraction of nodes. Another approach to obtain a time-dependent clustering employing static clustering methods is based on the *time-expanded graph*. Given a graph sequence $s\colon T \to \mathcal{G}$ the time-expanded graph $G(s) = (\mathcal{V}, \mathcal{E}, \widetilde{\omega})$, where the edge set is defined as the union of the two sets $\mathcal{E}_{\text{graph}}$ and $\mathcal{E}_{\text{time}}$.

$$\begin{aligned} \mathcal{E}_{\text{graph}} &:= \{\{(v,t),(w,t)\} \mid v,w \in V(t), \{v,w\} \in E(t), t \in T\} \\ \mathcal{E}_{\text{time}} &:= \{\{(v,t),(v,t')\} \mid v \in V(t), t' = \min\{t'' \mid t'' > t \wedge v \in V(t'')\}\} \quad , \end{aligned}$$

where $E(t)$ denotes the edge set of the graph $s(t)$. The weight of an *intra-graph* edge $\{(v,t),(w,t)\} \in \mathcal{E}_{\text{graph}}$ is give by the weight of $\omega_t(\{v,w\})$, where ω_t is the weighting function of graph $s(t)$. The weights on the *inter-time* edges, i. e., those

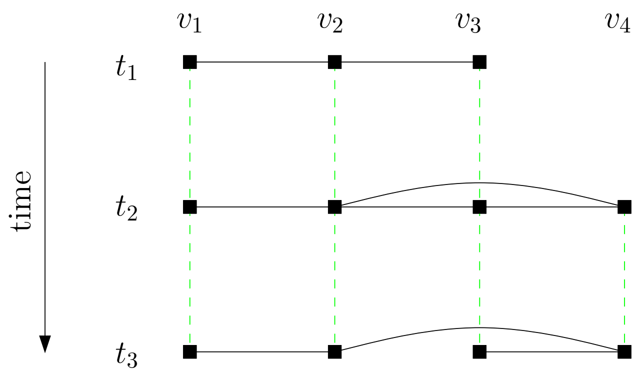


Figure 7.3.: The time-expanded graph of the sequence given in Figure 7.1.

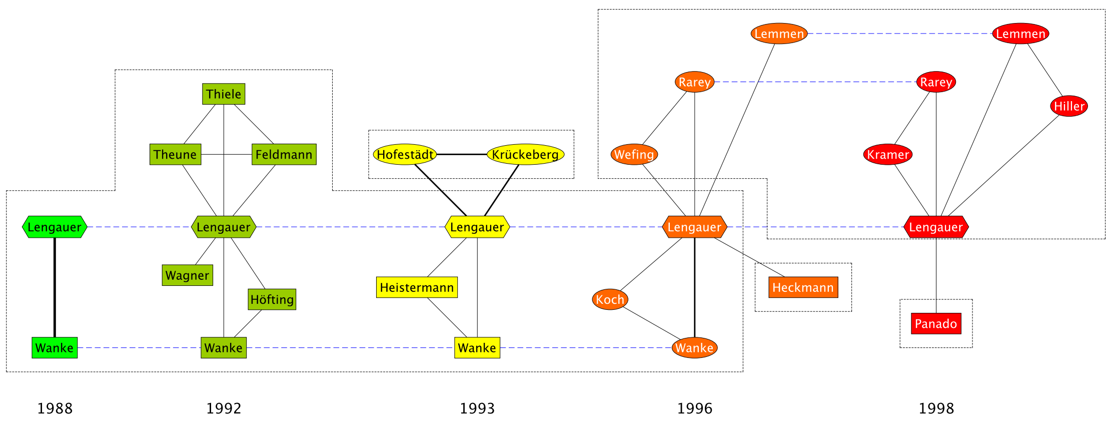


Figure 7.4.: An excerpt from the collaboration graph of T. Lengauer is shown for the years 1988, 1992, 1993, 1996, and 1998. Round shapes correspond to publications in the field of biology, and rectangular shapes to those in computer science. The time-expanded graph is clustered by the large boxes and inter-time edges are drawn dashed.

contained in $\mathcal{E}_{\text{time}}$, are an additional degree of freedom which can be tuned in order to obtain meaningful time-dependent clusterings. As an example, the time-expanded graph of the sequence given in Figure 7.1 is shown in Figure 7.3. Figure 7.4 shows an excerpt from the time-expanded graph of collaborations of the scientist Thomas Lengauer. In the 80's Lengauer concerned himself with algorithmic graph theory, focusing on planarity. However, in the early 90's he began collaborations in the field of bioinformatics and biology and then became an established scientist of the bioinformatics community. The clustering of the time-expanded collaboration graph clearly reveals this development, identifying the community transition in the 90's.

7.2. Structural Variations

Clusterings were introduced as partitions of the node set. In the following, we briefly discuss some extensions that still group the node set.

Fuzzy clustering relaxes the disjoint constraint, thus clusters can overlap each other. The basic idea is that bridging elements belong to adjacent clusters rather than build their own one. In order to avoid redundancy, one usually requires that a cluster is not contained in the union of the remaining clusters. Figure 7.5 shows such an example where the two groups have the middle node in common. It is

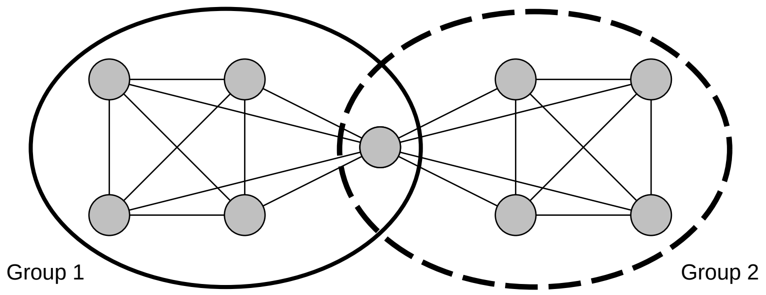


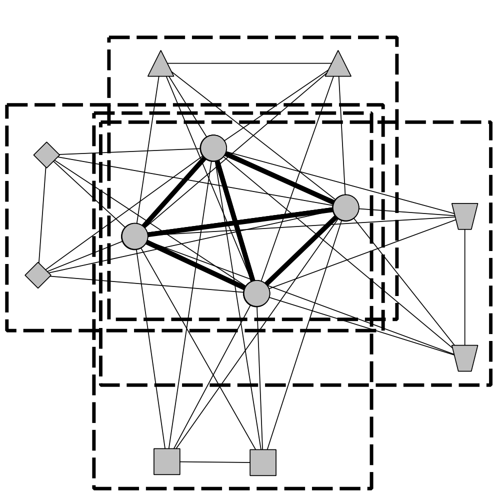
Figure 7.5.: Example of a fuzzy clustering

seldom used, due to its difficult interpretation. For example, it is very difficult to judge single nodes or small node subsets that belong to a relatively large number of (fuzzy) clusters. When the number of clusters is also restricted, then artefacts occur more frequently. For example, if the number is restricted to a constant k, then a difficult situation is where more than k cliques of size at least k have a large number of nodes in common (see Figure 7.6(a)). If the number can scale according to a node's degree, then sparse clusters can be torn apart. For example, a star with k leaves may be decomposed into k fuzzy clusters, each containing one leaf and the central node (see Figure 7.6(b)).

Another extension is to enhance *clusterings with representatives*. Each cluster has a representative. It is very similar to the facility location problems [37, 77, 89], where a candidate covers a subset of elements. This can be seen as 'representing' the group by one element. It is usually a node that is located 'in the center' of the cluster. This form of enhancement can be very effective when the graph is embedded in a metric or a vector space. In these cases the representative can also be an element of the space and not of the input. The concept is also used to perform speed-ups or approximate calculations. For example, if all the similarity/distance values between the nodes in two clusters are needed, then it can be sufficient to calculate the similarity/distance values between the representatives of the clusters.

7.2.1. Nested Decompositions

Nested clustering or *nested decompositions* represents a nested sequence of node subsets, i. e., a mapping $\eta\colon \mathbb{N} \to \mathcal{P}(V)$ such that:


(a) Four cliques of size six that have a common K_4. Each maximum clique is a fuzzy cluster.

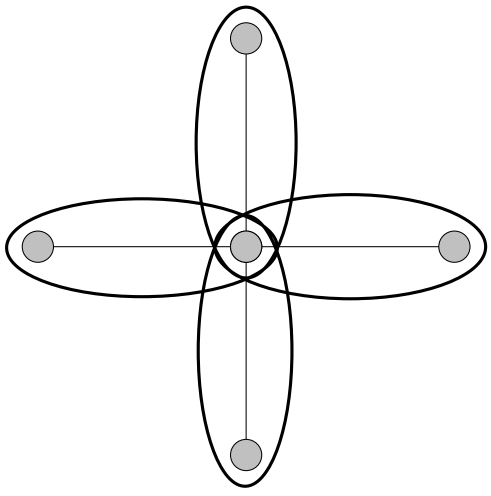
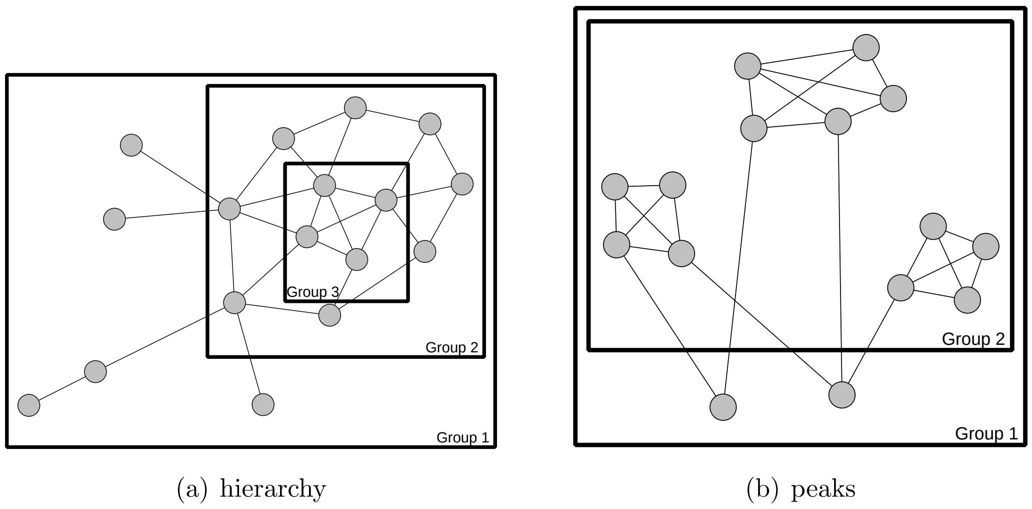

(b) A star with four leaves and each fuzzy cluster contains the center node and one leaf.

Figure 7.6.: Two examples of fuzzy clusterings where many clusters intersect each other

1. the subsets are nested, i. e.,
$$\forall\, i \in \mathbb{N}\colon \eta(i+1) \subseteq \eta(i)$$
2. and the sequence is finite, i. e.,
$$\exists\, k \in \mathbb{N}\colon \forall \ell > k\colon \eta(\ell) = \emptyset \ .$$

The smallest possible k is also called the *size* of the sequence, and $\eta(k)$ the top element. The *level* of a node is the maximum index i such that the node is contained in $\eta(i)$. The intuition behind this structure is that the top element $\eta(k)$ consists of locally maximal dense groups. The density of the subsets $\eta(i)$ also decreases with decreasing argument i. Therefore the argument i can be seen as degree of density. One can distinguish two extreme types: The first one is called *hierarchies*, where each $\eta(i)$ induces a connected graph. The corresponding graphs can be seen as onions, i. e., having a unique core and multiple layers around it with different density. The second type is called *peaks*, and is complementary to hierarchies, i. e., at least one subset $\eta(i)$ induces a disconnected graph. An appropriate example may be boiling water, where several hotspots exist that are separated by cooler parts. If the graph has a skew density distribution then a hierarchy type can be expected. In this scenario, it can reveal structural information, unlike standard clustering. If the graph has a significant clustering, then peaks are more likely to occur. In that case, the top element consists of the core-parts of the original clusters. Figure 7.7 shows such examples.



(a) hierarchy (b) peaks

Figure 7.7.: Examples of a nested clustering

Core Decomposition

Cores are an often used realization of nested decompositions. The concept was originally introduced in [91] and generalized in [11]. More precisely, the *k-core* of an undirected graph is defined as the unique subgraph obtained by iteratively removing all nodes of degree less than k. A node has *coreness* ℓ, if it belongs to the ℓ-core but not to the $(\ell + 1)$-core. The *ℓ-shell* is the collection of all nodes having coreness ℓ. The *core* of a graph is the k-core such that the $(k + 1)$-core is empty. The core decomposition can be computed in linear time with respect to the graph size [10]. Informally speaking, the coreness of a node can be seen as a robust version of the degree, i. e., a node of coreness ℓ keeps its coreness even after removing an arbitrary number of nodes of smaller coreness. An example is Figure 7.8.

Since cores are nested, we cannot directly use the notation of abstract graphs as defined in the preliminaries. However, given a core decomposition η of size k, we can associate two meaningful partitions in the following way: the *level view* uses the clustering $\mathcal{C}_\eta := \{V' \mid \exists\, 1 \leq i < k \colon \emptyset \neq V' = \eta(i) \setminus \eta(i+1)\}$ and the *level-component view* considers the refinement $\mathcal{C}'_\eta$ where each cluster corresponds to a connected component of a cluster in $\mathcal{C}_\eta$. Note that, the clusters in $\mathcal{C}_\eta$ group all elements having the same coreness. The level view and the level-component view of the example give in Figure 7.8 is just a path having three nodes. In Section 8.4.1 and Chapter 9, cores and the corresponding abstract graph are used visualization networks.

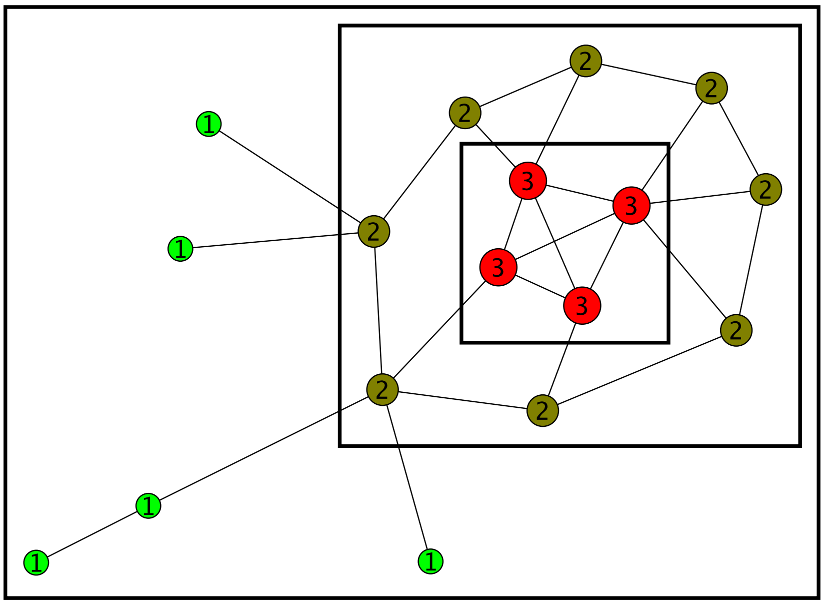


Figure 7.8.: Example of a core decomposition. The labels of the nodes represent their coreness, the boxes mark the individual cores.

Part II.

Applications in Network Analysis and Visualization

As previously mentioned, clustering techniques are fairly popular tools in the analysis and exploration of networks and data that can be modeled as graphs. Since such tasks naturally arise in many fields—such as project planning, transportation systems, communication networks, bioinformatics, hypertext and text analysis, information retrieval, data mining, complex systems, and social networks—the reasons for applying clustering are versatile. In the following, we briefly summarize the two main motivations which are the *organization of data* and the *identification of structure*. In our case study, which is presented in Chapter 8, both aspects are used in order to deepen the understanding of the considered network.

Organization of Data

Clustering was originally introduced in the field of data mining as the unsupervised classification of patterns into groups [65]. Thus, one of the main motivation for its application is the organization and reduction of data, i. e., by identifying elements that behave highly similar, one can abstract these elements into a single one. This technique is also known as *abstraction*, since redundant elements are removed and only few representatives are left. Furthermore, this schema is comparable to the standard technique of *Divide-and-Conquer*, which "breaks a problem into several subproblems that are similar, but smaller in size, solves them recursively, and then combines the solutions to a solution of the original problem" [30, p. 12].

An illustrating example is the segmentation of (textured) images. Such an image can be interpreted as a two-dimensional grid of pixels where the similarity between two neighboring pixels is defined by similarity of their color. Applying clustering techniques leads to a segmentation where clusters form local patches of almost the same color. In this way, the original image can be represented by the shape of the patches as well as their color, which can significantly reduce its size. More detailed examples can be found in [66, Ch. 5.3], [100, pp. 121], or [94].

Identification of Structures

In contrast to abstraction, the second aspect of clustering focuses on the structures of the individual clusters. Such a reduction can be useful in order to restrict one's attention to the relevant part of the network. For example in a recommendation system, it is sufficient to consider the items within the cluster that matches the user's interest. In addition, the study of the internal structure of a cluster provides valuable insights in the mutual relation of the contained elements. For example in the analysis of social networks, the study of clusters (also known as cohesive subgroups) and their internal structure is an important aspect. As social interaction often operates through direct ties (edges) and bears certain unreliabilities, clusters naturally form robust groups of local interaction. More details can be found in [104, Ch. 7].

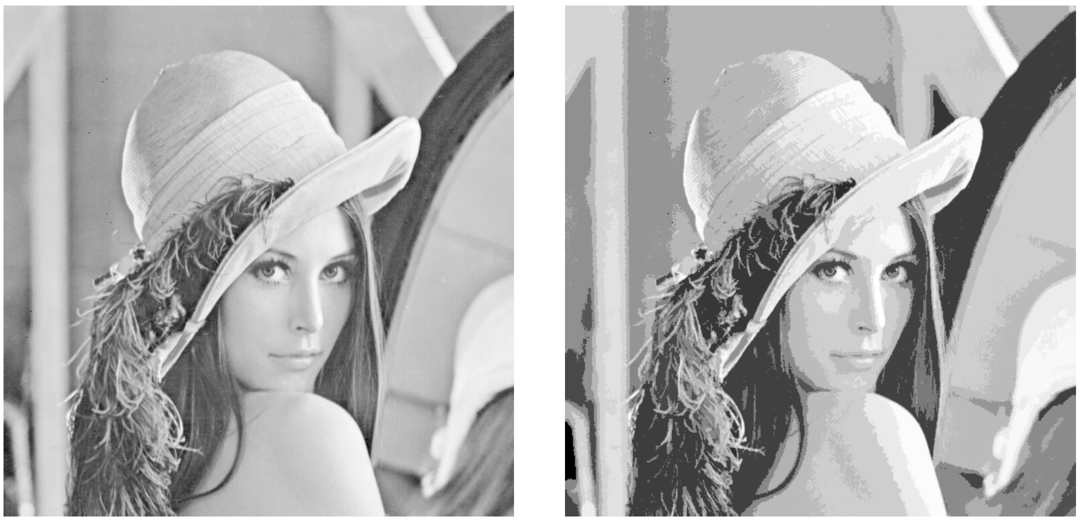

(a) original image (b) after segmentation

Figure 7.9.: Example of the segmentation of an image, which is named *Lena* and contained in standard benchmark suits, taken from [8].

For an illustration, consider the following excerpt from the recommendation system used by Amazon.com [5] given in Figure 7.10. The network was crawled from the web pages starting from the product ‘VW Beetle Restoration Handbook’ and following at most nine links. Inspecting the cluster with the dark blue nodes (in the middle of the right side and in Figure 7.10) as shown in Figure 7.11, reveals that it actually consists of two parts that are sparsely connected. When considering the book titles one observes that the left and bigger part contains technical books in contrast to the right and smaller part which is build of books covering the history of Volkswagen and the Bug. Since both parts have highly similar content (with respect to the whole system), they are unified in a cluster. Summarizing, the analysis of clusters can reveal additional information that need not be visible at the macroscopic level of the network.

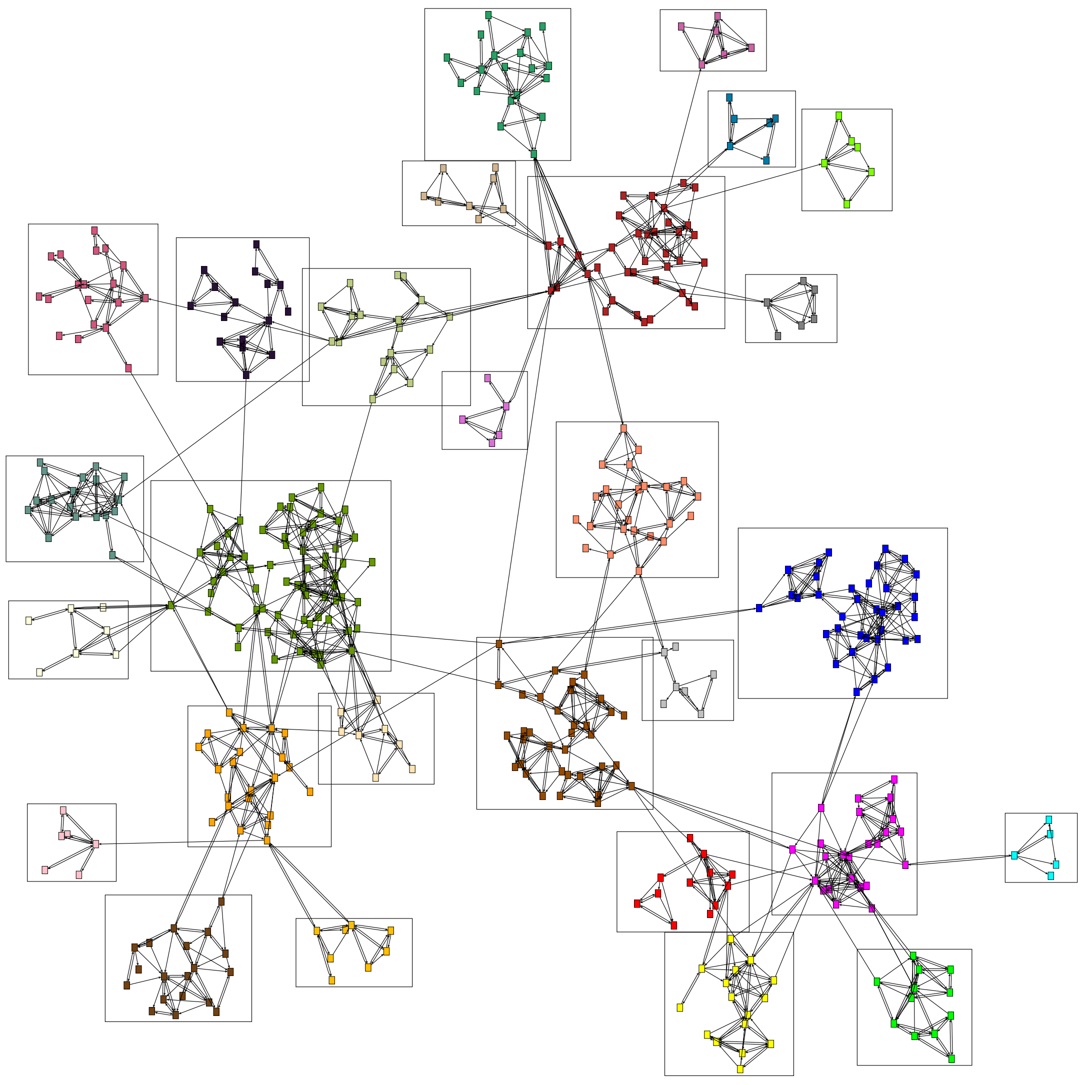


Figure 7.10.: Excerpt from the recommendation system used by Amazon.com.

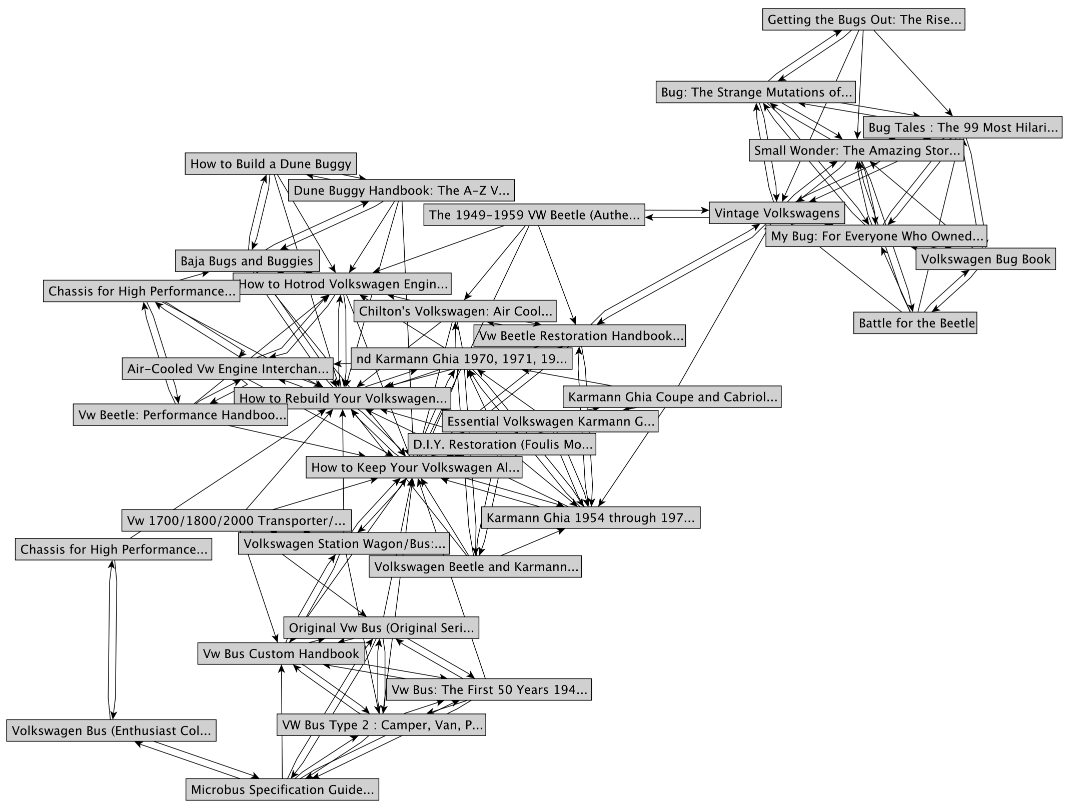

Getting the Bugs Out: The Rise...
Bug: The Strange Mutations of...
Bug Tales : The 99 Most Hilari...
Small Wonder: The Amazing Stor...
How to Build a Dune Buggy
Dune Buggy Handbook: The A-Z V...
The 1949-1959 VW Beetle (Authe...
Vintage Volkswagens
My Bug: For Everyone Who Owned...
Volkswagen Bug Book
Baja Bugs and Buggies
Chassis for High Performance...
How to Hotrod Volkswagen Engin...
Chilton's Volkswagen: Air Cool...
Battle for the Beetle
Vw Beetle Restoration Handbook...
Air-Cooled Vw Engine Interchan...
nd Karmann Ghia 1970, 1971, 19...
Karmann Ghia Coupe and Cabriol...
How to Rebuild Your Volkswagen...
Vw Beetle: Performance Handboo...
Essential Volkswagen Karmann G...
D.I.Y. Restoration (Foulis Mo...
How to Keep Your Volkswagen Al...
Vw 1700/1800/2000 Transporter/...
Karmann Ghia 1954 through 197...
Chassis for High Performance...
Volkswagen Station Wagon/Bus:...
Volkswagen Beetle and Karmann...
Original Vw Bus (Original Seri...
Vw Bus Custom Handbook
Vw Bus: The First 50 Years 194...
Volkswagen Bus (Enthusiast Col...
VW Bus Type 2 : Camper, Van, P...
Microbus Specification Guide...

Figure 7.11.: Close-up of one of clusters of the recommendation system given in Figure 7.10.

Chapter 8

Case Study: Autonomous Systems

In the late 90s of the last century, Faloutsos et. al. observed in [44] that the Internet (at the IP level as well as at an abstracted level) does not show characteristics of a random network, although no (central) organization manages it. Simultaneously, many other technical as well as natural networks which self-emerged or lacked a common management were studied, and slowly the interdisciplinary field of *complex systems* was born. Summarizing the main results, most unstructured, real-world networks do not have the characteristics of Erdős–Rényi networks, which are uniform random networks [41]. Furthermore, many of these networks exhibit *power-laws*: For example, Faloutsos et. al. observed that the degree distribution, i. e., node degree d versus the frequency f_d of this degree, follows a power-law such as $f_d \propto d^c$ for a constant c. On the one hand, these results indicated that "unstructured" or self-emerging networks exhibit an underlying structure that is different from random networks and need to be better understood. On the other hand, insights gained so far through random models, both theoretical and experimental in nature, became obsolete to a certain degree.

In the following, we consider an abstract view of the Internet as a case study for the analyses of complex networks. This abstraction, also known as the AS Network, holds several benefits over other networks: Foremost its size, which is comparable small with at most 20,000 nodes and 45,000 edges, and second, the availability of collected data through academic institutes.

After a brief summary of technical aspects of the AS Network, we present some results on filtering using the core decomposition which are required for our visualization technique. An analysis of the Peer-2-Peer (P2P) application Gnutella in the context of the AS Network concludes our case study. The results have been published in [49, 14, 3, 4].

8.1. Brief Technical Summary

An *Autonomous System* (AS) is a collection of routing devices and IP networks under the control of one entity that presents a common routing policy to the Internet. Each AS is uniquely identified by a 16-bit number. For further technical information see

RFC 1930 [82]. The *AS Network* is defined as a graph $G = (V, E)$ where a node represents an Autonomous System and two nodes are connected by an edge if the corresponding ASes directly exchange traffic. It is an abstract version of the physical Internet, where routing devices are interpreted as nodes and an edges connect two nodes if the corresponding devices can communicate directly.

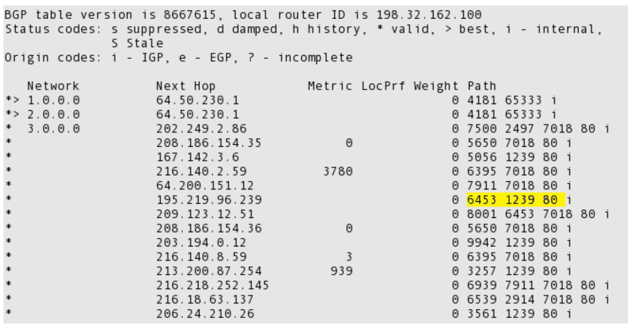
Since routing in the Internet is basically established by a distributed and dynamic version of a shortest-path algorithm (respecting not only path length, but also commercial interests), the size of the considered network is an essential issue in terms of runtime and convergency. More precisely, such a protocol would have a relatively low convergence rate when run on each router in the Internet individually. The current implementation (*Border Gateway Protocol* (BGP) [83]) considers only the AS Network and ensures that the information about reachability of one AS by the help of another AS is properly propagated; on the other hand, each AS has to ensure that this global information is correctly distributed inside its own network. Such multistage techniques are fairly often used in order to speed-up shortest path computations, see [62, 61, 86, 87] for examples.

8.1.1. Data Collection

Although the definition of the AS Network is precise and straightforward, the gathering of all traffic exchange agreements is hardly possible. The major reason is the protection of commercial interests, i. e., many ASes which operate as intermediate ASes that buy and sell Internet connectivity have to protect their exchange agreements in order to successfully compete in the market. Fortunately, some ASes are willing to provide their data and this information is collected by the Routeviews Project at the University of Oregon [85]. We refer to these ASes as *(observing) peers*. More precisely, all routing paths of the participating peers are stored. The union of these paths is used to obtain an estimation of the whole AS Network. An illustrating example is given in Figure 8.1. Note that this data collection can be interpreted as the exploration of an unknown network through a collection of single-source shortest path trees. In reality and due to commercial interests, the reported paths of an AS need not form a tree, but can contain cycles. A theoretical investigation of network discovery and verification by the help of shortest path trees was performed by Erlebach et. al. in [42]. Routeviews collect their data in two hour intervals and started in spring 2001.

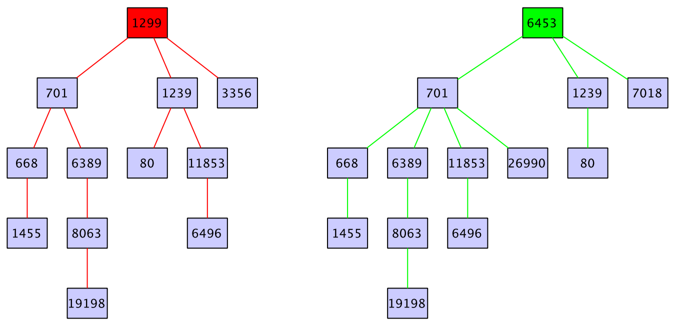
An alternative method to gather data about the AS Network is indirect through trace routes, i. e., collecting information about the path a packet traverses in the Internet. Since the quality and quantity of observed data directly depends on the number of trace routes and the number of pairs of endpoints of packets, it is fairly difficult to get useful data[1]. The project DIMES [36] considers an approach similar to SETI@home (`http://setiathome.ssl.berkeley.edu/`) where a light-weighted client should be downloaded by many people and in the idle time of their computers

[1]Also note technical issues: for example sending too many dummy packets from one source to many destinations could be interpreted as malicious attack and thus would correspondingly be blocked.

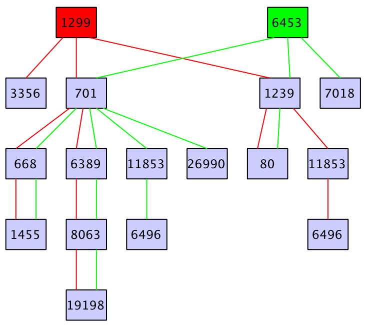


```
BGP table version is 8667615, local router ID is 198.32.162.100
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 1.0.0.0          64.50.230.1                            0 4181 65333 i
*> 2.0.0.0          64.50.230.1                            0 4181 65333 i
*  3.0.0.0          202.249.2.86                           0 7500 2497 7018 80 i
*                   208.186.154.35           0             0 5650 7018 80 i
*                   167.142.3.6                            0 5056 1239 80 i
*                   216.140.2.59          3780             0 6395 7018 80 i
*                   64.200.151.12                          0 7911 7018 80 i
*                   195.219.96.239                         0 6453 1239 80 i
*                   209.123.12.51                          0 8001 6453 7018 80 i
*                   208.186.154.36           0             0 5650 7018 80 i
*                   203.194.0.12                           0 9942 1239 80 i
*                   216.140.8.59             3             0 6395 7018 80 i
*                   213.200.87.254         939             0 3257 1239 80 i
*                   216.218.252.145                        0 6939 7911 7018 80 i
*                   216.18.63.137                          0 6539 2914 7018 80 i
*                   206.24.210.26                          0 3561 1239 80 i
```

(a) Extract of a routing table



(b) Partial routing tree of the ASes 1299 and 6453



(c) (Partial) Union of the routing trees given in Figure 8.1(b)

Figure 8.1.: Example of a routing table, the corresponding union of paths for two ASes and their union.

the experiments are performed. In this way a large number of different endpoints should be available.

In the following, we focus on the data of Routeviews due to their availability. We briefly compare the two data sources in Section 8.4.3. When speaking of the AS Network without further reference, we always consider the data obtained from Routeviews.

8.2. Preprocessing

As mentioned at the beginning of this chapter, the AS Network is relatively small compared with other technical networks such as the WWW. Still, the network grew from 11,000 nodes and 24,000 edges in 2001 to 21,000 nodes and 45,000 edges in 2005 (Figure 8.2). Intuitively, not all ASes are equally important, i. e., ASes could be divided according to local, regional, national, and international Internet Service Providers (ISPs) as well as backbones. There is also a more technical definition, namely *tiers*, based on the peering behavior of an AS. Without explaining technical details, note that this classification heavily depends on the current market situation, e. g., major ASes may stop peering with each other and thus loose their tier status

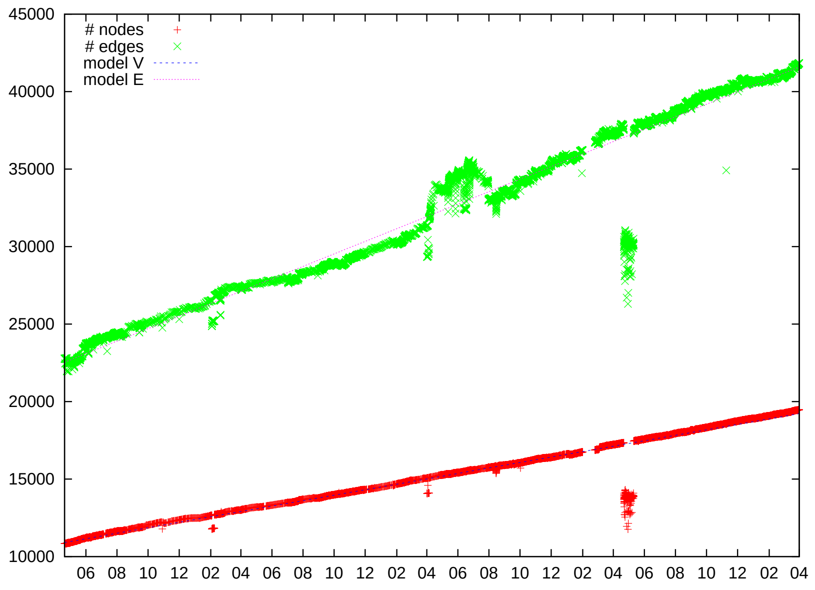


Figure 8.2.: Growth of number of nodes and edges in the AS Network beginning in April 2001 ending in March 2005. The model (dotted lines) predicts a linear growth of roughly 2100 nodes and 4800 edges per year; and matches quite well.

– for example, consider the quarrel between Level3 and Cogent [59].

There are successful attempts to recover the commercial dependencies in the AS Network based on structural properties. More formally, the undirected AS Network is transformed into a mixed graph, i. e., containing directed edges starting from a customer and pointing to its provider and undirected edges indicating a uniform peering relation. Furthermore, each original path (observed from Routeviews) starts with a sequence of forward edges, followed by at most one undirected edge, and ends with a sequence of backward edges. Such paths are also called *valley-free* and model the flow according to economic interests, i. e., providers route the traffic and no customer has to pay for traffic which is not intended for him/her. The associated decision problem, i. e., can all edges be oriented in such a way that all paths are valley-free, can be solved in polynomial time, the associated optimization problem, i. e., find an orientation of the edges such that the number of valley-free paths is the maximum possible, is $\mathcal{NP}$-hard [12]. Besides the theoretical analysis, many heuristics are known that compute very good orientations, i. e., more than 90% of all paths are valley-free, as an example consider [51]. Since most of these heuristics have many degrees of freedom, many good orientations exists [12], which render the commercial dependencies more or less useless for a filtering approach. In the following, we discuss a reduction technique based on density.

8.2.1. Filtering Based on k-Cores

As introduced in Section 7.2.1, the core decomposition is a reduction technique to obtain a nested decomposition of a network. Moreover, the coreness of a node only depends on the coreness in its neighborhood and is related to the degree. Since the core decomposition can be computed for every (undirected) graph, it can also be applied to the AS Network. However, due to the special structure, i. e., the AS Network being the union of paths routing the traffic between the ASes, a good filtering technique should respect these inherent path properties. Consequently, the quality of a filtering technique should be measured by the number of paths that remain connected in the reduced graph. For the k-core filtering, we define $\mu_k(i)$ to be the fraction of paths that have i connected components in the k-core. This number depends on the input parameter k. In order to avoid such a dependency and increase the insight into its compatibility, we will consider lower bounds for $\mu_k(1)$. Such a bound is given by the fraction of paths that remain connected in all k-cores and is denoted by $\mu(1)$. On the other hand, we also like to judge the degree of fragmentation. Therefore we count the fraction of paths that are cut into i components in a k-core for some value k and $i > 1$. This value, which is denoted by $\mu(i)$, gives an upper bound on $\mu_k(i)$. Intuitively speaking, a filtering technique is compatible with the path structure of the AS Network, if the paths are only shortened, but not split. Recall the notion of valley-free paths, if a valley-free path in the reduced graph is still connected then only customers and low-level providers are removed. Furthermore, the routing structure between high-level providers would remain. Thus the dependencies between (structurally) important ASes are still observable, while less important ASes are removed. As can be seen in Figure 8.3, more than 90% of all paths remain connected. The majority of the other paths split in two components and only a very small fraction of the paths fragment into more than two components. Summarizing, the k-core filtering preserves the intrinsic structure of the AS Network for the most part and the occurring fragmentation is very small. Thus, we can use this technique to reduce the size of the network and filter out irrelevant information. Further investigations about the effectiveness of the reduction are presented next.

8.2.2. Abstraction Based on k-Cores

As shown in the previous section, cores can be used to filter irrelevant information and thus reduce the size of the AS Network. In order to further judge the quality of the k-core reduction, we consider the resulting size distribution as well as connectivity aspects. Both issues provide valuable insight in the structure of the AS Network and result in a proper visualization technique which will be presented in Section 8.4.

Size Distribution

The size distribution, i. e., the coreness versus the number of nodes having a certain coreness, is very skew. In fact over 80% of the nodes have a coreness less or equal

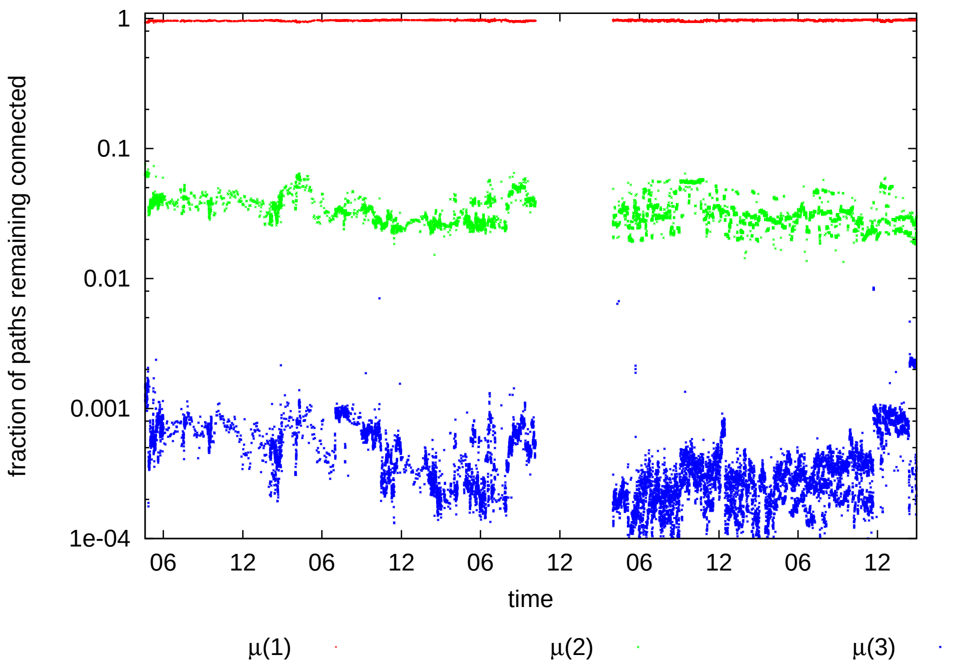


Figure 8.3.: Fraction of paths that are guaranteed to be connected and are split into two or three components over time (April 2001 till March 2006). Note the visible gap around December 2003 is due to data loss.

than three. Further, we observe that during the period of April 2001 to April 2005, the number of nodes in the AS Network increases by about 2000 nodes per year, the number of edges increases by 4800 edges per year and the maximum core number has increased from 18 to 26. Although the network grows in absolute terms and especially the individual core levels increase in size, their relative sizes remain stable. Similar to the rings of a tree trunk, Figure 8.4 illustrates the temporal evolution of the relative proportions of the k-shells, i. e., collection of nodes with coreness k. In this figure, the thickness of one strip corresponds to the fraction of nodes that have a given coreness. The lowest strip represents the maximum core while the highest strip reflects the 1-shell. One can clearly see the stability of k-shells with $k \leq 15$. It is also observable that the size and coreness of the maximum core increases over time. The growth in the coreness is not monotonic and has big fluctuations. White vertical strips indicate the absence of data due to technical problems in the external data collection process.

Connectivity

Beside the size distribution, the connectivity of the k-cores as well as between different shells is another important aspect. As mentioned in Section 7.2.1, a core decomposition need not yield connected k-cores (for every k) even if the input graph is connected. Thus it is fairly astonishing that all graphs in the period of April 2001 and April 2005 have core decomposition where each k-core is connected. On

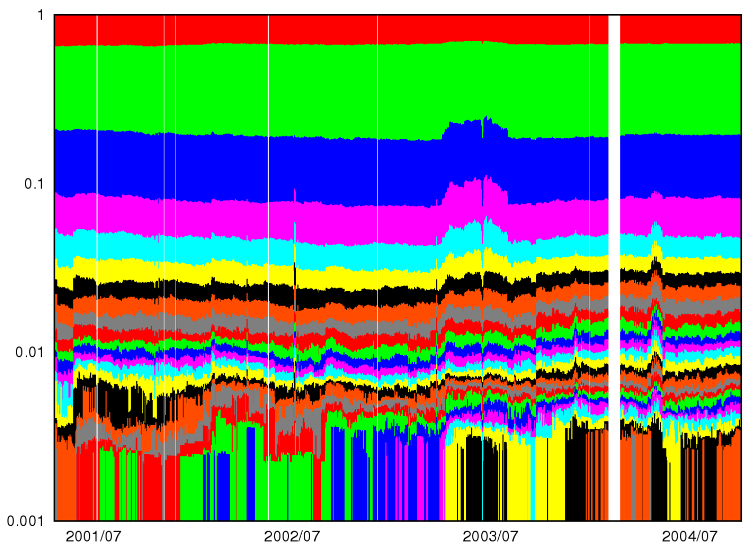
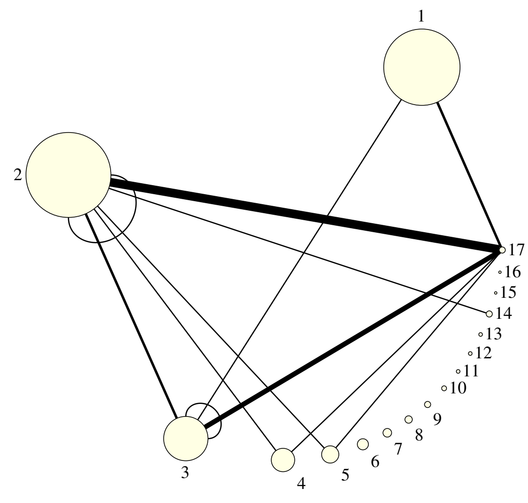


Figure 8.4.: Relative size of cores.

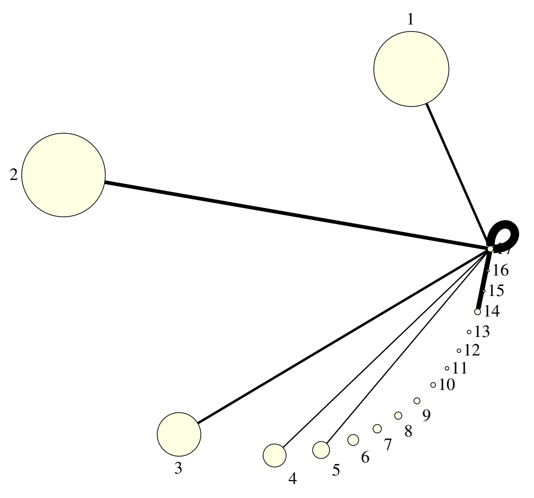
the other hand, this confirms our intuitive view of the hierarchical structure of the AS Network. More precisely, the so-called backbone of Internet, i. e., the collection of major ASes with high-capacity data connections that realize the global traffic exchange, is contained in the core of the AS Network and all customer ASes have small coreness.

Another related feature is the distribution of edges with respect to cores. More precisely, we are interested in structural properties of the graph defined by the level-view using uniform edge weights (see Section 7.2.1). At a first glance, the resulting graph is almost complete for all snapshots. After applying the logarithm of the edge weights and removing edges with very small weights, the global structure becomes more evident, see Figure 8.5(a). There are two different types of edges with large weights: edges connecting nodes with small coreness and edges that act as bridge between low-level shells and high-level shells (*transition edges*). This properly reflects the perception of the AS hierarchy, i. e., many small ASes exist and they need to be connected to the network either by other local providers or directly through the backbone.

The situation changes again when considering a different edge weighting, namely the number of paths that uses a certain edge. Again, a filtering is necessary, since otherwise the graph is almost complete. We use the same reduction as for the case of the uniform weights. In this case, the transition edges still exist, however, the edges inside the core contribute a lot of weight (see Figure 8.5(b) - comparing the thickness of the selfloop of the 17-shell with the other edges). This observation completes our intuitive view of the AS Network that many links are needed to connect the customers to their providers and that the links that occur more frequently in paths are links to or within the backbone and not inside the periphery.



(a) Uniform edge weight



(b) Edge weight induced by used paths

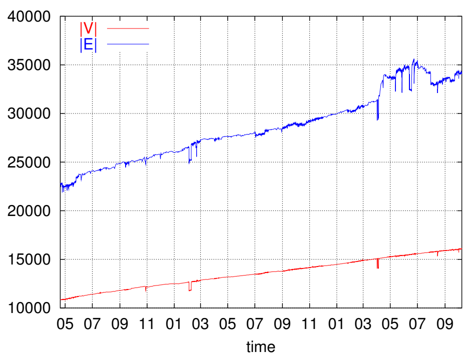
Figure 8.5.: Level-view of the AS Network (May 1st, 2001 0:00). The area of the nodes is proportional to the number of nodes having that coreness. The thickness of the edges is (logarithmically) proportional to their weight. Edges with very little weight are omitted. Note that the core (17-shell) is very small compared to the 1- and 2-shell.

8.3. Preliminary Dynamic Analysis

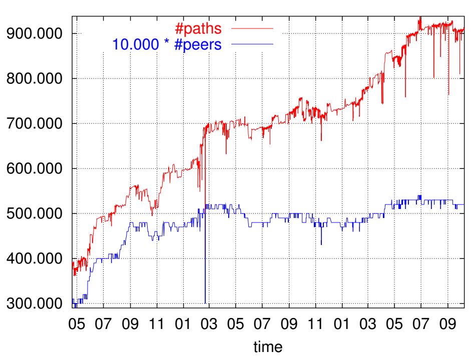
In the following, we present a preliminary study of the dynamic aspects of the AS Network. The study was performed in autumn 2003 in collaboration with the research group of Prof. Dr. Giuseppe Di Battista and is published in [49]. We investigated long-term and short-term features as current baselines, predicting trends, and analyzing anomalies.

8.3.1. Baselines and Trends

Determining the standard undisturbed behavior of the evolving network is one fundamental task of dynamic analysis. As an initial step we investigate the evolution of static indices. The temporal evolution is shown in Figure 8.6(a) and 8.6(b). The number of nodes and edges seem to be locally stable over time. In order to verify this observation, we calculated the standard deviations of time frames of different length. Both indices seem to be non-constant, therefore a larger time window results in a larger deviation. In Figure 8.7 the time frame length is plotted versus the average standard deviation (ASD). As expected, the ASDs are monotonically growing with the frame length, furthermore their increases can be expressed by a linear function, thus verifying the local stability. The slope for the number of edges is three times larger than for the number of nodes. This can be explained by the fact that edges are subject to more consistent changes and depend on the number of paths which in turn depend on the number of participating peers. For example, number of edges



(a) Number of nodes and edges



(b) Number of AS paths and peers

Figure 8.6.: Graph theoretic and domain specific measures. The x-axis represents time, starting in May 2001 till September 2003.

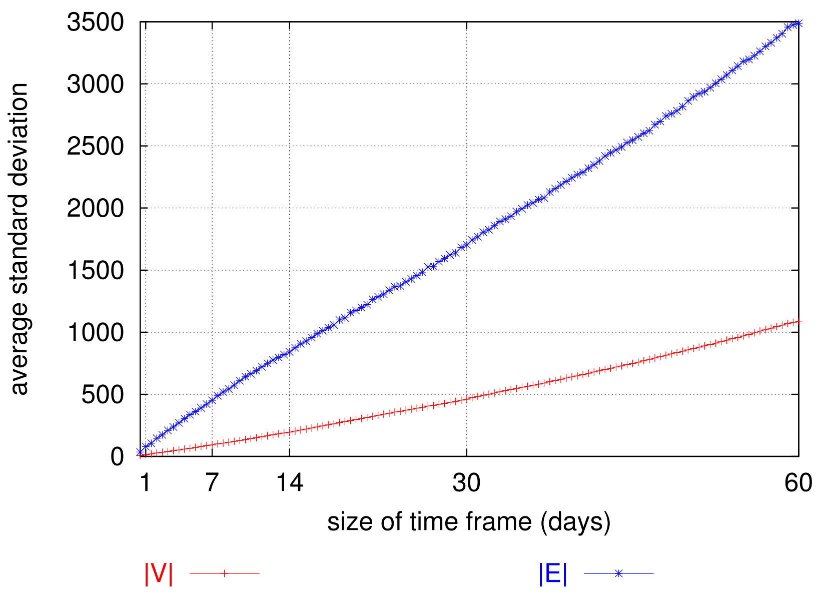
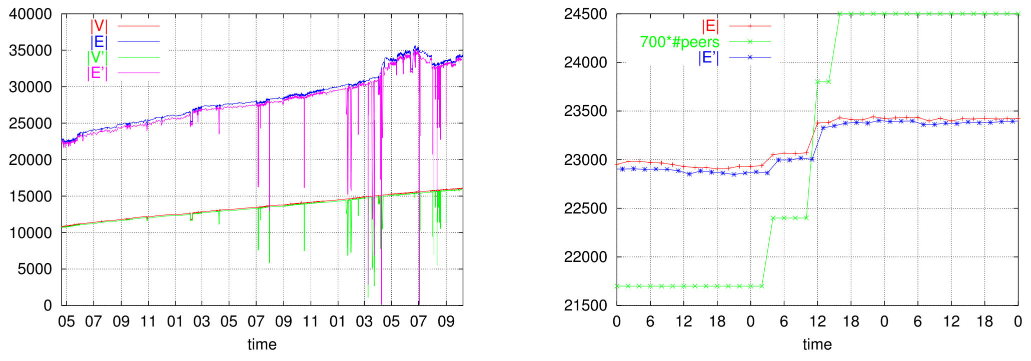


Figure 8.7.: Local stability of number of nodes and edges. The time frame length is plotted versus the average standard deviations.

and paths, as well as number of paths and peers are highly correlated (≈ 0.977 and ≈ 0.897, respectively).

In this special scenario, we can utilize the path structure to normalize the views by using the commonly observed AS Network of two consecutive points in time. The size of the nodeset and the edgeset will be denoted by $|V'|$ and $|E'|$. This enables us to decide whether a change in the number of nodes (or edges) is only due to a change in the set of observing peers or not. More precisely, we can judge if an increase in the number of nodes and edges is caused by a real growth of the the network or is just a fake growth where we would have observed the additional elements in previous points in time, if we had the additional observing peers. Figure 8.8(a) displays the evolution of the number of nodes and edges for the AS Network and the commonly observed graphs. Utilizing the detailed view of Figure 8.8(b) we conclude that the first two increases (2am and 10am) of peers led to a larger view (new edges were added) while the third increase (2pm) did not affect our view of the network (only redundant information was added). However, we have to be careful judging



(a) Evolution over time of the different measures

(b) Detailed view (2001 May from 24th till 27th).

Figure 8.8.: Sizes of AS Network and commonly observed graphs.

decreases, since the absence of vantage points can have many reasons. Some of these causes are independent of the (global) network status, like internal reorganization or connection failures during the collection process.

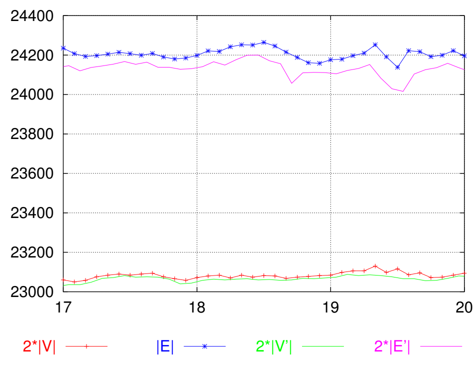
Long-term Patterns

Using these observed phenomena, we recognized two major patterns. The first one is the linear growth of the number of nodes and edges. Although it seems to be a trivial observation (Figure 8.6(a)), it is not at all expected. Many researchers have reported an exponential growth in the time frame of 1997 till 2000. Because both indices grow in a linear fashion, the ratio of observed edges to the total number of possible edges decreases. The second pattern is the distribution of growth. ASes are separated according to their importance in the core graph. By inspecting the evolution of the filtered graph (Section 8.2.1), we observed that most nodes enter the graph with a very low coreness score, while edges (in the path-weighted version) connect low-score nodes with high-score nodes or only high-score nodes with each other. This verifies the general intuition, that more customers than (high-level) providers enter the system and that more connections are established to connect the small providers with the backbone than to enlarge the backbone.

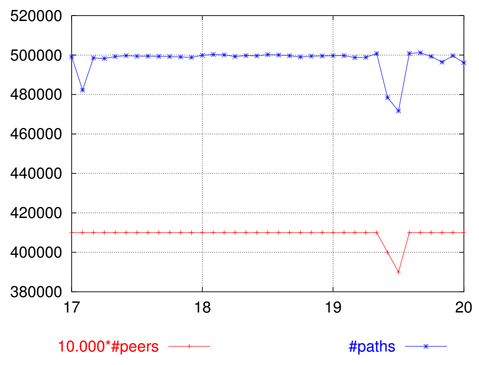
8.3.2. Short-Term Analysis – Anomalies

In this section we describe the results of short-term analysis. Namely, we measure the impact on our measurements of dramatic short-lived events that occurred in the network in the last few years. We used the GMC clustering approach on a k-core of the AS Network to identify potential changes. Due to the inhomogeneous density of the AS Network, we choose the minimum k such that the resulting k-core has at most 200 elements. Note that this choice is based on a rough estimate of the number of relevant/important ASes. Furthermore, the quality function of GMC is $\sqrt[4]{\mathrm{cov}\,(\mathcal{C}) \cdot \mathsf{perf}\,(\mathcal{C})^3}$. As already mentioned, events that last very short are problematic. In these cases a higher granularity of the data would be needed for better observations, but is often absent. However, our measurements suggest that events occurring on a small time-window may be effectively classified with our clustered analysis. In particular, we show that some kind of sporadic phenomena, like worm attacks or misconfigurations, even if they may have a dramatic effect from the user's point of view, do not seem to have a measurable impact on the overall network structure. On the other hand, some very specific events, notably DDoS attacks against DNS root servers, happen to cause a significant change of the structure of the clusters. Thus, this kind of analysis may be a promising tool, to be used as a litmus-paper to test if one of these specific events is occurring in the network.

BGP storm due to worm attack. On July 19th 2001, the `Code Red II` worm was spreading in the Internet exploiting the indexing service vulnerability in the Microsoft Internet Information Server MS-IIS ([29]). Although an exponential growth in the advertisement rate (of routing paths) was observed by [15], we could not assess a significant change in the number of nodes and edges. In fact, Figure 8.9 shows that the number of nodes and edges is quite stable while the drop in the number of peers accounts for the loss of paths. These missing peers - AS715 and AS19092 - usually contribute with more than 15,000 paths each. Using the clustering technique, we could not spot any significant changes. In fact, about 75% of all nodes were either perfectly stable or switched between two clusters, the other nodes blinked in and out of our observed view or could not be associated to one cluster. The number and sizes of the clusters were stable. A similar behavior could be observed when on



(a) Number of nodes and edges

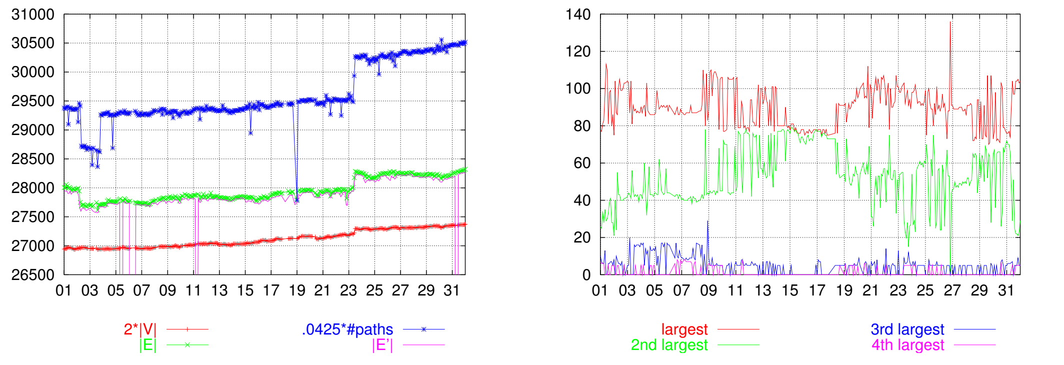


(b) Number of peers and paths

Figure 8.9.: Evolution around July 19th 2001

September 18th 2001, a extremely virulent worm, called `W32.NIMDA`, spread throughout the Internet using multiple methods to inflect both Windows servers and user machines ([102]).

Shutdown of KPNQwest/Ebone and RIPE NCC Test Traffic measures a 50% increase of alarms. In July 2002, KPNQwest, one of Europe's largest Internet backbone provider had a time-out. The company went offline on the 3rd and returned on the 25th ([74]). At the beginning of this event, the RIPE NCC measured a 50% increase of alarms on their TTMs (Test Traffic Measurement Boxes, [99]). We observed a drop in the number of peers, paths and edges. While the former two lasted only a few days, the latter one was present during the whole period. Figure 8.10 shows this evolution. The clustering is rather stable when we considered the three time periods (before, during and after the shutdown) separately. However, we could observe temporal migrations (small subsets formed new clusters or were swallowed up), new ASes 'entered' the system and old ASes 'left' during the transitions. The few drops in the number of edges of the commonly observed graph (see Figure 8.10(a)) are probably due to technical issues and do not reflect changes.

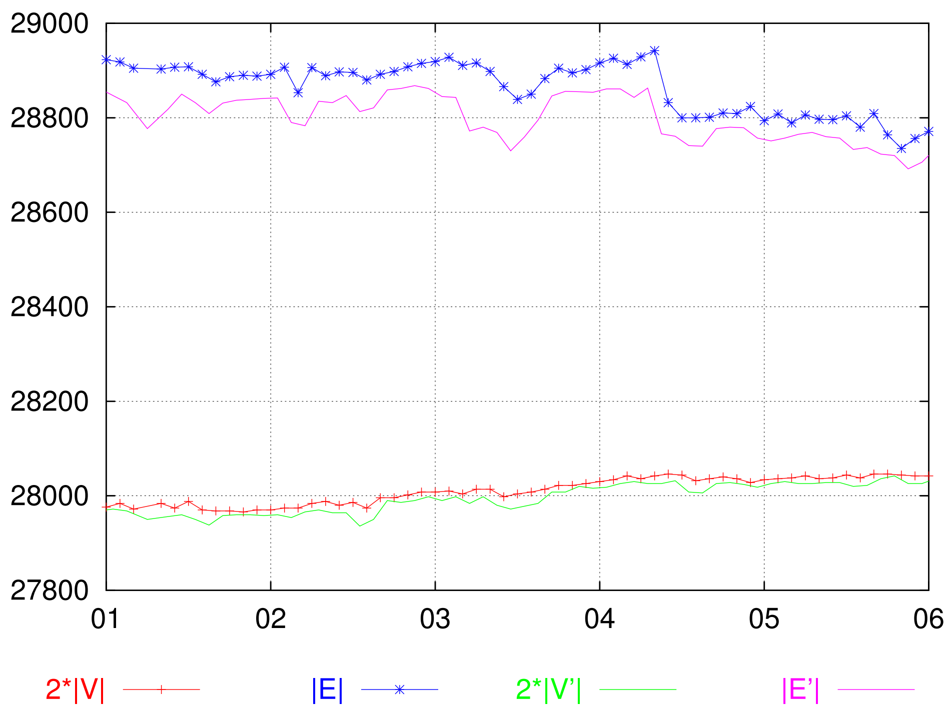


(a) Number of nodes, edges and paths (b) Sizes of clusters (sorted according to size)

Figure 8.10.: Evolution during July 2002

Misconfiguration in UUNet. UUNet (WorldCom) stopped talking to the rest of the Internet on October 3rd 2002 between 12am and 9pm (UTC), due to misconfiguration in some of their routers. This caused increases of respond times, delays, and packet losses ([80]). It affected many other ASes, because UUNet carried half of the world's Internet traffic. Although the number of edges seems to be affected, we could not observe a significant change (see Figure 8.11).

DDoS Attacks against 13 Internet Root Servers. On October 21st 2002, a series of well-coordinated, simultaneous DDoS attacks were launched from various points around the world, against each of the 13 Root Servers that are used for the Internet's Domain Name System (DNS) ([80]). The attack, which disabled nine of the 13 Root Servers, started at 8:45pm (UTC) and lasted approximately two hours. We could only partially observe the event, because *Oregon Routeviews* had a 6-hours blackout, due to connection time-outs, starting at October 22nd 2am (UTC) and the peer AS701 disappeared, causing a loss of approximately 17,600 paths earlier on the 21st (8am UTC). It never returned as an observing peer. Thus, we can only analyze the structure of the network before and during the attack, but not immediately afterwards. Also, the loss of paths limits the viewpoint. Similar to the



(a) Number of nodes and edges

Figure 8.11.: Evolution around October 3rd 2002

previous worm attacks, the number of nodes and edges is stable (see Figure 8.12). The clusterings are very stable, except for the point in time of 21st 8pm. In that instant a kind of splitting occurred. A subset of 29 elements and a single node emerged from the largest cluster and built their own clusters. For simplicity we denote the remaining part of the largest cluster by 1' and the other two with 1" and 1"' respectively, according to their size. Table 8.1 contains the geographic distribution of the ASes located in those clusters. Most of the elements in 1" and 1"' were perfectly stable while belonging to cluster 1 during the other time frames. The four Asian ASes – AS2518, AS4143, AS4728, and AS4766 – are a kind of exchange servers. The cluster 1" contains also several well-know ASes like AT&T, Cable and Wireless, former Genuity, Globalcrossing, Sprintlink, UUnet, Verio, Microsoft (three times) and Google. A more fascinating fact is that in each cluster 1' and 1" an unaffected or a less affected DNS root server (AS297 and AS3557) is present. This is a strong indicator that the clustering was affected. However, the degree of interaction remains open. The current granularity is not high enough for a further and more detailed analysis.

Sapphire worm attack. In the morning of 25th January 2003, the `Sapphire worm` (also known as `SQL slammer`) was released on the Internet. Exploiting a vulnerability in Microsoft SQL server, it multiplied itself rapidly and soon spread out over networks worldwide. From news headlines and activity on mailing lists it was clear the attack had an impact on the Internet's performance. Similar to the *DDoS Attack* our data source *Oregon Routeviews* had a blackout from 8am till 12am due

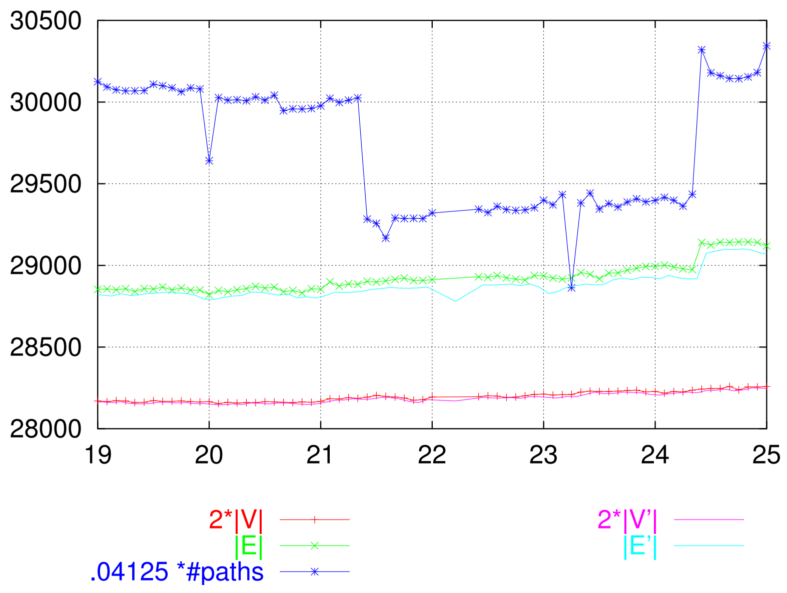


Figure 8.12.: Evolution around October 21st 2002

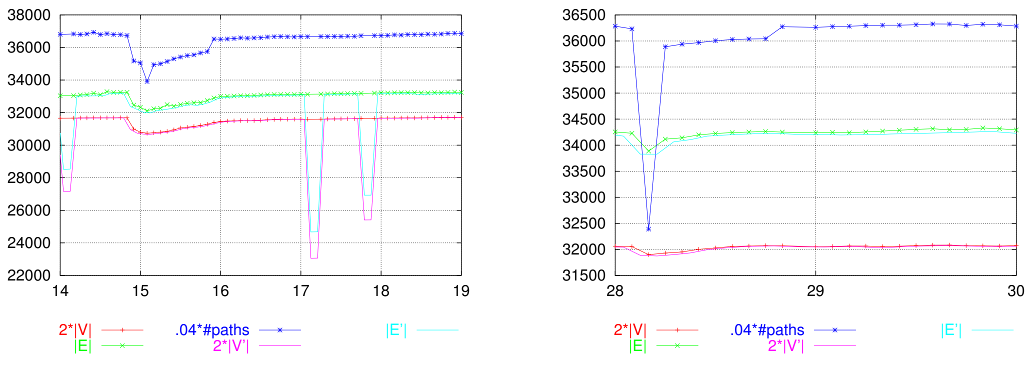
countries	clusters			
	1	1'	1"	1"'
US	74	48	26	-
Europe	9	9	-	-
Asia	11	7	3	1
Africa	1	1	-	-
Australia	1	1	-	-
unknown	1	1	-	-

Table 8.1.: The distribution of countries of the cluster and its segmentation.

to connection timeouts. Also two large peers - AS7660 and AS8297 - were absent on 26th. Each contributed with more than 16,000 paths. Thus our view point is very limited and can hardly be used to distinguish between worm attack and peer loss.

DDoS Attack on the RIPE NCC. Starting from 2pm (UTC) February 27th, 2003, the RIPE NCC network suffered a large DDoS attack (a distributed ICMP echo attack). Their network structure was affected not only by ICMP traffic. Network condition returned back to normal the same day at 4:30pm UTC. Shortly before (2am till 2pm) the peer AS5459 did not contribute, causing a loss of $\approx 7,100$ paths. Nothing unusual could be observed.

Blackouts. During August and September 2003, several blackouts happened in different geographic regions. On August 14th, around 6pm (UTC) started the cascading chain of events that lead to a gigantic blackout in the north-eastern part of America and Canada (refer to [1] for details). A 40-minute blackout took place in London on August 28th (around 6 pm). The last one struck parts of France, Italy and Switzerland on September 28th. It started around 2:30 am and power returned during the afternoon. The event in London could not be observed, since it was too



(a) US and Canada (b) France, Italy and Switzerland

Figure 8.13.: Evolution of different blackouts

short. The other two exhibited a similar pattern: a large drop in the number of paths occurred at the start of the event. This drop also caused a drop in the static indices (number of nodes and number of edges). The recovery phase was very short and the indices quickly got back to their old scores. The drops in the number of nodes and edges of the commonly observed graph (Figure 8.13(a)) is not due to the blackout itself. They are results of the disruption in the collection process. The reduced data set was relatively stable. Most nodes were present more than 80% of the time. Only a small set ($\approx$ 20 nodes) appeared only once or twice. The clustering itself is relatively stable. Figure 8.14 shows the temporal development of the sizes of the clusters. Smaller migrations, splittings or merging phenomena could be observed. Although there is no clear pattern or regularity apparent, it is a fact that these changes occurred more frequently during the recovery phase, thus giving a good indicator that clustering reflects aspects of the event. Similar observations could be made during the blackout in Europe. However, the impact on the structure was not comparable with respect to size and the recovery time much shorter.

8.4. Visualizations

After extensively studying structural properties of the AS Network, we present a visualization technique that emphasizes the hierarchical nature and provides further insight in the evolution. Standard visualization techniques, which include force-directed approaches [17] and spectral embeddings, usually capture the essential structure of a network quite well without the requirement of a priori knowledge. Unfortunately, these methods fail in the case of the AS Network and result only in 'hairy balls' as can be seen in Figure 8.15. The major reason for this behavior is the inhomogeneous connectivity pattern. The core of the network is dense but relative small and many edges link low-level shells with the core. As observed in the Section 8.2.2 about the abstraction, the AS Network can (very) roughly be seen

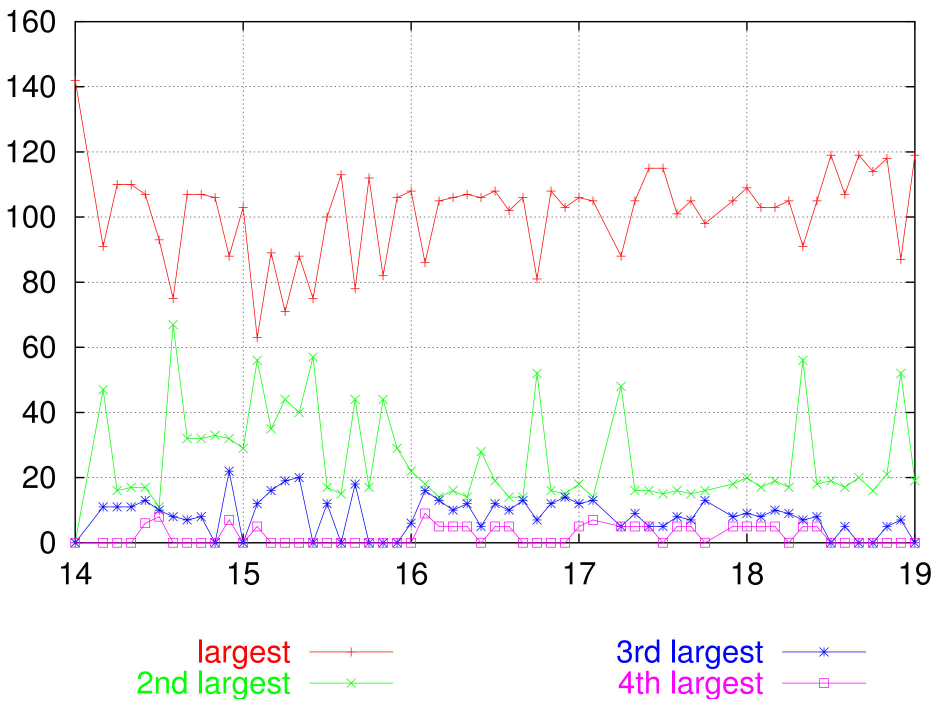


Figure 8.14.: Sizes of the different clusters during the blackout in the US and Canada

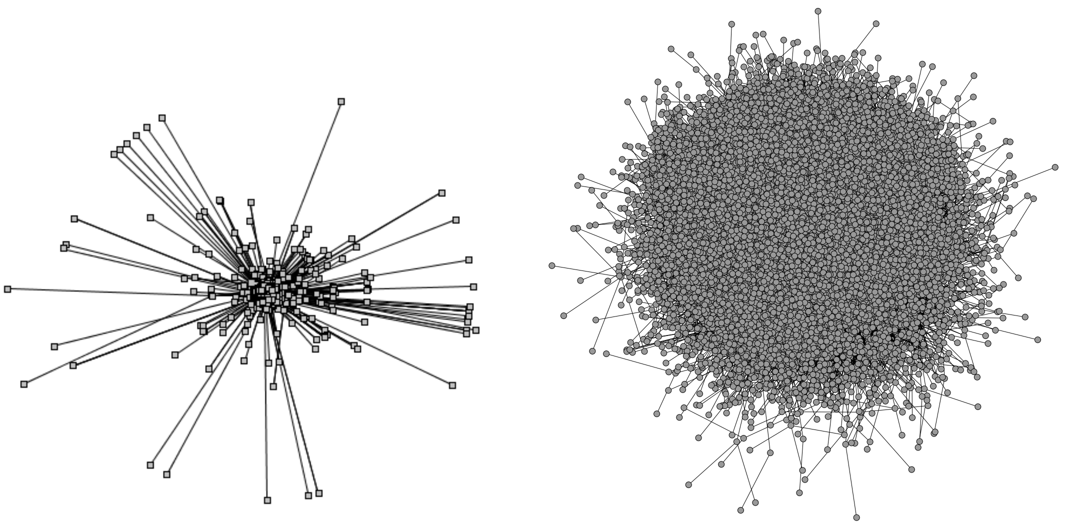
as a star. The standard techniques reflect this fact properly. In order to judge this artifical behavior further, we study Laplacian embeddings of the individual k-cores. See Figures 8.16 and 8.17 for an example. Note, that we used spectral embeddings since these layouts are deterministic and independent from initialization steps. As can be observed from the figures, the utilization of the drawing area improves with increasing the core level. In other words, the star-like structure is mainly caused by nodes having low coreness. Based on the hierarchy induced by the cores, we defined an incremental layout.

8.4.1. Layout for Hierarchies

We present an incremental algorithm to obtain a two and a half dimensional layout obeying the following criteria:

- All nodes and edges are displayed.
- The levels of hierarchy are highlighted.
- Inter- and intra-level edges are emphasized.

Two and a half dimensional visualizations have been proposed frequently for network data from various applications. Related methods use the third dimension to display a graph hierarchy [16, 38] or graphs that evolve over time [18]. In our layouts, the third dimension will also correspond to the hierarchy, i. e., nodes with the same level


(a) force-directed approach (with node overlaps)

(b) force-directed approach (without node overlaps)

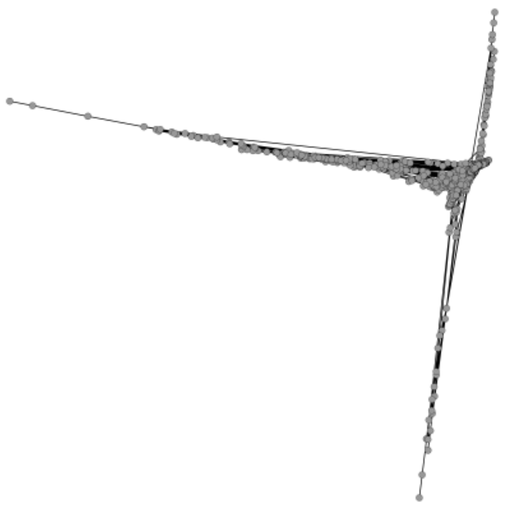
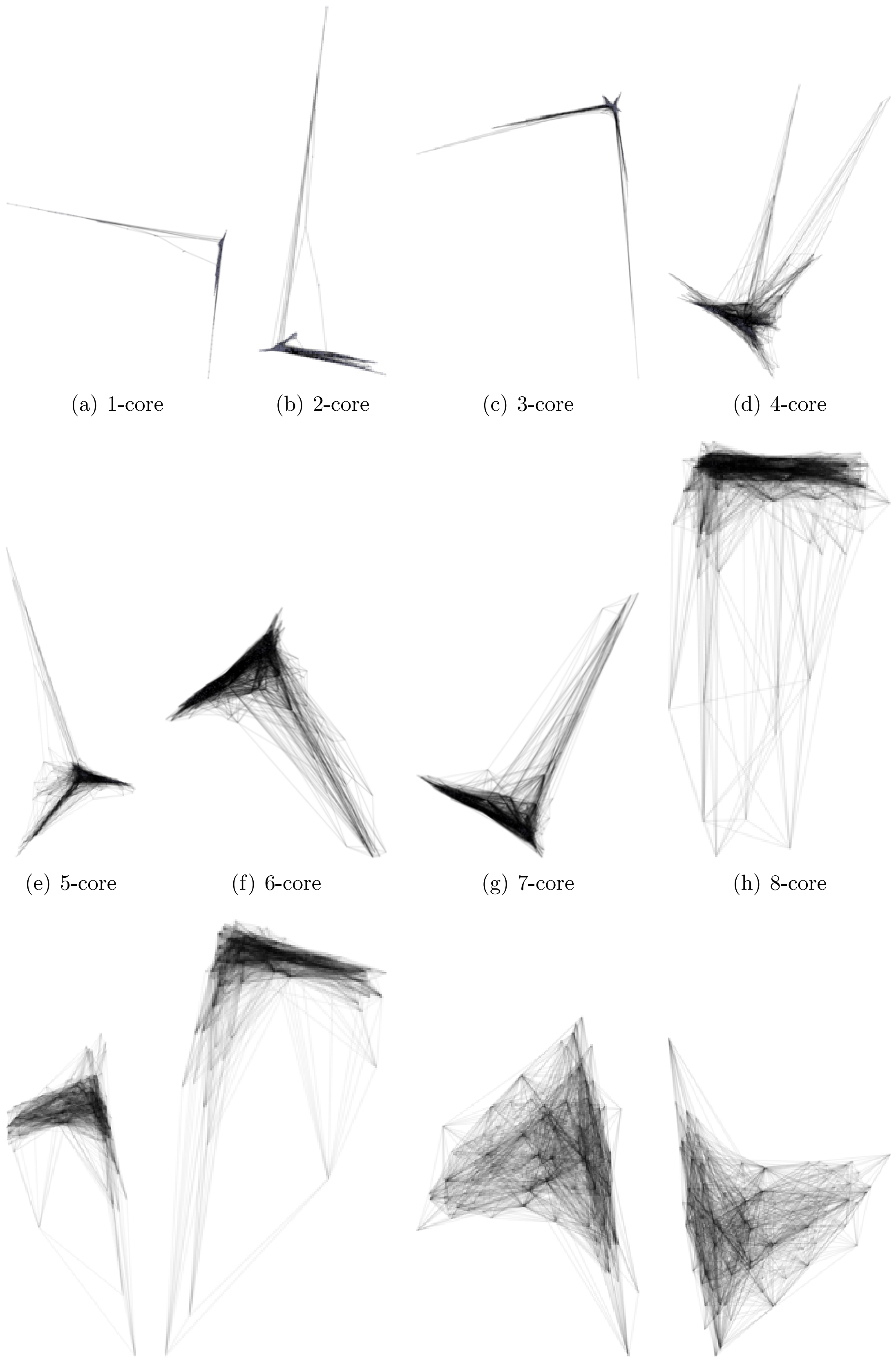

(c) standard Laplacian embedding

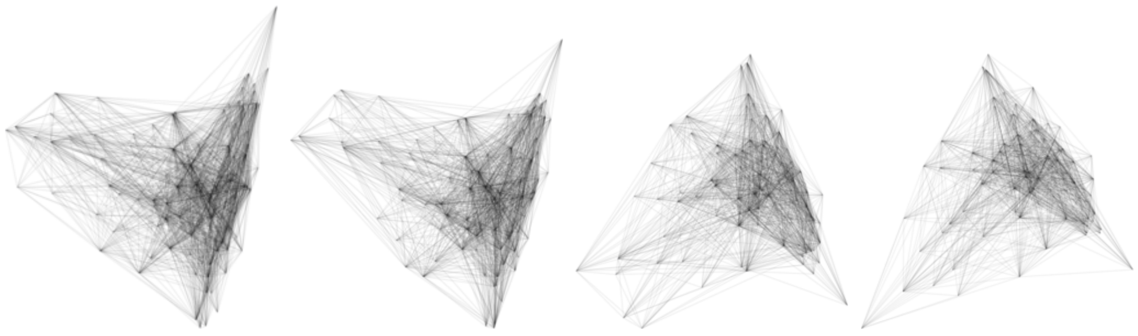
Figure 8.15.: Standard visualization techniques applied to an instance of the AS Network (January 1st, 2003).


(a) 1-core (b) 2-core (c) 3-core (d) 4-core

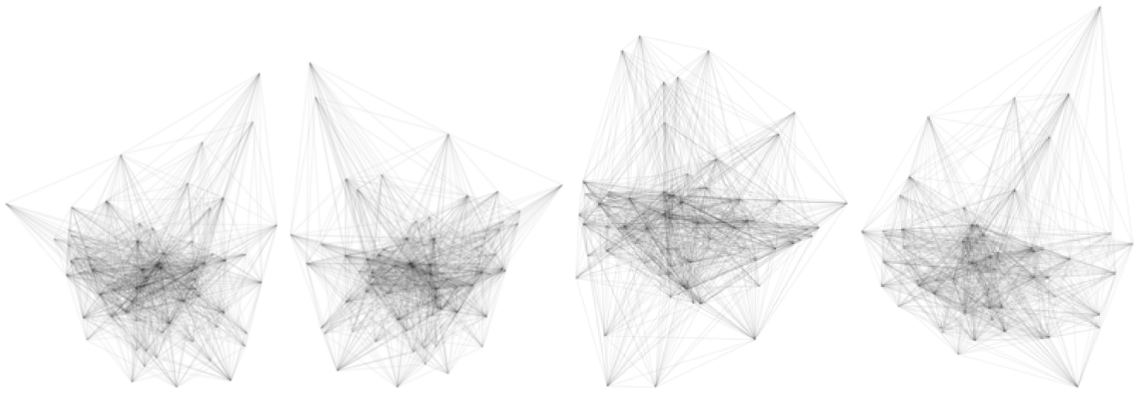
(e) 5-core (f) 6-core (g) 7-core (h) 8-core

(i) 9-core (j) 10-core (k) 11-core (l) 12-core

Figure 8.16.: Spectral Layouts of the cores of the AS Network (01/01/2003)



(a) 13-core (b) 14-core (c) 15-core (d) 16-core



(e) 17-core (f) 18-core (g) 19-core (h) 20-core

Figure 8.17.: Spectral Layouts of the cores of the AS Network (01/01/2003)

(in the hierarchy) will have the same z–coordinate. The remaining two-dimensional layout is incrementally created by first initializing the top element of the hierarchy with a spectral embedding and then iteratively placing nodes in lower levels using force-directed techniques.

More detailed, the algorithm for the two-dimensional layout is divided into two phases. In the first part, the top element of the hierarchy has to be properly layouted. The characteristics of this initialization will essentially influence the whole layout. We chose a spectral embedding due to several reasons. The main benefits are its deterministic nature and the relation between the distances in the embedding and graph-theoretical distances [56, 27]. In the second phase, the lower levels are added using a combination of barycentric and force-directed placement. More precisely, when a new level of the hierarchy, e. g., $\eta(i) \setminus \eta(i+1)$ for some i, is added, the barycentric layout assigns preliminary coordinates to nodes with level i considering only neighbors in the same or in higher levels. Note that this assignment is deterministic, since there is only one connected component and some nodes are already fixed. Unfortunately, barycentric layouts also have a number of drawbacks. First, nodes that are structurally equivalent in the subgraph induced by $\eta(i)$ are assigned to the same position. Second, all nodes are placed inside the convex hull of the already positioned nodes. In particular, this means that the outermost nodes are those belonging to the top level of the hierarchy which is clearly contradictory to the intuition of the importance. To overcome these difficulties, we use the barycentric layout as an initial placement for a subsequent force-directed refinement step, where only newly added nodes are displaced. In addition, a force-directed approach is applied for all nodes in order to relax the whole layout. However, the number of iterations and the maximum movement of the nodes is carefully restricted not to destroy the previously computed layout. A special feature of this relaxation step is the use of non-uniform natural spring lengths $l(u, v)$, where $l(u, v)$ scales with the smaller level of the two incident nodes u and v. Thus, the effect of a barycentric layout is modeled, since edges between nodes of high coreness are longer than edges between nodes of low coreness. Accordingly, these springs prevent nodes with high coreness from drifting into the center of the layout. The pseudo code is given in Algorithm 7. Beside the choice of the underlying hierarchical decomposition, the

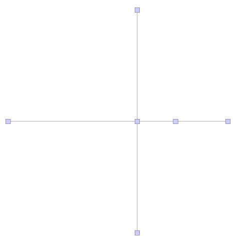
Algorithm 7: Generic hierarchy layout algorithm

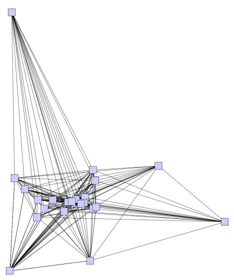
```
Input: graph $G = (V, E)$, hierarchy $\eta$
Output: two-dimensional layout for $G$

$k \leftarrow$ size of hierarchy $\eta$
for $\ell \leftarrow 1, \ldots, k$ do
    $V_\ell \leftarrow \eta(\ell) \setminus \eta(\ell + 1)$
    $G_\ell \leftarrow G[V_\ell]$

calculate spectral layout for $G_k$
for $\ell \leftarrow k - 1, \ldots, 1$ do
    calculate barycentric layout for $V_\ell$ in $G_\ell$, keeping $G_{\ell+1}$ fixed
    calculate force-directed layout for $V_\ell$ in $G_\ell$, keeping $G_{\ell+1}$ fixed
    calculate force-directed layout for $G_\ell$
```

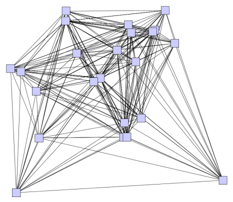
algorithm offers a few more degrees of freedom that allow an adjustment to a broad range of applications. In the next section, we fix these degrees in order to obtain a suitable layout for the AS Network.

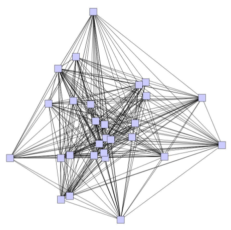
8.4.2. Fitting the Parameters

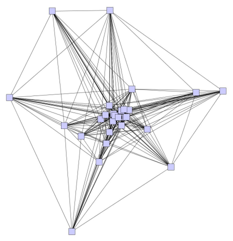
As extensively discuss in Section 8.2.2 for example, the core decomposition is a natural candidate for the hierarchy. For the spectral layout we propose a modified Laplacian matrix $L' = 1/4 \cdot D - A$ already considered in [19]. Our experiments showed that the application of the normalized adjacency matrix results in comparably good layouts while the standard Laplacian matrix performs significantly worse. An example is given in Figure 8.18, where the Laplacian, the modified Laplacian, and the normalized adjacency matrix is used for the spectral embedding of the core of the AS Network at three different points in time. The force-directed placement is performed by a variant of the algorithm from [45]. Unlike the original algorithm, we calculate the displacement only for one vertex at a time and update its position immediately. Furthermore, we use the original forces but with non-uniform natural edge lengths $l(u, v)$ proportional to $\min\{\text{level}(u), \text{level}(v)\}^2$. For the refinement step

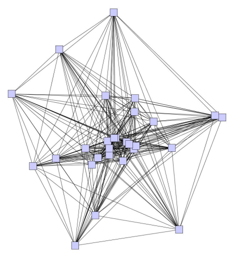

(a) normalized adjacency matrix (2001)

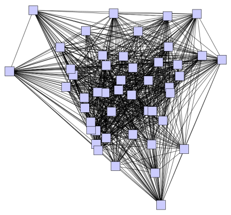

(b) Laplacian matrix (2001)

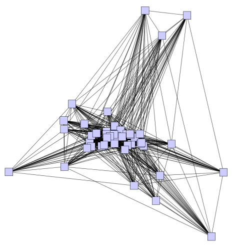

(c) modified Laplacian matrix (2001)


(d) normalized adjacency matrix (2002)


(e) Laplacian matrix (2002)


(f) modified Laplacian matrix (2002)


(g) normalized adjacency matrix (2003)


(h) Laplacian matrix (2003)

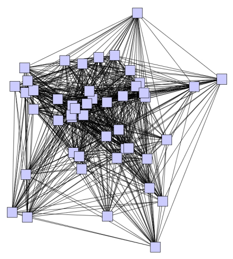

(i) modified Laplacian matrix (2003)

Figure 8.18.: Different spectral layouts for the core of the AS Network taken at different points in time (always June 1st).

where only new nodes are displaced using forces, we perform at most 50 iterations. The second application of forces to relax the whole layout is already stopped after 20 iterations.

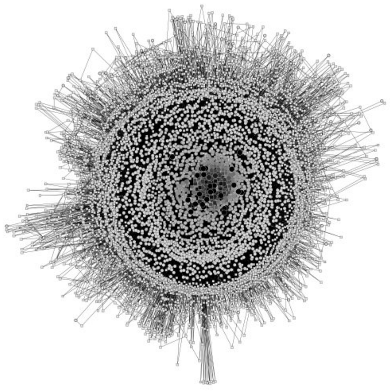
8.4.3. Result and Evaluation

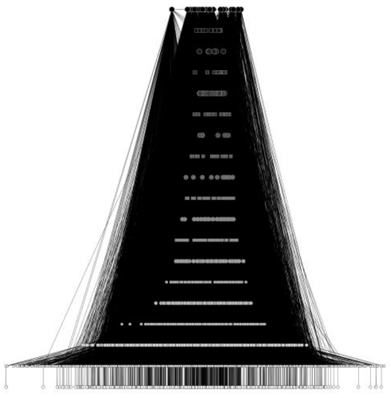
Examples of the final layout for different points in time are given in Figure 8.19. On the left hand side, Figures 8.19(a), 8.19(c), and 8.19(e), show the two-dimensional layout, i. e., the x-y–projection, while the right, Figures 8.19(b), 8.19(d), and 8.19(f), display a level projection, in our case the projection is done into the x-z–plane. A 'full' three-dimensional view is given in Figure 8.20. Due to the prominent form induced by the network, we refer to it as *volcano-like shape*.

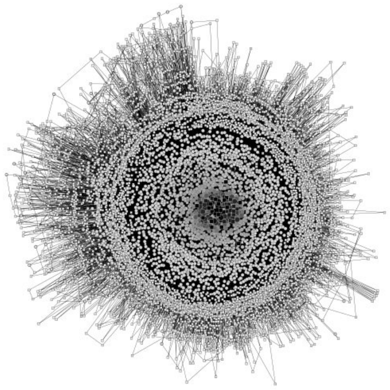
While techniques like force-directed or spectral methods [35, 17] place the majority of the nodes at a central position, here only 6-10% of the nodes are placed close to the center, constituting a peak (as in Figure 8.21). Interestingly, the nodes with coreness two or three, which are the majority of nodes, are placed in a concentric annulus around the peak. A closer examination reveals three almost separated radial areas around the center. The first one mainly contains the 3-core shell, while the 2-core shell forms the second and third area that are distinguished by their density. Figures 8.24(a)-8.22(c) show the corresponding nodes in the lower shells. Using several snapshots over time, we found a positive correlation of $0.67 - 0.78$ between the distance from the center and the coreness. The significance of this correlation is increased by the following facts that conceptually inhibit it: first, nodes in the 1-core shell are placed very close to their anchor nodes in higher core-levels which can be quite scattered or close to the center (see Figure 8.24(a) as an example), and second, nodes with coreness two or three constitute the majority of nodes and occupy a broad annulus rather than a ring. Especially for the 2- and 3-core shell, we can further identify two classes of nodes. In both cases there is a relatively uniform class of nodes drawn towards the center and a second class of nodes in the periphery. Nodes of medium and large coreness (greater than five) are contained in the convex hull of the core which documents the relation between importance and coreness. These properties can be observed over a time period of four years. The well-known growth of the AS Network hardly affects the layout, although we observed a slight decreasing spatial distance between the 2- and the 3-core shell. In order to further evaluate our technique, we consider the DIMES data set and artifically generated networks.

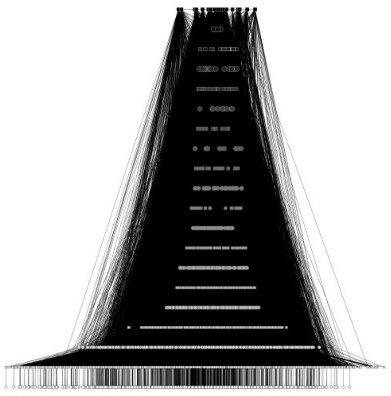
Generated Graphs

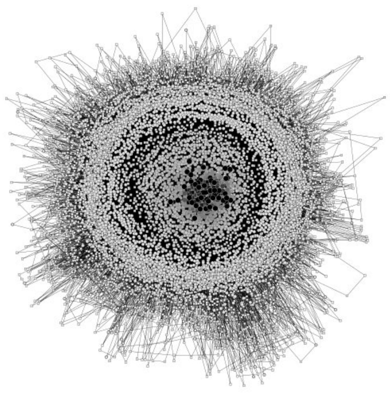
We used INET 3.0 [67] to generate artifical graphs that should exhibit a similar AS Network topology. The example we will explicitly discuss is a simulation of the AS Network of June, 1st 2001, which contained 11,211 nodes, 23,689 edges and has maximum coreness of 19. However, the generated graph with 11,211 nodes has almost 12,000 edges more than the original network, but the maximum coreness is only eight. Similar discrepancies were observed for instances taken at other points in


(a) 2D layout (2001)


(b) level projection (2001)


(c) 2D layout (2002)


(d) level projection (2002)


(e) 2D layout (2003)

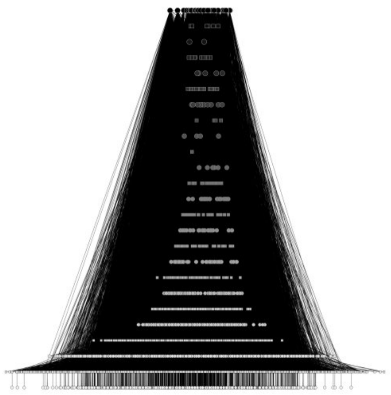

(f) level projection (2003)

Figure 8.19.: Two-dimensional layout and level projection of the AS Network. The instances are always taken on June 1st.

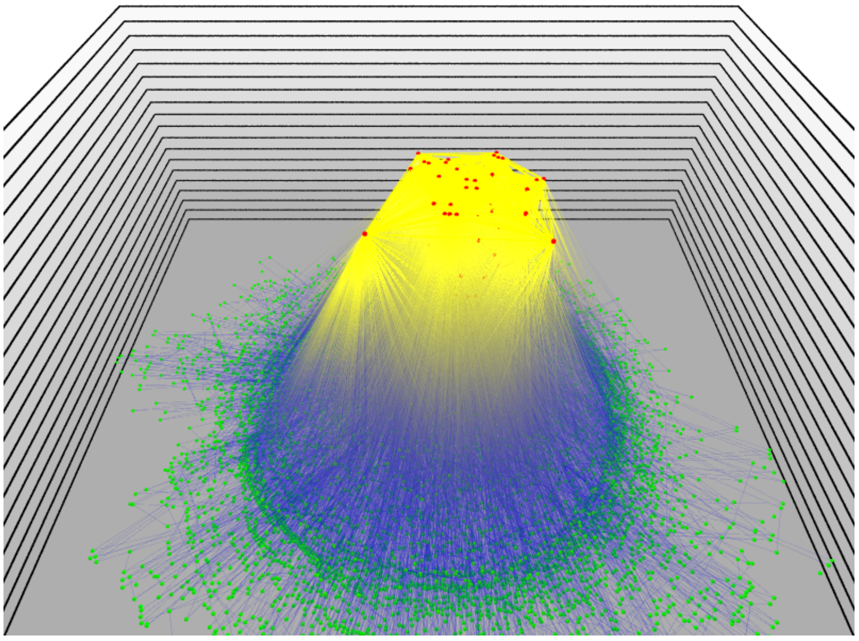


Figure 8.20.: Final layout of the AS Network (June 1st, 2001)

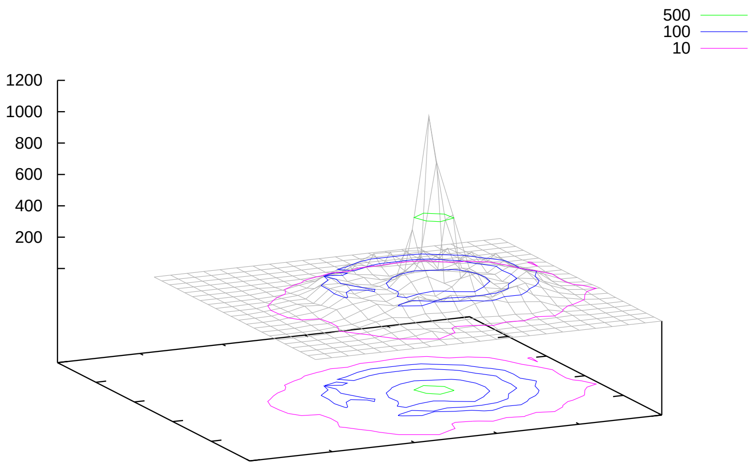


Figure 8.21.: Two-dimensional histogram of the nodes in the AS Network (January 1st, 2005) in a final layout.

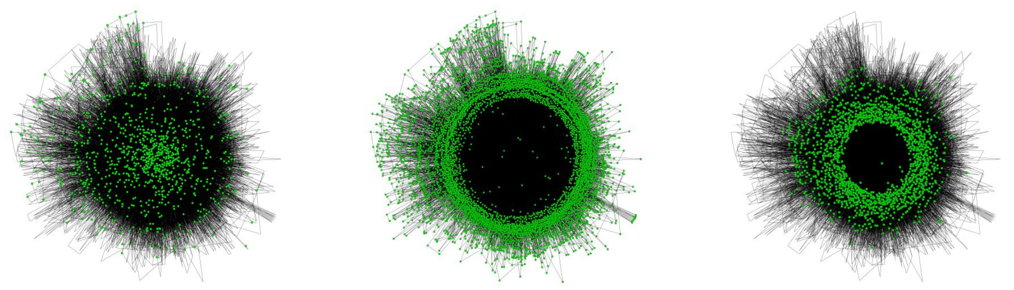

(a) restricted to 1-core shell (b) restricted to 2-core shell (c) restricted to 3-core shell

Figure 8.22.: Two-dimensional layout where only nodes in certain cores are drawn (06/01/02).

time. An obvious difference of the generated graph induced by this fact is the more

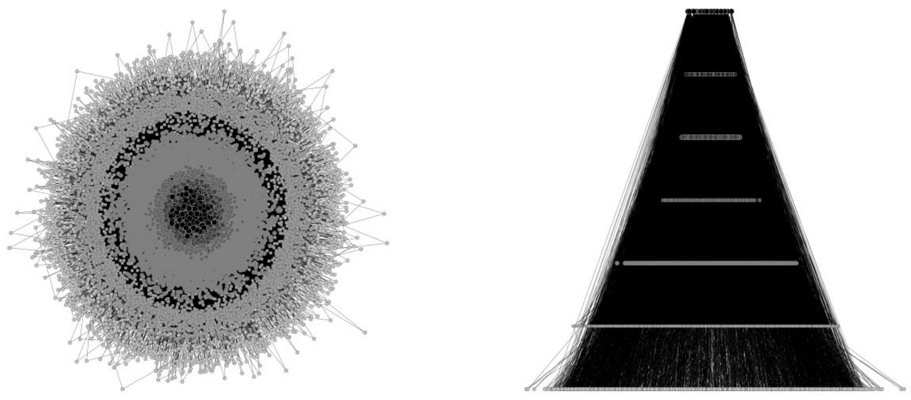

(a) 2D layout (b) level projection

Figure 8.23.: Figures of the generated graph with 11,211 nodes

uniform distribution of cardinalities of the core shells. Accordingly, in the layouts the separation of the different core shells is less significant. However, the 2D layouts are still satisfactory with respect to the illustration of the core hierarchy. Also the characteristic annulus-shapes of the 2- and 3-core shell are also present, as can be seen in Figure 8.24.

DIMES

The data sets correspond to the period of March to June 2005. We obtain 48, 073 edges (corresponding to 20, 406 ASes) from Routeviews and 38, 928 edges (corresponding to 14, 154 ASes) from DIMES. Of these, 21, 725 edges exclusively belong to Routeviews, and 12, 580 edges exclusively belong to DIMES. The rest of the edges are common to both data sets. The union of the two data sets thus results in 60, 653 unique edges (corresponding to 20, 612 ASes). Figure 8.25 visualizes the networks obtained from Routeviews and DIMES data sets. In order to highlight structural

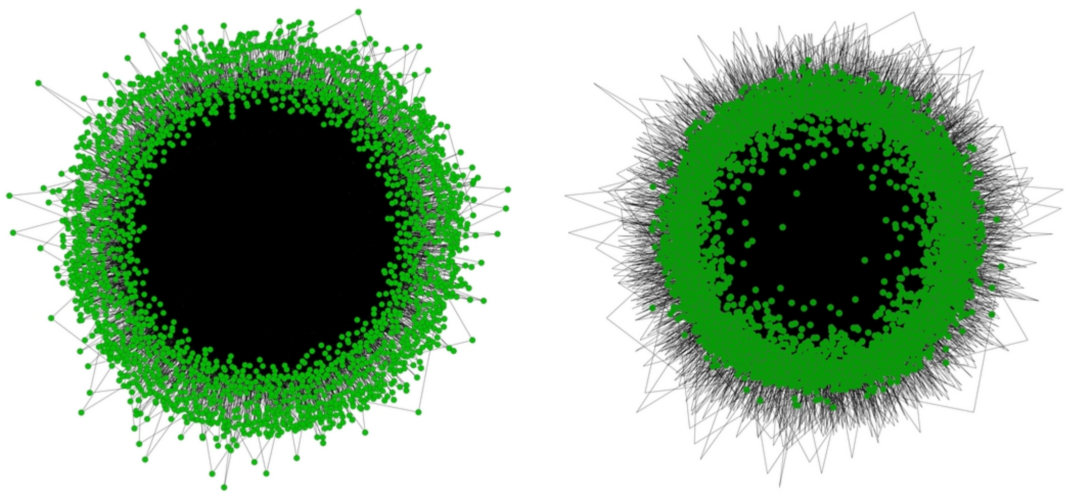
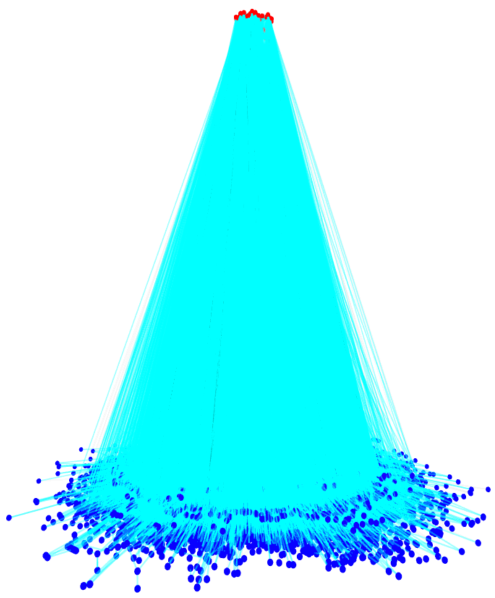

(a) restricted to 2-core shell (b) restricted to 3-core shell

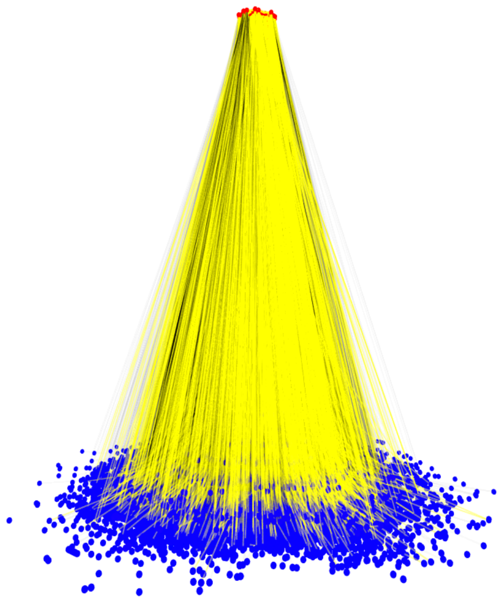
Figure 8.24.: Two-dimensional layout where only nodes in certain cores are drawn.

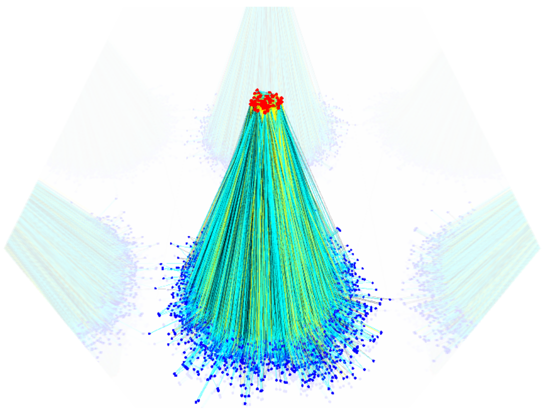
properties, we color edges that are obtained only from Routeviews as cyan, edges that are obtained only from DIMES as yellow, and edges that are present in both data sets as black.

A first glance shows that while the visualizations are very similar, the DIMES data set is slightly smaller and the geometric difference of the set of collected edges of Routeviews and DIMES is surprisingly large (about 42% overlap). In other words, 58% of the edges appear in only one data set. An interesting observation is that many edges that are only discovered by DIMES are incident to the core. Figure 8.26 shows the plots of the coreness of the end-nodes of the edges versus their rank, i. e., positioned in the non-decreasing sorted sequence. The coreness is calculated in the graph that consists of the union of the two data samples. This enables us to set up a less biased comparison. The Routeviews data sample is plotted as a solid line, while the DIMES sample is dotted. Figure 8.26(a) plots a data point for each edge belonging to Routeviews or DIMES using the maximum coreness of the end-nodes (as y-axis), while Figure 8.26(b) shows the same scenario using the minimum coreness. A similar comparison is made in Figures 8.26(c) and 8.26(d) where the common edges are omitted. Thus the solid lines represent the distribution of edges that are exclusively observed by Routeviews, and the dotted lines correspond to the exclusive part of the DIMES sample. In principle, the distributions of Routeviews and DIMES are very similar, except for the broad tail of the Routeviews distribution observed in Figure 8.26(c), which is an interesting observation requiring further investigation. However the overall similarity of the plots together with the resembling visualizations reveal that our visualization technique is not only applicable to Oregon Routeviews data and that data obtain from DIMES or Routeviews both contain similar intrinsic patterns. A deeper analysis of the difference of the two extraction methods was not possible due to a lack of data from DIMES.

This concludes our structural investigation of the AS Network. The remaining


(a) Routeviews


(b) DIMES


(c) Union of Routeviews and DIMES

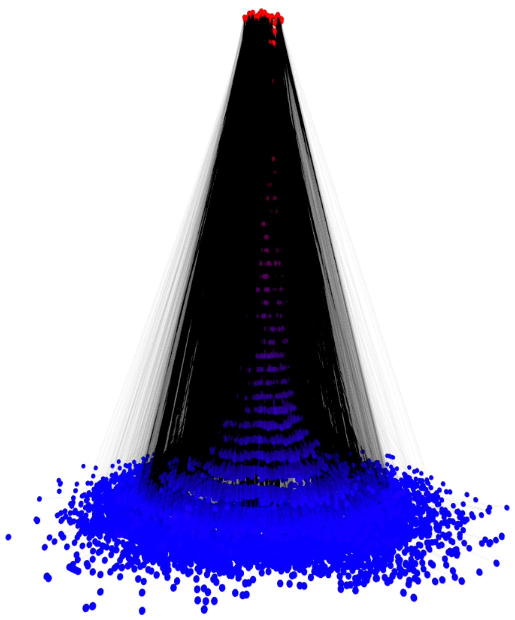
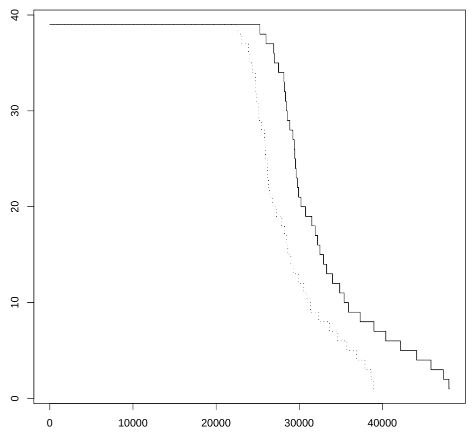
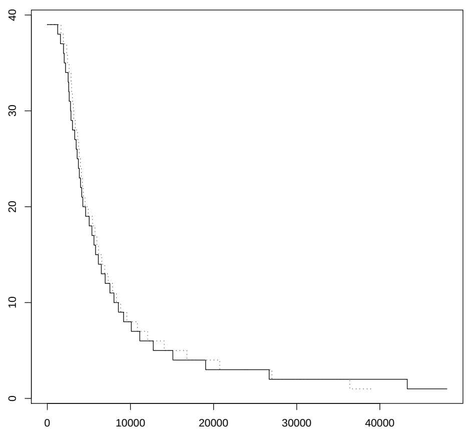

(d) Intersection of Routeviews and DIMES

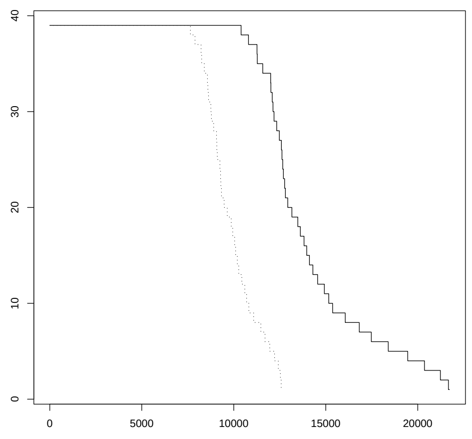
Figure 8.25.: Two and a half dimensional layouts of the AS Network obtained from Routeviews and DIMES. Color code: cyan-Routeviews, yellow-DIMES, black-both Routeviews and DIMES



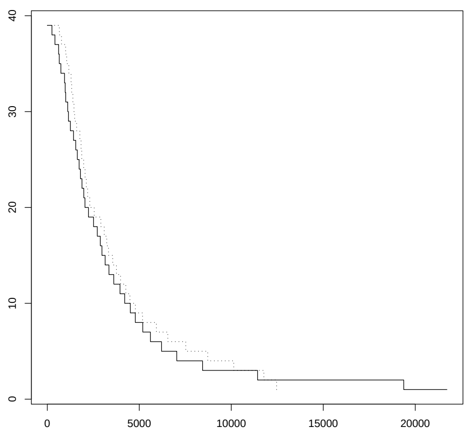
(a) maximum coreness



(b) minimum coreness



(c) maximum coreness



(d) minimum coreness

Figure 8.26.: Comparison of coreness distributions of the edges. Figure (a) and (b) compare Routeviews (solid) with DIMES (dotted), while Figure (c) and (d) compare the exclusive sets. The x-axis denotes the number of edges, and y-axis the minimum or maximum coreness.

part of the is chapter focuses of the relation between the AS Network and an overlay generated by the popular file sharing application Gnutella.

8.5. Analysis of Overlay-Underlay Topology Correlations

The recent growth of Peer-2-Peer (P2P) file sharing applications with respect to total Internet traffic [92, 68] has led to an unprecedented interest in their analysis [26, 2]. The P2P file sharing applications create overlay networks on top of the Internet underlay. In other words, the P2P file sharing systems create overlay topologies on top of the AS Network. When constructing their topology by forming neighborhoods, most structured and unstructured P2P protocols do not explicitly take the underlay routing into account. Yet as the underlay connectivity determines the overlay performance, understanding the correlation between routing in the overlay and underlay layers is crucial. There have been some investigations on such correlations recently. Using game theoretic models, Liu et.al. study the interaction between overlay routing and traffic engineering within an AS [76]. Ratnasamy et.al. present in [81] a node-partitioning scheme that allows overlay nodes to choose peers that are relatively close in terms of network latency. An analysis of routing around link failures [90] finds that tuning underlay routing parameters improves overlay performance. Most investigations tend to point out that the overlay topology does not appear to be correlated with the underlay, e. g., [2], but the routing dynamics of the underlay do affect the overlay in ways not yet well understood.

In the following, we analyze the correlation between overlay and underlay topologies of the Gnutella [55] network using the above visualization technique.

8.5.1. Formalization

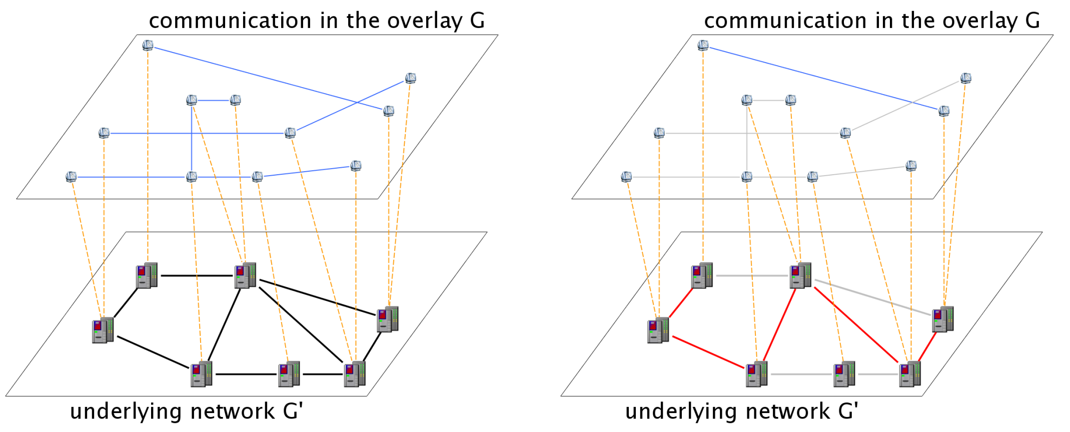
Before presenting our approach, we give a formal description of the problem.

Definition 8.1. *An* overlay *is given by a four-tuple* $\mathcal{O} := (G, G', \varphi, \pi)$*, where*

- $G = (V, E, \omega)$ *and* $G' = (V', E', \omega')$ *are two weighted networks with* $\omega\colon E \to \mathbb{R}$ *and* $\omega'\colon E' \to \mathbb{R}$*,*
- $\varphi\colon V \to V'$ *is a mapping of the nodes of* G *in the nodeset of* G'*, and*
- $\pi\colon E \to \{p \mid p \text{ is a (un-/directed) path in } G'\}$ *is a mapping of edges in* G *to paths in* G' *such that*

$$\operatorname{source}(\pi(\{u,v\})) = \varphi(u) \text{ and } \operatorname{target}(\pi(\{u,v\})) = \varphi(v) \text{ or vice versa }.$$

The interpretation of Definition 8.1 is that G models the overlay itself, the network G' corresponds to the underlay, and the two mappings establish the connection between the two networks, i. e., map nodes to their correspondings and overlay edges


communication in the overlay G
underlying network G'
communication in the overlay G
underlying network G'

(a) Both networks G and G' with the mapping φ.

(b) Highlighting one edge e in G and the corresponding path $\pi(e)$ in G'.

Figure 8.27.: Example of an overlay $\mathcal{O} := (G, G', \varphi, \pi)$. The mapping φ is represented by dash lines between nodes in G and G'.

which coincides with direct communication in the overlay to the actual path which realizes the communication. Thus the problem of analyzing the correlation of two overlays is reduced to find matching matters between G and G'. Since the underlay networks for two different overlays need not be the same, e. g., sampled at different points in time or location, four graphs have to be related to each other. Our approach provides a potential solution by mixing information of the overlay and the corresponding underlay. More precisely, the overlay is reduced to a composition of elements in the underlay, see Definition 8.2.

Definition 8.2. *Given an overlay $\mathcal{O} := (G = (V, E, \omega), G' = (V', E', \omega'), \varphi, \pi)$. The* induced overlay $\widetilde{\mathcal{O}} := H := (V'', E'', \omega'')$ *is a weighted network, where*

- $V'' := \{v \in V' \mid \exists\, e \in E\colon \pi(e)$ *contains* $v\}$,
- $E'' := \{e' \in E' \mid \exists\, e \in E\colon \pi(e)$ *contains* $e\}$, *and*
- $\omega''(e') := \sum_{e \in E} \omega(e) \cdot [e'$ *contained in* $\pi(e)]$.

The weight function ω'' is also called appearance weight.

In other words, the induced overlay corresponds to the subgraph of the underlay that is required to establish the communication in the overlay. Note that the defined weight can be interpreted as the load caused by the communication and thus is independent of a weighting in the underlying network. The problem of analyzing the correlation is now reduced to find the patterns between a subgraph and its parenting graph that occur in both induced overlays.

8.5.2. Case Study: Gnutella

The following case study has been performed in collaboration with the group of Prof. Dr. Anja Feldmann at the Technische Universität München (TUM) who provided the technique to crawl the Gnutella network and a data set. The results are also published [3, 4]. Our aim is verify or falsify if Gnutella has a random topology. We briefly summarize the required technical aspects.

Crawling Data Sets

Gnutella is a popular file-sharing network with 2 million users [96], and has attracted a healthy interest from researchers [97, 84, 88]. The Gnutella [54] network comprises of agents called servents, who can initiate as well as serve requests for resources. When launched, a servent searches for other peers to connect to by sending Hello-like `Ping` messages. The `Pings` are answered by `Pong` messages, which contain address and shared resource information. Search queries are flooded within the Gnutella network using `Query` messages, and answered by `Query Hits`. To limit flooding Gnutella uses TTL and message IDs. Each answer message (`Query Hit`/`Pong`) traverses the reverse path of the corresponding trigger message. While the negotiation traffic is carried within the set of connected Gnutella nodes, the actual data exchange of resources takes place outside the Gnutella network, using the standard HTTP protocol. Due to scalability problems, later versions of Gnutella [55] introduced a hierarchy which elevates some servents to ultrapeers, while others become leaf nodes. Each leaf node connects to a small number of ultrapeers while each ultrapeer maintains a large number of neighbors, both ultrapeers and leafs. To further improve performance and to discourage abuse, the `Ping`/`Pong` protocol underwent semantic changes. Answers to `Ping`s are cached and too frequent `Ping`s or repeated `Query`s may cause termination of connection.

In order to analyze overlay structure, a representative set of edges in the P2P network need to be identify. An edge means a direct P2P connection between two overlay nodes. The most obvious way of finding edges in a P2P network is to create some by participating. Yet these are not representative as they are highly biased by the location and software of the participant. Rather, we wish to identify edges in the P2P network where neither of the two nodes is controlled by us. We refer to such nodes are remote neighbor servents.

Due to the introduction of Pong caching scheme and the rapid fluctuation in Gnutella networks (we measured the median incoming/outgoing connection duration to be 0.75/0.98 seconds), the simple crawling approach used in [88] is no longer sufficient. One cannot assume that answers to `Ping`s with TTL 2 (namely crawler Pings) contain still active servents. They should, however, have been remote neighbor servents at some point.

To cope with these complications, we deploy a combination of active and passive techniques to explore the Gnutella network. Our passive approach consists of an ultrapeer based on the GTK-Gnutella [57] program. The goal is to have an ultrapeer that behaves like a normal node in the network, yet worthwhile to connect to. It shares 100 randomly generated music files (totaling 300 MB in size) and maintains 60 simultaneous connections to other servents. To derive various statistics the servent is instrumented to log per-connection information augmented with a packet level trace. The passive approach gives us a list of active servents.

The active approach consists of a multiple-client crawler that uses `Ping`s with TTL 2 to obtain a list of candidate servents. Using `Query`s with TTL 2 allows us to get a set of remote neighbor servents. These servents are then contacted actively to further advance the network exploration. This approach allows us to discover Gnutella edges that existed at a very recent point of time. When interacting with other servents, our crawler pretends to be a long- running ultrapeer with an acceptable querying scheme. It processes incoming messages and has a non-intrusive `Ping`/`Pong` behavior. For instance, the servent issues `Query`/crawler Pings only to those peers that have already responded with a `Pong`, `Ping`s are issued only to those servents that send one themselves. This pragmatic behavior seems to avoid bans. The client uses `Query` messages with catchwords like mp3, avi, rar. One can expect `Query`s to yield only a subset of neighbors due to the presence of free-riders [88].

Our combined active/passive approach integrates the crawler into the ultrapeer. Experiments with the unmodified and modified ultrapeer confirm that the changes did not alter the characteristics of incoming connections.

Evaluation

We collected Gnutella logs for one week in April 2005, and map the IP addresses of Gnutella nodes to ASes using the BGP table dumps offered by Routeviews [85] during the week of April 14, 2005. This results in 2964 unique AS edges involving 754 ASes, after duplicate elimination and ignoring P2P edges inside an AS. For the random network, we pick end-points at the IP level by randomly choosing two IP addresses from the whole IP space. These edges are then mapped to ASes in the same manner as for the Gnutella edge. The considered instance which we use for the visualization and the diagrams has 4975 unique edges involving 2095 ASes. The different size of the graphs are a result of the generation process: we generated the same number of IP pairs for the random networks as observed in the Gnutella sample, and applied the same mapping technique to both sets. This way, the characteristics of Gnutella are better reflected than by directly generating a random AS network of the same size as the induced overlay of Gnutella. Using the visualization techniques for the AS Network presented in Section 8.4.1, we create a visualization for the induced overlays by first applying the layout to the corresponding underlying network and then draw only those edges contained in the induced overlay. The results are shown in Figure 8.28. We can observe that while both networks have many nodes with large degrees in the center, the random network possesses several nodes with large degree in the periphery. Gnutella, on the other hand, has almost no such nodes.

This visualization indicates that Gnutella may differ from random topologies. To verify this claim, we analyze structural dependencies between the induced overlay and the underlying AS Network. Edges in the underlay network are not equally loaded as some edges appear in more communication paths than other. As it is not

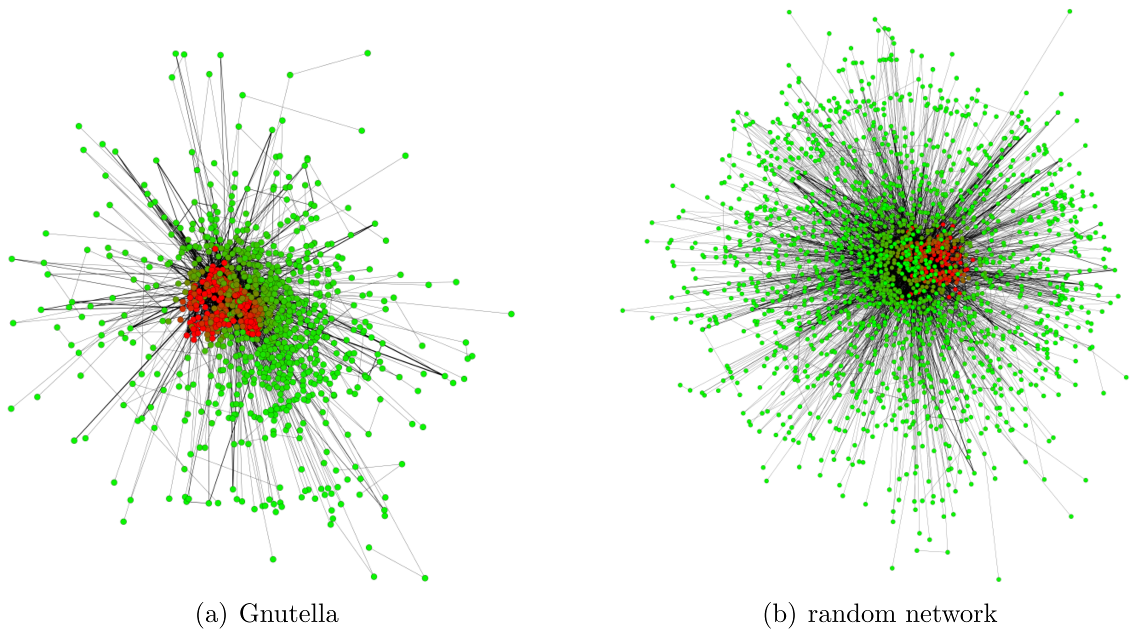

(a) Gnutella (b) random network

Figure 8.28.: Visualization of the induced overlay where the node positions are calculated with the above layout technique for the AS Network and only edges present in the induced overlay are drawn. Node color codes the coreness of the ASes in the underlay.

possible to measure the actual traffic on the individual edges (the edge weight ω), we consider a simplified model where a single communication causes one unit of traffic to be routed. Thus, the appearance weight of an edge corresponds to its load. The real load of an edge in the underlay network (including all the traffic caused by other applications) naturally is larger. Comparing these two loads reveals whether the P2P communication has characteristics similar to the accumulated load. This helps understanding and enhancing the underlay network topology and application level routing techniques. However, measuring the traffic load in the underlay network is not trivial. Even in a simplified model where we consider the load to be equal to the number of appearances in router-path announcements, the measurement heavily depends on the collection of routers and the collection process and is thus biased. Hence, we compare the appearance weight with node-structural properties of the corresponding end-nodes in the original underlying network. We focus on the properties degree and coreness, as both have been successfully applied for the extraction of customer-provider relationship as well as visualization [98, 49], as these properties reflect the importance of ASes. We establish the relation between the edge weight and the node structural measurements by systematically comparing the weight with both the smaller and the larger value of end-nodes. Figure 8.29 (upper four) shows the plots of the weights versus the degree. From the plots it is apparent that the appearance of an edge and its end-nodes' degrees are not correlated in both the Gnutella and the random network, as no pattern is observable. Also, the distributions are similar as the majority of edges are located in the

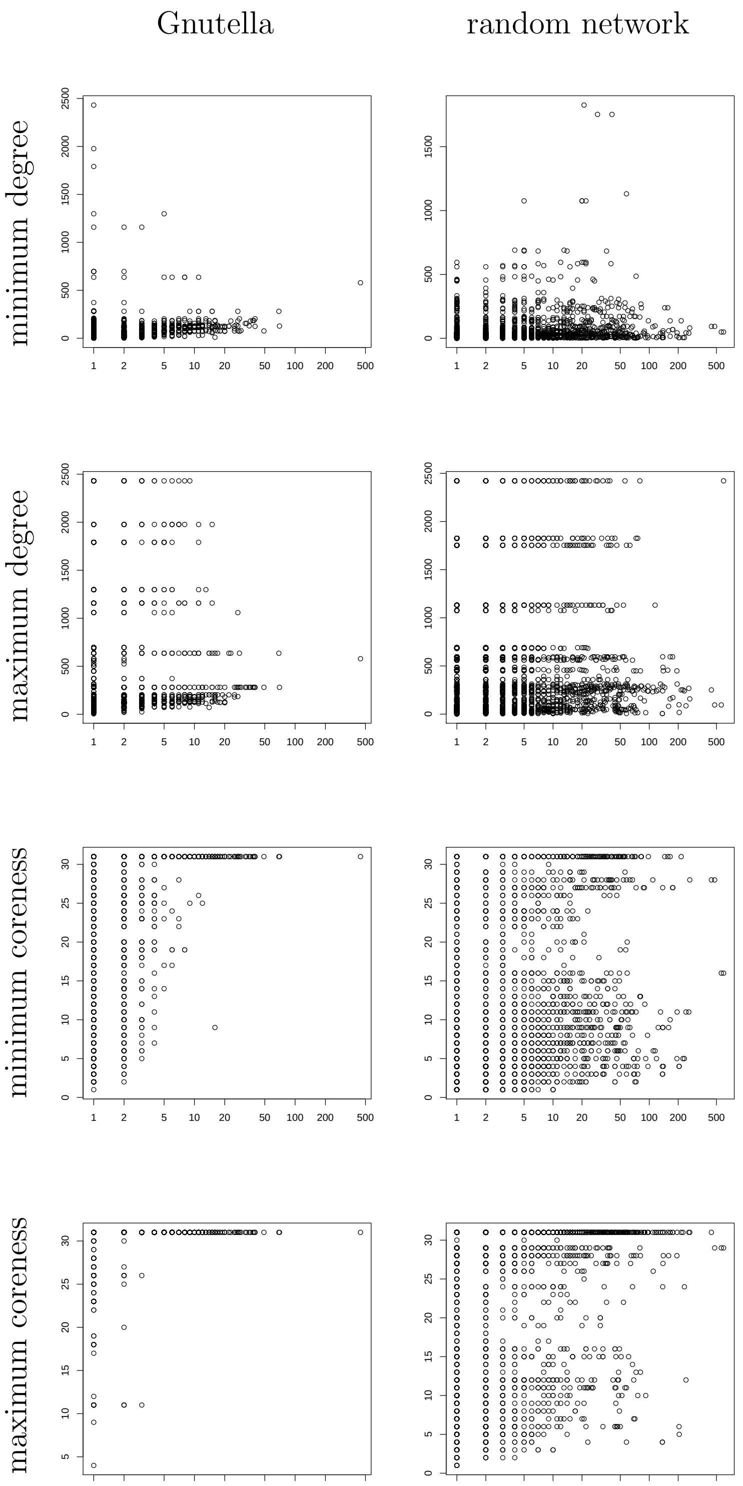


Figure 8.29.: Comparing appearance weight with minimum and maximum degree and coreness of the corresponding end-nodes in Gnutella and the random network. Each data point represents an edge, the x-axis denotes the appearance weight and the y-axis reflects the degrees (coreness) of the end-nodes. All axes use logarithmic scale.

periphery of the network where the maximum degree of the end-nodes is small. We thus hypothesize that the relation of load in the P2P network and node degree in the underlying network is the same in both the Gnutella and the random network.

However, the situation changes when considering the coreness instead of the degree. From Figure 8.29 (lower four), we can observe that although there is no correlation in any of the two networks, the distributions are different. In the random network the distributions are very uniform, which reflects its random nature. In the case of Gnutella almost no heavy edge is incident to a node with small coreness, as can be seen in the minimum-coreness diagram. Positively speaking, most edges with large weights are incident to nodes with large minimum coreness. Interpreting coreness as importance of an AS, these edges are located in the backbone of the Internet. The same diagram for the random network does not yield a similar significant distribution, thus denying a comparable interpretation. For instance, in the random network, there exist edges located in the periphery that are heavily loaded. As an aside, backbone edges need not necessarily be heavily loaded in either network.

These observations lead us to conclude that the Gnutella network differs from random networks and appears to be correlated to the underlay network. We are not the first researchers, who discovered that Gnutella differs from a random topology [90, 2]. However, we illustrated that the analysis of correlation between overlay and underlay can be significantly supported by applying a proper visualization technique to the underlay and study the layout for the induced overlay. Although, our technique may not replace a sound mathematical investigation, it can provide suitable visual indicators and properties to search for.

8.6. Summary

We presented an extensive case-study of applying clusterings techniques and visualization in the analysis of the AS Network: In the first part, we used the aspect of data reduction of clustering in order to obtain a filtering technique. We experimentally verified its compatibility with the special graph structure. Based on the removal of irrelevant information, we extracted baselines, long- and short-term patterns. In the second part, we focused on a proper visualization. We developed a new technique that incorporates the notion of importance defined by the previous filtering technique. The combination lead to analytic layouts that could be used to distinguish between the real networks and artifically generated networks having similar properties. The final part contained the analysis of the embedded application Gnutella. Although Gnutella operates as an independent service, it relies on the routing properties of the Internet. We introduced a novel technique to analyze such underlay-overlay relations. As a main result, we were able to show that the network of Gnutella and randomly generated networks share some properties, but are otherwise not (linearly) correlated. Furthermore, we observed that Gnutella and the underlying AS Network are sightly correlated.

Chapter 9

Landscape Like Visualization Techniques

In the previous chapter, we introduced a visualization technique for the AS Network (Section 8.4.1, Algorithm 7). The method does not explicitly utilize any special properties of the AS Network, such as the fact that it is the union of paths, and only relies on the fact that the core decomposition is a hierarchy (as defined in Section 7.2.1). In the following, we sketch and discuss our first approaches to extend the visualization technique. The goal is to obtain methods which can draw arbitrary arbitrary nested decompositions. We keep the focus on large networks.

9.1. Founding Paradigm

Visualizations of large networks, i. e., those with more than 10,000 nodes or edges, usually have a trade-off between the details of visually shown elements and the amount of represented information. For example, drawing all nodes and edges can lead to crowded layouts, while restricting to subgroups of nodes or edges is normally accompanied by loss of information. In contrast to previous attempts, our goal is to emphasize global structural dependencies with respect to a given nested decomposition, while representing all nodes and edges. More precisely, important parts of the decomposition and their mutual relations should be visually highlighted.

Commonly, layered drawings are handled by a Sugiyama approach, for an overview see [35, 9]. Such layouts are typically two-dimensional and the layer information is represented in the y–coordinate of the nodes. Optimization goals include crossing reduction and width, i. e., maximum difference of x–coordinates. Motivated by the size issues of large networks, a two-dimensional layout will probably not suffice. For example, in the AS Network more than 60% of the nodes have coreness two, thus more than 7,000 nodes would be placed on a single line implying a bad resolution. More over, in such a one and a half dimensional layout, certain details could not be hidden. Thus, we used a corresponding three-dimensional layout, where the z–coordinate reflects the layer index which corresponds in our scenario to the importance of nodes. Features such as perspective and the fact that the interior of three-dimensional objects are not visible now enable us the masking of irrelevant details. More precisely, a suitable placement in the x-y–plane can then be interpreted as a landscape of mountains, i. e., each group of high importance is a clearly visible peak and different peaks are separated by valleys. Note, that a large absolute value of importance of a node is not sufficient to be visible, since it may be covered inside a peak of higher importance in its neighborhood. On the other hand, in a valley, nodes of small absolute importance can be visible due to the fact that they connect different groups and thus are located on the surface. An example is given

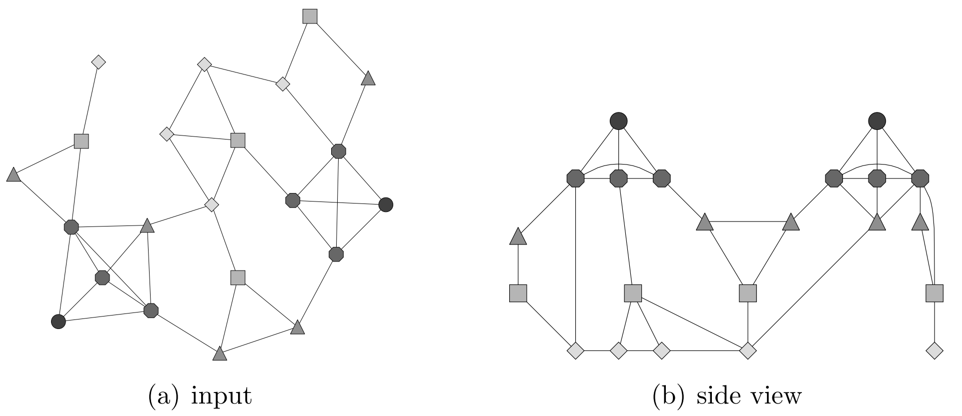

(a) input (b) side view

Figure 9.1.: Input network and a side view of the importance landscape.

in Figure 9.1; the color and the shape of the nodes correspond to the importance. The triangular nodes in the middle are clearly visible, while some nodes of higher importance (octagonal and circular) cover other triangular nodes in the right peak.

Such landscape-metaphor paradigms are frequently used for visualizing bibliographic networks [106, 25]. In general the landscape is produced simply by overlaying a triangulated grid, where grid points are elevated according to the density of data points in their vicinity. This can be seen as a histogram for two dimensional data. In our scenario, the height of the points/nodes is already fixed. The landscape is induced by the structural important elements, which will automatically conceal inferior parts. The layout models this effect by placing nodes and edges accordingly. Furthermore, it binds the landscape to the global shape of the network. In contrast to the original landscape metaphor, no explicit surface is added. If the edge set is sufficiently large, then it forms an implicit landscape surface as has been observed in the case of the AS Network.

9.2. Multi-Peak Layouts

As mentioned in Section 8.4.1, Algorithm 7 is not suitable to handle arbitrary nested decompositions. The problem are disconnected layers, i. e., layers in which a connected component has only neighbors in layers with smaller index. Since no anchor points are known for such components, they will be placed randomly. However, this can significantly influence the overall quality of the final layout. In the following, we discuss two potential approaches: The first one is a more or less straightforward extension that uses a pre-computation. In contrast, the second method recursively identifies and layouts peaks and their mutual dependencies. Both techniques produce proper visualizations and work as a proof of concept that the original (landscape) paradigm can be extended to handle arbitrary nested decompositions. A disadvantage is the loss of simplicity as both methods require a certain degree of parameter tuning.

9.2.1. Extension using Pre-Computation

As mentioned above, disconnected layers are problematic to handle, since no anchors are placed beforehand. Thus, it would be sufficient to know a rough estimate of the position of anchorless components. Recall the definition of the level-component view as defined in Section 7.2.1 and the corresponding abstracted graph (Section 2.1), then a suitable two and a half dimensional layout of this abstracted graph can be used as a schematic layout of the original graph. More precisely, we can use the calculated coordinates for the abstracted nodes as reference coordinates for initializing an anchorless component. The pseudo-code is given in Algorithm 8.

Algorithm 8: Layout algorithm for (general) nested decompositions

```
Input: graph $G = (V, E)$,
nested decomposition $\eta = (V =: V_0 \supsetneq V_1 \supsetneq \cdots \supsetneq V_k \supsetneq \emptyset)$
Output: 2.5D layout of $G$
calculate connected components in the individual layers, i. e., $V_i = \biguplus_{j=1}^{p_i} V_i^j$
calculate 2.5D layout $\mathcal{L}$ for level-component abstraction $G_\eta$
calculate initial layout for each $G(V_k^j)$ with $j = 1, \ldots, p_k$
        and place the components according to $\mathcal{L}$
for $\ell \leftarrow k - 1, \ldots, 0$ do
    for $j \leftarrow 1, \ldots, p_\ell$ do
        if $V_\ell^j$ is anchorless then
            calculate initial layout for each $G(V_\ell^j)$ and place it according to $\mathcal{L}$
        else
            calculate barycentric layout for $V_\ell^j \setminus V_{\ell+1}$ relative to $\mathcal{L}$ in $G$,
                keeping $G(V_{\ell+1})$ fixed
            calculate force-directed layout for $V_\ell^j \setminus V_{\ell+1}$ relative to $\mathcal{L}$ in $G$,
                keeping $G(V_{\ell+1})$ fixed
    calculate force-directed layout for $G(V_\ell)$
```

We evaluated our technique using the recommendation network of Amazon [5]. Amazon offers for each product a list of similar items that have been frequently purchased together. This list usually contains four items, and the recommendation property is asymmetric. We express this relation by weights on undirected edges. An edge between product A and B has weight two, if A recommends B and vice versa, and weight one, if only one recommendation exists between them.

We crawled several samples of this network in 2003. In the following, we discuss one of them, which started with the product 'VW Beetle Restoration Handbook' and has a depth of nine. The resulting network contains 468 products and 1702 recommendations. The part of the network that contains the initial node and its neighborhood is given in Figure 9.2. The boxes indicate the disconnected components in the core layers. The upper-most large box contains the initial product

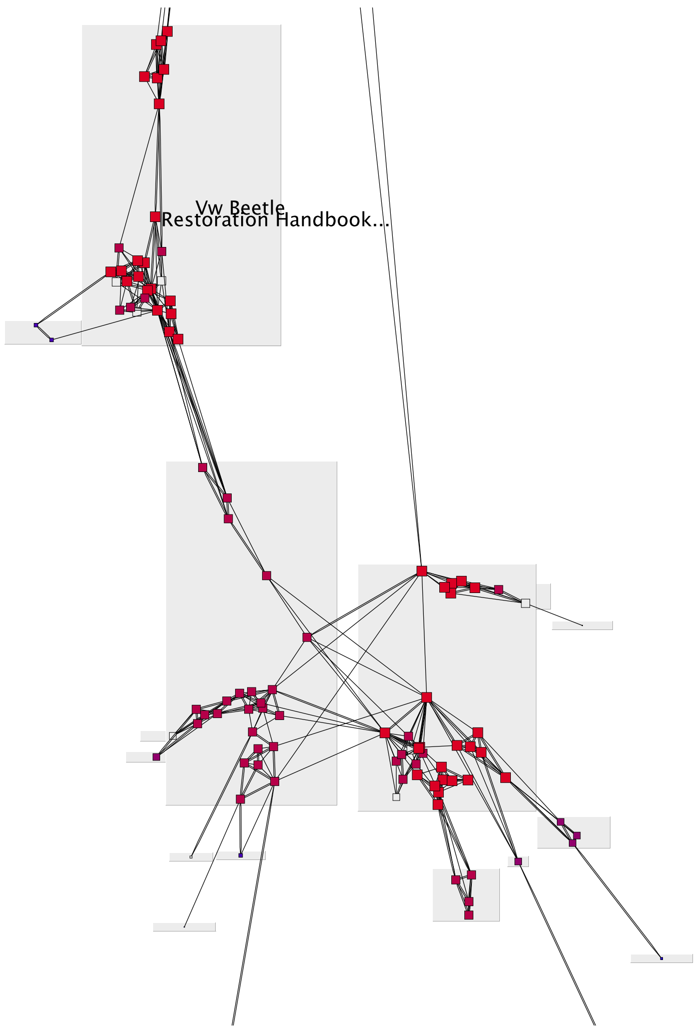
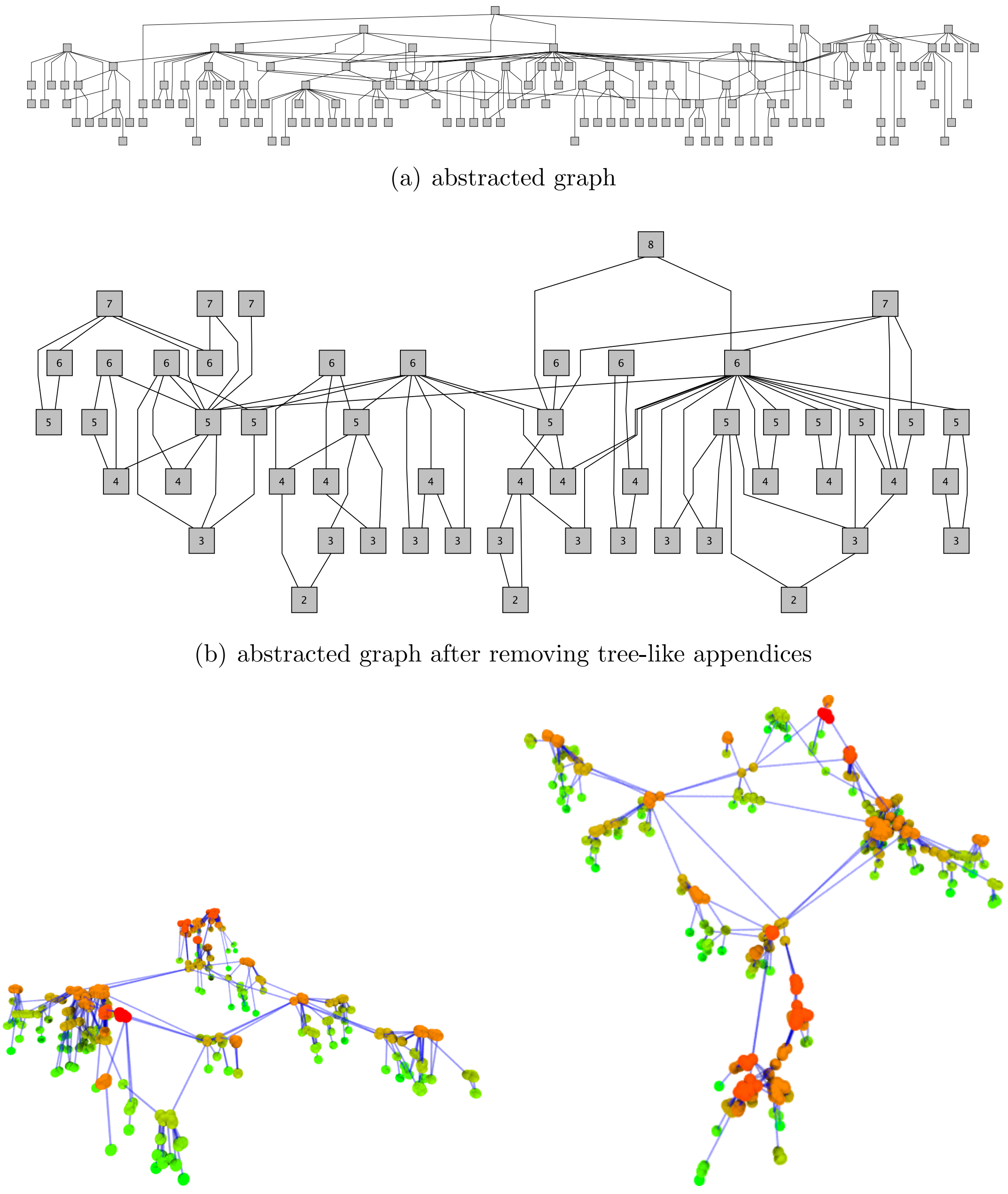


Figure 9.2.: Example of recommendation network of Amazon.com

together with other VW related books, the two other large components are about repairing and tuning vehicles, i. e., cars and motorcycles. Thus related topics about car and restorations are spatially close to the product 'VW Beetle Restoration Handbook'. The level-component abstraction of our sample (Figure 9.3(a)) indicates the existence of further components. More precisely, neither the initial node nor one of its related topics form the maximum core. The 2.5D layout is given in Figure 9.3(c) and 9.3(d) and shows that at least three other independent groups of products exist with topics: food, history, industrial welfare and novels. Indeed, the top layer is formed by the following novels: *Austerlitz*, *After Nature*, *On the Natural History of*


(a) abstracted graph

(b) abstracted graph after removing tree-like appendices

(c) side view

(d) top view

Figure 9.3.: Example of recommendation network of Amazon.com

Destruction, *The Emigrants*, *The Rings of Saturn* and *Vertigo*. One explanation for the unrelated top layer is the crawl depth of nine. However, even a more limited crawl could have lead to unrelated topics. Popular items, like novels or guidebooks, are commonly purchased with other products and thus induce recommendations. In the top layer of this network, the books are all written by the same author (W.G. Sebald) and are well-known.

In our layout (Figure 9.3(d)) the different topics are spatially separated and the components occupy areas corresponding to their size. Only nodes of low degree/coreness are detached from their group if they act as bridging elements between two topics. One example of such an element is 'High and Mighty: SUVs – The World's Most Dangerous Vehicles and How They Got That Way', which links the car topic to cyber-culture via a recommendation to 'Cybertypes: Races, Ethnicity, and Identity on the Internet'.

Summarizing, the presented extension (Algorithm 8) is able to produce suitable layouts, i. e., different peaks are clearly separated and their low-level interdependencies are highlighted. However, the size of the (level-component) abstraction is often correlated with the size of the input graph. Thus, this approach seems to work only for small graphs or graphs where the abstraction graph is small. In our sample, the original abstracted graph has 150 nodes and 180 edges. After removing tree-like appendices that do not influences the initial pre-computed layout, the remaining graph (Figure 9.3(b)) had only 53 nodes and 83 edges. Note that such a drastic reduction may not always be possible. Another issue is the local and global relaxation: On the one hand, repulsive forces are needed to ensure a certain distribution of the nodes and thus a high utilization of the available drawing area. On the other hand, the repulsive forces may not interfere with the pre-computed scheme of the level-component abstraction. Since the repulsive forces have such a big influence, it might be hard to find suitable settings without further knowledge. The given layout of our sample (Figure 9.3) still exhibits a high similarity with the layout of the abstraction, although the forces have been tuned. Next, we present an extension that further reduces the abstraction graph and defines a recursive approach.

9.2.2. Extension using Refactoring

The previous approach requires a suitable layout of the level-component abstraction. Thus the quality and computational time highly depends on the size of the abstraction graph and the quality of the pre-computed layout. Recall the above example. The removal of tree-like appendices drastically decreased the size of the abstraction. Furthermore, not all of the remaining nodes are required to estimate the relative positions of the peaks. More precisely, all nodes that can be uniquely associated to a single peak, can be omitted.

More formally, we interprete the level-component abstraction as a directed acyclic graph, where the source of an edge is always in the lower layer. Thus the top-element of every peak corresponds to a sink in the graph. Furthermore, a node is uniquely assigned to a peak, if every directed path starting at that node leads to the same peak. A node is a *connector*, if there are at leat two directed paths starting at that node and leading to different peaks. Consider the example given in Figure 9.4. The

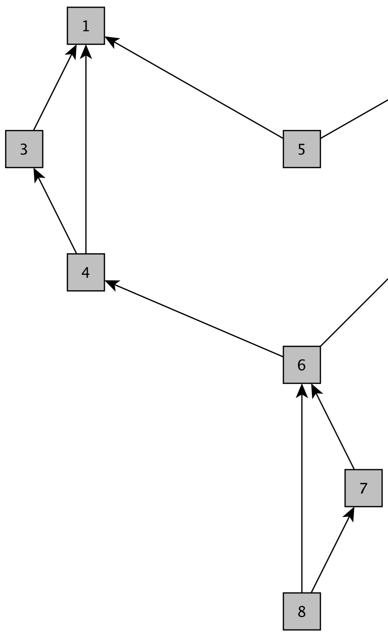


Figure 9.4.: An example of a (level-component) abstraction. The y–coordinate represents the layer level. The labels are arbitrary.

nodes 1 and 2 are the top-elements of the two peaks. Node 3 and 4 are uniquely assigned with peak 1, node 9 is uniquely attached to peak 2 and all other nodes are connectors. Note that node 7 and 8 inherit their status as connectors from node 6. Thus, in order to determine the relative positions of the peaks, only node 5 and 6 are necessary.

Definition 9.1. *An* essential connector *of the peaks u and v is a connector with the following property: there exists at least one path to one of the peaks u or v such that all internal nodes can only reach one of the nodes u or v.*

In the previously given example (Figure 9.4) only node 5 and 6 are essential connectors. The two nodes 7 and 8 are not, since all paths to one of the peak 1 or 2 contain node 6. Informally speaking, essential connectors form the hull of the connectors of two peaks. Note that this intuition is very imprecise and counter-intuitive situations may easily arise as given in Figure 9.5. Node 1, 2, and 3 are peaks and node 4 and 5 are essential connectors with respect to the pairs of peaks $(1, 2)$ and $(2, 3)$. In addition, node 6 is also an essential connector for the pair of peaks $(1, 3)$. This is simply due to the fact that neither node 4 nor 5 can reach both peaks 1 and 3.

We use the concept of essential connectors to further reduce the size of the level-component abstraction. The formal definition is given in Definition 9.2.

Definition 9.2. *Given a directed acyclic graph $G = (V, A)$. Let P be the set of all sinks (peaks) and C be the set of all essential connectors for every pair of*

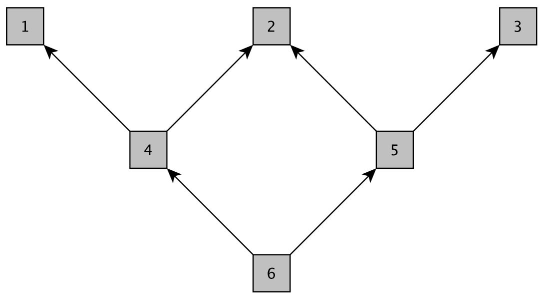


Figure 9.5.: An example with counter-intuitive essential connectors. The y–coordinate represents the layer level. The labels are arbitrary.

nodes contained in P. Then the essential peak graph *is defined as the undirected graph $H := (P \cup C, A')$, where two nodes are connected, if there is a directed path that connected the two nodes in G.*

A proper layout of such an essential peak graph provides relative positions of the peaks and an estimate of the essential connectors. In contrast to the previous approach, information about other connectors and uniquely assignable nodes is not available. Note that additional size constraints are needed to obtain a suitable layout in order to guarantee that peaks are well separated. Since such a layout contains sufficient information to roughly place anchorless components, we could directly use Algorithm 7 with the enriched information. On the other hand, sinks and essential connectors may be only the 'top' element of a more complex structure. Due to our initial goal of revealing structural dependencies, we further extend a layout for an essential peak graph using recursion and refactoring, respectively. Consider Figure 9.6 as an example. Nodes 1 and 2 are the top elements of peaks, nodes 3 and $5-8$ are uniquely assignable, nodes 4 and 9 are essential connectors, and all other nodes are connectors. Further, the essential connector 9 tops two disjoint parts (nodes $10, 12$ and $11, 13$, respectively), while the essential connector 4 has none. Similarly, both peaks have one node 8 and 5, respectively, that links them with essential connectors, while the peak associated with node 1 has a additional independent component (nodes $3, 6, 7$). Our basic idea is to refactor the graph into smaller pieces, namely the components that were topped by peaks and essential connectors. These components either form a hierarchy, i. e., they are uniquely assigned to their top element and are connected, or again have several disjoint components that need to be properly placed.

We propose the following procedure in order to obtain a proper layout for the essential peak graph and the level-component abstraction: estimate for each peak and essential connectors the space it requires. Such an estimate could be the square-root of the maximum numbers of nodes in a level and being associated with it. Use a force-directed layout to establish first relative coordinates such that peak nodes are not overlapping and peak nodes and essential connectors only touch each other, if they are related to each other. Due to the size- and overlapping-constraints, each node can be placed inside a cone that has its top at the peak and its basis at that

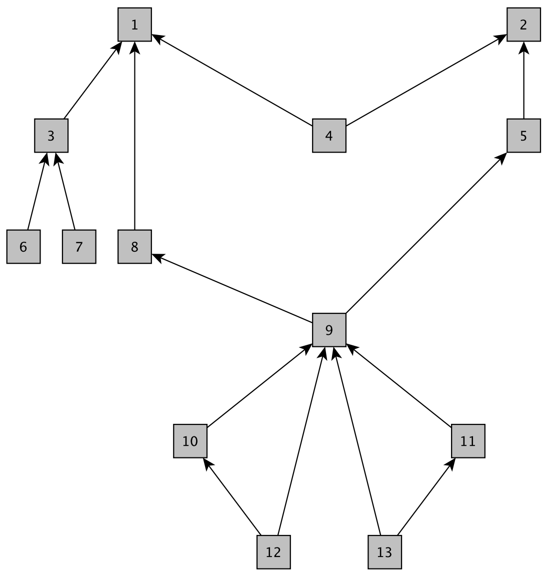


Figure 9.6.: An example of a level-component abstraction, where peaks and essential connectors 'hide' internal structural dependencies. The y–coordinate represents the layer level. The labels are arbitrary.

level which defined the size. If the component has more nodes in lower levels (as the one which defined the size), the cone is extended by a cylinder. The layout of the essential peak graph enriched with the geometric object is then expanded into a layout for the level-component abstract. More precisely, all nodes other than peaks and essential connectors can be uniquely assigned to one of the peaks or essential connectors. If the collection of nodes associated with one peak or essential connector is connected after removing the peak or essential connector, then the collection can be layouted using our original layouter for hierarchies. On the other hand, if the collection has multiple components, we identify local peaks and essential connectors and iterate this process until only hierarchies are found. In this way, our new extension just finds the 'building' hierarchies, layouts them, and merges the local layouts into a global one.

The decomposition of the example given in Figure 9.6 is shown in Figure 9.7. In contrast to the previous pre-computation-based approach, this expansion operates locally. On the other hand, this locality can introduce certain drawbacks such as the global layout potentially appearing too schematic. To our current experience and knowledge, the achieved esthetics and usability of the produced layouts heavily depend on the chosen parameters.

The final layout for the level-component abstraction for the 'VW Beetle Restoration Handbook' previously used is shown in Figure 9.8(a) and the final expanded layout for the whole network is given in Figure 9.8(b). As can be seen, the individual peaks are clearly visible in both visualizations. Furthermore, their occupied volumes are well separated and the interconnection realized by (essential) connectors are emphasized. On the negative side, the final layout has still many characteristics

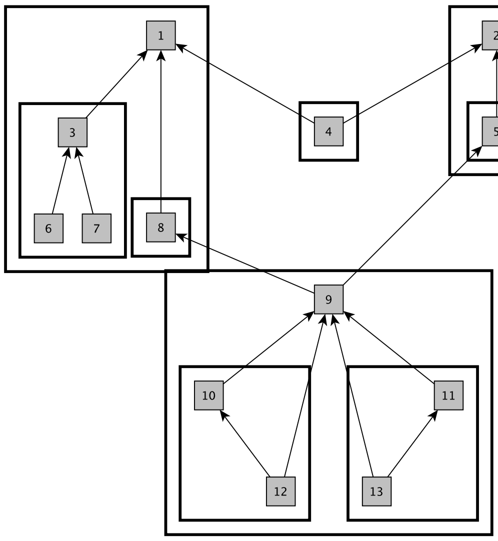
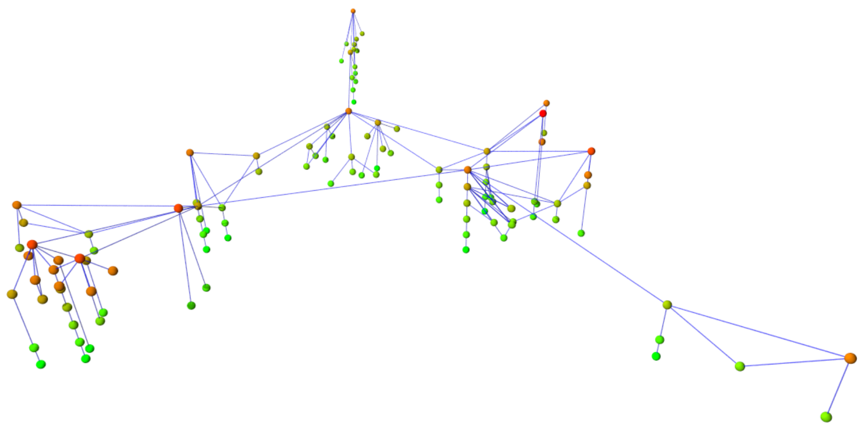


Figure 9.7.: Decomposition of the example given in Figure 9.6. The boxes show the different groups. The y–coordinate represents the layer level. The labels are arbitrary.

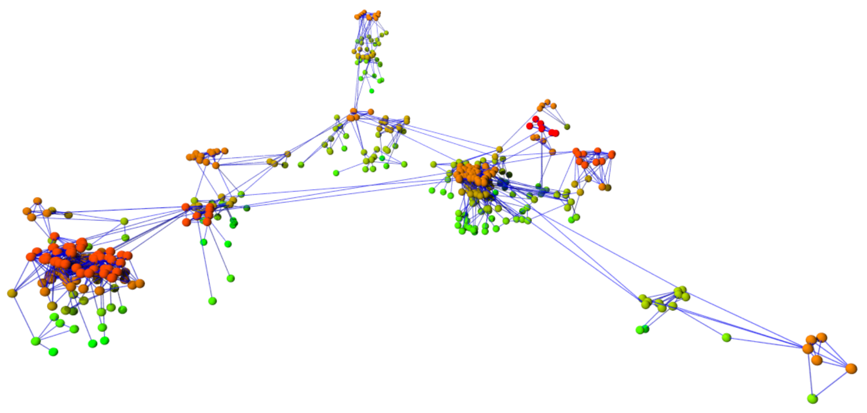
in common with the level-component abstraction. Although, this might improved the readability and reveal certain dependencies at a glance, it also decreases the recognizability of the interior of the peaks. In our example, this might be due to size issues, i. e., some nodes in the level-component abstraction represents lots of nodes in the original graph, while others corresponds to just a few, and the fact that our implementation of the hierarchical approach favors compacts peaks. Comparing the visualization with the pre-computational approach, the degree of abstraction in the final layout is smaller. More precisely, comparing Figure 9.3(d) and 9.8(b), the distribution of nodes associated with a peaks has improved. Another benefit of the refactoring method is its fast computation, we observed a speed-up of two to six on various instances.

9.3. Outlook

Summarizing, we presented two ideas on generalizing the visualization technique for the AS Network to arbitrary nested decompositions. Both proposed algorithms fully incorporated the original ideas and produced (almost) the same results for hierarchies. While the obtained results are satisfactory both with respect to esthetic criteria such as utilization of drawing area, and to analytic properties such as the readability of the decomposition, the techniques have some drawbacks: First, the version using a pre-computation can require a significant amount of time calculating the initial sketch due to the size of the level-component abstraction. Furthermore, the final layout can still look like an abstraction, i. e., nodes represented by the same



(a) final layout of the level-component abstraction



(b) final layout of the whole graph

Figure 9.8.: Layout of the level-component abstraction and the whole graph using the refactoring method applied to the 'VW Beetle Restoration Handbook' example. Colors code layers, i. e., red is top, while green corresponds to bottom layer.

node in the level-component graph can be distributed in a very small area. Although this artifact emphasizes the interdependencies of the components in the decomposition, the internal structure is obstructed due to the small resolution. Second, the technique built on refactorization is fairly sensitive to the methods placing the inner components and their expansion. Although the presented techniques have drawbacks and may require a fair amount of fine tuning of the parameters, the obtained results are a proof of concept, that the underlying idea is appropriate.

We conclude this part about landscape-like visualization techniques with a brief summary about further extensions. First of all, note that the founding paradigm does not rely on the fact, that the input (nested) decomposition is associated with the graph. More precisely, every pre-defined decomposition would suffice as well; the only important aspect is, that the visualization clarifies the given structure with respect to the network. Thus, the obtained layout can only be as meaningful as the decomposition is. Such pre-defined decompositions can be meta-information that is not expressed in the graph structure. Consider the case of collaboration networks, where nodes model authors and edges represent common publications. In order to study the mutual relations of a set S of author, one can consider the direct neighborhood of S in the collaboration network and define the decomposition accordingly to the numbers of connection to authors in the given input set S. More precisely, the set of authors form the top shell of the decomposition and let k be the index; the $(k-i)$th shell contains all those nodes having edges to $k-i+1$ nodes in the top shell[1]. Thus the different layers separate the coauthors of the authors in the given set with respect to their interaction with them. Note that such information is not available through the network structure itself. An example is given in Fig-

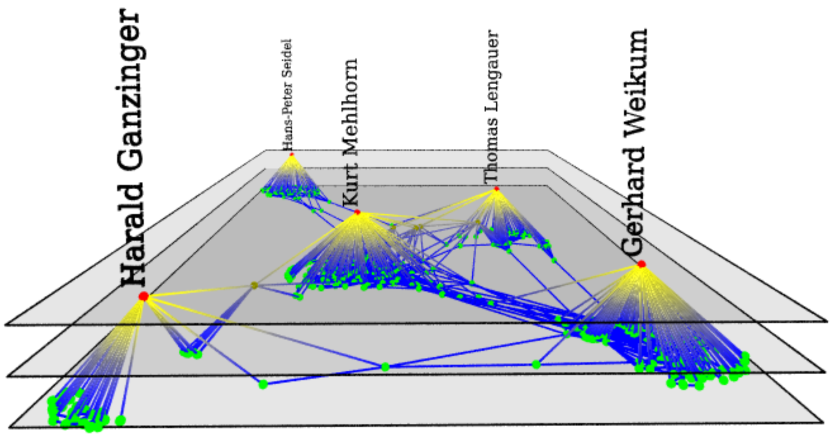
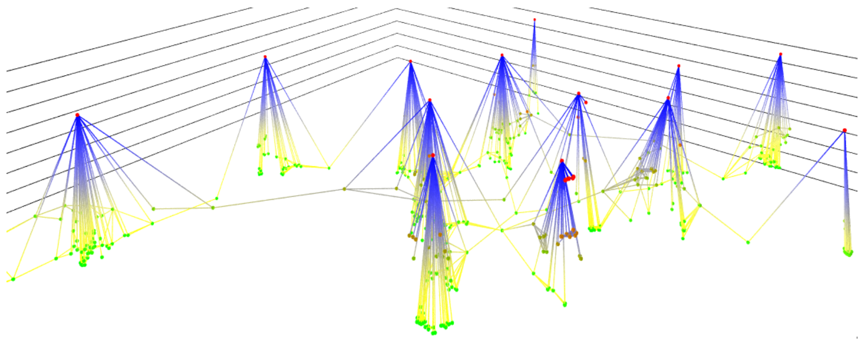


Figure 9.9.: The collaborations of the five heads of the MPII's departments in Saarbrücken. Data were collected in mid 2004.

ure 9.9, where the five heads (H. Ganzinger, T. Lengauer, K. Mehlhorn, H. Seidel, G. Weikum) of the departments of the Max-Planck-Institut für Informatik (MPII) Saarbrücken form the initial set. It is immediately recognizable that most of the people collaborated with only one head. Although the hierarchy is very flat, the layout is not crowded while the relative size of each neighborhood and their interaction is clearly obtainable. Another natural nested decomposition in this context is the *earliest collaboration decomposition*. Again, the initial node set forms the top layer. The depth of a node corresponds to the age of the earliest collaboration with a member of the initial set. This layout focuses on the evolution of recent collaborations, therefore the significant part of the network is formed by nodes placed in intermediate layers and nodes representing well established or former collaborations are situated in lower layers. An example is given in Figures 9.10 and 9.11, where the heads of the computer science department of the Universität Karlsruhe (TH) form the initial set. The time hierarchy spans a period of 35 years, i. e., between 1970 and 2004, and the network consists of 20 heads, 580 coauthors and over 1700 collaborations. Our method therefore exhibits that most of the collaboration within the department is well established and only few new interactions appeared recently.

[1]empty shells will be removed

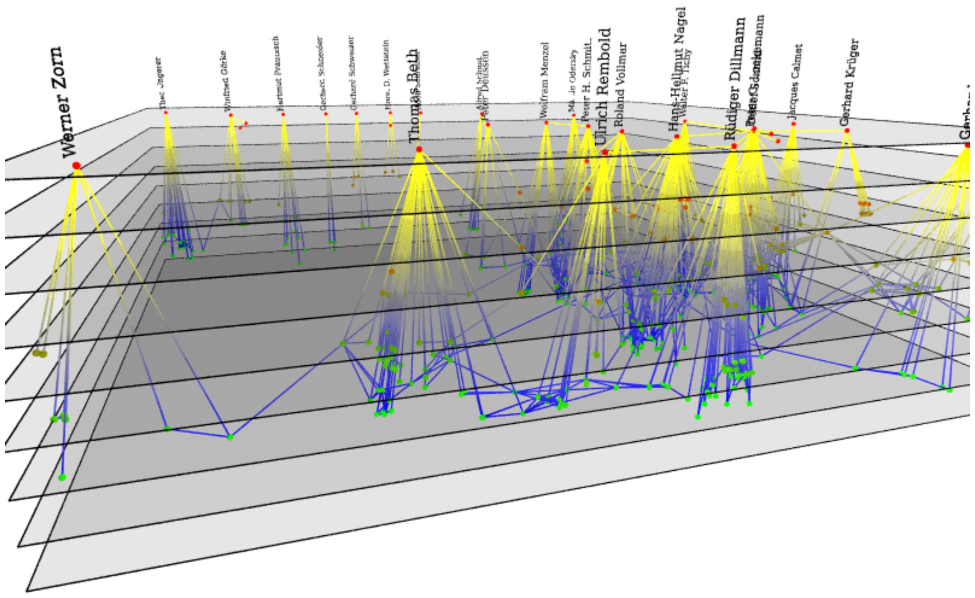


(a) 2004 (top view and clipped)

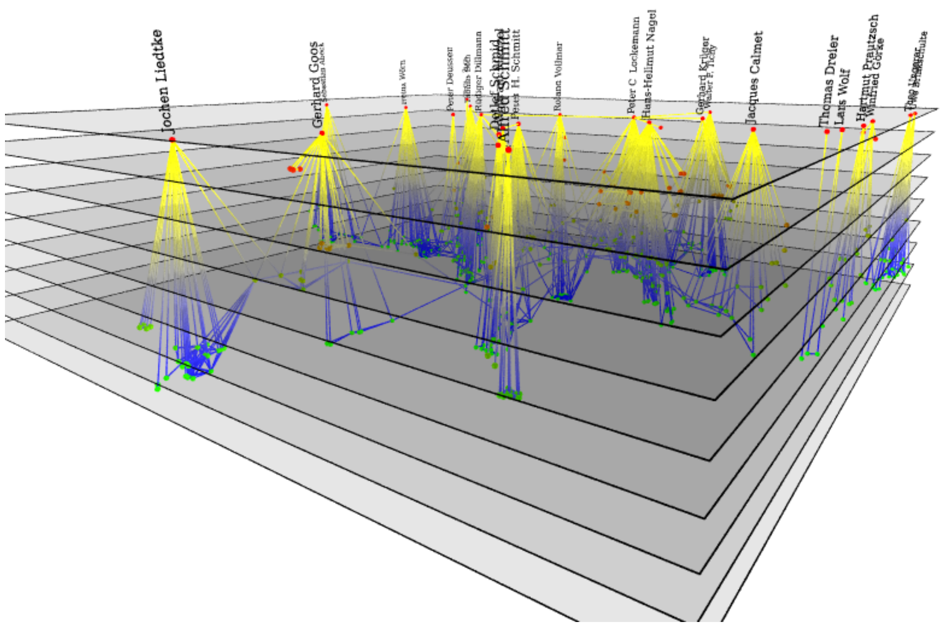
Figure 9.10.: Collaborations within a computer science department.

The second direction is the integration of temporal data. In the example of the earliest collaboration decomposition, the layers corresponded to different points in time. In [50], we introduced a similar paradigm for drawing evolving and dynamic graphs. The temporal evolution of the graph is usually given as a sequence of graphs, each representing a snapshot at a particular point in time. While the visualization of individual points in time helps to understand the current situation, a visualization of the whole sequence can reveal information about the evolution in general. So far most visual representations either use a static cumulative view of the sequence or a dynamic animation. By layouting the time-expanded graph (as defined in Section 7.1.3) where the decomposition is defined by time, both aspects are unified, i. e., a suitable layout for each snapshot is still available and, in addition, their combination shows the ongoing evolution in the network. Since time-expanded graphs tend to be fairly large, the resulting layouts may easily look crowded. An example is given in Figure 9.12 and shows the evolution in collaboration network over 15 years. The colors of nodes and edges code time, while the size represents the amount of (shared) publication in a particular year for edges and an accumulated count of publication for nodes. It can be observed, first, that nodes either remain constant in size or growth over time, and second, that those fat nodes roughly keep their positions. In addition, the top view reveals that most collaborations are repeated.

Concluding, our paradigm of landscape-like visualization offers an interesting perspective for analytically layouting graphs. More precisely, the given decomposition, either defined by the graph structure or given by the user, is emphasized and dependencies between individual layers and components are revealed. Although, the current implementation has certain drawbacks, it illustrated the usefulness.

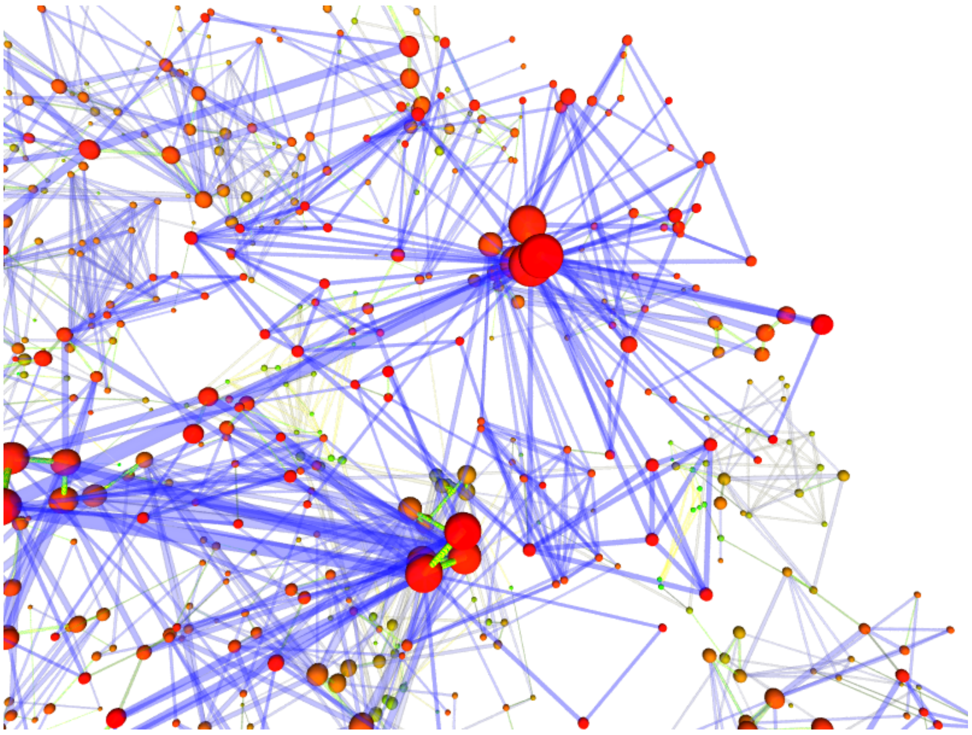


(a) 1994 (side view)

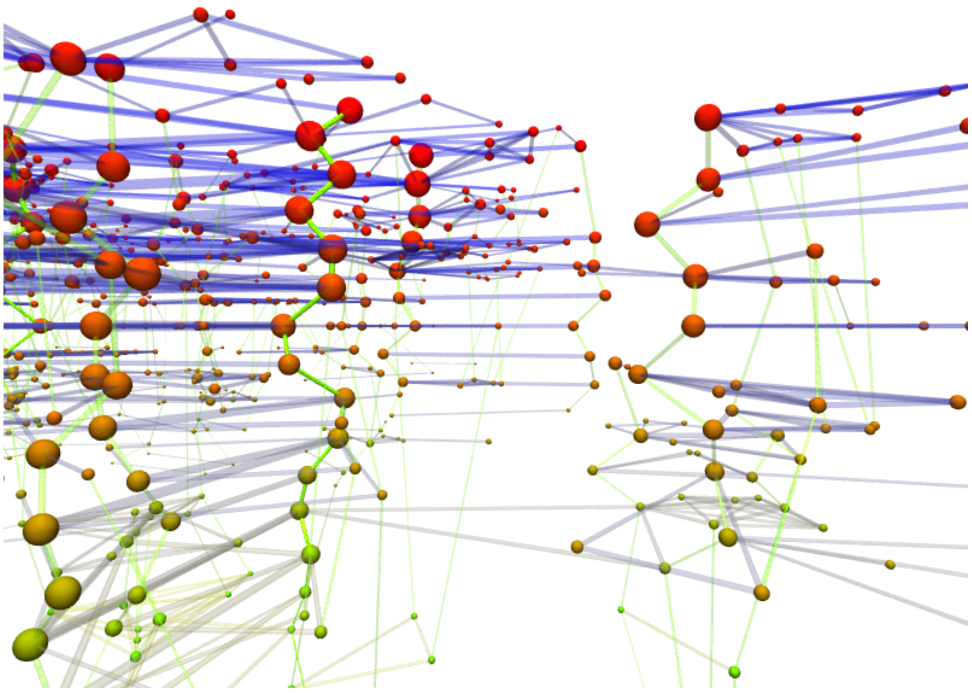


(b) 1999 (side view)

Figure 9.11.: Collaborations within a computer science department (continued).



(a) top view



(b) side view

Figure 9.12.: Evolution in a collaboration network over 15 years (clipped). Color codes the time, i. e., red nodes are in the newest time frame and green in the oldest, while blue edges connect red nodes and yellow edges the green ones. Green edges connect different time-copies of the same node and edges

Chapter 10

Conclusion

We conclude this thesis by summarizing the achieved results and give some remarks about open problems.

10.1. Theory and Experimental Evaluation

In the first part, we focused on various algorithmic aspects of density-based clusterings techniques. More precisely, we introduced several methods for measuring the quality of clusterings, comparing different clusterings with each other, generating graphs with a known, significant clustering, and finding clusterings.

Beside the presentation of a general framework for expressing quality measures based on the paradigm of intra-cluster density versus inter-cluster sparsity, we studied five measures in detail. These were coverage, performance, intra-cluster conductance, inter-cluster conductance, and modularity. We gave a characterization of clusterings having maximum score with respect to these indices in general and for some special graph families, respectively. In addition, we pointed out artifacts and counter-intuitive behavior. A brief summary of measures for comparing two clusterings concluded the part about quality and structural measures for clusterings. For the comparison, we presented lattice-based approaches that have been established in literature as well as some extensions that generalize the underlying ideas in order to incorporate the graph structure.

A discussion about general concepts for finding clusterings preluded the main topic of this first part, namely the design and evaluation of clustering techniques. First, fundamental concepts and potential realizations were presented in a uniform way. We focused on well-known concepts such as greedy and local searches. Second, four implementations were introduced that were the main target in the following experimental evaluation. Since our evaluation was based on artifically generated data, we introduced several concepts for generating graphs with known, significant clustering. Our experiments were composed of small unit tests which are comparable to those from software engineering. These tests formalize basic ideas and intuitions of clustering techniques. In this way, we could easily reduce biases and incorporate application-specific requirements. In the experimental study, we verified the usability of our presented techniques for graphs containing a highly significant clustering and highlighted artifacts for very sparse and dense graphs.

The first part was concluded with an overview of variations of clusterings. We modeled two problems concerning dynamic graphs. The first one was the update problem, where one wishes to maintain features of a clustering when the underlying graph changes. The second problem was the clustering of a sequence of graphs in order to identify stable groups over time. Furthermore, some structural extensions were stated such as fuzzy or hierarchical clustering.

Open Questions

Beside the obvious hunt for better measures and algorithms, there are two general and opposing directions: first, specified clustering and, second, an axiomatic approach.

As it is very unlikely that a “perfect” and general clustering technique will ever exist, a feasible solution is to specify a certain set of properties or features as input for the clustering technique which would then self-adapt to the selection. For example, in our experimental evaluation we considered only graphs having a significant clustering where the cluster sizes are distributed homogeneously. Our results regarding the usability need not carry over to graphs where the distribution of cluster sizes is very skew. Partially, such an approach has been considered before, for example, when the number of clusters or bounds on the number of nodes inside a cluster were given. However, these approaches failed when the “intuitive” or desired clustering did not meet these constraints. A possibility to avoid such a pitfall is to consider the additional information as advice or soft-constraints which should be met, but a violation may be tolerated. In this way, partition-based clustering could be able to handle networks with hierarchical structure by allowing dense connections between some clusters.

The opposite way to prespecify desired features is an axiomatic approach, where one defines a set of properties every clustering technique has to fulfill. Kleinberg presented in [72] that a selection of three intuitive axioms already denies the existence of corresponding clustering algorithms. On the other hand, Single Linkage, which is a greedy agglomerative approach, with different stopping criteria meet every selection of two axioms and a basic relaxation of the original axioms as well. Although such an approach sounds promising, it would be fairly difficult to select proper axioms that lead to useful clustering techniques and, up to now, nobody proposed a promising approach. Nevertheless it is an interesting idea and would certainly remove a lot of the vagueness associated with clustering, that is still present.

10.2. Applications in Network Analysis and Visualization

The second part contained an extensive case study of the network of the Autonomous Systems which is an abstract view of the physical Internet. The study was structured into three sections: First, we performed a conditioning of the collected data. More precisely, we defined and evaluated a filtering technique to remove structurally unimportant elements. In addition, as the network is evolving over time, we gave a preliminary analysis of long- and short-term features such as baselines, predicting trends, and analyzing anomalies. After this preparation of the data and our deepened knowledge of the network, we designed, implemented, and evaluated a visualization technique that emphasizes the intrinsic structure of the network in the second section. This layout approach combines the gained insight of the filtering technique with standard concepts from graph drawing. In the evaluation, we were able to show that a standard generator that should produce networks having a similar topology to the AS Network failed to match essential features. Albeit this structural incompatibility, the resulting visualizations of generated networks and the original one had the same global appearance, which indicated the applicability of the generator to some extent. In the final section, we focused on the application Gnutella, which is a Peer-2-Peer file sharing application operating on top of the Internet. Since it utilizes the ability of the Internet to route traffic, the analysis of its performance should take the Internet into consideration. We presented a basic method founded on projections and visualization to study such overlay-underlay relations. As a result, we verified that the topology of Gnutella is not random and found a certain kind of correlation with the topology of the Internet (at the AS level).

In the final section of this part, we discussed several possibilities to extend our visualization technique to arbitrary networks and arbitrary hierarchical decompositions. We briefly evaluated these extension using various (real-world) networks such as recommendation systems for books and collaboration data. The part was concluded with an outlook how to handle dynamic networks using a highly similar concept.

Open Questions

As we improved our understanding of the AS Network by applying various clustering techniques, for some issues we just scratched the surface. For example in the preliminary analysis of dynamic features, we observed stability in terms of static measurements such as the number of nodes, edges, and peers and in a microscopic part of the network, namely its core. In this context, it would be interesting to extend this analysis and to include dynamic clustering in order to identify stable groups over time in the major part of the network. Also, the study of long- and short-term phenomena could benefit from a broader view of the network. For example, how does the connectivity of a small customer change during a global virus spread that causes tremendous load in the Internet or how does general competition affect a customer's choice of his/her providers.

On the other hand and independent of application-specific context of ASes, there are two interesting problems that originate from the presented visualization technique. First, the extension of the landscape-like visualization techniques remains open. Our presented approaches are fairly promising, yet they are sometimes sensitive with respect to choices of the parameters. The second problem is the development of a suitable visualization technique for the analysis of overlay-underlay relations. In our case study, we profited from a proper interpretation of the positions of nodes. Furthermore, it would be beneficial to include knowledge of the structure of the (induced) overlay in the computation of the layout. In graph drawing, the problem is known as *simultaneous embeddings* and so far has only been considered for sparse graphs such as trees or planar graphs, see [43] for an example.

Bibliography

[1] A timeline of the 2003 blackout [online]. Available from World Wide Web: `http://www.cnn.com/2003/US/08/16/blackout.chron.ap/index.html`. 138

[2] Vinay Aggarwal, Stefan Bender, Anja Feldmann, and Arne Wichmann. Methodology for Estimating Network Distances of Gnutella Neighbors. In *GI Jahrestagung*, pages 219–223, 2004. 153, 159

[3] Vinay Aggarwal, Anja Feldmann, Marco Gaertler, Robert Görke, and Dorothea Wagner. Analysis of Overlay-Underlay Topology Correlation using Visualization. Technical Report 2005-31, ITI Wagner, Faculty of Informatics, Universität Karlsruhe (TH), December 2005. 12, 125, 155

[4] Vinay Aggarwal, Anja Feldmann, Marco Gaertler, Robert Görke, and Dorothea Wagner. Analysis of Overlay-Underlay Topology Correlation using Visualization. In *Proceedings of the 5th IADIS International Conference WWW/Internet Geometry*, 2006. 12, 125, 155

[5] Amazon.com. Available from World Wide Web: `http://www.amazon.com`. 9, 122, 163

[6] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander Flows, Geometric Embeddings and Graph Partitioning. In *Proceedings of the 36th Annual ACM Symposium on the Theory of Computing (STOC'04)*, pages 222–231. ACM Press, 2004. 24

[7] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, and Alberto Marchetti-Spaccamela. *Complexity and Approximation - Combinatorial Optimization Problems and Their Approximability Properties.* Springer-Verlag, 2nd edition, 2002. 24

[8] A. Baraldi, P. Blonda, F. Parmiggiani, and G. Satalino. Image Segmentation through Contextual Clustering. Technical Report TR-98-009, Berkeley, CA, 1998. Available from World Wide Web: `http://citeseer.ist.psu.edu/baraldi98image.html`. 122

[9] Oliver Bastert and Christian Matuszewski. Layered Drawings of Digraphs. In Kaufmann and Wagner [71], pages 87–120. Available from World Wide Web: `http://www.springerlink.com/content/xl91uu3fwk0fhejj/`. 161

[10] Vladimir Batagelj and Matjaž Zaveršnik. An $\mathcal{O}(m)$ Algorithm for Cores Decomposition of Networks. Technical Report 798, IMFM Ljublana, Ljubljana, 2002. 117

[11] Vladimir Batagelj and Matjaž Zaveršnik. Generalized Cores. Preprint 799, IMFM Ljublana, Ljubljana, 2002. Available from World Wide Web: `http://vlado.fmf.uni-lj.si/pub/networks/doc/`. 117

[12] Giuseppe Di Battista, Thomas Erlebach, Alexander Hall, Maurizio Patrignani, Maurizio Pizzonia, and Thomas Schank. Computing the Types of the Relationships between Autonomous Systems, 2007. Available from World Wide Web: `http://www.cs.le.ac.uk/people/te17/papers/ton-tor.html`. accepted for publication and scheduled to appear in the August 2007 issue. 128

[13] Michael Baur, Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. Drawing the AS Graph in Two and a Half Dimensions. Technical Report 2004-12, ITI Wagner, Faculty of Informatics, Universität Karlsruhe (TH), June 2004. 12

[14] Michael Baur, Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. Drawing the AS Graph in 2.5 Dimensions. In GD'04 [53], pages 43–48. Available from World Wide Web: `http://springerlink.metapress.com/link.asp?id=xfqay6gkcychr4hf`. 12, 125

[15] Code Red II and Nimda Worms and BGP Instability [online]. Available from World Wide Web: `http://www.renesys.com/projects/bgp_instability`. 135

[16] Ulrik Brandes, , and Falk Schreiber. Visual Understanding of Metabolic Pathways Across Organisms Using Layout in Two and a Half Dimensions. *Journal of Integrative Bioinformatics*, 002, 2004. 140

[17] Ulrik Brandes. Drawing on Physical Analogies. In Kaufmann and Wagner [71], pages 71–86. Available from World Wide Web: `http://www.springerlink.com/content/8w9xeajyn1ql4kxq/`. 139, 146

[18] Ulrik Brandes and Steven R. Corman. Visual Unrolling of Network Evolution and the Analysis of Dynamic Discourse. *Journal of Information Visualization*, 2(1), 2003. 140

[19] Ulrik Brandes and Sabine Cornelsen. Visual Ranking of Link Structures. *Journal of Graph Algorithms and Applications*, 7(2):181–201, 2003. 144

[20] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Höfer, Zoran Nikoloski, and Dorothea Wagner. Maximizing Modularity is hard. arXiv physics/0608255, 2006. 12, 34

[21] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Höfer, Zoran Nikoloski, and Dorothea Wagner. On Modularity - NP-Completeness and Beyond. Technical Report 2006-19, ITI Wagner, Faculty of Informatics, Universität Karlsruhe (TH), 2006. 12, 34

[22] Ulrik Brandes and Thomas Erlebach, editors. *Network Analysis: Methodological Foundations*, volume 3418 of *Lecture Notes in Computer Science*. Springer-Verlag, February 2005. Available from World Wide Web: `http://springerlink.metapress.com/openurl.asp?genre=issue&issn=0302-9743&volume=3418`. 15, 185

[23] Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. Engineering Graph Clustering: Models and Experimental Evaluation. *ACM Journal of Experimental Algorithmics*. accepted for publication. 12, 83

[24] Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. Experiments on Graph Clustering Algorithms. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03)*, volume 2832 of *Lecture Notes in Computer Science*, pages 568–579, September 2003. Available from World Wide Web: `http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=2832&spage=568`. 12, 25, 67, 77, 83, 90, 112

[25] M. Chalmers. Using a landscape metaphor to represent a corpus of documents. In *Proceedings of the European Conference on Spatial Information Theory (COSIT'93)*, volume 716, pages 377–390. Springer-Verlag, 1993. 162

[26] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P Systems Scalable. In *ACM SIGCOMM*, 2003. 153

[27] Fan R. K. Chung. *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997. 63, 69, 143

[28] Aaron Clauset, Mark E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(066111), 2004. Available from World Wide Web: `http://link.aps.org/abstract/PRE/v70/e066111`. 32, 70

[29] The Code Red Worm [online]. Available from World Wide Web: `http://www.ciac.org/ciac/bulletins/l-117.shtml`. 135

[30] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001. 121

[31] Daniel Delling. Analyse und Evaluierung von Vergleichsmaßen für Graphclusterungen. Diplomarbeit, Institut für Theoretische Informatik - Universität Karlsruhe (TH), February 2006. 45, 46, 50, 51, 96

[32] Daniel Delling, Marco Gaertler, Robert Görke, Zoran Nikoloski, and Dorothea Wagner. How to Evaluate Clustering Techniques. Technical Report 2006-24, ITI Wagner, Faculty of Informatics, Universität Karlsruhe (TH), 2006. 12, 83

[33] Daniel Delling, Marco Gaertler, Robert Görke, and Dorothea Wagner. Experiments on Comparing Graph Clusterings. Technical Report 2006-16, ITI Wagner, Faculty of Informatics, Universität Karlsruhe (TH), 2006. 12, 46, 51, 83, 96, 111

[34] Daniel Delling, Marco Gaertler, and Dorothea Wagner. Generating Significant Graph Clusterings. In ECCS'06 [39]. 12, 83

[35] Guiseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing - Algorithms for the Visualization of Graphs.* Prentice Hall, 1998. 146, 161

[36] DIMES – Distributed Internet MEasurements & Simulations. Available from World Wide Web: `http://www.netdimes.org/`. 126

[37] Zvi Drezner and Horst W. Hamacher, editors. *Facility Location: Application and Theory.* Springer-Verlag, 2002. 115

[38] Peter Eades and Qing-Wen Feng. Multilevel Visualization of Clustered Graphs. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*, volume 1190 of *Lecture Notes in Computer Science*, pages 101–112. Springer-Verlag, January 1997. 140

[39] *Proceedings of the European Conference of Complex Systems ECCS*, September 2006. 184, 185

[40] David Eppstein. Fast Hierarchical Clustering and Other Applications of Dynamic Closest Pairs. *ACM Journal of Experimental Algorithmics*, 5:1–23, 2000. 55, 58

[41] Paul Erdős and Alfred Rényi. On Random Graphs I. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959. 77, 125

[42] Thomas Erlebach, Alexander Hall, Michael Hoffmann, and Matúš Mihal'ák. Network Discovery and Verification with Distance Queries. In *Proceedings of the 6th Conference on Algorithms and Complexity (CIAC)*, volume 3998 of *Lecture Notes in Computer Science*, pages 69–80. Springer-Verlag, 2006. 126

[43] Cesim Erten and Stephen G. Kobourov. Simultaneous Embedding of Planar Graphs with Few Bends. In GD'04 [53], pages 195–205. 180

[44] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the Internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 251–262. ACM Press, 1999. 125

[45] Thomas M. J. Fruchterman and Edward M. Reingold. Graph Drawing by Force-Directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991. 144

[46] Marco Gaertler. Clustering with Spectral Methods. Diplomarbeit, Fachbereich Informatik und Informationswissenschaft, Universität Konstanz, March 2002. 67

[47] Marco Gaertler. Clustering. In Brandes and Erlebach [22], pages 178–215. Available from World Wide Web: `http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=3418&spage=178`. 12, 16, 19, 53, 112

[48] Marco Gaertler, Robert Görke, Dorothea Wagner, and Silke Wagner. How to Cluster Evolving Graphs. In ECCS'06 [39]. 12, 109

[49] Marco Gaertler and Maurizio Patrignani. Dynamic Analysis of the Autonomous System Graph. In *IPS 2004 – Inter-Domain Performance and Simulation*, pages 13–24, March 2004. Available from World Wide Web: `http://w3.tmit.bme.hu/ips2004/index.html`. 12, 125, 132, 157

[50] Marco Gaertler and Dorothea Wagner. A Hybrid Model for Drawing Dynamic and Evolving Graphs. In *Proceedings of the 13th International Symposium on Graph Drawing (GD'05)*, volume 3843 of *Lecture Notes in Computer Science*, pages 189–200. Springer-Verlag, January 2006. Available from World Wide Web: `http://www.springerlink.com/content/711061543691865v/`. 12, 173

[51] Lixin Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, 2001. 128

[52] Michael R. Garey and David S. Johnson. *Computers and Intractability. A Guide to the Theory of $\mathcal{NP}$-Completeness.* W. H. Freeman and Company, 1979. 35

[53] *Proceedings of the 12th International Symposium on Graph Drawing (GD'04)*, volume 3383 of *Lecture Notes in Computer Science*. Springer-Verlag, January 2005. 182, 184

[54] Gnutella Protocol Specification v0.4. Available from World Wide Web: `http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf/`. 155

[55] Gnutella Protocol Specification v0.6 (RFC), 2002. Available from World Wide Web: `http://www.the-gdf.org/`. 153, 155

[56] Chris Godsil and Gordon Royle. *Algebraic Graph Theory.* Graduate Texts in Mathematics. Springer-Verlag, 2001. 143

[57] GTK-Gnutella. Available from World Wide Web: `http://gtk-gnutella.sourceforge.net/`. 155

[58] Erez Hartuv and Ron Shamir. A Clustering Algorithm based on Graph Connectivity. *Information Processing Letters*, 76(4-6):175–181, 2000. Available from World Wide Web: `http://citeseer.nj.nec.com/hartuv99clustering.html`. 22, 63

[59] Carrier-Krach: Level 3 peert vorläufig wieder mit Cogent. Available from World Wide Web: `http://www.heise.de/newsticker/meldung/64759`. [Online; accessed 19-July-2006]. 128

[60] Jeder vierte deutsche Haushalt ist mit DSL versorgt. Available from World Wide Web: `http://www.heise.de/newsticker/meldung/65827`. [Online; accessed 09-November-2005]. 9

[61] Martin Holzer, Frank Schulz, and Dorothea Wagner. Engineering Multi-Level Overlay Graphs for Shortest-Path Queries. In *Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX 2006)*, volume 129 of *Proceedings in Applied Mathematics*, pages 156–170. SIAM, January 2006. 126

[62] Martin Holzer, Frank Schulz, and Thomas Willhalm. Combining Speed-up Techniques for Shortest-Path Computations. In *Proceedings of the Third International Workshop on Experimental and Efficient Algorithms (WEA 2004)*, volume 3059 of *LNCS*, pages 269–284. Springer, May 2004. 126

[63] Kenneth E. Iverson. *A Programming Language.* Wiley, 1962. 27

[64] Jahr der informatik. Available from World Wide Web: `http://www.bundesregierung.de/-,413.942628/artikel/2006-ist-das-Jahr-der-Informat.htm`. [Online; accessed 16-February-2006]. 9

[65] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988. 15, 62, 121

[66] Anil K. Jain, M. N. Murty, and Patrick J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999. 60, 121

[67] Cheng Jin, Qian Chen, and Sugih Jamin. Inet Topology Generator. Technical Report CSE-TR-433, EECS Department, University of Michigan, 2000. 146

[68] T. Karagiannis, A. Broido, N. Brownlee, kc claffy, and M. Faloutsos. Is P2P dying or just hiding? In *Globecom'04*, 2004. 153

[69] George Karypis and Vipin Kumar. Analysis of Multilevel Graph Partitioning. In *Proceedings of the 1995 ACM/IEEE conference on Supercomputing.* ACM Press, 1995. 64

[70] George Karypis and Vipin Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998. Available from World Wide Web: `http://epubs.siam.org/sam-bin/dbq/article/28799`. 64

[71] Michael Kaufmann and Dorothea Wagner, editors. *Drawing Graphs: Methods and Models*, volume 2025 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001. Available from World Wide Web: `http://www.springerlink.com/content/xkru1gvnyh5p/`. 181, 182

[72] Jon M. Kleinberg. An Impossibility Theorem for Clustering. In *Proceedings of 15th Conference: Neiral Information Processing Systems, Advances in Neural Informatio*, 2002. Available from World Wide Web: `http://www.cs.cornell.edu/home/kleinber/kleinber.html`. 178

[73] Donald E. Knuth. Two notes on notation. *American Mathematical Monthly*, 99:403–422, 1990. 27

[74] KPNQwest limps back after shutdown [online]. Available from World Wide Web: `http://news.com.com/2104-1033_3-946262.html`. 136

[75] W. Lai, Peter Eades, K. Misue, and K. Sugiyama. Preserving the Mental Map of a Diagramm. pages 24–33. 111

[76] Y. Liu, H. Zhang, W. Gong, and D. Towsley. On the Interaction between Overlay Routing and Traffic Engineering. In *IEEE INFOCOM'05*, 2005. 153

[77] Georg L. Nemhause and Laurence A. Wolesy. *Integer and Combinatorial Optimization*. Wiley, 1988. 115

[78] Mark E. J. Newman. Fast Algorithm for Detecting Community Structure in Networks. *Physical Review E*, 69(066133), 2004. 64

[79] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(026113), 2004. Available from World Wide Web: `http://link.aps.org/abstract/PRE/v69/e026113`. 70, 112

[80] The internet outage and attacks of october 2002 [online]. Available from World Wide Web: `http://www.isoc-chicago.org/internetoutage.pdf`. 136

[81] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically aware overlay construction and server selection. In *IEEE INFOCOM'02*, 2002. 153

[82] Guidelines for creation, selection, and registration of an Autonomous System (AS), 1996. Available from World Wide Web: `http://tools.ietf.org/html?rfc=1930`. [Online; accessed 13-July-2006]. 126

[83] A Border Gateway Protocol 4 (BGP-4), 2006. Available from World Wide Web: `http://tools.ietf.org/html?rfc=4271`. [Online; accessed 23-October-2006]. 126

[84] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella Network: Properties of Large-Scale P2P Systems and Implications for System Design. In *IEEE Internet Computing Journal*, 2002. 155

[85] University of Oregon Routeviews Project. Available from World Wide Web: `http://www.routeviews.org/`. `http://www.routeviews.org/`. 126, 156

[86] Peter Sanders and Dominik Schultes. Highway Hierarchies Hasten Exact Shortest Path Queries. In *Proc. European Symposium on Algorithms*, volume 3669 of *LNCS*, pages 568–579. Springer, 2005. 126

[87] Peter Sanders and Dominik Schultes. Engineering Highway Hierarchies. In *Proc. 14th European Symposium on Algorithms*, volume 4168 of *LNCS*, pages 804–816. Springer, 2006. 126

[88] S. Saroiu, K. Gummadi, and S. Gribble. A Measurement Study of P2P File Sharing Systems. In *Multimedia Computing and Networking*, 2002. 155, 156

[89] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer-Verlag, 2003. 115

[90] S. Seetharaman and M. Ammar. On the Interaction between Dynamic Routing in the Overlay and Native Layers. In *IEEE INFOCOM'06*, 2006. 153, 159

[91] Stephen B. Seidman. Network Structure and Minimum Degree. *Social Networks*, 5:269–287, 1983. 117

[92] S. Sen and J. Wang. Analyzing Peer-to-Peer Traffic Across Large Networks. In *Internet Measurement Workshop*, 2002. 153

[93] Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster Graph Modification Problems. In *Proceedings of the 28th International Workshop on Graph-Theoretical Conecpts in Computer Science (WG*, volume 2573 of *Lecture Notes in Computer Science*, pages 379–390. Springer-Verlag, 2002. Available from World Wide Web: `http://www.springerlink.com/app/home/content.asp?wasp=5f8nqjqtlg5l7xxuueet&referrer=contribution&format=2&page=1&pagecount=12`. 28

[94] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. 121

[95] Alistair Sinclair. *Algorithms for Random Generation and Counting: A Markov Chain Approach.* Birkhäuser Verlag, 1993. 24

[96] Slyck. Available from World Wide Web: `http://www.slyck.com/`. 155

[97] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems. In *Internet Measurements Conference*, 2005. 155

[98] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz. Characterizing the Internet Hierarchy from Multiple Vantage Points. In *Proceedings of Infocom'02*, pages 618–627, 2002. 157

[99] Test Traffic Analysis Update [online]. Available from World Wide Web: `http://www.ripe.net/ripe/meetings/archive/ripe-43/presentations/ripe43-tt-analysis/`. 136

[100] Stijn M. van Dongen. *Graph Clustering by Flow Simulation.* PhD thesis, University of Utrecht, 2000. 27, 65, 66, 121

[101] Santosh Vempala, Ravi Kannan, and Adrian Vetta. On Clusterings - Good, Bad and Spectral. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS'00)*, pages 367–378, 2000. 24, 25, 63, 66, 112

[102] The W32.nimda Worm [online]. Available from World Wide Web: `http://www.ciac.org/ciac/bulletins/l-144.shtml`. 135

[103] Silke Wagner and Dorothea Wagner. Comparing Clusterings – An Overview. Technical Report 2006-04, ITI Wagner, Faculty of Informatics, Universität Karlsruhe (TH), 2006. 45

[104] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications.* Cambridge University Press, 1994. 121

[105] Wikipedia. Information society — Wikipedia, the free encyclopedia, 2005. Available from World Wide Web: `http://en.wikipedia.org/wiki/Information_society`. [Online; accessed 09-November-2005]. 9

[106] Thomas Willhalm and Ulrik Brandes. Visualization of Bibliographic Networks with a Reshaped Landscape Metaphor. In *Proceedings of the 4th Joint Eurographics - IEEE TVCG Symposium on Visualization (VisSym '02)*, pages 159–164. ACM Press, 2002. 162

[107] C.T̃. Zahn. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transactions on Computers*, C-20:68–86, 1971. 69

Index

Zusammenfassung

Der Themenkomplex *Clusterung und Clusteranalyse* beschäftigt sich mit dem Einteilen von Daten in natürliche Gruppen. Die Eingabedaten bestehen aus einer Menge von Objekten und einer Gewichtsfunktion, die die (Un-)ähnlichkeit von Paaren dieser Objekten bewertet. Ursprünglich entstammt das zugrunde liegende Problem den Gebieten der Klassifikation und Mustererkennung; im Laufe der Zeit wurden Clusterungstechniken in eine Vielzahl von weiteren Bereichen integriert. Heute ist Clusteranalyse ein essentieller Bestandteil der Erforschung, Bearbeitung und Analyse von großen und komplexen Daten, wie sie sich etwa in den Gebieten Verkehrs- und Transportplanung, Biologie, Geologie, Ökonomie und Kommunikationswissenschaften finden. Gerade durch den vielseitigen Einsatz von Clusterungen in sehr unterschiedlichen Szenarien konnte sich keine einheitliche Terminologie entwickeln. Hinzu kommt, dass viele Gebiete einen eigenen formalen Rahmen geschaffen haben und bestehende Techniken neu erfunden haben. Selbst der definierende Begriff der natürlichen Gruppe wird in (fast) jedem Bereich unterschiedlich ausgelegt.

In meiner Dissertation betrachte ich den Spezialfall des Graphenclusterns mit dem Fokus auf dessen algorithmischen Aspekten. Beim Clustern von Graphen stellen die Knoten die Objekte dar und die Gewichtsfunktion wird durch (gewichtete) Kanten modelliert. In vielen verschiedenen Anwendungsgebieten liegen die Eingabedaten als Netzwerk vor, etwa bei der Analyse von sozialen Interaktionen oder der Visualisierung von großen Graphen. Im Unterschied zu Eingabedaten, die in Gebieten wie Klassifikation, Mustererkennung und Data-Mining betrachtet werden, enthalten die Netzwerke wenige Kanten und lassen sich oft nicht vervollständigen. Als weitere Einschränkung werden dichtebasierte Techniken untersucht, d. h. die Gruppen innerhalb der Clusterungen sollen dichten Subgraphen entsprechen, die durch dünne Schnitte voneinander getrennt werden können.

Im ersten Teil der Arbeit werden theoretische Grundlagen und eine ausführliche experimentelle Auswertung für folgende Aspekte präsentiert: das qualitative Bewerten von Clusterungen, Ansätze für das Vergleichen von Paaren von Clusterungen, das Generieren von Graphen mit vorgegebener (signifikanter) Clusterung sowie das effiziente Berechnen von Clusterungen. Das Messen der Qualität einer Clusterung ist sehr eng an die (informelle) Definition einer Clusterung geknüpft, und so verwundert es nicht, dass kaum eine einheitliche und weitverbreitete Messmethode dafür existiert. Neben der Einführung einer Formalisierung, die viele existierende Bewertungsfunktionen abdeckt, werden fünf konkrete Maße bezüglich Komplexitätsstatus sowie charakteristischen Eigenschaften untersucht. Vergleichsmaße wurden bisher nur vereinzelt in der Literatur betrachtet. Es wird ein Überblick über verschiedene Ansätze und deren Erweiterungsmöglichkeiten für Graphclusterungen vorgestellt

und diskutiert. Des Weiteren werden grundlegende Prinzipien für Clusterungsalgorithmen vorgestellt und ihre Vor- und Nachteile beleuchtet. Neben der abstrakten Beschreibung von allgemeinen Techniken werden verschiedene Algorithmen aus der Praxis vorgestellt, die Teil einer ausführlichen Studie zur Auswertung von Clusterungstechniken sind. Für diese Studie wurden auch Generatoren entwickelt, die Graphen mit bekannter (signifikanter) Clusterung erzeugen. Diese experimentelle Auswertung gliedert sich grob wie folgt: Zuerst werden einfache Tests entwickelt, die die Nutzbarkeit einer Technik überprüfen. Diese Tests sind in Analogie der Unit-Tests aus der Softwareentwicklung einfach und allgemein gehalten, lassen sich aber an spezielle Situationen und Anwendungen anpassen. Anschließend werden die Algorithmen qualitativ evaluiert. Dabei werden die zuvor gewonnenen Ergebnisse integriert. Den Abschluss bildet eine Zusammenfassung über mögliche Erweiterungen für Clusterungen, etwa das dynamische Problem, Clusterungen nach änderungen zu aktualisieren, oder die Erweiterung der repräsentierenden Datenstruktur.

Der zweite Teil der Arbeit untersucht die praktische Anwendbarkeit von Clusterungstechniken. In meinem Fall dient die Visualisierung von hierarchischen Netzwerken als Grundmotivation. Es wird eine Fallstudie an Hand des Netzwerks der *Autonomen Systeme*, einer Abstraktion des physikalischen Internets, durchgeführt. Das Netzwerk hat eine moderate Größe mit 10.000–21.000 Knoten und 25.000–45.000 Kanten, enthält aber einige "‘irrelevante"’ Elemente, welche die Analyse von zentralen und strukturell wichtigen Teilen behindern. Die Studie verläuft grob in drei Phasen: Als erstes wird Clusterung zur Datenreduktion eingesetzt. Anschließend wird die Visualisierungstechnik entwickelt und an realen und generierten Instanzen getestet. Als ein Nebenergebnis konnte gezeigt werden, dass ein weitverbreiteter Generator die Netzwerkstruktur homogener erzeugt, als sie in der Realität vorliegt. Im Anschluss daran wird die Visualisierungstechnik bei der Analyse der Dateitauschbörse *Gnutella* eingesetzt. Gnutella ist eine Anwendung, die bestehende Kommunikationsnetzwerke wie das Internet ausnutzt, um ihren eigenen Service anzubieten. Als Ergebnis konnte gezeigt werden, dass Gnutella einige Eigenschaften mit zufälligen (Overlay-)Netzwerken gemeinsam hat, allerdings in anderen abweicht. Des Weiteren konnte eine gewisse Korrelation zu dem zugrunde liegenden Internet (hier auf der Ebene der Autonomen Systeme) nachgewiesen werden. Eine Diskussion über die Erweiterbarkeit der Visualisierungstechnik beendet den zweiten Teil der Arbeit.

Zusammenfassend wird in meiner Dissertation der Dialog zwischen theoretischen und praktischen Aspekten des Graphenclusterns untersucht. Auf der einen Seite werden verschiedene Werkzeuge wie Qualitätsmessung, Generierung und Berechnung, systematisch dargestellt, ausgewertet und ihre wechselseitigen Abhängigkeiten beleuchtet. Auf der anderen Seite werden dieselben Techniken eingesetzt, um das Verständnis von realen Netzwerken zu verbessern. Ein Beispiel der Visualisierung des Netzwerkes der Autonomen Systeme ist in Abbildung 11.1 zu sehen. Die prominente vulkanartige Form läßt schon auf die hierarchische Struktur schliessen.

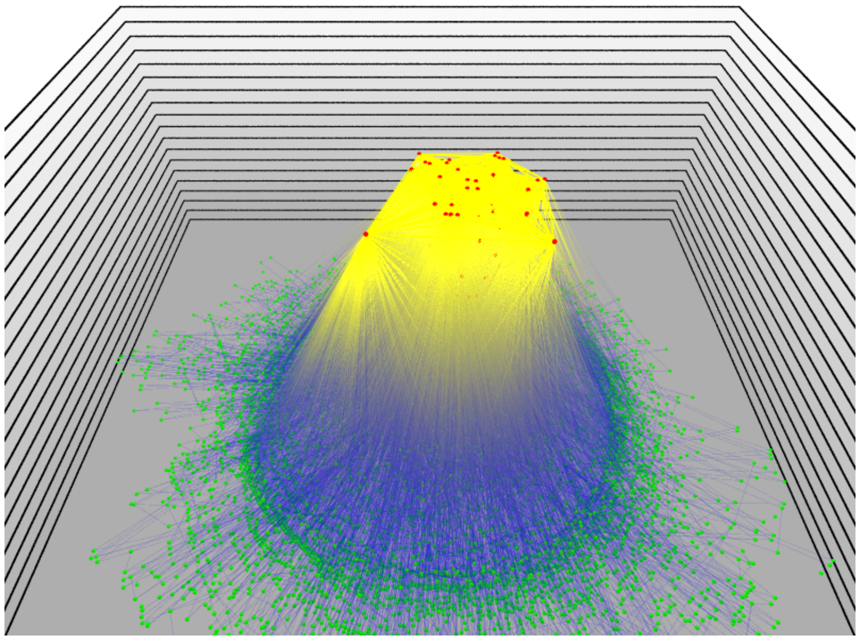


Figure 11.1.: Visualisierung des Netzwerks der Autonomen Systeme zum Zeitpunkt 01.06.2001. Die Höhe eines Knotens spiegelt seine Wichtigkeit wieder: wichtige oben und unwichtige unten.