

# Community Analysis in Dynamic Social Networks

## Dissertation

zur Erlangung des akademischen Grades

## Doktoringenieurin (Dr.-Ing.)

angenommen durch die Fakultät für Informatik  
der Otto-von-Guericke-Universität Magdeburg

von Dipl. Wirt.-Inf. Tanja Falkowski  
geb. am 18. Oktober 1972 in Göttingen

Gutachter:

Prof. Dr. Ulrik Brandes

Prof. Dr. Rudolf Kruse

Prof. Dr. Myra Spiliopoulou

Magdeburg, den 14. Mai 2009



# Contents

<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of Algorithms</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Online Communities - A Spreading Phenomenon . . . . .	1
1.2 Addressing Real World Problems . . . . .	5
1.3 Structure of the Thesis . . . . .	6
<b>2 Fundamentals</b>	<b>9</b>
2.1 Social Network Analysis . . . . .	9
2.1.1 Types of Social Networks . . . . .	9
2.1.2 Analyzing Networks . . . . .	11
2.2 Communities in Networks . . . . .	16
2.2.1 Subgroups in Social Network Analysis . . . . .	16
2.2.2 Real-World and Virtual Communities . . . . .	19
2.3 Graph Clustering Techniques . . . . .	22
2.3.1 Partitioning Methods . . . . .	23
2.3.2 Hierarchical Methods . . . . .	24
2.3.3 Density-based Methods . . . . .	26
2.4 Evaluating Graph Clusters . . . . .	27
2.4.1 Resilience Measure for Graph Clusters . . . . .	27
2.4.2 Quality Measures for Graph Clusterings . . . . .	28
2.5 Tracking Temporal Dynamics of Communities . . . . .	32
2.5.1 Network Evolution Models . . . . .	33
2.5.2 Community Evolution Models . . . . .	33
2.5.3 Models for the Evolution of Clusters . . . . .	34
2.5.4 Community Evolution in an Organizational Context . . . . .	35
<b>3 Community as a Subgraph Formation Over Time</b>	<b>39</b>
3.1 The Community Discovery Model . . . . .	39
3.1.1 Partitioning the Time Axis and Building the Graph of Interactions	41
3.1.2 Clustering Users into Community Instances . . . . .	44
3.1.3 Evolution of Community Instances . . . . .	46
3.1.4 Community Survival Graph . . . . .	48

3.1.5	Communities in the Survival Graph . . . . .	49
3.2	Visualizing Temporal Dynamics in Communities . . . . .	50
3.2.1	Actor-oriented Visualization of Community Instance Evolution . . .	50
3.2.2	Group-oriented Visualization of Community Instance Evolution . . .	52
3.3	Application: Studying Communities with Fluctuating Members . . . . .	55
3.3.1	The <i>IKUS</i> Data Set . . . . .	56
3.3.2	Determining Parameters . . . . .	57
3.3.3	Data Set Characteristics . . . . .	58
3.3.4	Fluctuating Members . . . . .	59
3.3.5	Conclusion . . . . .	63
3.4	Application: Representativeness of Communities for Members . . . . .	63
3.4.1	Users within Evolving Communities . . . . .	63
3.4.2	Relevance of a Community for a User . . . . .	65
3.4.3	Conclusion . . . . .	67
3.5	Related Work . . . . .	67
3.5.1	Modelling Community Dynamics . . . . .	67
3.5.2	Visualizing Community Dynamics . . . . .	69
3.6	Concluding Remarks . . . . .	69
<b>4</b>	<b>Community as a Subgraph that Evolves Over Time</b>	<b>73</b>
4.1	DENGRAPH: Density-based Graph Clustering . . . . .	73
4.1.1	Density-Reachability in a Graph . . . . .	74
4.1.2	Graph Clustering for Overlapping Communities . . . . .	76
4.1.3	Complexity and Computation Time . . . . .	77
4.2	DENGRAPH-IO: Incremental Graph Clustering . . . . .	77
4.2.1	Cluster Changes . . . . .	78
4.2.2	Correctness of the Incremental Updates . . . . .	83
4.2.3	Complexity and Computation Time . . . . .	87
4.3	Evaluating Graph Clusters . . . . .	88
4.3.1	Graph Cluster Stability . . . . .	89
4.3.2	Graph Cluster Quality . . . . .	90
4.4	Application: Graph over Interactions . . . . .	93
4.4.1	<i>Enron</i> Data Set . . . . .	93
4.4.2	Defining Proximity in the <i>Enron</i> Graph . . . . .	95
4.4.3	Experiments and Results . . . . .	96
4.5	Application: Graph over Similarity . . . . .	115
4.5.1	<i>Last.fm</i> Data Set . . . . .	116
4.5.2	Defining Similarity in the <i>Last.fm</i> Graph . . . . .	116
4.5.3	Experiments and Results . . . . .	119
4.6	Characteristics of the DENGRAPH-IO Algorithm . . . . .	132
4.6.1	The Impact of the Parameters on the Clustering . . . . .	132
4.6.2	Characteristics of Core and Border Vertices . . . . .	133
4.6.3	Relevant Edge-Changes for the Clustering . . . . .	134
4.6.4	Impact of Positive and Negative Changes on Running Time . . . . .	134

4.6.5	Cluster Resilience and DENGGRAPH-Stability . . . . .	135
4.7	Related Work . . . . .	136
4.8	Concluding Remarks . . . . .	137
<b>5</b>	<b>Conclusion</b>	<b>139</b>
5.1	Main Contributions . . . . .	139
5.2	Future Work . . . . .	140
<b>A</b>	<b>Pseudocode</b>	<b>143</b>
<b>B</b>	<b>Experimental Results</b>	<b>147</b>
	<b>Bibliography</b>	<b>157</b>



# List of Figures

3.1	Community Discovery Roadmap . . . . .	40
3.2	Partitioning the Time Axis. . . . .	41
3.3	Building a Graph for Each Time Window. . . . .	44
3.4	Edge Betweenness Clustering. . . . .	45
3.5	Dendrogram-cut Based on Highest Modularity. . . . .	47
3.6	Comparing Community Instances . . . . .	51
3.7	Community Survival Graph . . . . .	54
3.8	Temporal View of Community Survival Graph with Zoom . . . . .	55
3.9	Cut-out of Community Survival Graph . . . . .	56
3.10	<i>IKUS</i> : Degree Distribution . . . . .	58
3.11	<i>IKUS</i> : Dendrogram (Example) . . . . .	59
3.12	Actor-oriented Visualization . . . . .	60
3.13	<i>IKUS</i> : Structural Breaks in the Community Evolution . . . . .	62
3.14	Example for Community Instance . . . . .	65
4.1	DENGRAPH: Concepts of Connectivity . . . . .	75
4.2	DENGRAPH-IO: The <i>update</i> method . . . . .	80
4.3	DENGRAPH-IO: New Close Relation and Lost Close Relation . . . . .	82
4.4	DENGRAPH-IO: Detecting a Cluster Split . . . . .	84
4.5	Modularity Calculation for Clusters with Overlaps . . . . .	91
4.6	<i>Enron</i> : Data Set Statistics . . . . .	94
4.7	<i>Enron</i> : Influence of Parameters $\epsilon$ and $\eta$ ( <i>Enron-intern, week</i> ) . . . . .	98
4.8	<i>Enron</i> : Influence of Parameters $\epsilon$ and $\eta$ ( <i>Enron-intern, month</i> ) . . . . .	99
4.9	<i>Enron</i> : Influence of Parameters $\epsilon$ and $\eta$ ( <i>Enron-total, week</i> ) . . . . .	100
4.10	<i>Enron</i> : Influence of Parameters $\epsilon$ and $\eta$ ( <i>Enron-total, month</i> ) . . . . .	101
4.11	<i>Enron</i> : Graph in Week 49/2000 ( <i>Enron-intern, week</i> ) . . . . .	104
4.12	<i>Enron</i> : Cluster Resilience and DENGRAPH-Stability ( <i>Enron-intern, week</i> ) . . . . .	105
4.13	<i>Enron</i> : Graph in Month 1/2001 ( <i>Enron-intern, month</i> ) . . . . .	106
4.14	<i>Enron</i> : Cluster Resilience and DENGRAPH-Stability ( <i>Enron-intern, month</i> ) . . . . .	107
4.15	<i>Enron</i> : Graph in Week 50/2001 ( <i>Enron-total, week</i> ) . . . . .	108
4.16	<i>Enron</i> : Cluster Resilience and DENGRAPH-Stability ( <i>Enron-total, week</i> ) . . . . .	109
4.17	<i>Enron</i> : Graph in Month 4/2001 ( <i>Enron-total, month</i> ) . . . . .	110
4.18	<i>Enron</i> : Cluster Resilience and DENGRAPH-Stability ( <i>Enron-total, month</i> ) . . . . .	111
4.19	<i>Enron</i> : Clustering Results ( <i>Enron-total, week</i> ) . . . . .	114
4.20	<i>Last.fm</i> : Influence of Parameters $\epsilon$ and $\eta$ . . . . .	119
4.21	<i>Last.fm</i> : Cluster Resilience and DENGRAPH-Stability (week 10/2007) . . . . .	122

LIST OF FIGURES

---

4.22	<i>Last.fm</i> : Cluster Resilience and DENGRAPH-Stability (week 14/2007)	124
4.23	<i>Last.fm</i> : Cluster Evolution	126
4.24	<i>Last.fm</i> : Clustered Graph in Week 8/2007	127
4.25	<i>Last.fm</i> : Clustered Graph in Week 9/2007	128
4.26	<i>Last.fm</i> : Clustered Graph in Week 10/2007	129
4.27	<i>Last.fm</i> : Clustered Graph in Week 13/2007	130
4.28	<i>Last.fm</i> : Clustered Graph in Week 14/2007	131
B.1	<i>Last.fm</i> : Cluster Resilience and DENGRAPH-Stability, Cluster 1-4 (10/07)	147
B.2	<i>Last.fm</i> : Cluster Resilience and DENGRAPH-Stability, Cluster 5-7 (10/07)	148
B.3	<i>Last.fm</i> : Cluster Resilience and DENGRAPH-Stability, Cluster 1-6 (14/07)	149
B.4	<i>Last.fm</i> : Cluster Resilience and DENGRAPH-Stability, Cluster 7-10 (14/07)	150
B.5	<i>Enron</i> Clustering Results (intern, weekly)	151
B.6	<i>Enron</i> Clustering Results (intern, monthly)	152
B.7	<i>Enron</i> Clustering Results (total, weekly)	153
B.8	<i>Enron</i> Clustering Results (total, monthly)	154
B.9	<i>Last.fm</i> : Clustering Results ( $\epsilon = 0.02$ , $\eta = 5$ )	155

# List of Tables

1.1	Two Frameworks for the Analysis of Community Dynamics . . . . .	6
2.1	Types of Cluster Evolution . . . . .	36
3.1	Representativeness of Communities for Members . . . . .	66
3.2	Temporal Analysis with Aggregates and Snapshots . . . . .	72
4.1	<i>Enron</i> : Data Set Information . . . . .	94
4.2	<i>Enron</i> : Data Set Statistics . . . . .	95
4.3	<i>Enron</i> : Clustering Statistics . . . . .	102
4.4	<i>Enron</i> : Cluster Resilience and DENGGRAPH-Stability ( <i>Enron-intern, week</i> )	105
4.5	<i>Enron</i> : Cluster Resilience and DENGGRAPH-Stability ( <i>Enron-intern, month</i> )	107
4.6	<i>Enron</i> : Cluster Resilience and DENGGRAPH-Stability ( <i>Enron-total, week</i> )	109
4.7	<i>Enron</i> : Cluster Resilience and DENGGRAPH-Stability ( <i>Enron-week, month</i> )	111
4.8	<i>Last.fm</i> : Clustering Statistics . . . . .	120
4.9	<i>Last.fm</i> : Cluster Resilience and DENGGRAPH-Stability (week 10/2007)	121
4.10	<i>Last.fm</i> : Cluster Resilience and DENGGRAPH-Stability (week 14/2007)	123
4.11	<i>Last.fm</i> : Statistics; Exemplary for Five Weeks . . . . .	125



# List of Algorithms

2.1	Partitional Clustering: k-means . . . . .	23
2.2	Hierarchical Agglomerative Clustering . . . . .	25
2.3	Hierarchical Divisive Clustering . . . . .	25
3.1	Edge Betweenness Clustering . . . . .	46
4.1	DENGRAPH: Core . . . . .	78
4.2	DENGRAPH-IO: Incremental Clustering (overview) . . . . .	85
4.3	DENGRAPH-IO: <i>UpdateClustering</i> (u) Function (overview) . . . . .	86
4.4	DENGRAPH: Stability . . . . .	90
A.1	DENGRAPH-IO: <i>DensityConnectedNodes</i> (u,v) Function . . . . .	143
A.2	DENGRAPH-IO: <i>DensityConnectedNodes</i> (u) Function . . . . .	143
A.3	DENGRAPH: Non-incremental Clustering . . . . .	144
A.4	DENGRAPH-IO: Incremental Clustering . . . . .	145
A.5	DENGRAPH-IO: <i>UpdateClustering</i> (u) Function . . . . .	146



## Abstract

Due to the growing popularity of social networking platforms, the analysis of online communities became in recent years a popular topic in different research fields. However, an aspect that has received only little attention is how the temporal aspects of social networks can be studied. This thesis bridges the gap and deals with the analysis of community structures in large social networks and their temporal dynamics. Two clustering techniques are proposed to detect communities in social networks and to study the evolution of these structures over time. The two approaches basically differ in the underlying definition of what constitutes a community over time: In the first case, a community is considered a subgroup that can be observed over time and a hierarchical edge betweenness clustering approach is proposed to detect such communities. In the second case, a community is defined as a subgroup that evolves over time and an incremental density-based clustering algorithm is proposed to detect and track these evolving communities. The applicability of the proposed approaches is evaluated by applying them to different real world data sets. The obtained results indicate that the introduced clustering techniques are appropriate to efficiently detect online communities in large social networks and to track their evolution over time.

## Kurzfassung

Die Analyse von Gruppen in sozialen Netzwerken hat in den letzten Jahren in verschiedensten Forschungsbereichen zunehmende Bedeutung erlangt. Ein Aspekt der hierbei bislang wenig untersucht wurde, ist die Analyse der zeitlichen Entwicklung der Gruppen ("Communities"). Die vorliegende Arbeit untersucht diese Fragestellung und stellt zwei Verfahren vor, die eine temporale Analyse von Gruppenstrukturen in Netzwerken ermöglichen. Die vorgeschlagenen Verfahren unterscheiden sich in der Art der Definition einer Community: Im ersten Fall wird eine Community als eine Gruppe angesehen, die über die Zeit hinweg beobachtbar ist. Für die Erkennung wird ein hierarchisches Clustering-Verfahren basierend auf einem Zentralitätsmaß für Kanten vorgeschlagen. Im zweiten Fall ist eine Community als eine Gruppe definiert, die sich über die Zeit entwickelt. Zur Entdeckung und zeitlichen Beobachtung wird ein inkrementelles dichte-basiertes Clustering-Verfahren entwickelt. Die Methoden werden auf verschiedene reale Datensätze angewandt und ihre Eignung evaluiert. Die Ergebnisse lassen den Schluss zu, dass beide Verfahren geeignet sind, den jeweiligen Community-Typ zu erkennen. Insbesondere durch das inkrementelle Clustering wird eine effiziente Verfolgung der zeitlichen Änderungen von Gruppenstrukturen erzielt: Hierdurch wird eine Analyse temporalen Entwicklungen von Communities auch in sehr großen Netzwerken ermöglicht.



# 1 Introduction

This thesis deals with the analysis of community structures in social networks and their temporal dynamics. For this, methods to detect communities in social networks are proposed as well as approaches to track the evolution of these structures over time. To verify their applicability, the presented methods are applied to different real world data sets and the obtained results are evaluated. In this first chapter, the motivation for studying evolving communities in social networks is discussed. Furthermore, based on the extracted real world problems, the goals of this thesis are presented. The chapter closes with an outline of the thesis structure.

## 1.1 Online Communities - A Spreading Phenomenon

The proliferation of online communities, in the WWW as well as in organizations, is a phenomenon worth studying: Community platform providers are interested to learn more about their members, for example, to enhance their services and since recent research indicate the positive impact of communities on the performance of organizations, the analysis of intra-organizational communities is of interest as well.

### Online Communities in the WWW

The World Wide Web (WWW) brings people together that share ideas, search for information or discuss about topics they are interested in. On so-called *social networking sites* people provide personal information, for example, in the form of user profiles containing data such as name, gender, age and location. Additionally, also implicit personal information which can be inferred from messages, for example, in discussion fora or guestbooks are available. A social networking site is a Web-based service that allows individuals (i) to construct a public or semi-public profile, (ii) to assemble a list of users with whom they share a connection and (iii) to view their list of connections and those made by other users (on their list) [20]. The semantic of the connection varies from site to site such as friendship, business relation, political opinion or more generally a common interest (music, photography, etc.).

There are hundreds of social networking sites, providing a platform for a wide range of interests and practices. Some of these platforms, such as *MySpace*<sup>1</sup> and *Facebook*<sup>2</sup>, have grown dramatically in recent years, with more people joining every day. For millions of people, online social networking has become an integral part of their live and is seen

---

<sup>1</sup><http://www.myspace.com/>

<sup>2</sup><http://www.facebook.com/>

as an extension of our means of communication, created by the Internet. Currently, a quarter of all U.S. surfers regularly visit social networking sites, according to a statistic released in the Consumer Internet Barometer by The Conference Board.<sup>3</sup> About fifty percent of social networkers visit these sites on a daily basis, while some of them even log in several times a day. Teenagers (aged 12 to 17) are more likely to be daily users. Besides interaction with friends, about half of all users report using networks sites to maintain their own online profile.<sup>4</sup>

Among the working-age population (aged 30 and over), the Consumer Internet Barometer reports one out of eight uses social networking sites to conduct business. Based on this observation, The Conference Board predicts that the next step will be to expand and incorporate these networks into the corporate lives of more people. A growing number of users in existing business networks supports this projection. *LinkedIn*<sup>5</sup> and *XING*<sup>6</sup> are two examples for popular platforms for professionals to connect with business partners. *LinkedIn* is an U.S. platform founded in 2003 with today around 34 million users and *XING* is a networking platform mostly frequented by German-speaking people. The *XING AG* (formerly *openBC*), founded in 2003, has according to company's information over 6.5 million users, 300 million page impressions per month by 20 million visitors. Both platforms offer a free and a paid account that provides more tools for finding and reaching people. However, so far only few social networking platforms charge money for membership. This may be because service providers seek a large number of members in order to increase their income from online advertisements. Hence, charging for membership would be counter productive.

The growing popularity of social networks can also be deduced from the increasing amount of money companies spend for online marketing campaigns on social networking sites. A study published by eMarketer<sup>7</sup> shows that one out of four businesses in the U.S. plans on spending in 2009 more money for online marketing on social networking sites. This is the highest increase compared to other channels such as e-mail, company Web site, search engines, etc.

Mainly because of the advertisement-driven revenue model, providers of social networking platforms are interested to gain a deeper understanding about their users which will allow in the end much better targeted advertising based on the users' preferences. The analysis of patterns in social networking sites can support this understanding, for example, to learn about the importance and influence of people or groups in social networks. Besides improving advertisements, knowledge revealed by data analysis about platform users could be of interest for service providers to face challenges such as the following:

---

<sup>3</sup>The Consumer Internet Barometer is based on a quarterly survey of 10,000 households. The latest survey was conducted during the second quarter of 2008. <http://www.consumerinternetbarometer.us/>, as of March 13, 2009.

<sup>4</sup>An extensive study on undergraduate student participation in social networking sites can be found in [144].

<sup>5</sup><http://www.linkedin.com/>

<sup>6</sup><http://www.xing.com/>

<sup>7</sup><http://www.emarketer.com/>

- Enhancing the load balancing system to offer a better service to premium users
- Maintaining a positive environment free of harassers and offenders so as to make users feel comfortable
- Anticipate new forum topics to stimulate discussion and interaction

### **Online Communities in an Organizational Context**

Driven by the growing popularity of social networking sites, several industries started using social-networking techniques mainly to establish a dialog with their customers, to reach potential clients with highly targeted advertisements and to build a brand affinity. Besides open platforms that support social networking among individuals in general or customers in particular, interactions between professionals are also observable, mostly in a more unstructured manner, in every organization.

Especially in knowledge-intensive sectors, personal relationships between members of an organization are important to obtain relevant information and to do their job. This information resides somewhere in the organization, perhaps in the mind of an expert who works in another department and/or at a different geographic location. Despite growing databases it is much more likely that people turn to friends or colleagues for information instead of electronic information sources [40]. These networks provide strategic and operational benefits to organizations by enabling an effective collaboration. The problem grows even more acute in large corporations, when people must be able to reach knowledgeable colleagues to perform their respective duties. Organizations are thus interested in promoting and supporting these networks.

A study conducted by the National Learning Laboratories for Applied Behavioral Science supports this observation. Companies are more and more aware of the importance to provide knowledge sharing opportunities for their employees. The methods with the highest information retention rate are: to teach others (80%), practice by doing (75%) and discussion groups (50%). Therefore, among others, companies need to set up the general conditions that allow employees to meet and discuss. This is an important prerequisite for experience sharing and problem solving.

These discussions may take place as personal meetings (same time and space) or as technologically supported meetings (same time and different space or different time and space). Online platforms can serve as a medium for the meeting and the communication of these groups. These platforms gain importance as it is known that a great portion of a person's personal network is based on informal relationships that are not based on formal organizational boundaries but rather on previous work or project relationships [141]. It has been reported in several studies that individuals working together virtually (also called *virtual teams*) are important organizational structures (see, e.g., [69, 75, 96, 132]).

Several recent studies report that the structure of a person's social network has a significant impact on his information acquisition (see, e.g., [18, 41, 39, 42, 83]). Especially in large complex organizations some employees may be at a disadvantage in searching for information. Singh et al. conducted a study at a large multinational consulting firm with more than 50 offices in 34 countries [152]. In such an environment, professionals

routinely rely on help from their colleagues, to ask for advice on certain topics, markets, industries or companies. Thus, taking the shortest path to the right person can save a lot of time and in the end greatly increases the operational efficiency of an organization. Besides of this study, many other reports are available that support the assumption that an organization supporting employees in building networks will increase its performance (see, e.g., [14, 124, 140]).

Considerable research has focused on internal and external factors that affect virtual teams (see, e.g., [86, 93, 98, 117, 135]). Internal factors such as hierarchies, departments and rewards as well as external factors influence the “performance” of online communities. This raises the question what the key characteristics of an organizational environment are that permit a successful community work and how the performance of virtual teams can be enhanced. A related question that arises is how do the structures of cooperation in an organization relate to the creation and development of creative ideas.<sup>8</sup>

Lesser and Storck [114] reported that the support of groups in particular in large organizations is one strategy to improve organizational performance. The kind of group the authors observed has become known as a *community of practice* (CoP). A community of practice is a group of people whose members are engaged in discussions based on their common goals and interests [167]. In contrast to teams that are established by the organization, CoPs are formed around practice. In the beginning, a CoP was recognized as being valuable for the individuals in the group, whereas nowadays, organizations observe a positive impact on their organizational performance. CoPs appear to be more flexible to handle unstructured problems and to come up with new ideas for products and services. Furthermore, knowledge is shared between employees outside of defined processes and a long-term organizational memory is maintained which results among others in a decreased learning curve of new employees [114]. For more information on the impact of communities of practice on the financial and innovative performance of organizations see, e.g., [143, 166, 48, 76, 167].

Analogously, to the social networking sites mentioned in the previous section, organizations must ensure to motivate users to contribute to the platform. Ensuring that members who truly stand out stay active is critical for keeping the platform attractive. Davenport and Prusak studied the reasons for individuals to contribute in an environment that aims to exchange knowledge and identified three motivations: Some people just like to help other users (“altruism”), others expect recognition by the site and other peers for their contribution (“reputation”) and others expect special treatment by the platform provider such as special access to information that is not accessible otherwise (a form of “reciprocity”) [43]. Clary et al. studied the motivations of volunteers and identified six motivational categories: “career”, “social”, “values”, “enhancement”, “protective” and “understanding”. These categories are basically refinements of the motivations proposed by Davenport and Prusak [37]. Nov reported that these factors can also explain the contribution of online content, in his case, to the Wikipedia [131]. It is important to

---

<sup>8</sup>Amabile defines creativity as the production of novel and useful ideas in any domain [5]. Creativity is seen as the basis for innovation and so innovation is defined as the successful implementation of creative ideas within an organization.

identify and assess the impact of these motivations on the contribution to an online platform to motivate and retain users who contribute to the community,

Organizations and community providers could be supported in their task to foster community building by finding patterns of contribution behavior for example to assess the importance of individuals or groups to a community. The aim of this thesis is to develop such methods to study the temporal evolution of online communities.

## 1.2 Addressing Real World Problems

In the last section we discussed the usefulness of communities in different contexts. To assist organizations or service providers in analyzing “their” communities, different questions and problems need to be addressed. Those which are discussed in this thesis are presented in the following.

### Growing Network Sizes with “Casual” Actors

An important characteristic of social networks that needs to be considered is their size. The growth of online social networks (measured in number of subscribed users) is impressive. In January 2009, *MySpace US* reported 76 million members, with a U.S. growth rate of 0.8% per month.<sup>9</sup> Besides a high number of subscribed users, social networking platforms also encounter a high popularity observable also in a high number of unique visitors. In January 2009, according to Nielsen Online<sup>10</sup>, *Facebook* was on rank six in terms of unique visitors in the US with 22 million users in one week. In December 2008, *LinkedIn* was visited by 6.3 million and *MySpace US* by 69 million unique guests.

The distinction between the number of “subscribed users” and “active users” is very important as the fraction of “passive users” is usually very large. Therefore, most data sets of online communities entail a huge number of actors with a very low activity level who only casually log in or contribute to the community. Active participation plays a key role in determining whether an individual will experience benefits from group membership [119]. Considering the growing size of social platforms, an important task is therefore to develop efficient algorithms to handle the data of large communities with a high fraction of inactive user.

### Networks Evolve Over Time

In the past, social networks have been already studied widely, however, an important property has been largely ignored: social networks change over time. The observed relationship such as the interaction between users in a social network is in most cases not stable over time but fluctuates. Taking for example large online communities such as *Facebook* or *XING* where frequently new users sign-up, old users leave and the intensity of interaction changes. Similarly, changes can be observed in an e-mail network of an

---

<sup>9</sup>According to Comscore via <http://www.techcrunch.com/2009/01/13/social-networking-will-facebook-overtake-myspace-in-the-us-in-2009/>, as of March 13, 2009.

<sup>10</sup><http://www.nielsen-online.com/>, as of March 13, 2009

organization. We can think of several reasons why the interaction between employees changes over time: new cooperations between departments are established, old projects are finished and new ones started with different teams, employees take on new responsibilities, etc. All these organizational changes result in an altered communication network. To learn about how the network changes over time it is important to develop methods that allow to study the dynamic aspects of social networks.

### Defining a Community Over Time

A problem we encountered is that there is no accepted definition of what constitutes a community. It is not worthwhile to try to find an answer as there is no right or wrong here. The definition depends on the context and the goal of the analysis. We present different approaches to define a community in Chapter 2, all of which have in common that they define a community as something based on a static graph. The question is, how a community relates to the temporal dimension. We propose solutions for two perspectives that one can have on communities: (i) a community is a structure that is observable over time and (ii) a community is a structure that evolves over time.

## 1.3 Structure of the Thesis

The aim of this thesis is to develop an approach for the analysis of community dynamics in large, noisy social networks that explicitly uses available temporal information to study subgroup evolutions. For this, we propose two frameworks for efficiently identifying communities in dynamic social networks and observing their temporal evolution. The two frameworks differ in the underlying definition of a community. In Table 1.1 the two approaches are shown.

Table 1.1: Two Frameworks for the Analysis of Community Dynamics

	Chapter 3	Chapter 4
<b>Definition of a Community</b>	Community as a structure that is observable over time	Community as a structure that evolves over time
<b>Community Detection Method</b>	Hierarchical clustering on graph of similar community instances	Density-based clustering
<b>Community Dynamics Analysis</b>	Matching cluster snapshots	Incremental cluster updates

The thesis is structured as follows. Chapter 2 provides an introduction to the fundamental terms and concepts of social network analysis, community research, cluster analysis and cluster evaluation that are used in this thesis.

In Chapter 3, a method to observe communities from a macroscopic view is presented. In this chapter we define a community as a cluster of similar community instances that are observable over time. The communities are detected by defining a similarity measure on community instances and applying a hierarchical edge betweenness clustering algorithm. To analyze the temporal dynamics we introduce a visualization technique that allows to detect structural changes in the evolution of communities. This approach is applied to a data set from an online student community. We report on results that show that we are able to provide insights into the evolution of communities and furthermore to identify triggers for structural changes.

A microscopic view on the evolution of communities is proposed in Chapter 4. This model can be used to analyze the internal structural changes of communities that occur over time. For this, we define a community as a dense subgraphs that evolves over time. At first, DENGGRAPH - a DENsity based GRAPH clustering algorithm - is proposed to extract dense subgraphs from social networks. The algorithm is extended to detect overlapping clusters, thus, taking into account that actors are often participating in several communities parallel instead of just one. Furthermore, an incremental version of DENGGRAPH is proposed which deals efficiently with changes in large networks (DENGGRAPH-IO<sup>11</sup>). Additionally, we introduce two measures to assess the quality of the obtained clusterings. The usability of the proposed approach and its characteristics are evaluated using two real world data sets.

The thesis closes with a summary of the contributions and examples for future work in Chapter 5. In the Appendix, additional material concerning the algorithm and the experimental results is added. In Appendix A the pseudocode for the proposed algorithms are presented. Furthermore, results of the experiments in the form of statistics are presented in Appendix B.

---

<sup>11</sup>“IO” stands for “Incremental” and “Overlapping”



## 2 Fundamentals

As discussed in the introduction, methods for the analysis of temporal changes in community structures are of interest in several application areas. In Chapter 3 and 4, two approaches are proposed and the underpinning concepts from social network analysis and data mining are discussed in this chapter. At first, the principles of social network analysis are discussed (Section 2.1). The observable network types in the WWW or in an organization are presented, followed by a discussion of network analysis methods to learn about their characteristics. Furthermore, we explore the concept “community” (Section 2.2). The section includes a discussion about structures on the group-level that are known in social network analysis and a working definition for the network structure “community”. Thereafter, clustering methods known from data mining and their applicability for community detection are discussed (Section 2.3). The evaluation of the obtained clusterings is an important task, therefore, the chapter closes with a presentation of quality measures for graph-based community detection methods (Section 2.4).

### 2.1 Social Network Analysis

Social networks represent relationships among social entities. Examples for often studied relationships are communications among members of a group or economic relations between corporations. Social network analysis focuses on the structural analysis of networks, for example, to explain social behavior. The methods are traditionally widely used in the social and behavioral sciences but nowadays also applied in economics, marketing and other areas as well. In this section, we present types of (social) networks and methods from social network analysis to study them.

#### 2.1.1 Types of Social Networks

In social network analysis, networks are categorized by the nature of the actor set and the properties of the relations between them (see, e.g., [161, 149, 66, 164, 31]). The *actors* or *entities* can be of a variety of types such as individuals, organizations or a collection or aggregate of people or organizations. An collection of people could be, for example, a group of students attending the same lecture and an aggregate of organizations could be for example a set of nation-states. The *relations* observed at the level of pairs of actors can be of different types as well:

- Individual evaluations are sentiments articulated by people such as friendship or trust (see, e.g., [161])<sup>1</sup>
- Transfer of tangibles such as buying water or lending books
- Transfer of intangibles such as transmitting information or giving advice
- Physical interactions between actors such as attending the same party

The *mode* of a network is defined as the number of types of entities (actors) on which the structural variable (relation), such as interactions of people or friendship between people, are measured. The structural variables are measured on pairs of entities. If these pairs of entities belong to a single set, the network is called a *relationship* or *one-mode network*. If the structural variables are measured on two sets of entities, we refer to it as an *affiliation* or *two-mode network* (see for example [161]). A two-mode network consists of two distinct sets of actors or of a set of actors and a set of events. In the first case, the network is called *dyadic network* and a link between two actors from different sets represents the relation between them. An example of such a network is a data set consisting of two sets of employees that work in different geographic locations. A link between the two sets could be the participation in a project.

A *relationship network* states a social relationship between actors. This relationship can be a co-authorship, a power relation within an organization, friendship or trust between people, collaborative relations, contractual relationships, etc. Many of these relationships are difficult to measure as they are usually not explicit and only part of an individual's knowledge (see, e.g., [161]). *Interaction networks* represent one type of relationship networks where the modeled relation could be an interaction between two actors representing, for example, a contact (e.g., via email or telephone) or a transmission (e.g., a disease). In interaction networks, often a transfer of "things" can be observed, such as information or resources. This particular form of interaction networks is called *transmission networks* where a tangible (gas, water, etc.) or intangible material (money, information, etc.) - which is generally measurable - is transmitted between nodes.

In contrast to relationship networks, *affiliation networks* are two-mode networks, traditionally representing a set of actors and a set of "social occasions" which define events. In [161], the authors stress that actors are linked through their joint participation in social events. However, one can also build affiliation networks based on relations that are not necessarily explicitly defined but obtained implicitly by observation (see, e.g., [153]).

For the experiments presented in Chapter 3 and 4, we use three different data sets and build relationship networks that represent a social relationship between individuals. The relationship is in one case the posting of messages in public guestbooks of students (*IKUS* data set, see Section 3.3.1), in the other the e-mail exchange between employees in a company (*ENRON* data set, see Section 4.4.1) and the similarity of users regarding their music listening behavior (*Last.fm* data set, see Section 4.5.1). The characteristics of the data sets are discussed in the respective chapters.

---

<sup>1</sup>Historically, these relations have been studied the most.

## 2.1.2 Analyzing Networks

In social sciences, the analysis of networks has a long tradition [67]. The main objective of social network analysis is to detect and interpret patterns of social relations among actors. However, recently the field also gained popularity in many research areas such as immunology, transportation systems, molecular biology, information systems, computer systems and organizational science to name a few. Even though the domain of application determines the appropriate form of analysis, the methods that are prevalent in network analysis can be distinguished by the level of analysis. Brandes and Erlebach [25] propose three levels of analysis: (i) element-level analysis, (ii) group-level analysis and (iii) network-level analysis. In this subsection, we first introduce the preliminaries and discuss afterwards the concepts of network analysis that will be used in this thesis, following the structure of the three levels of analysis.

### Preliminaries

*Networks* are usually modeled as *graphs*. A *graph*  $G = (V, E, \omega)$  is an object in which  $V$  denotes the set of *vertices* (nodes, actors),  $E$  the set of *edges* (links, relations) that connect pairs of vertices and  $\omega : E \rightarrow \mathbb{R}^+$  represents the strength of the relation modeled by the edges. The graphs we observe are *simple* (no parallel edges; each edge is contained in  $E$  only once), *undirected* (the vertices connected by an edge have no order), *weighted* (numerical values are assigned to edges) and *loop-free* (no edge joins a vertex to itself). An undirected edge connecting vertices  $u, v \in V$  is denoted by  $(u, v)$ . Each edge  $e \in E$  is assigned a *weight*  $\omega(e)$ . The weight quantifies the relationship between two nodes depending on the context. This could be the “number of messages exchanged” in case the relation quantifies the interaction strength based on the e-mail exchange between individuals or the “similarity of users” if the relation models for example the music listening preferences of individuals. If the relation does not allow a quantification, an edge weight of 1 is assumed if an edge exists.

A graph  $G' = (V', E')$  is a *subgraph* of graph  $G = (V, E)$  if  $V' \subseteq V$  and  $E' \subseteq E$ . The subgraph is called *induced subgraph*, if  $E'$  contains all edges  $e \in E$  that join vertices in  $V'$ .

A *walk* from  $v_1$  to  $v_k$  in an undirected graph  $G = (V, E)$  is an alternating sequence of vertices and edges  $v_1, e_1, v_2, e_2, v_3, \dots, e_{k-1}, v_k$ , where  $e_i = (v_i, v_{i+1})$ , in which each vertex is incident with the edges following and preceding it in the sequence. The *length* of the walk is defined by the number of edges in it.

A *path*  $p$  is a walk in which all vertices and all edges are distinct:  $e_i \neq e_j$  and  $v_i \neq v_j$  for  $i \neq j$ . A path  $p$  from  $u$  to  $v$  in a graph  $G = (V, E)$  is a *shortest path* or *geodesic* if its weight  $\omega(p)$  is the smallest possible among all paths from  $u$  to  $v$ , where  $\omega(p)$  is defined as the sum of all weights of the edges on  $p$ . The length  $d(u, v)$  of the shortest path is called the *shortest-path distance* or *geodesic distance*. If there is no shortest path between two vertices then the distance between them is infinite.

An undirected graph  $G = (V, E)$  is *connected* if there is a path between all pairs of vertices, i.e. if every vertex can be reached from every other vertex. Graphs that are not connected are called *disconnected*. A *connected component* of  $G = (V, E)$  is an induced subgraph  $G' = (V', E')$  that is maximal which means there is no connected subgraph

$G'' = (V'', E'')$  with  $V'' \supset V'$ . In other words, a connected component is a subgraph in which there is a path between all pairs of vertices and there is no path between a vertex in the component and any vertex not in the component.

### Element-Level Analysis

A fundamental question in element-level analysis is the relevance of a network element, i.e. of a vertex or an edge: “How important is this vertex in the network?” Typical measures to assess the “importance” of a vertex are based on its *centrality* in the network.

In the area of Web search, two prominent examples for indices to determine the centrality of a vertex in a network are the *PageRank* measure [27] and the *Hubs and Authorities* scores [103]. The *PageRank* index measures the centrality of a vertex in a directed Web graph in which Web documents are connected via hyperlinks. The assumption that underlies the PageRank measure is that a link corresponds to a recommendation for a relevant document and that referring documents are relevant themselves. In the model of Kleinberg [103], *authorities* are pages that are linked by many *hubs* and the score estimates the value of the content of the page. That is, a vertex is given a high authority score if many pages that are recognized as hubs link to it. Hubs are pages that link to many authorities and the score estimates the value of its links to other pages. That is, a vertex is given a high hub score if it links to vertices that are considered to be authorities. The hub and authority scores for a Web page are calculated with the Hyperlink-Induced Topic Search (HITS) algorithm [103]. The PageRank as well as the hub and authority scores are based on the linkage of the documents on the Web. However, in contrast to PageRank, the hub and authority scores are calculated for each query as they are topic dependent and vary therefore depending on the query.

The centrality of elements in social networks can be determined using different local statistics such as the in/out-degree or the clustering coefficient to assess the centrality of a vertex and the edge betweenness centrality to assess the importance of an edge in the network. These measures are introduced in the following.

**Local Statistics** Let  $\mathcal{G}$  be a class of graphs,  $P$  a set of parameters and  $Y$  a set of values. Furthermore, let  $X_G$  be a set of graph elements in  $G$ , which may consist of vertices, edges, subgraphs, paths, etc. A *local statistic*  $\lambda_G$  assigns a single value  $\lambda_G(x) \in Y$  to a certain graph element  $x$  of a given graph  $G \in \mathcal{G}$ . Examples are in/out-degree, edge weight, distance, clustering coefficient of a vertex and centrality measures.

- **Degree of a Vertex** The *degree* of a vertex  $v$  in an undirected graph  $G = (V, E)$ , denoted by  $deg(v)$ , is the number of edges in  $E$  that have  $v$  as a vertex. The degree of a vertex indicates its centrality in a graph. In a directed graph  $G = (V, E)$ , the *out-degree* of  $u \in V$ , denoted by  $deg^+(u)$  is the number of edges that have their origin in  $u$ . The *in-degree* of  $u \in V$ , denoted by  $deg^-(u)$ , is the number of edges that have their destination in  $u$ .
- **Clustering Coefficient of a Vertex** A question that is often asked in social network analysis is: What is the likelihood that two friends of mine know each other?

The *clustering coefficient of a vertex*  $c(v)$  is a measure that represents the likeliness that two adjacent vertices of  $v$  are connected [162]. It is obtained by dividing the number of actual links between friends of an actor by the number of links that they could have if they were all friends with each other. In [28], the clustering coefficient is defined in terms of triangles and triples. A *triangle*  $\Delta = \{V_\Delta, E_\Delta\}$  is a complete subgraph of  $G$  with exactly three nodes.  $\lambda(G)$  is the number of triangles in  $G$ .  $\lambda(v) = |\{\Delta \mid v \in V_\Delta\}|$  is the number of triangles of a node.

A *triple* is a subgraph with three vertices and two edges and the number of triples  $\tau(v)$  at a node  $v$  depends on its degree  $\text{deg}(v)$ :

$$\tau(v) = \frac{\text{deg}(v)^2 - \text{deg}(v)}{2} \quad (2.1)$$

The number of triples for the whole graph is

$$\tau(G) = \sum_{v \in V} \tau(v). \quad (2.2)$$

The *clustering coefficient of a vertex*  $v$  is defined as

$$c(v) = \frac{\lambda(v)}{\tau(v)} \quad (2.3)$$

A low clustering coefficient indicates that an actor's friends are rather loosely connected, whereas a highly clustered network has a high clustering coefficient. In our experiments we use, among others, the clustering coefficient of the graphs as an indicate for social network structure.

- **Betweenness Centrality of an Edge** The edge betweenness measure assesses the centrality of an edge in a network. It is calculated by summing up the number of geodesics that pass the observed edge.

The *edge betweenness* of an edge  $e$  in a graph  $G(V, E)$  is defined as the number of shortest paths along it. The proportion of shortest paths between  $s$  and  $t$  that pass edge  $e$  is defined as

$$\delta_{st}(e) = \frac{\sigma_{st}(e)}{\sigma_{st}}. \quad (2.4)$$

The *betweenness centrality*  $c_B(e)$  of an *edge*  $e$  is

$$c_B(e) = \sum_{s \in V} \sum_{t \in V} \delta_{st}(e). \quad (2.5)$$

Granovetter argues that the bridges between clusters, so-called *weak ties*, are responsible for the efficient diffusion of information [78]. This is because bridges connect different parts of a social networks, it is through these weak ties that diffusion will occur most efficiently. When we assume, that all communication is conducted along shortest paths,

the edge betweenness centrality can be interpreted as the probability that an edge  $e$  is involved in a communication between  $s$  and  $t$ .  $\delta_{st}(e)$  can be interpreted as the amount of communication that passes if one unit is sent from  $s$  to  $t$ .

The edge betweenness centrality measure is applied in Chapter 3 to detect communities in networks.

### Group-Level Analysis

A common observation in social networks is the separation of *groups* where people are connected via strong edges. For example in interaction networks, people forming groups interact more closely with each other than with other individuals in the networks. A group is characterized (among else) by strong relations among its members. The subgraph induced by this group has a higher connectivity between each pair of members inside the group compared to vertices outside the group. The starting point for all of these concepts is the idea of *cohesive subgraphs*. The most often encountered cohesive subgraph concepts are *cores* and *cliques* [161, 149].

Besides patterns of strong connections, a common question in social network analysis is to partition the group of actors into *positions* with the same or similar relational patterns. The research method is called *blockmodeling* [161].<sup>2</sup> Two vertices are *structurally equivalent* to one another if they occupy the same position in the network. In social network analysis it is argued that the formation of distinct social roles are central elements and that types of social relations are maintained by particular categories of actors. A similar approach to blockmodeling is proposed by Milo et al. [122]. The authors introduced the term *network motif* and define it as a connected subgraph that occurs more often than in a random network of same size and degree distribution.

In the area of graph mining, a similar approach is taken to extract frequent graph patterns such as frequent subgraphs. Common applications are, for example, in the area of biology, the detection of frequent subgraphs in chemical components (see, e.g., [19, 45]) or in protein structures (see, e.g., [91]).

Since the focus of this work is on group dynamics in social networks, we will elaborate on definitions for groups and methods to detect them in more detail in the next Section 2.2.

### Network-Level Analysis

The focus of an analysis on the network-level is on properties of the network as a whole. Network properties are first indicators to assess the similarity of networks and obtained network statistics might eventually reflect characteristic traits of networks, e.g., in a certain application domain. Furthermore, observed changes in the properties of a network might provided important indications for the interpretation of the temporal analysis. We therefore introduce global statistics and global distributions that are assigned to graphs which are used in the following to describe graph characteristics.

---

<sup>2</sup>The disjoint subsets of  $V$  are called *positions*. The *blockmodel* is thus a simplified version of  $G$  whose elements are the positions.

**Global Statistics** Let  $\mathcal{G}$  be a class of graphs,  $P$  a set of parameters and  $Y$  a set of values. Furthermore, let  $X_G$  be a set of graph elements in  $G$ , which may consist of vertices, edges, subgraphs, paths, etc. A *global statistic*  $\gamma$  assigns a single value  $\gamma_G \in Y$  to each graph  $G \in \Gamma$ . Examples are number of vertices/edges, diameter, density, clustering coefficient and mean geodesic distance of the graph.

- **Diameter** The *diameter* of a graph is the maximal distance between two vertices:

$$\text{diam}(G) := \max \{d(u, v) \mid u, v \in V\}, \quad (2.6)$$

where  $d(u, v)$  is the *distance* between two vertices defined as

$$d(u, v) = \min \{|P| \mid P \text{ is a path from } u \text{ to } v\} \quad (2.7)$$

- **Density** The *density* of a graph is the proportion of the number of edges present to the maximum possible. The density of a graph  $G$  is defined as

$$\Delta(G) = \frac{m}{n(n-1)/2}, \quad (2.8)$$

where  $n$  is the number of vertices,  $m$  the number of edges and  $n(n-1)/2$  is the maximum number of edges that can be present in an undirected graph.

- **Clustering Coefficient of a Graph** The *clustering coefficient of a graph*  $C(G)$  is the average clustering coefficient taken over all vertices.

$$C(G) = \frac{1}{|V'|} \sum_{v \in V'} c(v) \text{ with } V' = \{v \in V \mid \text{deg}(v) \geq 2\} \quad (2.9)$$

where  $c(v)$  is the clustering coefficient of node  $v$  as defined in Equation 2.3

- **Mean Geodesic Distance** The *mean geodesic distance*  $\bar{d}(G)$  of a graph is defined as

$$\bar{d}(G) = \frac{1}{\frac{1}{2}n(n+1)} \sum_{i \geq j} d(u, v) \quad (2.10)$$

Social networks often exhibit a high clustering coefficient. In combination with a short average path length, these networks are called “small world” networks (see, e.g., [121]).

**Global Distributions** Let  $\mathcal{G}$  be a class of graphs,  $P$  a set of parameters and  $Y$  a set of values. Furthermore, let  $X_G$  be a set of graph elements in  $G$ , which may consist of vertices, edges, subgraphs, paths, etc. A *global distribution*  $\Gamma$  is a mapping  $\Gamma_G : P \rightarrow Y$  for each graph  $G \in \mathcal{G}$ .

- **Average Degree** The *average degree* is denoted by  $\overline{\text{deg}}(G) = \frac{1}{|V|} \sum_{v \in V} \text{deg}(v)$ . In a directed graph  $G = (V, E)$ , the *out-degree* of  $u \in V$ , denoted by  $d^+(u)$  is the number of edges that have their origin in  $u$ . The *in-degree* of  $u \in V$ , denoted by  $d^-(u)$ , is the

number of edges that have their destination in  $u$ . Let  $p_k$  be the fraction of vertices in the graph that have degree  $k$ .  $p_k$  is then the probability that a randomly chosen vertex has degree  $k$ . A plot of  $p_k$  for any given network can be formed by making a histogram of the degrees of the vertices. The cumulative distribution function  $P_k = \sum_{k'=k}^{\infty} p_{k'}$  is the probability that the degree of a vertex is greater or equal to  $k$ .

Social networks in the Web are often characterized by a right-skewed in- and/or out-degree distribution. This indicates that these networks contain few highly interconnected nodes while the vast majority of nodes are only weakly connected (see, e.g., [113]).

## 2.2 Communities in Networks

In the following Section 2.2.1, we review definitions for dense subgroups which have been proposed in the area of social network analysis and discuss their applicability for our thesis. Furthermore, we present definitions for “communities” which are proposed, among others, in the field of computer science in Section 2.2.2. The section closes with a presentation of our view on the concept of “community”.

### 2.2.1 Subgroups in Social Network Analysis

Networks representing relationships between individuals usually exhibit group formations. Vertices and edges are not distributed uniformly but show locally dense groups. Typical reasons for the implicit or explicit formation of groups are for example common interests or goals, friendship or other similarities between actors. Even though, the notion of subgroups is widely used in social sciences, often no formal definition is given as it is assumed that everybody knows what a social group is, as noted by [65]. Several network researchers have studied methods for *cohesive subgroups* and attempted to formalize the notion of a *social group* (see, e.g., [4, 30, 142, 65]).

As discussed in the previous section, in social sciences, two major perspectives are prevalent to describe the characteristics of a social group in a relationship network: (i) strong relations among its members and (ii) similar relations among members. Thus, a subgroup  $SG \subseteq V$  in a social network is formalized in [161] by the *cohesion* among a subset of actors, where the cohesion is determined based on specific properties of the edges among subsets of actors.

**Definition 2.1** *The cohesion of a subgroup  $SG$  is defined as*

$$\text{cohesion}(SG) = \frac{\sum_{e \in E(SG)} \omega(e)}{n_{SG}(n_{SG}-1)}, \quad (2.11)$$

$$\frac{\sum_{e \in \overline{E}(SG)} \omega(e)}{n_{SG}(n-n_{SG})}$$

where  $n$  is the number of vertices in the network,  $n_{SG}$  is the number of vertices in the subgroup  $SG$ ,  $E(SG)$  is the set of edges in the subgroup  $SG$ ,  $\overline{E}(SG)$  is the set of vertices with origin in  $SG$  and destination outside the subgroup and  $\omega(e)$  is the edge weight.

The numerator of the ratio is the average strength of edges within the subgroup and a measure for the cohesiveness of the subgroup. The denominator is the average strength of the edges from subgroup members to outsiders and a measure for the sparsity of edges outside the subgroup. If the ratio is close to 1, the strength within the subgroup does not differ significantly from the strength of the edges outside the subgroup. If the ratio is greater than 1, the edges inside the subgroup are stronger than the edges outside.

According to Wasserman and Faust [161], four general properties of cohesive subgroups have influenced the formalization of the concept in social network analysis. In the following, we briefly discuss the four properties and present afterwards examples for formalizations.

- **Mutuality of ties:** If all pairs of subgroup members are adjacent, the subgroup is called a *clique*. A clique structure is only rarely observable in real networks. Thus, relaxations for cliques such as *k-plexes* and *k-cores* have been proposed.
- **Reachability:** All subgroup members are reachable by each other. Subgroup members are not necessarily directly connected but if they are not adjacent, then the paths between them are relatively short, i.e. the geodesic distances among members are small. Restricting the length of the geodesic distance is the basis for the definition of an *k-clique*. As the definition of *k-cliques* for a subgroup may be seen as too loose, further restrictions for the geodesic path can be applied by requiring a maximal diameter for the subgraph. *k-clubs* and *k-clans* are possible definitions that possess the desired restriction.
- **Separation:** Subgroups based on separation consist of a subset of members which have numerous ties among members inside the group. Examples are *k-plexes* and *k-cores*. Unlike the clique definition which requires all members inside the group to be adjacent to all other subgroup members, these relaxations require all members to be adjacent to some minimum number of other subgroup members.
- **Density:** Subgroups based on a relatively high density inside the subgroup compared to the rest of the network. Unlike the other three properties for subgroups that focus on properties of ties within the subgroup, this definition incorporates the idea that a subgroup has strong, frequent and dense ties inside the group but relatively weak, infrequent and sparse ties outside the subgroup.

Authors studying social groups, often look at *structural properties* of subsets of actors in the network. Their notion of a subgroup is formalized by properties that members of a subgroup fulfill. In contrast, subgroups can be formalized based on *statistical properties*. Here, the requirements for a subgroup are not based on properties that all members of a group must fulfill but the respective measure is calculated as an average over all members. Furthermore, we can distinguish *local measures* and *global measures* that are used to assess subgroup structures.

In general, it is more difficult to use statistical measures than structural in order to detect subgroups, as will be explained later, but they are more flexible since they do not impose a requirement on each individual. Therefore, statistical measures are preferable

for subgroup detection however, due to size and the characteristics of the analyzed data sets, one major concern of current research is to propose efficient measures to cope with large networks.

### Subgroup Detection Based on Local Information

A graph structure that can be determined using local information is a *clique*; a maximal complete subgraph consisting of three or more vertices. In a clique all members have ties with each other member.

**Definition 2.2** *Let  $G = (V, E)$  be an undirected graph, the subset  $U \subseteq V$  is a clique if and only if  $G[U]$  is a complete graph.*

The structural properties of a clique are strong: it is perfectly dense, perfectly compact (the diameter of a clique is 1) and perfectly connected. All members of the group have to fulfill the same requirements to be a member of the group. Since the requirements are very restrictive it is not likely to observe these structures in real networks very often.

Therefore, several relaxations of the clique concept have been proposed such as *k-plexes* and *k-cores*. A *k-plex* is a group in which each actor is connected to all but *k* of the other actors.

**Definition 2.3** *Let  $G = (V, E)$  be an undirected graph and  $k \in \mathbb{N}$ . A subset  $U \subseteq V$  is a *k-plex* if and only if the minimum degree  $\delta(G[U]) \geq |U| - k$ .*

A *k-core* is a subgraph in which each member is adjacent to at least a minimum number of *k* other members. In contrast to a *k-plex* which specifies the maximal number of connections that may be absent from the subgraph, a *k-core* specifies the minimum of connections that must be present in the subgraph:

**Definition 2.4** *Let  $G = (V, E)$  be an undirected graph and  $k \in \mathbb{N}$ . A subset  $U \subseteq V$  is a *k-core* if and only if the minimum degree  $\delta(G[U]) \geq k$ .*

The presented concepts of subgroups are (i) cliques which are based on cohesiveness and (ii) relaxations of a clique based on a reduction of structural requirements. But relaxations based on structural densities (such as *k-plexes* and *k-cores*) still require that all members of a certain group fulfill the requirements. In contrast, by applying statistical relaxations, a property for being a group must not be satisfied by all members but is averaged over all members. At this point it become obvious that the calculation of statistical measures is computational more complex than the verification of structural properties as the averages of the respective property must be calculated over all combinations of possible subgroups.

Using a local statistical property to determine subgroup structures means to assign a value to a certain graph element (such as an edge or a vertex). Typical local statistical measures are in/out-degree, edge weight and clustering coefficient of a vertex. A detailed review on structural and statistical local density measures is given in [106].

In Chapter 4 we propose a method to find dense subgroups which is based on a local property: The number of neighbors of an actor that lie in a predefined neighborhood. This measure corresponds basically to the degree of a vertex when only neighbors are

considered that are close enough, while closeness is determined using two parameters. If the neighborhood is dense enough, a cluster is build. The density-based cluster algorithm DENGRAPH is described in detail in Chapter 4.

### Subgroup Detection Based on Global Information

The approaches discussed so far are based on properties of vertices within the subgroup. As we will see in the following Section 2.2.2 in more detail, subgroups are intuitively defined by a relative strength, closeness or similarity within the group, but also by a relative weakness or sparseness of edges to vertices outside the group. Thus, both aspects should be considered when determining subgroups in graphs. Borgatti et al. use the notion of *edge connectivity* between vertices to define subgroups (*Lambda Sets*) [17]. The edge connectivity is the minimal number of edges that must be removed from the graph in order to disconnect the two vertices. The smaller the value, the more likely is a disconnection of the vertices after the random removal of an edge. In contrast to the properties demanded for the types of subgroups discussed above, for lambda sets no requirements regarding the cohesiveness in terms of adjacency or geodesic distance are imposed since there is no restriction on the length of the paths between vertices in the set.

The detection based on global information relies on measures that can only be assessed by considering the whole graph structure. Other examples for global measures are the various centrality measures such as edge or vertex betweenness centrality which are calculated based on the shortest paths between all pairs of nodes (see, e.g. [64]). In Chapter 3, we apply a hierarchical clustering procedure based on edge betweenness centrality to determine subgroups in social networks.

## 2.2.2 Real-World and Virtual Communities

There are various notions of what constitutes a community. According to the Oxford English Dictionary, the term *community* has its origin in the Old French word *comunete* which again has its origin in the Latin word *communis* meaning “fellowship, community of relations or feelings”. In medieval Latin it was, like *universitas*, used concretely in the sense of “a body of fellows or fellow-townsmen”.

### Definition of the Term “Community”

The Merriam-Webster Online Dictionary offers several definitions for the term “community” such as “the people with common interests living in a particular area”, “an interacting population of various kinds of individuals (as species) in a common location”. These examples stress two important and typical characteristics: (i) common interest and (ii) interacting individuals.<sup>3</sup>

Hillery [87] found through a meta-analysis of 94 definitions of the term community that 69 definitions included the following aspects: social interaction, common ties and a geographic criterion as characteristics of the concept. The geographic aspect attributes

---

<sup>3</sup><http://www.merriam-webster.com/dictionary/community>, as of March 9th, 2009

to the fact that it has been observed that people's lives were intertwined with the local institutions and that a complex network of people, institutions, shared interests, locality and a sense of psychological "belonging" had been identified. The community concept originates from this observation [160]. In other words, people come together, physically, or by other means, because they have something in common, a shared purpose, and this results in shared activities and social interactions.

Depending on the context, the noun community has several meanings such as:

- a group of people whose members reside in a specific locality, share government, and often have a common cultural and historical heritage (see, e.g., [163])
- a locality inhabited by such a group (see, e.g., [34, 111])
- a group of people with a common religion, race, or profession (e.g., the scientific community [126])
- a group of people sharing certain attitudes and interests in common (community of interest) (see, e.g., [159, 94, 137])
- a group of people who share a concern, a set of problems or a passion about a topic and who deepen their knowledge and expertise by interacting on an ongoing basis (community of practice) (see, e.g., [166])
- a group of associated nations sharing common interests or a common heritage (e.g., the Western European Community [21])
- a group of interdependent plants or animals growing or living together or occupying a specified habitat (see, e.g., [107])

### Communities in the Virtual World

Even though originally the term community was used for groups of individuals living or interacting in a physical environment, it is nowadays often also applied to groups of people interacting virtually. In the beginning of this transition, authors used the term *online community* or any of its synonyms such as *virtual community*, *mediated community*, *electronic community*, *cyber community* or *Internet community* to distinguish from physical communities.

Howard Rheingold is credited with inventing the term *virtual community*. In 1993, he published a book called "The Virtual Community" [139] in which he describes the cultural and political implications of a new communication medium - a computer conferencing system called the WELL (Whole Earth 'Lectronic Link). The WELL was founded in 1985 and its goal was to attract interested people into online conversation with each other. In his book he defined a *virtual community* as follows: "Virtual communities are social aggregations that emerge from the *Net* when enough people carry on those public discussions long enough, with sufficient human feeling, to form webs of personal relationships in cyberspace." [139]. Rheingold uses the term *Net* for loosely interconnected computer networks that use computer-mediated communication technology to link people

around the world into discussions. He was one of the first people, if not the first, discussing his own experiences and observations with relationships being build virtually.

Later on, Wellman and Gulia elaborated in [165] on whether on-line relationships are comparable to those established in “real life”. One out of seven questions they discuss is “Are Virtual Communities *Real* Communities” and answer the question by putting together observations and anecdotes. The authors assess that the *Net* maintains strong, supportive community ties especially between people who are physically dispersed. They notice that online relationships are based more on shared interest than on shared social characteristics. An interesting observation is that while online relationships may be specialized, a virtual community as a set of ties, provides a wide range of support. Even though the question remains in the end unanswered, besides particularities that have been observed, in several aspects, virtual communities obviously have strong similarities with “real world” communities.

A precise definition of what a “community” really is does not exist yet but the usage of the term “online community” does not differ significantly from that of a physical community. More recently, Preece described an online community as a structure that consists of “people who interact socially as they strive to satisfy their own needs or perform special roles; a shared purpose that provides a reason for the community; policies that guide people’s interactions; and computer systems to support and mediate social interaction and facilitate a sense of togetherness” [137].

Preece accommodates the lack of a specific geographical area by expanding the community definition with a technological communication infrastructure capable of enabling and supporting the community. Since the technological support to facilitate the social interaction is indispensable for online communities, the community provider must offer an appropriate infrastructure whose features will depend on the type of community, its (prospective) participants and its purpose.

In the computer science “community”, one of the most widely accepted and used definition is that given by Newman and Girvan in [128]: “A community is a subgraph containing nodes which are more densely linked to each other than to the rest of the graph or, equivalently, a graph has a community structure if the number of links into any subgraph is higher than the number of links between those subgraphs.” The authors assume that the links between actors are the representation of the people’s interaction. This definition goes back to the description of a cohesive subgroup by Seidman who defined cohesive subgroups as “...sets of individuals more closely tied to each other than to outsiders” [150].

In computer science the term community is also often applied to networks that represent “interactions” between computers or Web sites. For example, Kumar et al. view communities as a dense bipartite subgraph of the Web graph built upon hyperlinks [110].<sup>4</sup> Gibson et al. analyze the link topology of the WWW and view communities as structures consisting of a core of central “authoritative” pages linked together by “hub” pages [72]. Flake et al. define a Web community as a set of Web pages that link (in either direction) to more Web pages in the community than to pages outside of the community [61]. Adamic and Adar

---

<sup>4</sup>A bipartite graph is a graph whose set of vertices can be split into two subsets A and B in such a way that each edge connects a vertex in A and a vertex in B.

use hyperlinks and information on a homepage as reflections of social interactions a user has in the real world [2]. The information such as common in- or out-links is used to predict relationships between individuals. Park follows a similar approach and interprets the social or communication structure among social actors based on the hyperlink structure among Web sites [134]. Even though the detection of communities was not the main aim, the application of social network analysis methods to study hyperlink networks was proposed already in the late nineties by Jackson [95]. The idea was to analyze the communication structure between individuals based on the hyperlink network in the WWW.

As we have seen, the term community is used in many disciplines and the definitive driver for a community is that all individuals have something in common. We also observe that the assessment whether this communality is fulfilled is done based on a static view of a graph. In the presented approaches, the observer determines whether a community exists or not only based on a snapshot of interactions taken at a certain point in time. An aspect that is rarely discussed in the literature is how a community relates to temporal aspects.

In this thesis, this traditional static view on communities is extended to capture temporal evolution. We investigate communities over time taking two perspectives which allow us to capture different aspects of community dynamics:

1. We define a community as a group of individuals observed at different time points.
2. We define a community as a group of individuals that evolves over time.

A community mining approach to detect communities as objects that can be observed over time is presented in Chapter 3. There, a community is defined as a constellation that is observable over time. This constellation consists of sets of similar subgroups obtained by a hierarchical clustering procedure. In Chapter 4, an approach to detect and observe communities defined as groups of people that evolve over time is discussed.

## 2.3 Graph Clustering Techniques

In the research field of data mining, *clustering* techniques are known that partition a set of data in a set of meaningful sub-classes, called *clusters*. A *cluster*  $C$  is a subset  $C \subseteq D$  of data points. Each cluster represents a collection of data objects that are similar to one another and are thus regarded as a group. The entire collection of clusters is referred to as a *clustering*  $\zeta = \{C_1, \dots, C_t\}$  (see, e.g., [156]).

Since the aim of clustering corresponds to our goal to detect groups of “similar” or “close” actors we discuss in the following clustering techniques and their applicability for community detection.<sup>5</sup>

To cluster similar data points into groups a measure of *similarity* or *dissimilarity* is needed. The similarity between two points is a numerical measure of the degree to which both objects are alike. Accordingly, dissimilarity is a numerical measure of the degree

---

<sup>5</sup>For a more detailed discussion about the various other data mining techniques refer to, e.g., [156, 168, 81, 16, 82].

to which both objects are different. The more alike the objects are, the higher is the similarity and the less similar the points are the higher the dissimilarity. Usually, when speaking of dissimilarity, the term *distance* is used.

A distance  $dist(x, y)$  between two points  $x$  and  $y$ , for example the Euclidean distance, fulfills the following properties (cf., e.g., [156]):

- $dist(x, y) \geq 0$  for all  $x$  and  $y$
- $dist(x, y) = 0$  only if  $x = y$
- $dist(x, y) = d(y, x)$  for all  $x$  and  $y$  (Symmetry)
- $dist(x, z) \leq dist(x, y) + d(y, z)$  (Triangle Inequality)

Distance measures that satisfy all properties are called *metrics*. According to [156], these properties can be useful for the application, for example if the triangle inequality holds, a clustering can be performed more efficiently. Each type of clustering presented in the following, can be adapted to graph structures since they are based on a more general distance or similarity measure. We will discuss their applicability briefly and refer to subsequent chapters in which the methods are presented in more detail.

### 2.3.1 Partitioning Methods

A partitional clustering is a method to divide data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset.

The k-means algorithm is a prominent example for a partitioning clustering approach [116]. It assigns each point to the cluster whose center (or centroid) is nearest. Each cluster is associated with a centroid (center point). The centroid is the point generated by computing the arithmetic mean for each dimension separately for all points in the cluster. The number of clusters  $k$  must be specified. The basic structure of the clustering procedure is shown in Algorithm 2.1.

---

**Algorithm 2.1:** Partitional Clustering: k-means

---

```

1 begin
2   Randomly select  $k$  points as the initial centroids
3   repeat
4     Assign each point to the nearest cluster center
5     Recompute the new cluster centers
6   until (Some convergence criterion is met (usually until the centroids not not change))
7 end

```

---

The main advantages of this algorithm are its simplicity and speed, which allows it to run on large datasets. Due to its random initialization it does not yield the same result with each run of the algorithm. Rather, the resulting clusters depend on the initial assignments. The k-means algorithm maximizes inter-cluster (or minimizes intra-cluster)

variance, but does not ensure that the solution given is not a local minimum of variance. To ensure that the optimal partitioning is computed, all clustering combinations need to be calculated. Thus, heuristics are used to determine the best partitioning. The algorithm is known to be good for spheric clusters but has problem when the data has outliers or when the clusters have different sizes or densities. Derivative algorithms have been proposed such as k-medians [129] and CLARANS [130] to overcome some of these problems.

The idea of partitional clustering can also be applied to graph structures. The approach is called “Graph Partitioning”: The graph is partitioned into a predetermined number of groups where each actor in the network is assigned to one and only one group. The goal is to minimize the number of edges between partitions and on the other hand to balance the edges weight or the number of nodes over all partitions (see, e.g., [52, 97]). The number of groups has to be determined a-priori.

A graph partitioning proposed by Flake et al. is based on minimum cuts within the graph [61]. Given is a graph  $G = (V, E)$ , with non-negative edge capacities  $c(u, v)$ , and two vertices  $s, t \in V$ . The main idea is to find the maximum flow that can be routed from the source  $s$  to the sink  $t$  that obeys all capacity constraints. A *cut*  $(S, T)$  of the network  $G$  is a partition of  $V$  into  $S$  and  $T = V \setminus S$  such that  $s \in S$  and  $t \in T$ . A *minimum cut* of a network is a cut whose capacity is minimum over all cuts of the network. The max-flow-min-cut theorem of Ford and Fulkerson proves that the maximum flow of the network is identical to the minimum cut that separates  $s$  and  $t$  [62]. The main idea behind maximum flow clustering techniques, is to create clusters that have small inter-cluster cuts and relatively large intra-cluster cuts to reach a strong connectedness within the clusters. Note that clusters are not allowed to overlap, which is problematic for domains in which it seems more natural to assign a vertex to multiple cluster. Its applicability to social networks is furthermore limited as (i) the number of communities is not known a-priori and (ii) many noise nodes are “forced” in partitions.

### 2.3.2 Hierarchical Methods

A hierarchical clustering produces a set of nested clusters organized as a hierarchical tree. The result can be visualized as a dendrogram; a tree like diagram that records the sequences of merges or splits (see, e.g., [156]). The strength of hierarchical methods is that the number of clusters must not be known beforehand. Any desired number of clusters can be obtained by “cutting” the dendrogram at the proper level.

Two main types of hierarchical clustering are *agglomerative* and *divisive* methods. Agglomerative algorithms start with the points as individual clusters and build the hierarchy from the individual elements by progressively merging the closest pair of clusters until only one cluster is left (see Algorithm 2.2). The key operation of the algorithm is the computation of the proximity of two clusters. Therefore, approaches differ in the way they define the distance between clusters.

*Chameleon* is a agglomerative graph clustering technique [101]. The basic idea is that two clusters are only merged if the similarity of the resulting cluster to the two original clusters is high. The algorithm consists of three steps. The first step is to generate a

---

**Algorithm 2.2:** Hierarchical Agglomerative Clustering

---

```

1 begin
2   | Compute the proximity matrix
3   | Let each data point be a cluster
4   | repeat
5   |   | Merge the two closest clusters
6   |   | Update the proximity matrix
7   | until (Only a single cluster remains)
8 end

```

---

$k$ -nearest neighbor graph, i.e. a graph is built which contains only edges between vertices and its  $k$ -nearest neighbors. By this, the size of the graph can be significantly reduced. Chameleon starts then to bisect the largest current subgraph until no cluster has more than a predefined number of vertices. Afterwards, the partitions are merged to clusters that best preserve the cluster self-similarity until no more clusters can be merged. The process has some drawbacks: (i) Besides the value  $k$  also the minimal size of the cluster must be determined, (ii) for high-dimensional data the time complexity for determining the  $k$ -nearest neighbor graph is  $\mathcal{O}(m^2)$ , where  $m$  is the number of edges, (iii) the process does not remove noise objects, but assigns them to clusters (see, e.g., [156]).

Divisive algorithms start with one cluster and split at each step a cluster until all clusters contain only one point. The procedure works as shown in Algorithm 2.3.

---

**Algorithm 2.3:** Hierarchical Divisive Clustering

---

```

1 begin
2   | Compute the distance matrix
3   | Put all data points in one cluster
4   | repeat
5   |   | Choose a cluster to split
6   |   | Replace the chosen cluster with sub-clusters
7   | until (Each data point is one cluster)
8 end

```

---

A widely used divisive hierarchical graph clustering technique is the so-called edge betweenness clustering algorithm proposed by Girvan and Newman [73]. The underlying assumption of their approach is that when a graph is made of tightly bound clusters which are loosely interconnected, all shortest paths between clusters have to go through the few inter-cluster connections. Thus inter-cluster edges have a high edge betweenness (cf. Definition 2.5) and these edges are candidates for inter-cluster bridges. We discuss the algorithm in Chapter 3 in more detail and discuss how we apply this algorithm to detect communities as clusters that are observable over time.

### 2.3.3 Density-based Methods

In density-based clustering, a cluster is a dense region of points that is surrounded by a region of low density. This definition of a cluster is applicable when the clusters have an irregular shape and when noise points are present. Ester et al. proposed DBSCAN, a density-based algorithm that produces a partitional clustering [50]. DBSCAN stands for “Density-Based Spatial Clustering of Applications with Noise”. For DBSCAN, a cluster is a continuous area of arbitrary shape that is more dense than its surroundings. To capture this into a cluster, DBSCAN scans the data points in the dataset and computes *neighborhoods*. A neighborhood has a given radius and must contain a minimum number of points. A data point that has such a neighborhood around it is termed a *core point*. A data point that has no such a neighborhood is a *noise point*, unless it is itself located in the neighborhood of a core point; then, it is a *border point*. The two thresholds, the radius and the minimum number of points ensure that neighborhoods are dense areas. DBSCAN builds clusters by connecting adjacent neighborhoods. If a sub-area is not dense enough to form a neighborhood it is not considered as a cluster. DBSCAN is known to have problems, if different clusters have very different densities or if they form a hierarchy (see, e.g, [156]).

Another connectivity based clustering algorithm named OPTICS was introduced by Ankerst et al. [7]. The purpose of this cluster analysis is to create an ordering of the data point with respect to its density-based clustering structure. Thus, the algorithm does not produce a clustering of a data set explicitly but the cluster-ordering contains information which is equivalent to the density-based clustering corresponding to a broad range of parameter settings. The procedure groups points by their density connectivity and build hierarchies of clusters.

Hinneburg and Keim introduce a density-based clustering algorithm called DENCLUE [88]. The density of a set of points is modeled as the sum of *influence function* associated with each point. An *influence function* describes the impact of a data point within its neighborhood. The overall *density function* of the data space can be calculated as the sum of the influence function of all data points. Clusters can be determined by identifying local maxima of the overall density function. DENCLUE shares many of the strengths and weaknesses of other density-based algorithms: It efficiently deals with noise points and finds clusters of arbitrary shapes and sizes but has problems with high-dimensional data and data with varying densities.

In [49], Ester et al. extended DBSCAN to an incremental algorithm to handle dynamic data sets. The motivation for this work was to apply DBSCAN to large data sets that are regularly updated such as data warehouses in organizations. Due to the size of these data sets and the high number of updates, the aim is to perform these updates incrementally. IncrementalDBSCAN considers *insertions* of new objects and *deletions* of old objects and identifies the neighborhood of the object that is affected by the update. A neighborhood may have more than the minimal number of points in it after the arrival of the new object and a new cluster is created. Accordingly, after the deletion of an object, a neighborhood may have less than the minimal number of objects and a cluster is removed. These local effects imply that clusters may grow, shrink, merge or split. IncrementalDBSCAN

inherits the robustness of DBSCAN towards noisy data and is quite effective in adjusting the clusters.

The density-based clustering algorithms discussed so far have been designed for spatial data. For the proposed incremental graph mining algorithm we transfer the idea of density-based incremental clustering to social network (graph) structures. The procedure is described in detail in Chapter 4.

## 2.4 Evaluating Graph Clusters

Clusters are “meaningful” if they reflect the natural structure of the data set. The quality of a clustering therefore depends on how good the natural cluster structure of the data is captured by the chosen similarity measure. The evaluation of cluster detection methods depends on the definition of what actually constitutes a cluster. In the following, we present indices to determine the resilience of clusters and the quality of clusterings.

### Preliminaries

Let  $G = (V, E)$  be an undirected graph. A cluster  $C_i \subseteq V$  is a non-empty subset of vertices. A *clustering*  $\zeta = \{C_1, \dots, C_k\}$  of  $G$  is a partition of all vertices into clusters  $C_i$ . The set  $E(C_i, C_j)$  consists of all edges that have their origin in  $C_i$  and their destination in  $C_j$ .  $E(C_i)$  is the set of edges that have their origin and destination in  $C_i$ .  $E(\zeta) := \bigcup_{i=1}^k E(C_i)$  is the set of *intra-cluster edges* and  $\overline{E(\zeta)} := E \setminus E(\zeta)$  the set of *inter-cluster edges*. The number of intra-cluster edges is denoted by  $m(C)$  and the number of inter-cluster edges by  $\overline{m}(C)$ . A cluster  $C_i$  is identified with the induced subgraph of  $G$ , i.e., the graph  $G[C_i] := (C_i, E(C_i))$ .

### 2.4.1 Resilience Measure for Graph Clusters

Najjar and Gaudiot [125] propose a measure to assess the robustness of a network to disconnection if network components are removed.<sup>6</sup>

**Definition 2.5** *The disconnection probability of a network is defined as*

$$P(G[C_i], j) = \Pr[G[C_i] \text{ is disconnected exactly after the removal of the } j\text{-th edge}]. \quad (2.12)$$

The larger the connectivity of a network in terms of the vertex degree  $k$ , the more removals are needed until a disconnection occurs.

<sup>6</sup>The authors refer to computer networks and propose a measure of network resilience as a probabilistic measure of network fault tolerance. This is used to evaluate the effects of the disconnection probability on the reliability of a computer system.

**Definition 2.6** Let  $G[C_i]$  be an undirected subgraph induced by the cluster  $C_i$  with  $n$  edges. The network resilience is the largest number of edges removed such that  $G[C_i]$  is still connected with probability  $1 - p$ . The relative network resilience relates  $\text{clusres}(G[C_i], p)$  to the size of  $G[C_i]$ :

$$\overline{\text{clusres}}(G[C_i], p) = \frac{\text{clusres}(G[C_i], p)}{n} \quad (2.13)$$

The network resilience is based on the idea of connectivity, thus, as soon as one vertex is separated, the network is considered as unconnected. Since a disconnection of a vertex must not necessarily result in a dissolution of cluster, we introduce in Section 4.3 a measure to assess the stability of a graph cluster to perturbations of the input graph.

## 2.4.2 Quality Measures for Graph Clusterings

In the data mining literature, evaluation or validation measures for clusterings are traditionally classified into the following three types (see, for example, [156]):

**Unsupervised** Unsupervised quality measures assess the “goodness” of a clustering structure without considering any external information. In data mining, unsupervised measures for cluster validity are often further divided into two classes: (i) Measures of *cluster cohesion*, which determine how closely related the objects in the cluster are and (ii) measures of *cluster separation*, which determine how distinct or well separated a cluster is from other clusters. Unsupervised measures are often called *internal indices* because they use only information that is contained in the data set.

**Supervised** Supervised measures assess to what extent the discovered clustering structure matches some external structure. An example of a supervised index is the entropy, which measures, e.g., how well cluster labels match externally supplied class labels. Supervised measures are often called *external indices* because they use information not present in the data set.

**Relative** Relative measures compare different clusterings or clusters. A relative cluster evaluation measure is a supervised or unsupervised evaluation measure that is used for the purpose of comparison.

To assess the applicability of cluster evaluation measures, we first have to define what characteristic a desirable clustering has. As discussed in Section 2.2.2, we define a cluster as a group that is internally dense and that is only sparsely connected to the rest of the graph. A “good” clustering is therefore a partition of a graph where the members inside a group are densely connected (*intra-cluster density*) but only sparsely connected with members from other groups (*inter-cluster sparsity*). Given the discussed characteristics, we specify the following requirements of a quality measure. The measure should

- reward internal edges,

- penalize missing edges in the same cluster,
- penalize inter-cluster edges and
- reward missing inter-cluster edges.

Secondly, to determine the type of quality measure (unsupervised or supervised), we have to assess whether only internal information is available or also external. For example, the data sets we analyze in Chapters 3 and 4 do not contain any information that could be used to evaluate the proposed clustering procedure, therefore only unsupervised measures are applicable. However, besides assessing whether the demanded cluster characteristics are fulfilled by using unsupervised clustering measures, we present in Chapter 4 an application where cluster labels can be extracted and also supervised measures can be applied to compare clusters based on label information.

### Unsupervised Cluster Quality Measures

In [70], three unsupervised indices are proposed that are designed to evaluate clusters as subgraphs: (i) coverage, (ii) conductance and (iii) performance. The range of the indices is normalized to the unit interval, where zero indicates worst structural behavior and one means best possible structural behavior. Following the argumentation of intra-cluster density and inter-cluster sparsity two functions  $f, g : A(G) \rightarrow \mathbb{R}_0^+$  are defined, where  $f$  measures the density inside the clusters and  $g$  the sparsity between clusters and  $A(G)$  is the set of all possible clusterings. The functions are then combined in the following way

$$index(\zeta) := \frac{f(\zeta) + g(\zeta)}{\max \{f(\zeta') + g(\zeta') : \zeta' \in A(G)\}}. \quad (2.14)$$

#### Coverage

The *coverage*  $\kappa(\zeta)$  measures the weight of intra-cluster edges, compared to the weight of all edges [70].

**Definition 2.7** *The coverage  $\kappa(\zeta)$  is defined as*

$$\kappa(\zeta) := \frac{\sum_{e \in E(\zeta)} \omega(e)}{\sum_{e \in E} \omega(e)} \quad (2.15)$$

Since the coverage measures only the accumulated density within the clusters, it is possible that an individual cluster is sparse or that the number of inter-cluster edges is large. A clustering has a coverage with value 1 if the inter-cluster set is empty, i.e., if  $\zeta = \{V\}$  or if all edges in the inter-cluster set have the weight 0. Because of these characteristics, coverage should not be used as the only quality measure of a clustering.

## Conductance

Besides a high connectivity inside a cluster, a good clustering is characterized by a low degree of connectivity between two clusters. The *conductance* compares the weight of the *cut* with the edge weight in either of the two induced subgraphs. A *cut*  $\zeta'$  is a partition of a vertex set  $V$  of a graph  $G$  in two non-empty subsets  $(C'_1, C'_2)$ , i.e.,  $C'_2 = V \setminus C'_1$ . The weight of the cut is the sum of the weights of the edges with one endpoint in each of the two disjoint vertex sets  $C'_1$  and  $C'_2$ :  $\omega(C'_1, V \setminus C'_1)$ . The *conductance weight*  $a(C'_1)$  of a cut side is defined in [70] as

$$a(C'_1) := \sum_{(u,v) \in E(C'_1, V)} \omega(u, v) \quad (2.16)$$

**Definition 2.8** *The conductance  $\phi(\zeta')$  of a clustering is defined as*

$$\phi(\zeta') := \begin{cases} 1, & \text{if } C'_1 \in \{\emptyset, V\} \\ 0, & \text{if } C'_1 \notin \{\emptyset, V\}, \omega(\overline{E(\zeta)}) = 0 \\ \frac{\omega(\overline{E(\zeta)})}{\min(a(C'_1), a(C'_2))}, & \text{otherwise} \end{cases} \quad (2.17)$$

*The conductance of the graph  $G$  is the minimum conductance over all the possible cuts:*

$$\phi(G) = \min_{C_1 \subseteq V} \phi((C_1, V \setminus C_1)) \quad (2.18)$$

## Performance

The *performance* index combines two functions for the density measure  $f$  and the sparsity measure  $g$ . Observed are pairs of nodes. A pair of nodes may increase the density of a cluster if both nodes are in one cluster and are connected by an edge (counted by  $f$ ) or it increases the sparsity of inter-cluster edges if both nodes lie in different clusters and are not connected by an edge (counted by  $g$ ). In both situations the paradigm of intra-cluster density and inter-cluster sparsity is supported.

$$f(\zeta) := \sum_{i=1}^k \omega(E(C_i)) \quad (2.19)$$

and

$$g(\zeta) := \left( \sum_{u,v \in V} M \cdot [(u, v) \notin E] \cdot [u \in C_i, v \in C_j, i \neq j] \right) + \underbrace{M \cdot |E(\zeta)| - \omega(E(\zeta))}_{g_\omega(\zeta)} \quad (2.20)$$

The definition of  $g(\zeta)$  is given, as used in [70], in Iverson notation. The statement in the squared brackets is interpreted as a logical statement. The term is 1 if the statement is true and 0 if it is false. By this, all non-existent inter-cluster edges are counted. The additional term  $g_\omega(\zeta)$  corresponds to the weight difference that would be counted if no inter-cluster edges were present and the weight that is assigned to the actual inter-cluster

edges ( $\overline{E(\zeta)}$  the set of *inter-cluster edges*). By this we achieve that inter-cluster edges with low weights are punished to a lesser extent than edges with high weights and high similarity. It is assumed that all edges have a meaningful edge weight of  $M$ . This value can not be predetermined but depends on the input graph. In our case, the maximum edge weight is  $M = 1$  as edge weight are normalized to  $0 < w(e) \leq 1$ .

**Definition 2.9** Considering  $f(\zeta)$  and  $g(\zeta)$  the performance is defined as

$$performance(\zeta) = \frac{f(\zeta) + g(\zeta)}{1/2(n(n-1))M} \quad (2.21)$$

The performance measure tends to favor large clusters with less inter-cluster edges over smaller clusters. Furthermore, the interpretation of the clustering performance should also include an analysis of the noise ratio since noise vertices are not considered when calculating the clustering performance.

**Performance for Overlapping Clusters** To adapt the presented measures for the clusterings with overlapping clusters we consider only vertices that belong to just one cluster. All vertices belonging to more than one cluster and the respective edges are neglected in the calculation.

## Modularity

The *modularity*  $Q$  is an unsupervised measure for cluster quality. The index, proposed by Newman [127], compares the partition of the clustered graph with a random graph. It is assumed that a random graph does not show a clustering structure and that a cluster is significant if the number of edges within the cluster is higher than the expected value that would randomly fall within the cluster.

For unweighted, undirected graphs, the modularity is defined by Newman and Girvan [128]. Brandes et al. [23] follow the definition by [128] and propose the following notation:

$$Q(\zeta) = \sum_{C \in \zeta} \left[ \frac{|E(C)|}{m} - \left( \frac{\sum_{C' \in \zeta} |E(C, C')|}{2m} \right)^2 \right], \quad (2.22)$$

where  $m$  is the total number of edges in the graph.

The function  $Q$  compares the fraction of edges within a community to those that lead to vertices outside the community.  $Q$  favors graph partitionings that consist of dense subgraphs and is thus a measure of graph modularity: Values close to  $Q = 1$  indicate strong community structure.<sup>7</sup> However, modularity implies a fundamental trade-off: Maximizing  $Q$  means to maximize the first term and to minimize the second term. The first term increases with a growing number of edges with high weights in clusters, whereas the minimization of the second term is achieved by splitting the graph into many clusters

<sup>7</sup>According to [128], values greater than 0.3 indicate significant community structure.

containing nodes with small weights. Brandes et al. [24] showed that in an undirected and unweighted graph for a given clustering  $\zeta$ ,  $-\frac{1}{2} \leq Q(\zeta) \leq 1$  holds. The minimal value of the first term is zero if the clusters contain no edges, the second term is maximized if all edges are between clusters; then the value of  $Q$  is  $-\frac{1}{2}$ . A value of 1 is only reached in the case that the graph does not contain any edges because the coverage is then defined to be 1.

**Remark** Modularity is one of the most widely used quality measures for graph clusterings. However, maximizing modularity is  $\mathcal{NP}$ -complete and the runtime of exact algorithms is prohibitive for large graphs (see, e.g., [24]). The edge betweenness algorithm based on modularity which is applied in Chapter 3 to cluster graphs is a heuristic to cope with the problem but does not ensure an optimal solution in all cases. However, the index is useful to serve as a quality measure for comparing clusterings. Therefore, we apply modularity for assessing the quality of a clustering obtained with the proposed clustering procedures, rather than using it for optimization purposes. For the quality assessment of overlapping clusters we discuss the adaptation of the modularity measure in Section 4.3.

### Supervised Clustering Measures

If the clustering of a data set is known a-priori or if clustering results should be compared, a measure of agreement between two clusterings is needed. The Jaccard coefficient serves this purpose.

**Jaccard Coefficient** The Jaccard coefficient is a statistic used for comparing the similarity of sample sets  $X$  and  $Y$ .

**Definition 2.10** *The Jaccard coefficient is defined as the size of the intersection of sets  $X$  and  $Y$  divided by the size of the union of the sample sets:*

$$J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \quad (2.23)$$

*The Jaccard distance, which measures dissimilarity between sample sets, is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from 1.*

$$J(X, Y)_{dist} = 1 - J(X, Y) \quad (2.24)$$

The Jaccard distance ranges in the interval  $[0, 1]$ . If the two sets have no overlap, the Jaccard index is 0 and the distance is 1. Two sets are equal if their Jaccard distance is 0, i.e. if  $J(X, Y) = 1$ .

## 2.5 Tracking Temporal Dynamics of Communities

We present in the following, the most popular approaches to study the dynamics of networks. After the discussion of models for network evolution we present models to study the evolution of communities in networks. Furthermore, we describe the implications of community evolutions from an organizational perspective.

### 2.5.1 Network Evolution Models

Networks tend to evolve (gradually) over time. Barabási et al. [11] proposed a model for the temporal evolution of networks called *preferential attachment*. The hypothesis is that strongly connected nodes increase their connectivity faster than less connected actors. This behavior is typical for social networks and under suitable circumstances, power law degree distributions are generated. In [99], the authors showed in experiments the existence of preferential attachment for some real evolving networks.

The dynamic perspective of networks is studied by Leskovec et al. [113], who observe evolution in graphs. They study a range of different networks, from several domains, and observe how fundamental network properties vary over time. Their methodology delivers insights to changes and trends concerning the graph properties, such as average vertex degree, with some emphasis on large networks (e.g., the Internet). Empirical observation revealed on the one hand a densification of the networks and on the other hand a shrinking diameter. Thus, networks are becoming denser over time, with the average degree increasing and the effective diameter is, in many cases, actually decreasing as the network grows.

Kumar et al. studied the evolution of structures within two social networks (*Flickr* and *Yahoo! 360°*) [109]. They observed the overall network properties and the evolution of “giant components” which represent a large group of people who are connected to one another. The study showed that the density of the network follows the same pattern in both networks: rapid growth, decline and then slow but steady growth. The authors assume that this behavior is due to the activities of “early adopters” who create significant linkages with other individuals which results in a rapid growth. This period is followed by a period in which new members join more quickly than friendships can be established, settling finally into a period in which both membership and linkage increases. Concerning the giant component, the authors observed an decreasing average distance between users over time. (This observation has also been reported by [113], as presented in the previous paragraph.)

The dynamics of social networks are complicated because “network effects” are typically endogenous feedback effects, e.g., reciprocity, transitivity, popularity, subgroup formation. Therefore, statistical models usually do not make justice to the complexity of the network structure and can only reflect a part of a complex reality.

### 2.5.2 Community Evolution Models

The membership of actors to communities shifts over time, resulting in growing and declining communities. Cortes et al. [38] propose a data structure that captures a dynamic graph over time. The data structure arises from a bottom-up representation of the large graph as the union of small subgraphs which they introduce as “Communities of Interests” centered on every node. These subgraphs consist of a node and its directed top- $k$  edges to other nodes. The meaning of “top” is relative to the aggregation function applied to transactions associated with each edge, and therefore, the value of  $k$  determines how well the data structure approximates the true graph. To accommodate the time evolution of

the graph, the data structure is updated at fixed time steps by aggregating the transactions and updating the subgraph accordingly. A community is defined as the neighborhood of a chosen vertex, subject to the vertex's connectivity to other vertices.

A method for the discovery of communities in data streams has been proposed by Aggarwal and Yu [3]: They focus on indicators of a given community's evolution and use them to detect subgroups with the highest increase, resp. decrease of interaction within a graph to detect expanding, contracting and stable communities. The interactions are considered as a data stream captured at certain time points. This data record is an interaction, weighted with a value and consists of the newly arrived data and of old data which is subject to a decay function. At each time point, the "frame" of the graph of interactions is built. An evolving community is then defined as a cluster that shows a different level of evolution than its surroundings and Aggarwal and Yu propose an algorithm that traces these evolving communities.

Backstrom et al. [10] aim to determine the critical factors for group evolution by focusing on a small set of network snapshots. They analyze the role of common friends in community formation regarding the motivation of an individual to join a community. Their goal is to identify structures that influence the decision of individuals in joining the community and to understand the evolution of a community and its interplay (overlap of members) with other communities. The authors define a community as a subgraph of interacting members and introduce what they call a "fringe": It consists of individuals that interact with at least  $k$  community members but are not community members themselves. Based on this model, features are identified that influence members of the fringe to move inside the community. Backstrom et al. study the Live Journal-Community an online blogging site where users can create friendship links to other users. The authors identified two variables that make a good predictor of the propensity of an individual to join a given community: (i) the likelihood of joining a community increases with the number of friends already in it, but is very noisy for individuals with many friends and (ii) the existence of friendships among friends contributes to this likelihood.

Tang et al. [157] propose a model to identify dynamics in multi-mode networks by adopting a spectral clustering framework. By observing a series of network snapshots, community evolution is detected using an evolutionary clustering approach.

### 2.5.3 Models for the Evolution of Clusters

In order to develop a model for the detection of possible transitions a community may undergo and the factors that evoke these changes, it is necessary to recognize and classify situations where changes occur. In addition, the model for the evolution of communities depends on the definition of a community and the approach that is applied to detect communities:

- In Chapter 3 we define a community as a subgraph that evolves over time and detect this structure by re-clustering graph snapshots and determine matches between clusters obtained in preceding clusterings.

- In Chapter 4 we denote a community as dense subgroups that evolve over time and observe the evolution of subgroups by adapting clusterings over the observation period. Changes here are considered as transitions of communities.

In Chapter 3, changes are considered as cluster transitions upon data that are collected and clustered at time points whereas in Chapter 4 as the evolution of a community.<sup>8</sup> In the following, we propose models for both types.

**Model for Community Evolution in Chapter 3** To model the transition of clusters between time points, the MONIC framework is adapted [155]. It assumes the re-clustering of data rather than cluster adaptation at each time point, so that changes between cluster instances from one time point to another can be monitored. The model is described in detail in Section 3.1.3.

**Model for Community Evolution in Chapter 4** When observing changes via cluster adaption, we distinguish six types of transitions [49, 53]: (1) *Creation* of a new cluster, (2) *Absorption* of a vertex which results in an increased cluster size, (3) *Merge* of two or more clusters, (4) *Removal* of clusters, (5) *Reduction* of a cluster which results in a decreased cluster size and (6) *Split* of a cluster into two or more clusters. In Table 2.1, these temporal dynamics of clusters are shown.

A formal definition for the transition types and a description of the incremental algorithm which detects these transitions is given in detail in Section 4.2.

## 2.5.4 Community Evolution in an Organizational Context

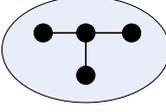
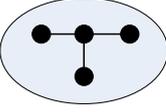
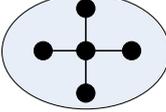
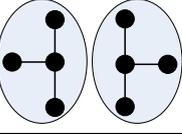
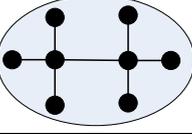
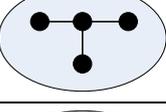
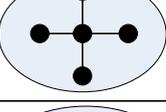
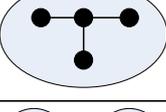
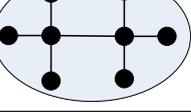
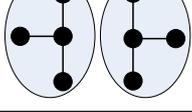
Considerable research has focused on internal and external factors that affect the evolution of teams working together virtually (see, e.g., [86, 93, 98, 117, 135]). The reasons why a community structure changes are complex but can be divided into two categories: (i) internal and (ii) external influences:

- Internal Influences
  - Common Ground: Common ground is the sum of “mutual knowledge, mutual beliefs, and mutual suppositions” shared by individuals [36]. It provides the basis for communication between individuals. The size of the community is an important factor that influences the probability of a common ground. It can be observed that rapidly growing communities “loose” their common ground which influences the evolution of the group (usually negatively).
  - Information Overload: The communication activity is an important indicator for the attractiveness of the community as members would not participate actively in the community if the benefits of the community do not outweigh the costs of the membership [32]. With costs, not only monetary amounts

---

<sup>8</sup>For the sake of simplicity we use the term *community* if we talk about the impact or trigger of the evolution and *cluster* if the methodology is in focus.

Table 2.1: Six types of cluster evolutions.

	Period t	Period t+1
Creation of a new cluster		
Absorption of a vertex		
Merge of clusters		
Removal of a cluster		
Reduction of a cluster		
Split of clusters		

are meant, but the resources in general that a member provides for the community (in particular the time he spends). However, an increasing number of community members and transactions yield to an information overload which members might not be able to cope with. Therefore, the positive and negative consequences of growing communities need to be balanced.

- External Influences

- Advertising or publicity in the mass media may influence the development of a community. The influence can be either positive or negative. In the first case, a positive report about a community in the media or an award can be very motivating, a negative publicity or a slating review can have devastating effects.
- Seasonal variations can be observed, depending, e.g., on the domain of the community. A political community might be more active in the run-up to elections, a community of sailors during summer time.

The two basic relationship changes that can be observed in communities are (i) new relationships emerge and (ii) old relationships disappear. A new relation can result in a

new cluster (Creation), a growing cluster (Absorption) or merging clusters (Merge). A lost relation may cause a shrinking cluster (Reduction), a split of clusters (Split) or even the death of a cluster (Removal).

**A New Relation: Creation, Absorption or Merge** From an application-oriented perspective in an organizational context the question to decide whether a community is created and alive is: Does the community maintain a coherent, ongoing identity known both to itself and the larger organization, and does it function as a goal-directed, self-managing system?

The proposed model is able to assess whether a community is created and matures. Questions regarding their performance can only be indirectly answered by these observations. An interaction is an indicator for knowledge transfer, it can be assumed that the performance of a community increases with a growing number of participants and interaction number. To assess which size of a community is desirable is hard to decide. It depends on the environment and the goal of the community. In [80], a model has been developed for the assembly of teams in which the selection of the members of a team depends upon three parameters one of which is the team size.<sup>9</sup> They find that team assembly mechanisms determine both the structure of the collaboration network and the team performance. In an organizational context, large communities tend to be less effective, unless coordination is enhanced. However, small communities may not have the resources needed to perform the task.

Gongla and Rizzuto distinguish two types of community mergers [77]:

- **Mergers of equal communities** Sometime communities find that they have a lot in common and individual members find themselves in both communities. The knowledge domain of each community is either similar or complementary to the other community. Even processes and technology use are similar and members recognize that the merger of both communities may enhance efficiency as resources may be shared. The two communities disappear and a new community is established in their place.
- **Mergers of unequal communities** If a community constitutes a specialized subdomain of a larger community it may join willingly the broader community or it may be absorbed by it. The specialized community disappears and takes on the identity of the broader community.

**A Lost Relation: Removal, Reduction or Split** “When we say that a community disappears, what we mean is that the community is no longer recognized as a separate, functioning system with a known, ongoing identity.” [77] Gongla and Rizzuto identified three factors that most commonly act as triggers to the disappearance of communities [77]: (i) Organizational Change, (ii) Knowledge Domain Change and (iii) Community Leadership Change. However, these transformations must not necessarily result in the

---

<sup>9</sup>The other two parameters are the fraction of newcomers and the tendency of incumbents to repeat previous collaborations.

removal of a community. As changes usually proceed gradually it is likely that they lead at first to a reduction of the community and potentially to splits.

- **Organizational Change:** Communities are usually sponsored by a particular business unit and are therefore to some extent aligned with the objectives of this part of the organization. A reorganization or realignment of this business unit that included for example new goals directly influences the function and efficiency of the community. Furthermore, a change of leadership of this unit may result in new priorities and a redeployment of resources [77].
- **Knowledge Domain Change:** A knowledge domain of a community is not necessarily static. Interests and objectives may change and communities may merge or disappear. This can be especially observed in technology-oriented communities, where knowledge may become obsolete or knowledge domains shift to more innovative methods.
- **Community Leadership Change:** A community most likely has core members who have a high influence on the development of the community. This can be a single person, but often it is a (small) group of people who possibly have established the community and feel a high responsibility for the community as a whole. This core makes decisions about the direction of the community and usually encourages and stimulates activity. Studies have shown that only a small minority of the users usually posts the majority of the messages (see, e.g., [148, 6]). If these leaders leave the community it is very important to find new active and passionate members who continue their work. If this fails, it is very likely that the community becomes less active and finally disappears. If the amount of core members increases, the leadership may not be active enough to stimulate the community. This may also happen if the number of core members stays the same but their desire or ability to commit time decreases significantly.

Communities usually do not disappear all at once but gradually drift into non-existence. It might take quite some time before its disappearance. During this drift phase, new members rarely join the community since the community is not visible to them and any overtures may get little response. If core members leave, they are often not replaced. Members stop identifying with the community and less and less interaction can be noticed. Since this is usually a very slow process it is hard to determine when exactly the community finally ceases to exist.

A community that is no longer observable, may have become a structural unit of the organization. This can be the case if for example the community's knowledge or influence has become very important to the organization. The mission is then determined by the organization and the members are expected to deliver certain results that are specified by the management. The same members may interact and the body of knowledge may still be enhanced and shared among the members, however the interaction channel may have changed to a new form.

# 3 Community as a Subgraph Formation Over Time

In this chapter, we define a community as a set of community instances that are observable over time. A community is thus a cluster of similar *community instances* and observing temporal changes allows a macroscopic view on the evolution of subgroups in networks. We propose a community discovery model (Section 3.1) including a statistical analysis technique to study the characteristics of community instances and a graph visualization to study communities as evolving graph structures (Section 3.2). Furthermore, we discuss results of experiments with longitudinal community data to study communities of fluctuating members (Section 3.3) and present an application of this procedure to the area of user modeling where knowledge about a community membership of an individual can be part of a user model. In this study, we analyze how the representativeness of a community for a member can be determined (Section 3.4). The chapter closes with a discussion of the related work (Section 3.5) and concluding remarks (Section 3.6).

## 3.1 The Community Discovery Model

Let  $G = (V, E)$  be a weighted, undirected graph in which  $V$  denotes the set of vertices, representing the actors and  $E$  the set of edges between actors  $(u, v)$ , with  $u, v \in V$ . A *cluster*  $C \subseteq V$  is a subset of vertices.  $G[C]$  is the graph induced by  $C$ . A graph cluster represents a collection of vertices that are similar to one another and are thus regarded as a group. A *clustering*  $\zeta = \{C_1, \dots, C_k\}$  of  $G$  is a partition of all vertices into clusters.  $A(G)$  is the set of all possible clusterings. A clustering is called *trivial* if either  $k = 1$  or  $k = n$ , where  $k = |\zeta|$  is the number of clusters in the  $\zeta$ .  $|C_i|$  denotes the number of vertices in a cluster  $C_i$ . Clusters of size 1 are called *singletons*.

The weight of an edge  $(u, v)$  represents the strength of the relation between vertices which is modeled by the edge. In a network of airports the weight could represent the flight time from one airport to the other or the number of daily flights between two airports. In the applications presented in this chapter, vertices represent actors and the relation between actors represent *interactions*. The edge weights represent the *interaction strength* between actors measured, e.g., in number of reciprocated e-mails messages. Therefore, the terms “edge weight” and “interaction strength” are interchangeable in the given context.

To discover and track communities over time, we first describe an approach for modeling and discovering communities as clusters of similar community instances which are present over time [55, 54]. The five-step process is illustrated in Figure 3.1.

1. Step: Since an aggregation of all interaction data to one graph would eliminate all temporal information about the network dynamics, we define time periods over which network information are aggregated. Therefore, we partition the observation period in equidistant *time windows*, for example, an observation period of one year is partitioned into twelve time windows, each covering the interactions collected over one month. For each time window, we build a graph  $G_t$  based on the observations made during that period, where  $t$  indicates the time period over which the information is aggregated.
2. Step: In the next step, we discover *community instances* in each time window by clustering the graph  $G_t$  using the hierarchical edge betweenness clustering algorithm.
3. Step: Afterwards, we link community instances which are detected in different time windows based on their similarity over the observation period.
4. Step: In the next step, a *community survival graph* is build representing community instances as vertices which are linked via edges based on their similarity. Here, the edge weight represents the similarity of both end-vertices. On this graph, a *community* is a dense subgraph of linked instances.
5. Step: We detect a *community* by clustering the community survival graph of similar community instances with the hierarchical edge betweenness algorithm.

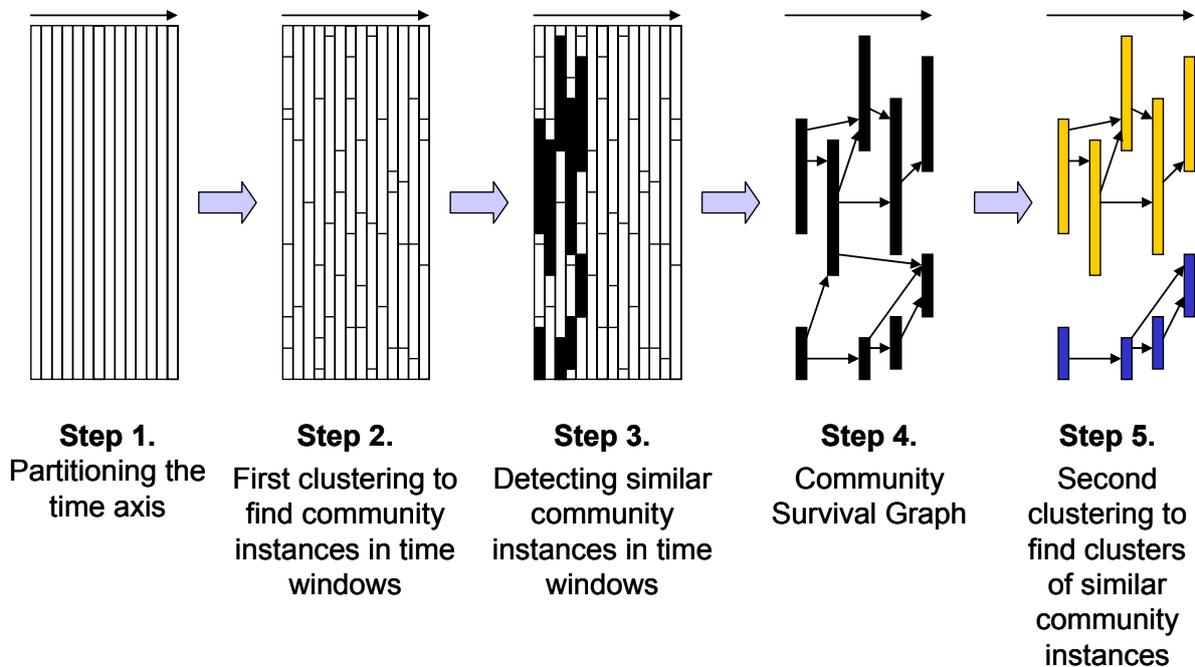


Figure 3.1: Community Discovery Roadmap

### 3.1.1 Partitioning the Time Axis and Building the Graph of Interactions

In the first step, we define how the time axis is partitioned to specify the periods that are considered for the dynamic graph  $G_t$  which is clustered later on (cf. Step 1 in Figure 3.1). We consider discrete time applications where information about transactions from time step  $t$  to  $t + 1$  only become available at the end of the time step. Furthermore, to build the graph, we define the edge weights in  $G_t$  by specifying the importance assigned to old vs. new interactions.

#### Sliding Window Approach

Since the interactions between actors (represented as edges) and the set of actors (represented as vertices) are not static but change over time, i.e. edges and vertices appear and disappear from the graph, an aggregation of all data would lead to a loss of all temporal information. An intuitive way to capture the dynamics of a graph is to aggregate edge weights, which represent in the given context interaction strength, only over a given period and extend this period to develop a discrete notion from continuous data (cf. Figure 3.2 (a)). In case of a community platform where members interact via messages, the aggregation could be to sum up the number of interactions over a given time period. However, this aggregation of the interactions over time would favor “old” members of the community over newcomers.

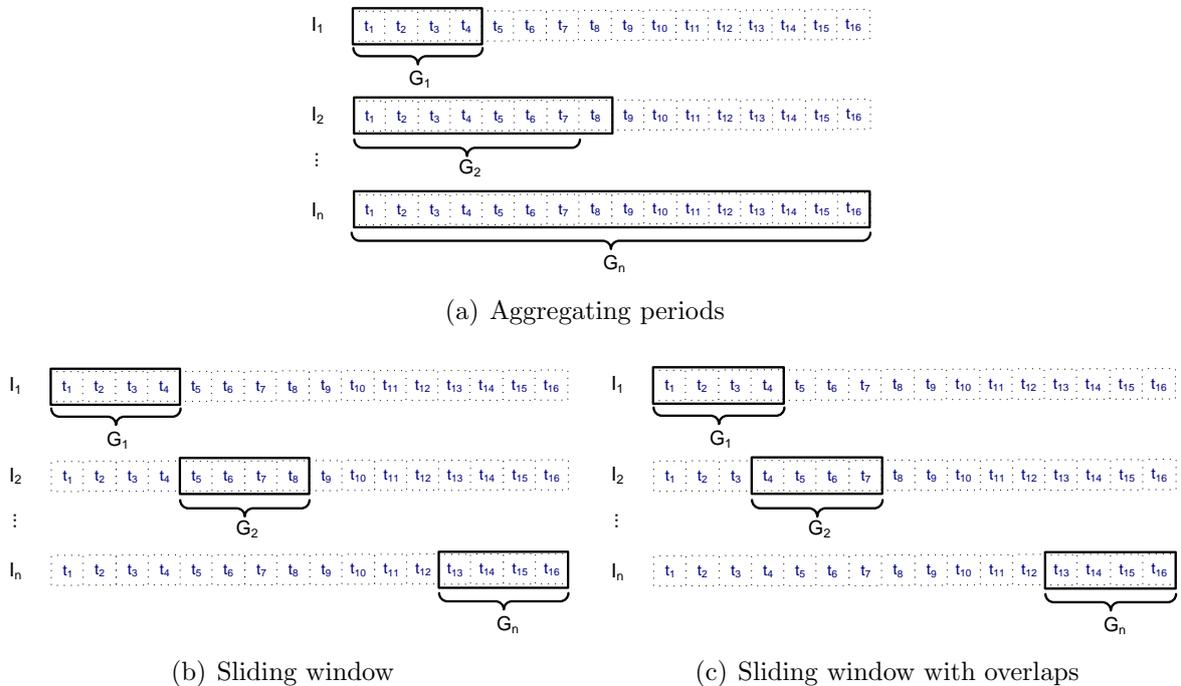


Figure 3.2: Partitioning the Time Axis.

To prevent the prevalence of “old” users that have performed many interactions over “new” ones that have not yet interacted with many people, a *sliding window* approach can be applied. Here a window of a predefined length  $k$  is moved over the data stream and specifies the intervals to analyze. Let  $T = \langle t_1, \dots, t_n \rangle$  be a sequence of points in time,  $I_i$  an interval  $[t_{i-k}, t_i]$  of length  $k$ , where  $0 < k \leq i$  and  $G_i$  is the graph accumulated during the interval  $I_i$ , beyond which interactions are forgotten. The sliding windows acts as a decay function which determines which data contributes to  $G_t$ .

After defining the interval we would intuitively partition the graph over time into equidistant time slots, each slot starting when the last slot finished (cf. Figure 3.2 (b)). This modus is called a *non-overlapping sliding window*. An *overlapping window* partially overlaps with the prior window (cf. Figure 3.2 (c)). The degree  $\delta$  to which it overlaps must be defined.

In the experiments we apply an overlapping window since it smoothes out the gaps that sometimes occur between two intervals. It can be interpreted as a moving average [123]. To find the right length of the window is very challenging. Each graph should show enough change, but not too much in order to give information about network dynamics. The question concerning the right rate of change can be described with respect to different levels (e.g., fast, medium, slow). The meaning of these terms depends on the context since the scale will vary across different relations. For example, the rate of change for the relation “e-mail-exchange” will be much shorter than for “trade relationship”. Chunks of time must be identified that account for the relation type and the nature of the events in the network.

After defining the partitioning of the time axis, the next step is to set up the graph of interactions.

### Defining Proximity over a Graph of Interactions

The first question that arises concerns the definition of  $G_t$ ; the graph  $G$  in time window  $t$ . Each edge is associated with a weight that is derived from an aggregation function applied to all transactions between a pair of nodes at time step  $t$ . Let  $g_t$  be the graph induced by the transactions during time step  $t$ .

If  $G_t$  is defined by  $g_t$ , the graph might change strongly at each time step. If, on the other hand, following the notation of [38],  $G_t$  is defined by summing up all interactions up to interval  $t$

$$G_t = g_1 \oplus g_2 \oplus \dots \oplus g_t = \bigoplus_{i=1}^t g_i \quad (3.1)$$

all information about the temporal development would be lost.

Cortes et al. propose a recursive definition to calculate the edge weights which allows to reduce the influence of “old” interactions [38]:

$$G_t = \theta(G_{t-1}) + (1 - \theta)g_t \quad (3.2)$$

Where  $0 \leq \theta \leq 1$ . The influence of interactions in “future” intervals decreases with a small  $\theta$ . In other words, more recent data contributes more heavily to  $G_t$  than older data. But,

this definition suffers from the problem of determining the parameter  $\theta$ : Typically, one determines the influence of old edges based on the average of all edge weights. However, by choosing the average, all edges below this average lose their influence on the weight in adjacent intervals. Thus, rather weak edges have no influence on future time steps.

We therefore define  $G_t$ , according to the sliding window method presented above, as a moving window over  $n$  time steps to allow the graph to track the dynamics of the interactions [123, 74]:

$$G_t = g_{t-n} \oplus g_{t-n+1} \oplus \dots \oplus g_t = \bigoplus_{i=t-n}^t g_i \quad (3.3)$$

By this we achieve that all interactions in the respective intervals have the same influence on  $G_t$ . The weight of an edge in  $t$  is the sum of the intensities in the interval  $[t-n, t]$ .

The edge weight  $\omega(u, v)$  in the graph  $g_t$  quantifies then the *proximity* of two members  $u$  and  $v$ . The weight is measured based on the *intensity* of the interaction.

**Definition 3.1** The “intensity” of the interaction between actors  $u, v$  is defined as

$$intensity(u, v) = \min((u \rightarrow v), (v \rightarrow u)). \quad (3.4)$$

where  $(u \rightarrow v)$  is the number of interactions  $u$  performed towards  $v$  measured in number of messages sent from  $u$  to  $v$  and vice versa.

The intensity is defined by the minimum number of reciprocal messages and the higher the intensity of message exchange, the higher the proximity:

**Definition 3.2** The proximity  $prox(u, v)$  between two actors  $u, v$  is defined as:

$$prox(u, v) = \begin{cases} 1 & u = v \\ 1 - \frac{1}{intensity(u, v)} & \exists(u, v) \in E \\ \text{undefined} & \text{otherwise} \end{cases} \quad (3.5)$$

The proximity function, where defined, ranges in  $[0, 1]$  and is symmetric. If only one interaction exists between them, then their proximity is zero. The proximity value increases asymptotically towards 1 as the number of interactions increases reciprocally.

The intensity is an appropriate measure for the strength of the relationship. It has been reported, for example, that the impact of individuals is greater the more the people communicate [112]. However, using the intensity, reciprocity regarding the number of interactions is of paramount importance here: If  $u$  addresses  $v$  1000 times but  $v$  addresses  $u$  only twice, the proximity of the two will be the same as if  $u$  had addressed  $v$  only twice. This notion of proximity suppresses the impact of broadcasts and is appropriate for networks where some nodes are characterized by broadcast behavior, e.g., a secretary

who regularly sends announcements to all company personnel. Such broadcasts should not be confused with person-to-person interactions.

Since proximity is only defined for pairs of actors that are connected with an edge, its 1-complement does not satisfy the triangle inequality and is thus not a distance function. Hence, we still do not have a measure over the graph as is required by conventional spatial clustering algorithms. However, this partial definition of proximity is adequate for computing the graph-based measures such as the edge betweenness centrality which is used for the clustering.

Although proximity can be used to assess similarity among users, it is stressed that proximity-by-interaction does not imply that the users are conceptually similar: Indeed, users with very different properties or original preferences may interact and influence each other, to the effect that their preferences coerce.

### Building the Graph of Interactions for Each Time Window

Using this notion of proximity, we build for each interval  $I_t$  the graph of interactions  $G_t$ . We assign the weights for each interval by updating the edge weight  $\omega(e)$  in  $I_{t+1}$  as the sum of the weights over the last  $n$  periods according to Equation 3.3. The edge weight is updated in  $I_{t+1}$  by adding the proximity in interval  $I_{t+1}$  to the current edge weight. The graph  $G_t$  is used as input for the hierarchical divisive clustering to extract the set of community instances  $\zeta$  (cf. Figure 3.3).

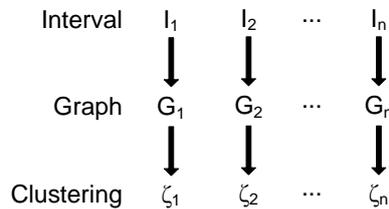


Figure 3.3: Building a Graph for Each Time Window.

### 3.1.2 Clustering Users into Community Instances

The transactions that occur in each time window are aggregated and considered as a static representation of the graph in the chosen interval. Next, the obtained graph  $G_t$  is decomposed into *community instances* of closely connected actors (cf. Step 2 in Figure 3.1). These clusters are defined as subsets of vertices within a graph with a high degree of interaction among the actors.

#### Hierarchical Divisive Edge Betweenness Clustering

To extract the community instances, we apply a hierarchical divisive clustering approach that partitions the graph into clusters by the iterative removal of edges (cf. Section 2.3).

The algorithm partitions the graph iteratively into denser subgraphs by deleting edges that do not belong to the dense subgraphs but rather serve to separate them.

To find these inter-community edges, Watts and Strogatz [162] have proposed a method which is based on counting short loops of edges. The motivation behind this method is that edges between communities belong to less short loops than intra-community edges as the completion of short loops requires a third actor who also interacts with a member from another community. To detect these inter-community edges one should look for edges with a low number of loops. This algorithm is suitable for dense networks with many triangles which social networks usually are. However, for sparse networks and networks with a lower transitivity the algorithm might not work well.

Therefore, we apply a method proposed by [73] to detect the edges to be deleted which is based on the edge betweenness. The betweenness of an edge is defined in Equation 2.5 as the number of shortest paths along it. The basic idea of the edge betweenness clustering is that when a graph is made of tightly bound clusters which are loosely interconnected, all shortest paths between clusters have to go through the few inter-cluster connections.

The hierarchical edge betweenness clustering procedure works as follows (cf. Algorithm 3.1):

- The edge betweenness values for each edge are calculated
- The edge with the highest edge betweenness is removed from the graph
- The process is repeated until no edges are left

In Figure 3.4 on the left, a graph is shown consisting of three clusters, where two of three clusters are connected with an edge. The three inter-cluster edges have a high edge betweenness and are deleted iteratively by the cluster algorithm. After three iterations, three edges are deleted resulting in three separated communities as shown in Figure 3.4 on the right.

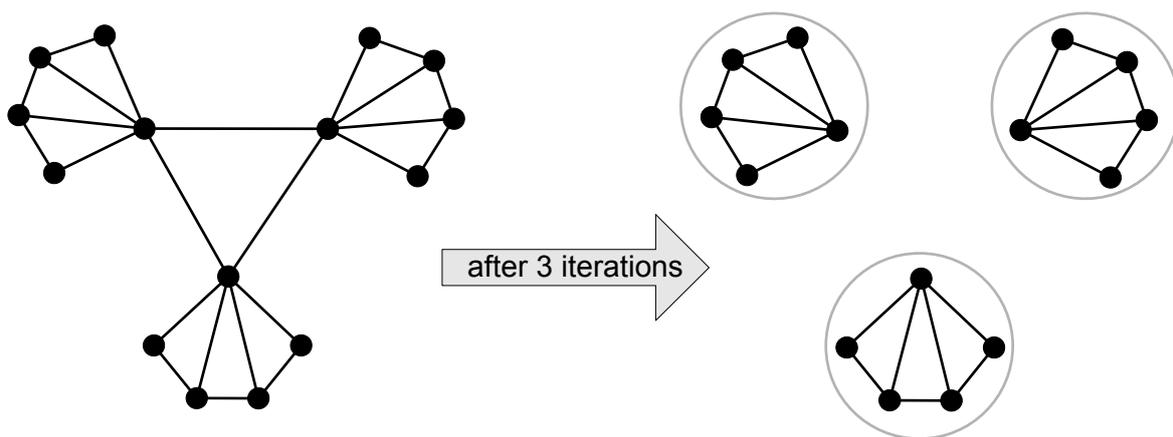


Figure 3.4: Edge Betweenness Clustering.

---

**Algorithm 3.1:** Edge Betweenness Clustering

---

```

1 Algorithm:EBC(Graph)
   input : Graph
   output: ClusterModel (Dendrogram)
2 begin
3   repeat
4     Compute edge betweenness for all edges
5     Remove edge with highest betweenness
6   until (no more edges in graph)
7   return Dendrogram
8 end

```

---

The algorithmic complexity of the edge betweenness centrality index is  $\mathcal{O}(nm+n^2\log(n))$  for weighted networks [105].

### Extracting the Clustering from Dendrogram

The output of the hierarchical divisive algorithm is a dendrogram; the root is the whole graph, the leaf nodes are individual vertices. From the hierarchical clustering structure, the clustering with the highest modularity  $Q(\zeta)$  is chosen as defined in Equation 2.22 (cf. Figure 3.5).

The function  $Q$  compares the fraction of edges within a cluster to those that lead to vertices outside the cluster.  $Q$  favors graph partitionings that consist of dense subgraphs and is thus a measure of graph modularity: Values close to  $Q = 1$  indicate strong cluster structure and values greater than 0.3 indicate significant cluster structure [128].

The obtained clusters are candidates for communities and referred to as *community instances* ( $CI$ ).

**Definition 3.3** *A community instance  $CI$  in  $t$  is a cluster  $C \in \zeta$  where  $Q(\zeta)$  is maximal.*

### 3.1.3 Evolution of Community Instances

Given a community instance found at period  $t$ , we are interested in its evolution in subsequent periods. For the conceptual definition of *survival*, we assume that a community instance is characterized by the people participating in it.<sup>1</sup> However, we tolerate a fluctuation of the community members [55].

As proposed in the MONIC framework for cluster evolution over a data stream [155], we juxtapose each community instance  $CI$  discovered at period  $t_i$  with each candidate cluster of the next period  $t_j$  (cf. Step 3 in Figure 3.1): We define similarity among community instances that have been discovered at different times as the overlap of members between the two community instances. Thus, two instances are similar if their overlap exceeds a

---

<sup>1</sup>This is an important distinction to the proposed incremental approach (Chapter 4) where clusters are tracked based on the similarity of actors.

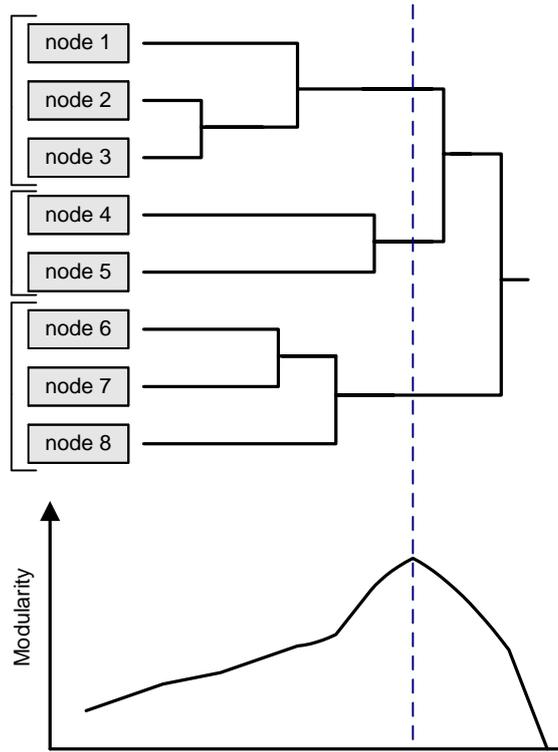


Figure 3.5: Dendrogram-cut Based on Highest Modularity.

given threshold. In particular, let  $t_i, t_j$  be two distinct time periods, let  $G_i, G_j$  denote the corresponding graphs of interactions and let  $\zeta(G_i), \zeta(G_j)$  be the corresponding clusterings. Further, let  $x \in \zeta(G_i)$  and  $y \in \zeta(G_j)$  be two community instances.

**Definition 3.4** *The overlap between two community instances  $x, y$  is defined as*

$$\text{overlap}(x, y) = \frac{|x \cap y|}{\min(|x|, |y|)} \quad (3.6)$$

where  $|\cdot|$  is the number of vertices in a community instance or intersection.

An overlap threshold  $\tau_{\text{overlap}}$  captures the tolerance to member fluctuation: It can be observed in some communities that the number of time intervals an individual is active in the community is small. Thus, former members might not show up in each interval or they participate temporarily in another community instance. Thus, the formation of the community instances is fluctuating over time and finding the same community instances over a longer period is rather unlikely. But even though the membership basis fluctuates over time, the community is still there.

This problem can be handled by demanding only for a low overlap of actors between community instances. Therefore it is not necessary that all individuals participate in each period in the same cluster instances.

The definitions of overlap and similarity allow for comparisons between community instances in arbitrarily remote time periods. It is of course possible to compute the similarity of a community instance to any past community instance. However, this is computationally expensive. Moreover, it may lead to noisy results in environments where individuals participate in several communities but are active in different communities at different times. Therefore, we introduce an upper boundary  $\tau_{periods}$  to the number of periods that may separate two potentially similar community instances; instances that are more than  $\tau_{periods}$  apart have zero similarity by default. This ensures (i) that similar community instances are connected to each other independently of their temporal proximity; subject only to  $\tau_{periods}$  and (ii) that communities that experience little change over time correspond to highly connected groups of community instances, i.e. to tight clusters.

**Definition 3.5** *Let  $t_i, t_j$  be two time periods, such that  $t_i < t_j$ , let  $G_i, G_j$  be the graphs of interactions in those periods and let  $x \in \zeta(G_i)$  and  $y \in \zeta(G_j)$  be two community instances. Using the notion of overlap and  $\tau_{periods}$ , we define a similarity function  $sim(x, y)$  as follows:*

$$sim(x, y) = \begin{cases} 1 & (\text{overlap}(x, y) \geq \tau_{overlap}) \\ & \wedge (t_j - t_i < \tau_{periods}) \\ 0 & \text{otherwise.} \end{cases} \quad (3.7)$$

Two community instances  $x$  and  $y$  are thus defined to be similar if (i) the overlap of their members reaches the threshold  $\tau_{overlap}$  and (ii) if both instances are at most  $\tau_{periods}$  apart from each other. Changing the threshold  $\tau_{periods}$  is like changing the resolution with which the community structure is investigated: by increasing the threshold the communities shrink and fall apart.

We use the notion of similarity in Equation 3.7 to create the community survival graph (cf. Step 4 in Figure 3.1).

#### 3.1.4 Community Survival Graph

To detect communities and to determine for example at which point in time a community dies, the development of a single community instance is rather irrelevant. One community can show different formations along the time axis. Tracking a specific community instance does not reveal information about the community structure. We therefore do not care about the development of a single community instance but for the persistent structure of similar community instances. Those structures can be revealed by detecting clusters of similar community instances. The borders of these clusters indicate that a community died or merged with another community.

Within each time window of  $\tau_{periods}$ , we compare community instances and connect matching instances with an edge, thus establishing a *community survival graph* (cf. Step 4 in Figure 3.1).

Formally, let  $G = (V, E)$  be the *community survival graph*. A vertex  $v$  has the form  $(C, t)$ , where  $C$  is a community instance discovered at period  $t$ . An edge  $e = (v, v') = ((C, t), (C', t'))$  denotes that cluster  $C$  has survived into  $C'$ , i.e.  $t < t'$  and  $\text{overlap}(C, C') > \tau_{\text{overlap}}$ . The threshold value  $\tau_{\text{periods}}$  ensures that  $|t' - t| < \tau_{\text{periods}}$  for all pairs  $(t, t')$  that appear in edges of  $E$ . An edge is further associated with a weight, defined as the similarity of the connected vertices (cf. Definition 3.5).

The obtained community survival graph is the input for the clustering to detect communities as described in the next section.

### 3.1.5 Communities in the Survival Graph

To discover groups of community instances upon the survival graph, we again use hierarchical divisive clustering: It uses the edge betweenness of the edges to eliminate edges which separate subgraphs that are denser than their surroundings (cf. Step 5 in Figure 3.1).

The algorithm proceeds analogously to Algorithm 3.1 but receives as input additionally a parameter  $k$  which determines the number of iterations to be performed; or equivalently the maximum number of edges to be removed from the graph.  $k$  is determined empirically and thus chosen by the user. The value of  $k$  depends on the size and the structure of the network. Changing  $k$  is like changing the resolution (as in a microscope) with which the community structure is investigated: by increasing  $k$  the communities are smaller and more disintegrated, but at the same time, also more cohesive.

We perform clustering upon the community survival graph to detect groups of similar community instances. As for the discovery of community instances in each time period, we weight the edges of the graph with edge betweenness and progressively remove edges as part of a hierarchical divisive clustering algorithm. In each round, the edge betweenness of the remaining edges is (re-) computed. The connected subgraphs retained after  $k$  iterations correspond to the communities found thus far. A connected subgraph returned at the end of the iterative process is a community, comprised of similar community instances.

**Definition 3.6** *A community, denoted as  $\chi$ , upon a community survival graph  $G(V, E)$  is a connected subgraph retained after  $k$  iterations.*

This definition observes dense substructures in graphs, in other words it is a macroscopic view that allows to discover superordinate community structures in the survival graph and relate them to their members. This perspective is especially useful if parallel evolutions of communities are of interest for the observer.

In communities with fluctuating members it is of interest to detect the gaps between clusters of communities instances.

**Definition 3.7** *Let  $G(k, V, E_k)$  be a community survival graph after  $k$  iterations, let  $t_1, \dots, t_n$  be the periods encountered in  $V$  and let  $\chi$  be a community found in  $G$ . The community  $\chi$  dissolves in  $t_d$ , if and only if there is no edge  $e = (v, v')$  in  $E_k$  with  $v = (C, t)$  and  $v' = (C', t')$  with  $t < t_d$  and  $t' \geq t_d$ . The time point  $t_d$  is the “dissolution point” or “gap” of community  $\chi$ .*

According to this definition, a gap separates two clusters of community instances in the sense that no instance observed before the gap has edges that cross the gap. In practice, the demand that no edge “crosses” the gap is too restrictive: We relax it by requiring that the number of edges belonging to an instance of  $\chi$  is minimal. Then, the “lifetime” of the community is the *timespan* $[t_{min}, t_{max}]$  between the first point it was encountered and the gap at which it has disappeared.

The localization of a structural gap corresponds to finding a decision boundary that minimizes errors using a method known as soft margin approach for linear support vector machines [156]. This method allows us to find a decision, in other words to determine a clearly-defined time period where communities are separated, even in situations where the clusters are not paraxial separable. In our case, we find a gap by minimizing the classification error, i.e. the number of misclassified community instances (those that are on the wrong side of the line).

## 3.2 Visualizing Temporal Dynamics in Communities

Studying the temporal dimension of graphs will be helpful to detect and observe current developments and to predict future trends. Depending on the granularity of the observation, different types of developments can be studied. It is necessary to identify the appropriate set of properties that indicate a changing behavior of community members, about community instances and about communities.

### 3.2.1 Actor-oriented Visualization of Community Instance Evolution

To observe community transitions, we track a detected community instance over time by measuring the structural equivalence with other community instances [58]. The development of a community instance can be described and assessed by measuring and interpreting the following measures: Stability, density and cohesion, Euclidean distance, correlation coefficient and group activity. For a chosen community instance  $x$  that has been detected in a given time window, the resulting comparisons are shown using different statistical measures and visualization techniques.

For comparison, all members of the chosen community instance are extracted and their interaction behavior in other periods is compared to the situation in the chosen period. Two types of comparisons are shown: the fixed and the periodic. For the *fixed* comparison, the chosen time period is compared to all other time periods (cf. Figure 3.6 (left side)). This allows to assess how the structure of the community instance varies over time compared to the initial period. The *periodic* chart starts from the first period and compares the interaction behavior of all members with the observation in the next period (cf. Figure 3.6 (right side)). By this, a period-by-period comparison is possible and for example periods with strong changes in the interaction structure can be identified.

The used measures and their interpretation are described in the following [54].

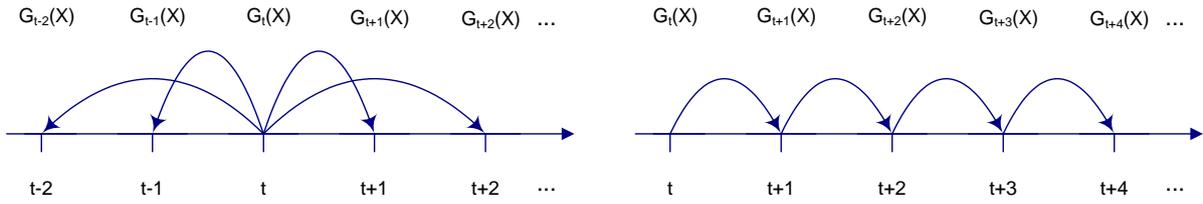


Figure 3.6: The interaction behavior of members of community instance  $x$  in time  $t$  are compared with instances in all other periods (left) and with the respective neighboring instances for all adjacent periods (right).

**Stability** The *stability* shows how stable the composition of the group is over time. The *fixed stability* indicates how many of the original members in the chosen time window are active in all other time windows. The set of members of a fixed period is compared to all other periods. If the value is one, all members of the chosen community instance are active. For the *periodic stability* the set of members is compared to the previous time window. A low curve indicates high changes in the membership of the community instance.

**Density and Cohesion** The *density* indicates the connectivity inside the group  $X$ . It is the proportion of the number of edges inside the group to the number of possible edges. The higher the density the more connected the group members are.

$$\Delta(X) = \frac{m(X)}{n(X)(n(X) - 1)/2}, \quad (3.8)$$

where  $m(X)$  is the number of edges in  $G(X)$  and  $n(X)$  the number of vertices inside the community instance.

The *cohesion* indicates how connected the group is to non-instance-members. It is obtained by dividing the average strength of the interaction inside the community instance by the average interaction strength to actors outside the subgroup.

$$v(X) = \frac{\frac{\sum_{e \in E(X)} \omega(e)}{n(X)(n(X)-1)}}{\frac{\sum_{e \in \overline{E(X)}} \omega(e)}{n(X)(n-n(X))}} \quad (3.9)$$

where  $E(X)$  is the set of edges that have their origin and destination in  $X$ ,  $\omega(e)$  is the weight of the edge,  $n$  is the number of vertices in  $G$  and  $n(X)$  the number of vertices inside the community instances. If the cohesion increases greater than the density, the connectivity to members outside the subgroup grows faster, resulting in a less stable subgroup.

**Euclidean Distance** The structural equivalence of two subgroups  $X$  and  $Y$  is measured by their *Euclidean distance*.

$$distance(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (3.10)$$

The larger the distance, the less equivalent the groups are.

**Correlation Coefficient** The *correlation coefficient* is obtained by dividing the covariance of the vector representation of the subgroups  $X$  and  $Y$  by the product of the standard deviations.

$$\rho_{X,Y} = \frac{cov(X, Y)}{\sigma_X \cdot \sigma_Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E^2(X)}\sqrt{E(Y^2) - E^2(Y)}} \quad (3.11)$$

where  $E$  is the expected value operator. The correlation coefficient takes a value between -1 and +1. The closer the coefficient is to either -1 or 1, the stronger the correlation between the variables. If the variables are independent then the correlation is 0. If two subgroups are structurally equivalent, the correlation is +1.

**Group Activity** The actual activity of the group members is measured in number of interactions in each time window. Shown are for example the *Internal Group Activity* and the *External Group Activity*. The *Min Internal Group Activity* illustrates the number of reciprocal interactions inside the group. Thus, by comparing the *Internal Group Activity* with the *Min Internal Group Activity* the reciprocity inside the group can be determined. The *Min External Group Activity* in comparison with the *External Group Activity* can be interpreted analogously.

### 3.2.2 Group-oriented Visualization of Community Instance Evolution

Research on the interactive, exploratory analysis of graph structures, poses large demands on visualization. Besides the reduction of the number of edge crossings a graph possesses and the emphasis on displaying symmetries of the graph, an even spacing between vertices is desirable. An important aesthetic criterion is a balance of the layout which is related to the individual characteristic of the given graph. Force-directed algorithms are a class of algorithms for drawing graphs with the purpose to position the nodes of a graph so that all edges are of more or less equal length and there are as few crossing edges as possible. At least for medium-sized graphs, the results obtained have usually very good aesthetic properties.

The algorithms by Fruchterman and Reingold [68] and Kamada and Kawai [100] are prominent examples for algorithms in the class of force-directed algorithms. Kamada-Kawai calculates the total balance of the graph, as the square sum of the differences between the ideal distance and the actual distance for all vertices. The process starts with an initial positioning of the vertices and in each iteration, the position of single node is changed. In contrast, Fruchterman-Reingold layouts the network as a whole and is less stable to changes in the initial layout as Kamada-Kawai (see, e.g., [44]).

The main disadvantage of force-directed layout algorithms are computational scaling problems. The typical force-directed algorithms are generally considered to have a running time equivalent to  $\mathcal{O}(n^2 + m)$ , where  $n$  is the number of vertices and  $m$  the number of edges. This is because in each step, the repulsive force between each pair of unconnected vertexes needs to be computed. For graphs containing thousand vertices and more, the running time of the layout computation can become prohibitive. The network we used in the experiments has several hundred vertices and the running time is acceptable.

### Visualizing Community Instances

The graph shown in Figure 3.7 consists of the community instances  $CI$  determined in a given interval  $G_t$  (cf. Section 3.1.2). The Kamada-Kawai layout is applied to display the community instances. The distance between instances is based on their similarity which is calculated as the overlap of members between community instances following Definition 3.4. Two community instances that are detected as similar according to  $\tau_{overlap}$  and  $\tau_{periods}$  are connected with an edge. The layout positions the vertices so that the distance between community instances is closer if they are more similar. The diameter of the vertex represents the size of the community instance logarithmically scaled: The more members, the bigger the node.

After clustering the community instances into communities, the color of the vertices is adapted to represent the membership to a community. Thus, community instances belonging to the same community are shown in the same color.

An example visualization of the graph of community instances using the Kamada-Kawai layout is shown in Figure 3.7. The screenshot on the left shows the layout after determining the community instances connected via edges based on their similarity. On the right side, the layout is shown after 48 clustering iterations. Four communities are detected and vertices belonging to the same community are displayed in the same color.

Please note that this graph visualization does not exhibit any temporal information about community evolution. This will be obtained by transferring the graph in the *community survival graph* representation as discussed next.

### Visualizing the Community Survival Graph

The goal of the community survival graph is to transform the Kamada-Kawai layout in a two-dimensional space which allows the analysis of the temporal evolution of communities. In Figure 3.8 (a) an example of a graph with six communities is depicted and the transformation in a temporal layout is shown in Figure 3.8 (b). The horizontal axis represents the temporal dimension. Each community instance is represented as a rectangle and the size of the rectangle shows the size of the community instance. The position of a rectangle on the x-axis shows in which interval the community instance was detected in.

The temporal evolution of the graph is shown from left to right, where community instances on the left show up in earlier periods. To the right, more recent community instances are displayed. By this, the sequence of occurrences of the community instances can be studied.

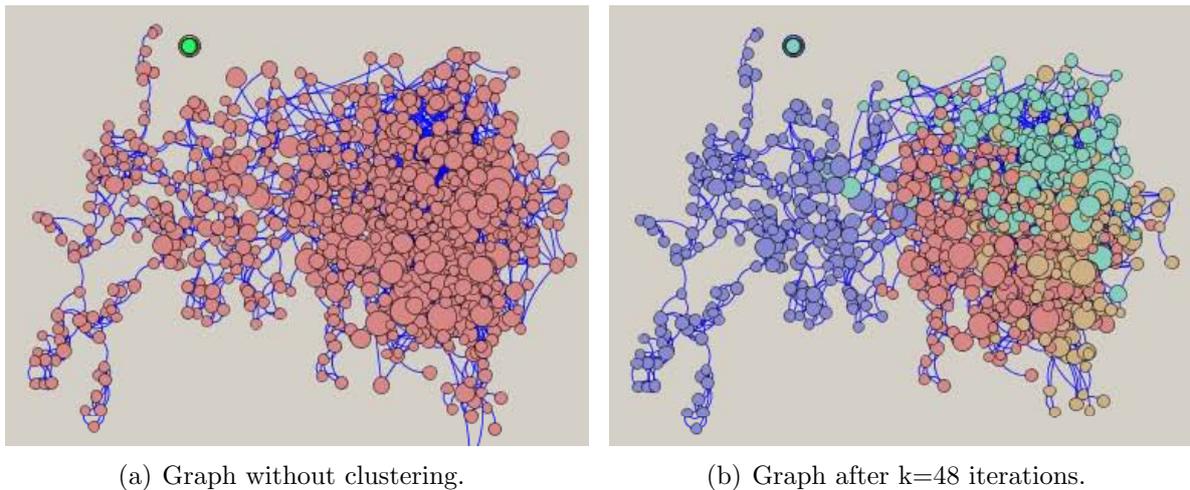


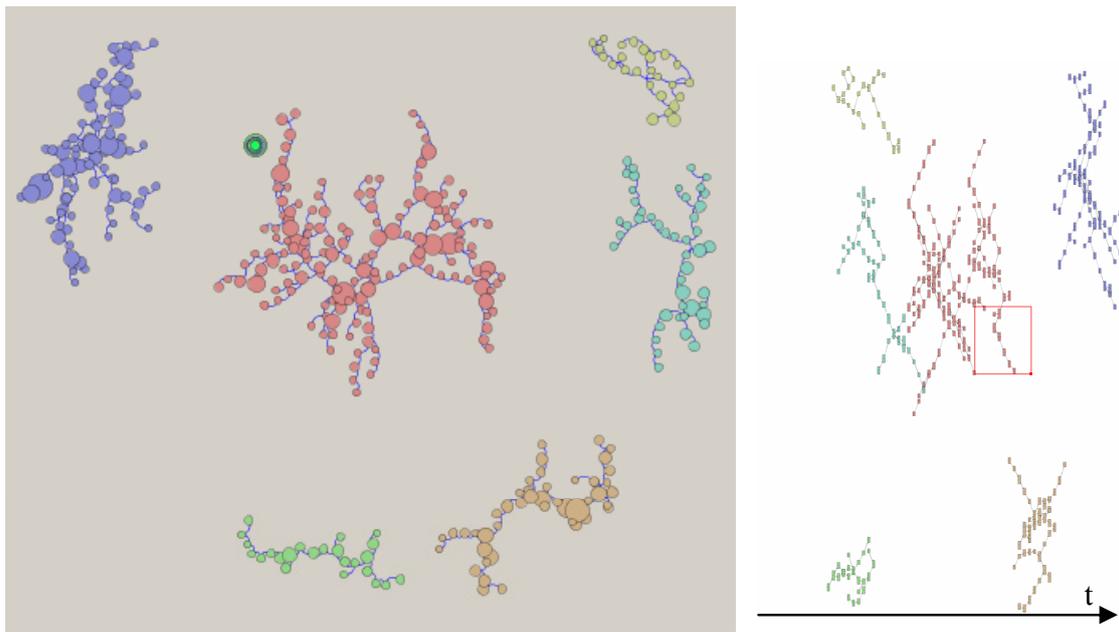
Figure 3.7: Graph of clustered community instances (a) without clustering and (b) after  $k=48$  cluster iterations (screenshots).

The position on the y-axis is taken from the Kamada-Kawai layout. Thus, vertices on the same height in the Kamada-Kawai layout are displayed on the same height in the temporal layout. Taking for example the blue and yellow community in Figure 3.8 (a). Both are in the upper part of the visualization. In the temporal view (Figure 3.8 (b)), both are again in the upper part, but their position on the x-axis has changed. The yellow community was active in earlier periods than the blue community.

Clusters of similar community instances are displayed in the same color. In the example in Figure 3.8, six separated communities are shown each having a different color. The colors of the communities in the temporal view are the same to ease the retrieval.

A cutout view of the graph (zoom in) shows the structure of the communities in more detail (see Figure 3.9 (b)). The edges between rectangles represent similar community instances. If for example a community instance has edges to more than one rectangle in a later period it shows that members have separated to different community instances. These might belong to the same community or members have separated to instances belonging to different communities. The latter case would be considered a split of a community instance.

The transformation of the survival graph to the temporal view in Figure 3.9 allows observing the evolution along the time axis because each community instance is displayed according to the period when it was observed. Now, the colors of the community instances indicate the borders of the clusters. The break between two differently colored clusters may show a period when a change in the community structure occurred. In the example, we see two breaks that separate three communities. The changes in the structure can have different reasons, e.g., the set of participants strongly fluctuated, the interaction behavior of the participants changed, or both. This visualization reveals periods that exhibit structural changes and thus offers a starting point for users to analyze the triggers for such developments.



(a) Kamada-Kawai layout of graph of similar community instances.

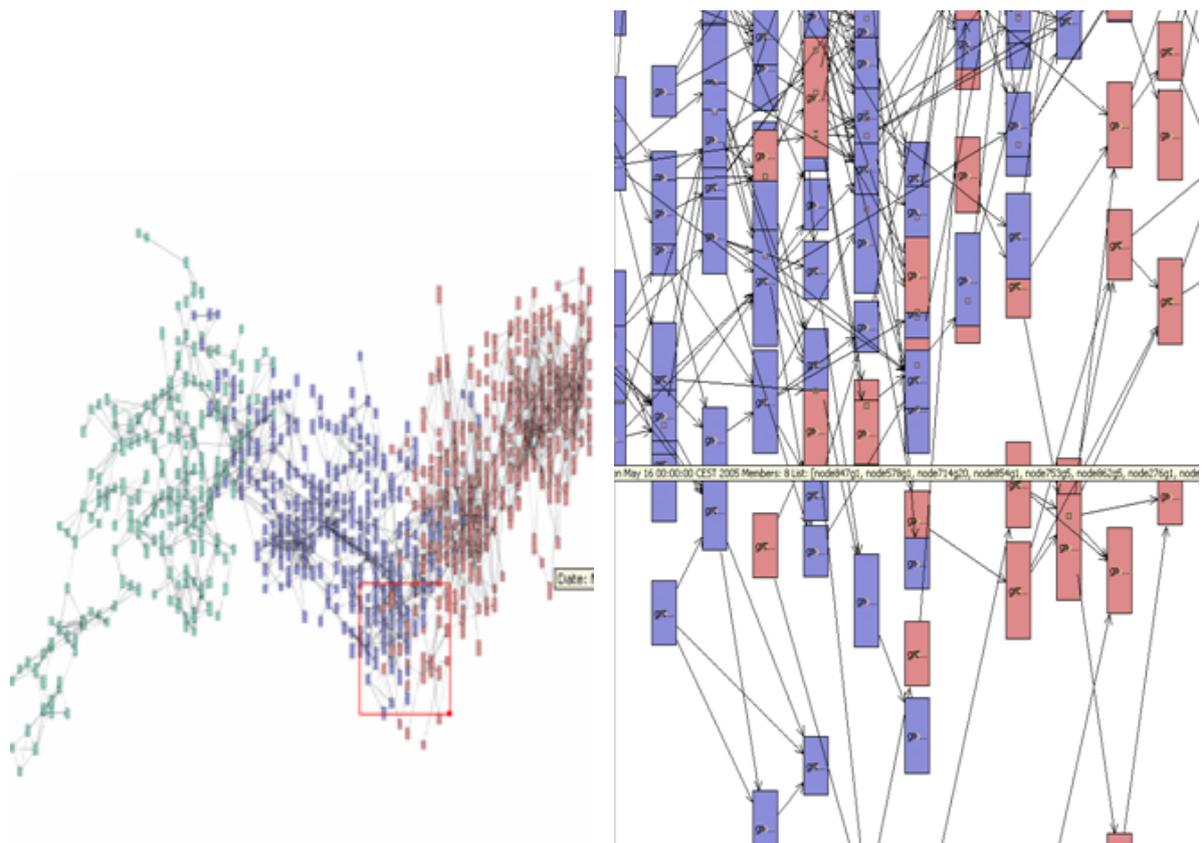
(b) Temporal view.

Figure 3.8: Graph of similar community instances (a) and temporal view in which the x-axis represents the time (b) (screenshots).

In analogy to an editing table which provides an overview of different scenes in a movie, but not of the activities of single actors, the community survival graph gives an overview of the temporal development of communities. It is particularly suited to display the overall (parallel) evolution of communities but it is less suited for the analysis of the behavior and roles of single actors. The incremental clustering approach presented in Chapter 4 is designed for the analysis of the internal evolution in communities.

### 3.3 Application: Studying Communities with Fluctuating Members

In this section, we present results of experiments with social network data and discuss the applicability of the previously proposed community detection procedure. We studied the temporal dynamics of the communication behavior of members of an online student community. The used data set consists of a large number of fluctuating members and we can show that the proposed approach reveals the gaps between community transitions [59].



(a) Community survival graph. The red rectangle indicates the cut-out which is shown on the right. (b) Cut-out view of community survival graph.

Figure 3.9: Temporal view of community evolution (a) and cutout view (b).

### 3.3.1 The IKUS Data Set

In the following, the data set that has been used for the experiments is described. The data set is taken from the online student community *IKUS*. *IKUS* (Die **I**nter**K**ulturellen **S**tudenten) represents an organization of students at the Otto-von-Guericke-University in Magdeburg. The main aim of the group is to assist foreign students during their stay in Germany. Established in 1998, at this time, thirty members of *IKUS* are offering voluntarily help to support foreign students in their needs such as financial questions and housing issues. In addition to their direct support, *IKUS* aims to foster contact between German and foreign students. To realize this, many events such as excursions, barbecues, film and game evenings are organized which are open to all students of the university.

In 2004, *IKUS* launched an online platform, which provides information for (prospective) students from abroad for example about the housing situation or insurance and finance issues. Besides this, the platform provides a Web space for each student. Students can set up an account and are entitled to access several applications. A profile page can be used to provide personal information, to link with friends or to search for language

tandems. Each member has a personal guestbook that can be used by other members to post public messages or comments. Most of the interactions between the students take place in these guestbooks, which work similar to email except that messages are not private but can be read by everybody. At the time of the analysis, the community had about 1,000 members from more than 50 countries.

The data set contains 250,000 guestbook entries which were collected during the observation period of 18 months (June 2004 until November 2005, 75 weeks). Each member is represented as a vertex and two vertices are connected with an edge if at least one bilateral message exchange took place (both members have written at least one message in the other member's guestbook). Members are made anonymous so that it is not possible to trace back the results of the experiments to a certain individual.

#### 3.3.2 Determining Parameters

**Window Length** There is several ways to partition the time axis into time windows. The length of the window size is primarily determined by the goal of the analysis. In the data set, we are interest in middle to long-term relationships between students. Choosing a short window length would reveal rather short-term interactions and very small groups.

To ensure that all periods show some interactions but no concentrations (see, e.g., [154]), we run several experiments with different time window lengths and analyzed statistics based on indices such as the density of the graph, the average degree and the degree variance. We found that for the data set a period length of 14-days exhibits a degree distribution with a rather low standard deviation. Furthermore, this window size allows to observe larger connected communities.

**Degree of Overlap** Since we apply an sliding window with overlaps to reduce the impact of fluctuations on the clustering, the size of the overlap must be determined. When choosing a window size of 14-days, the obtained graph shows in some intervals a high fluctuation. Depending on the time of the year, the number of vertices and edges increases or decreases significantly, resulting in clusterings which are hard to compare. To reduce this impact, we chose a window overlap of 50 percent (one week). Thus, each 14-day interval starts with the last week of the previous interval and the next week with "new" data is added.

**Edge Weights** As discussed in Section 3.1.1, the weight of an edge in the graph  $G_t$  is determined by aggregating the interaction strengths over a predefined number of periods (cf. Equation 3.3). We experimented with different settings and present results obtained when including the last six periods in the weight calculation.

**Similarity of Community Instances** In the experiments, the similarity threshold  $\tau_{overlap}$  is kept stable at 0.5 and  $\tau_{periods}$  is set to 6.

### 3.3.3 Data Set Characteristics

The analysis of the resulting graph revealed that the density of the graph is comparably high. In the *IKUS* network, the shortest path, as an indicator for the expansion of the network, varies between 1 and 4 with a mean of 2.48. Considering the size of the network, this is a low value. Typical for social networks, the *IKUS* graph shows a right-skewed degree distribution (cf. Figure 3.10).

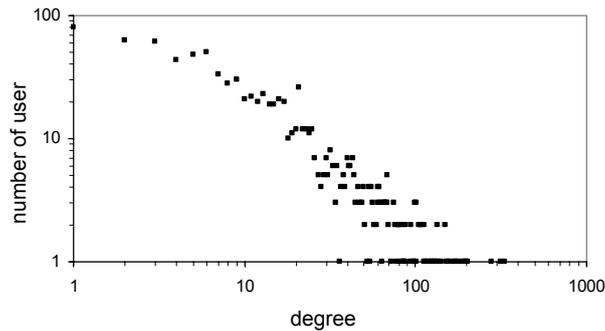


Figure 3.10: *IKUS* data degree distribution (log-log plot)

The clustering coefficient, as an indicator for the transitivity in a network, indicates how many friends of the friends are also friends of a given user. It is often observed that the clustering coefficient is much higher in social networks than in random networks and is therefore considered a good indicator for social network characteristic. In a fully meshed network the clustering coefficient is 1; all friends are friends with each other. In real networks such as collaboration or actor networks [162], WWW sites [1], Internet topology [170], ecological networks [46], economic networks [104], the clustering coefficient is reported to lie between 0.1 and 0.5. The mean clustering coefficient in the *IKUS* network is 0.51, indicating a highly clustered structure.

Without considering edge weights, the obtained network is rather dense as most pairs of people had at least one interaction with each other via their guestbooks.<sup>2</sup> However, it can be assumed that the quality and relevance of the relationship in the network varies considerably. We therefore propose to take the quantity of the relationship into consideration by modeling the graph using weighted edges. The basic idea is to reduce the denseness of the network by removing edges with a low weight.

In the experiment with the above presented data set, we obtained 1,181 community instances with more than three members. Ordered by the number of group members, the plot shows a Zipf distribution. The largest group has 45 members and the average group size is 6.4. Thus, the hierarchical edge betweenness clustering algorithm detects in the data set many small community instances. A dendrogram of a clustering is shown in Figure 3.11.

<sup>2</sup>An often observed behavior in online communities is that people welcome new members upon sign up. Others offer new people a mentor or guide to “show them around”. This results in the exchange of messages but must not necessarily result in a strong relationship.

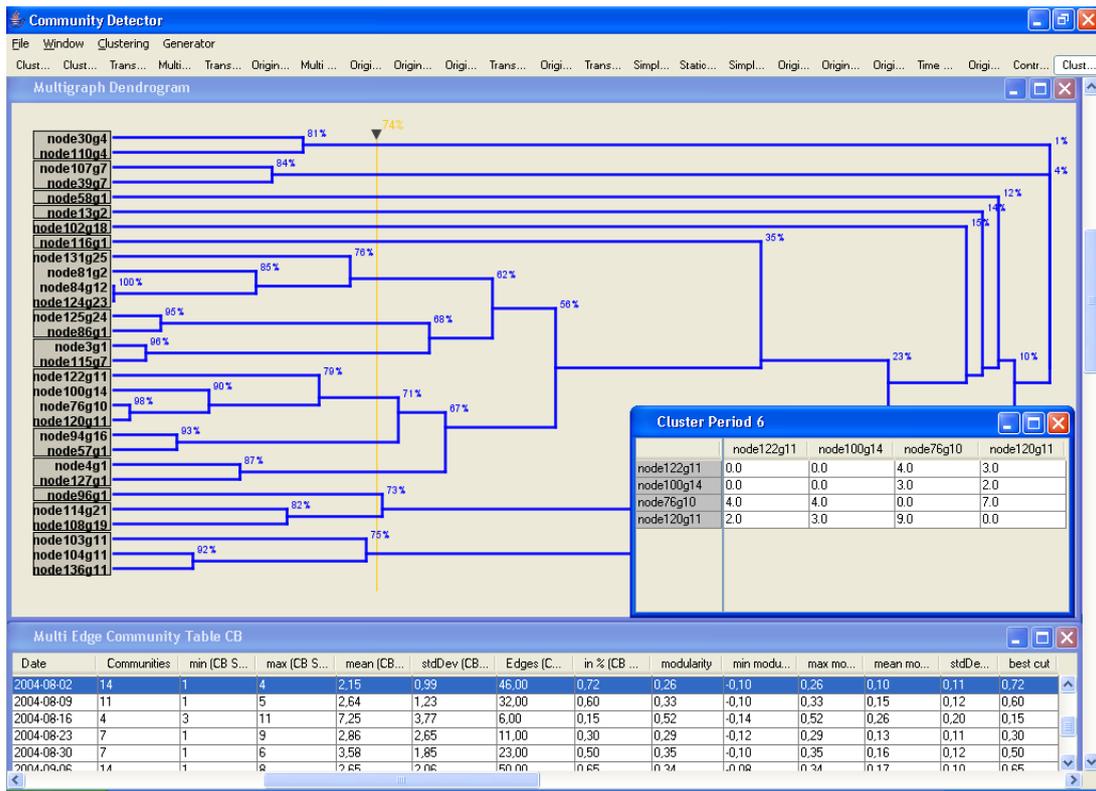


Figure 3.11: Output of hierarchical clustering procedure (screenshot)

In Figure 3.12 we chose a community instance that was detected in April 2004. By comparing the interaction behavior of all subgroup members in different periods, we observe that according to the *fixed correlation coefficient*, the *fixed euclidean distance*, the *cohesion* and the *density*, many subgroups show up only once in the observation period. The *group activity* and the *periodic correlation* indicate, that some of the members are interacting in earlier and later time windows. Thus, most subgroups are only active in one period.

In this data set, no long-term stable subgroups could be found; therefore tracking a community instance over time was not possible. However, by clustering the graph of similar community instances, subgroups are merged to traceable communities.

### 3.3.4 Fluctuating Members

In the following, we show that by clustering the community survival graph we can track evolving communities and detect breaks in this evolution. Choosing the similarity threshold  $\tau_{overlap} = 0.5$ , we obtain a graph of 1025 similar community instances in which only those community instances are included that are separated by maximal six periods ( $\tau_{periods} = 6$ ).

The given graph of similar community instances is iteratively clustered. The first change in the evolution is detected after 27 clustering iterations, the second after 38 iterations

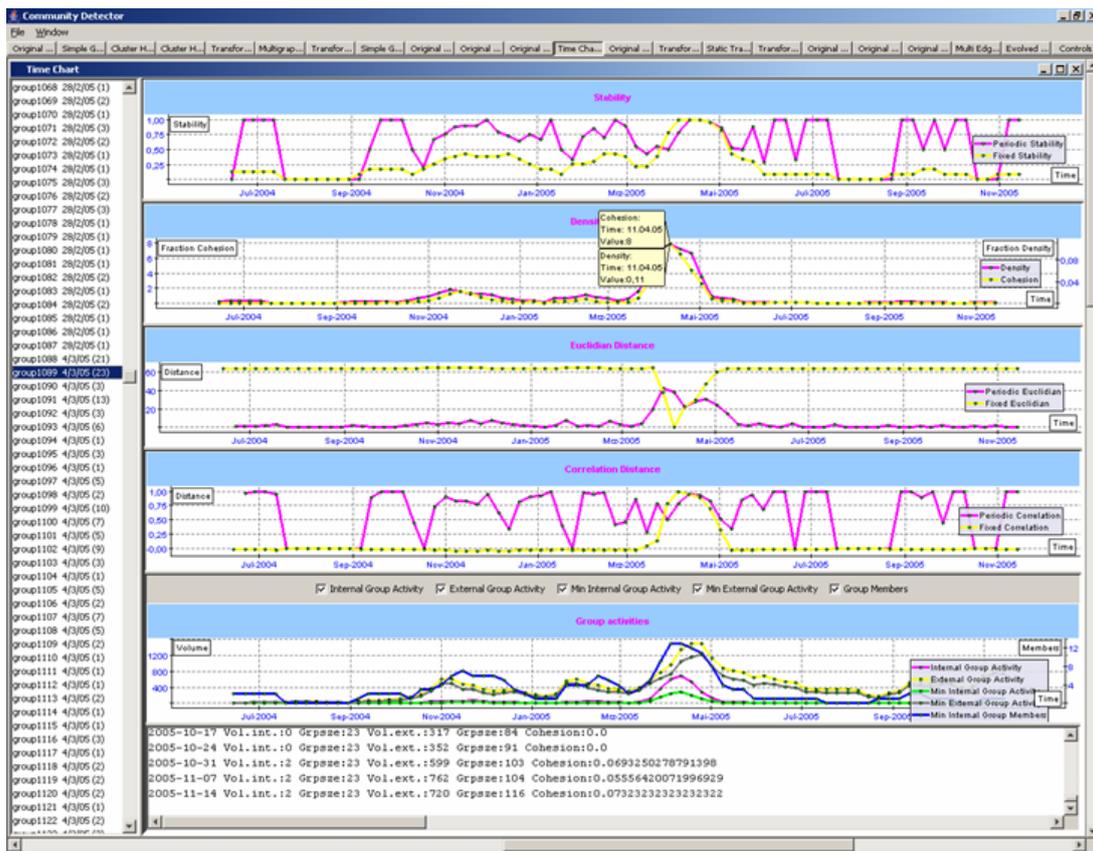


Figure 3.12: Actor-oriented visualization of community instance evolution (screenshot)

and the third after 48 iterations. The results of the graph clustering are shown in Figure 3.13. The figure consists of four screenshots that display the temporal graph after a number of clustering iterations. The screenshot in Figure 3.13 (a) shows the respective graph of similar cluster instances without clustering. The second screenshot (b) shows the graph when the first break in the community evolution was detected. The third screenshot (c) shows the second break and the fourth screenshot (d) the third. The revealed communities are displayed in different colors. To improve the readability, vertical bars indicate discontinuations of communities.

The obtained clustering results can be verified qualitatively in a way that “global events” can be related to the breaks between community clusters. The student community is especially oriented towards the integration of international students. Many international students stay at the University for one or two semesters only and this is usually the period when they participate in the community. Thus, the community members highly fluctuate over time and the highest fluctuation can be observed after the end of a semester and at the beginning of a new one.

This observation about fluctuating members corresponds to the results of the experiments. One break in the community structure can be observed during the summer break in 2005. This change can be attributed to the fact that at the end of the summer term

many students gradually leave the community because their semester abroad ends. Just a few students stay in touch for a longer time after leaving Germany. Thus, the set of participants as well as the interaction behavior change at this time.

A second structural change can be observed at the beginning of the winter term 2005/06. The start of a new semester is characterized by many new members and this is especially the case at the beginning of the winter term as most studies start in winter. We can observe that those new members form many smaller communities very fast. These new groups sometimes grow but many students break up and others join existing or new communities. This period thus exhibits a high fluctuation of members and interactions.

The third break can be attributed to the changing interaction behavior during the Christmas break in 2004. It could be observed that people who were active on the community platform during the Christmas break contacted students that were online too, even though they had not been interacting before. This resulted in a rapidly changing community structure, as many new edges appeared in a very short time.

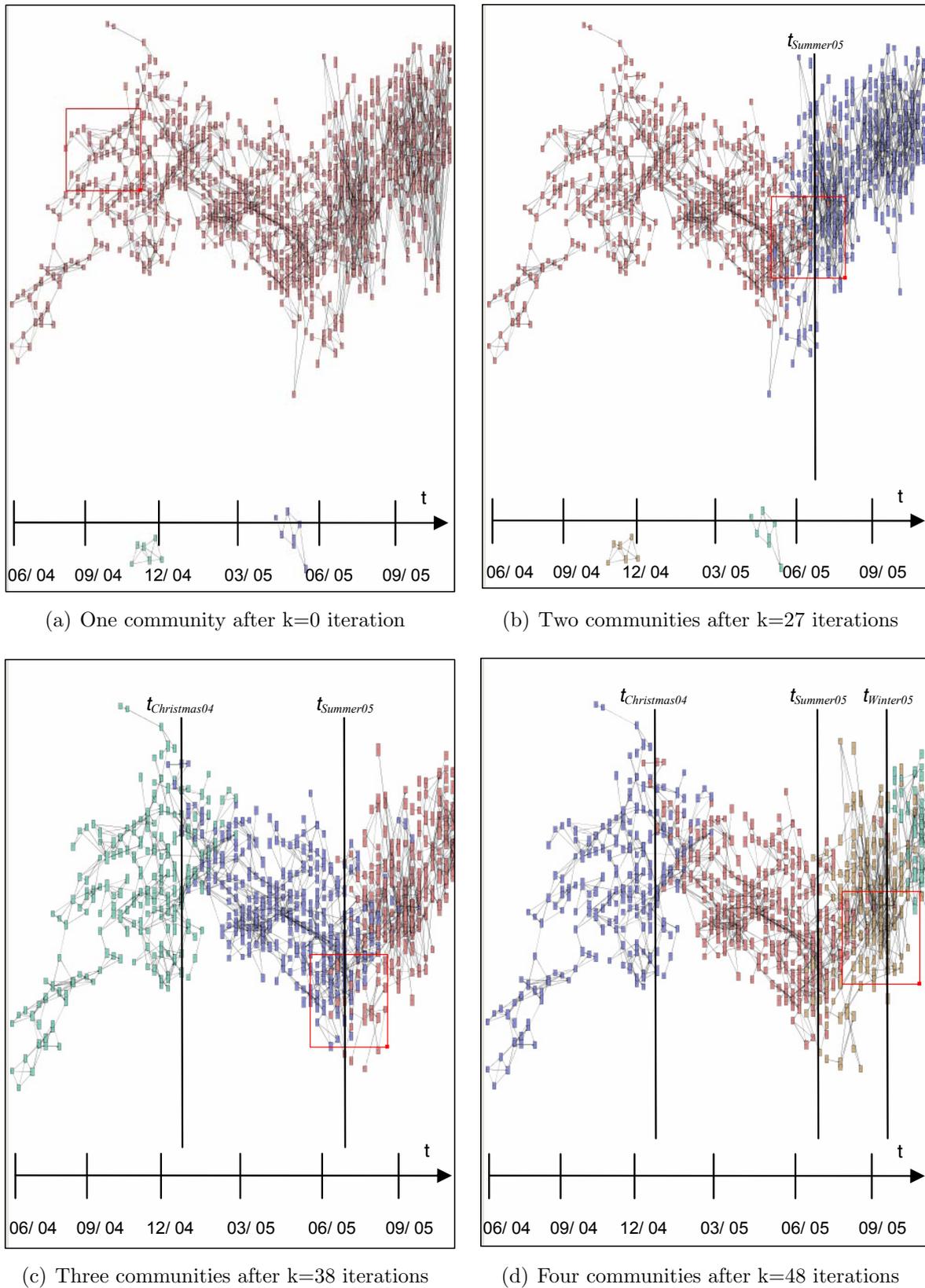


Figure 3.13: Structural breaks indicated by vertical lines in community evolution for  $\tau_{periods} = 6$ ,  $\tau_{overlap} = 0.5$ .

### 3.3.5 Conclusion

We have shown that a community over time must not necessarily consist of the same participants, rather members and interactions are fluctuating. By modeling a community as a persistent structure in a graph of interactions, we are able to detect communities even if parts of the membership fluctuate. Furthermore, we observe breaks between communities which indicate structural changes that might help to determine the impact of external or internal influences on the dynamics in communities.

## 3.4 Application: Representativeness of Communities for Members

Active participation of a person in a community is a powerful indicator of the person's interests, preferences, beliefs and (often) social and demographic context. Therefore, in [60] we argue that knowledge about the membership of an individual to a community is part of a user's model and can contribute to tasks like personalized services, assistance and recommendations. However, if a member is inactive at the time a recommendation should be made and the membership to a community is based on interactions in the past, to what extent is a community still representative of the interests of an inactive participant? The proposed model of a community is well suited to gain insights to this question since we observe a community as an evolving social structure comprised of static community instances and study the effects of member fluctuation.

### 3.4.1 Users within Evolving Communities

Active users contribute to the similarity of community instances, because they launch interactions with other users and thus increase their proximity to them. This agrees with the findings of [10], where it is stated that active participants of the studied community shape its topics of interest. Moreover, this extended notion of "survival" for a community instance allows us to tolerate short times of inactivity over the time span. Even the most active users inside a community may be temporarily inactive.

Following the community detection procedure describe in Section 3.1, we build a community survival graph. We have studied the issue of community relevance for a user participating in multiple communities, whereupon periods of active participation may be followed by inactivity. From the viewpoint of community evolution, this allows us to detect structural gaps, i.e. the discontinuation of a community's life span. From the viewpoint of community membership, the model allows us to assign for a certain period users to a most relevant community. The experiments with the IKUS data delivered some first insights on the number of clusters (dense subgraphs of similar community instances) to which users participate actively or inactively.

In each time period a user  $u$  belongs to one community instance. Within a community, i.e. a cluster of community instances upon the community survival graph, a user may appear more than once, as the result of interactions at different times. Obviously, a

user may belong to more than one community. For each community  $\chi$  and user  $u$ , the relevance of  $\chi$  for  $u$  is reflected in the ratio of his activity inside the community towards his overall activity. To compute this ratio, we first define the involvement of a user within a community instance:

**Definition 3.8** *Let  $u$  be a user and  $C$  be a community instance discovered in time period  $t$ . The “involvement” of  $u$  in  $C$  is the number of interactions that the user has performed inside this community instance in period  $t$ :*

$$\text{involvement}(u, C) = |\{e \in E_t \cap C : e = (u, v) \vee e = (v, u)\}| \quad (3.12)$$

Similar to the intensity of interaction specified in Definition 3.1, user involvement is also based on the number of exchanges performed. However, the intensity is used to discover the community instances upon the graph of interactions  $G_t$ , while the involvement is computed upon already derived instances.

On the basis of the involvement value for community instances, we can compute the following indicators of a user’s behavior towards communities:

**Definition 3.9** *The “participation” of user  $u$  in a community  $\chi$  with lifetime  $[t_{min}, t_{max}]$  is defined as the fraction of the involvement in community instance  $\chi$  divided by the involvement of the user in other community instances  $\Xi$  during the lifetime of the community:*

$$\text{participation}(u, \chi) = \frac{\sum_{C \in \chi} \text{involvement}(u, C)}{\sum_{C \in \Xi} \text{involvement}(u, C)} \quad (3.13)$$

User participation may be active or inactive, since the user may have non-zero involvement in some community instances and zero involvement value on other instances of the same community.

**Definition 3.10** *For a given time period  $T$ , the relevance of a community  $\chi$  for a user  $u$  is defined as:*

$$\text{rscore}(u, \chi, T) = \frac{\sum_{C \in \chi \wedge C \in \Xi_T} \text{involvement}(u, C)}{\sum_{C \in \Xi_T} \text{involvement}(u, C)} \quad (3.14)$$

where  $\Xi_T$  are all community instances during  $T$ . Thus, the relevance score is the fraction of the involvement of user  $u$  in community  $\chi$  during the given time period  $T$  divided by the involvement in other community instances during  $T$ . By this, the relevance of communities with lifetime outside  $T$  is zero.

The representative community for a user  $u$  in a period  $T$  is the community for which the  $\text{rscore}()$  value is maximum. We denote this community as  $\text{community}(u, T)$ . This definition allows us to identify a representative community for each user during a time period, even if the user was not active during the whole time period.

### 3.4.2 Relevance of a Community for a User

As discussed in Section 3.3, the IKUS network exhibits a high membership fluctuation. From the observed community instances we have selected one small community and studied the behavior of its users during its lifetime, using the measures as proposed above. In Figure 3.14 (left side) the  $k$ -communities of the social network are shown after  $k = 48$  iterations. We observe four communities that are depicted in the visualization in four different colors. To enhance visibility, the borders of the communities are also indicated by vertical lines. After further iterations we determine a small community of five community instances that is separated from the other communities (the respective instances are encircled in the upper part of the right side of Figure 3.14 ( $\chi = blue$ )). The community *blue* has a lifetime of four periods ( $t_{start} = 38, t_{end} = 41$ ).

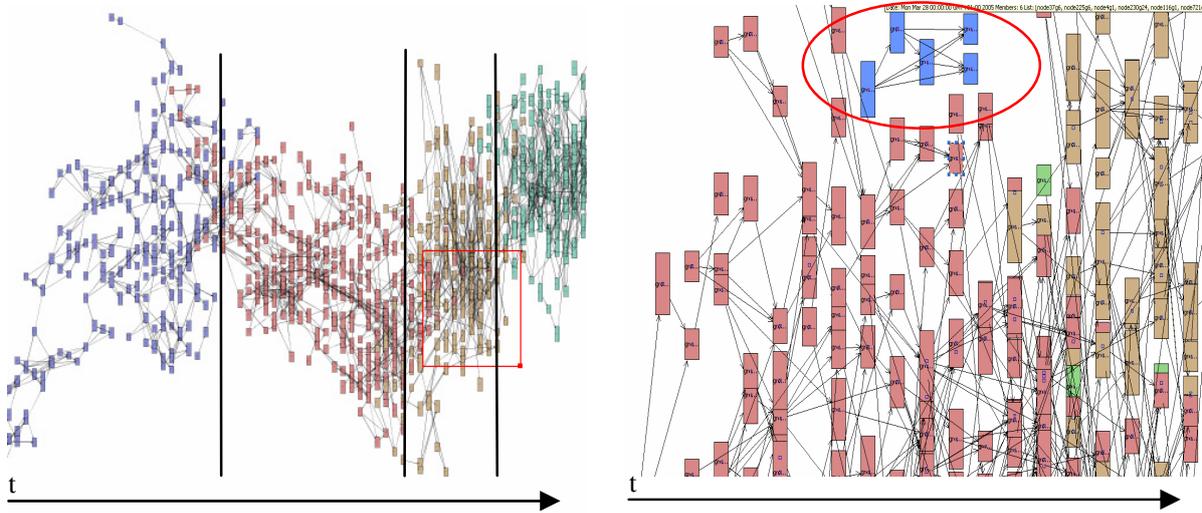


Figure 3.14: Left side:  $k$ -communities after 48 iterations. Right side: Separated  $k$ -community after further iterations. The encircled community instances in the upper part named in the following community “blue”.

The first instance of *blue* in  $t = 38$  consists of eleven members. The second instance consists of eight members: Five of the eight users were already participating in the first instance and three members are new to the community. The third instance in  $t = 40$  consists of seven members. All of them participated in  $t = 38$  and/or in  $t = 39$ . In  $t = 41$  two instances are assigned to *blue*. Both instances have three members and all members have already been participating in previous instances.

To assess the relevance of a community to a given user we observe the interaction behavior of the community members inside the community as well as interaction with members from other communities during the lifetime of the community under observation. We thus fix the time period  $T$  to the lifetime of *blue*, i.e. to the period 38-41, and measure the *involvement* and the *participation* for a subset of six members of the community. Since the observation period equals the lifetime of  $\chi$ , the measure for the participation

equals the *r*score measure. We can therefore use the participation measure to assess the relevance of a community to a user.

A user, who is assigned to a community, is most likely represented by this community; at least during the lifetime of the community he is assigned to. However, the involvement in the assigned community may vary over the lifetime of the community as shown in Table 3.1. No user is involved in community *blue* in all periods. Some users are only involved once (see  $v_{695}$ ,  $v_{685}$ ). Since we used an overlapping sliding window to partition the time axis, to smooth out gaps in the interaction behavior between periods, members might be assigned to a community instances, even though they had no interaction in the respective period because of former interactions. Thus, it is of interest to consider the involvement in other community instances. In the presented example, both users are not involved in other community instances, therefore it can be assumed, that they are still represented by community *blue* even in periods they are not active.

Table 3.1: Involvement and participation of six members of the community ‘blue’.

Type	User	$involvement(u,blue)$	$involvement(u,"parallel\_to\_blue")$
A	$v_{695}$	0,57	0 (not assigned to any other community instance in lifetime of <i>blue</i> )
A	$v_{685}$	0,33	0 (not assigned to any other community instance in lifetime of <i>blue</i> )
B	$v_{700}$	0,32	0 (not assigned to any other community instance in lifetime of <i>blue</i> )
C	$v_{525}$	0,45	0 (not assigned to any other community instance in lifetime of <i>blue</i> )
C	$v_{368}$	0,17	0 (not assigned to any other community instance in lifetime of <i>blue</i> )
C	$v_{58}$	0,28	In interval 41 assigned to another community instance <i>o</i> ; $involvement(u,o) = 0,38$

Users who are assigned to different community instances during the lifetime of the community *blue* might be still best represented by this community in the periods they are involved in *blue*; however users may not be represented best by this community over the whole period of their interaction. As shown in Table 3.1, users may change their interaction behavior during the lifetime of a community and switch to another community (see user  $v_{58}$ ). Thus, the participation in the assigned community is not maximal in all periods. User  $v_{58}$  was assigned to the community *blue* but shows in the last period of the community’s lifetime higher involvement in another community instance. This is reflected in the participation of the user for both communities: It is higher for community *o* (0.38 compared to 0.28 in  $\chi$ ). Thus, the respective time period should be considered when making a decision which community best represents a user. The relevance of a community for a user in a certain time period can therefore be assessed by determining the highest relevance score.

### 3.4.3 Conclusion

We studied the issue of community relevance for a user participating in multiple communities, whereupon periods of active participation may be followed by inactivity. We have modeled communities as dynamic structures comprised of static “community instances”, each of them characterized by its active users. From the viewpoint of community evolution, this allows us to detect “structural gaps”, i.e. the discontinuation of a community’s lifetime span. From the viewpoint of community membership, the model allows us to assign for a certain period users to a most relevant community. We have experimented with a real community that exhibits membership fluctuation and has known structural gaps. The experiment delivered some insights on the number of clusters (dense subgraphs of similar community instances) to which users participate actively or inactively.

## 3.5 Related Work

In the following section, we review related work regarding models for the analysis of dynamic networks and the visualization of communities.

### 3.5.1 Modelling Community Dynamics

**Framework by Berger-Wolf and Saia** Berger-Wolf and Saia propose a framework for the analysis of dynamic social networks [15]. The basic idea is, to represent the temporal information about changes in the model of the network. The authors assume that the input data for each time step is a set of *partitions* where each partition is a set of disjoint groups. Furthermore, a similarity measure which decides on the similarity of two groups must be given. Based on this input a concept called *metagroup* is defined. This metagroup is a sequence of groups such that (i) no two groups are in the same partition and the groups are ordered by the partition time steps and (ii) the consecutive groups in the metagroup are similar in that a similarity threshold is reached. Based on this graph the authors propose several statistical measures about the metagroups such as finding the largest or most persistent group.

In [158], the framework was used to detect communities as latent concepts in the graph of connected groups. The authors postulate five properties that describe the expected behavior of individuals and groups:

1. In each time step, every group is a representative of a distinct community.
2. An individual can be a member of only one group at a time while the membership may change over time.
3. An individual does not switch between communities frequently.
4. If an individual does change frequently, it will be an oscillation among a rather small number of communities; if he does change between many, he is not a member of any community.

5. An individual is frequently a member in the groups representing the same community.

A graph is built which consists of one vertex for each individual and one vertex for each group at each time step. Edges are added between individuals if they are active in two subsequent time steps and between individuals and groups whenever the individual is a member of a group. A *community interpretation* of the graph is calculated. The problem is modeled as a graph coloring problem and a community interpretation is deemed valid if and only if for each time step, no two groups share the same color. The expected properties are used to penalize violations with costs. The optimization problem is to find a community interpretation that is (i) valid and (ii) minimizes the total costs. Since the optimization problem is  $\mathcal{NP}$ -complete, the authors propose heuristic algorithms.

Even though the model as well as the community extraction method are different from ours, the perception of what constitutes a community is very similar. The concept of a *group* is basically the same as the concept of a *community instance*. It consists of individuals interacting with each other and at each time step an individual can be a member of only one group. The obtained graph can be interpreted similarly to the *community survival graph*. The set of groups forming a *community interpretation* resembles the *cluster of similar community instances*. While Berger-Wolf et al. obtain communities by minimizing costs resulting from penalties due to ‘offenses’ against rules, we obtain them by clustering the graph of similar community instances.

**Evolution Graph by Mei and Zhai** The presented community survival graph is similar to the *evolution graph* proposed by Mei and Zhai [120]: In both cases, nodes represent clusters found at different time periods, while edges denote that the clusters have a certain similarity. However, Mei and Zhai perform soft clustering upon documents and study matching cluster labels rather than the clusters themselves. In their work, the graph is used to predict the evolution of labels. We are rather interested in discovering superordinate community structures in the survival graph and relate them to their members.

**Evolutionary Clustering** A recent development in the area of data mining named “evolutionary clustering” might become of interest for community dynamics mining (see, e.g., [33, 35]). An evolutionary clustering should optimize two potentially contradicting criteria: first, the clustering at any point in time should remain faithful to the current data as much as possible; and second, the clustering should not shift dramatically from one time step to the next. In [33], a framework has been proposed and evolutionary versions of two clustering algorithms within this framework (k-means and agglomerative hierarchical clustering) have been discussed. The applied similarity measure to construct the clustering captures the history of the data to some extent. This “local similarity” is combined with a “temporal similarity” which is given by the correlation of the two time series to a “total similarity”. Given this similarity measure, at each time step  $t$ , the clustering algorithm produces a clustering and the user must specify a quality function that returns the quality of the clustering. The user must also provide a function that returns the historical cost of

the clustering at time  $t$ . The total quality of a cluster sequence is then calculated given these two functions. So far, this approach has not been applied to social networks.

### 3.5.2 Visualizing Community Dynamics

Many software tools are available for the visualization of static graphs such as NetMiner<sup>3</sup>, Pajek<sup>4</sup> [44], SONIVIS<sup>5</sup>, UCINET (NetDraw)<sup>6</sup>, visone<sup>7</sup> [26]. Most tools provide an interactive integration between visualizations and analytics allowing for an exploratory network analysis of static graphs. An overview of available programs and software toolkits can be found in [92]. Surveys about different aspects of graph visualization can be found in, e.g., [13, 102].

Software tools like SoNIA [123] and TeCFlow [74] support visualization of social networks across time by creating movies of graphs. They both work on the vertex and edge level and thus visualize changing behavior between single actors rather than the evolution of the community they belong to. It is thus not possible to explore the dynamics of subgroups. However, even though the individuals and the interactions between individuals change over time, the community might be considered by an observer as the same. In contrast, we propose to observe the temporal changes on the community level to allow for an exploration of community dynamics.

## 3.6 Concluding Remarks

As shown in the last section, the presented approach provides insights into the evolution of dense substructures in graphs. These insights are valuable for community providers to manage community building, for example to foster intra-organizational knowledge sharing or for online shops to make recommendations to users.

**Community Definition** The community here is defined as an evolving structure and we offer a method that can provide insights into the evolution of dense substructures. Furthermore, we can identify triggers for structural changes. However, due to the definition of the object to observe, namely a community is a constellation that is observable over time, internal changes in the community structure are not easily observable. The approach takes a macroscopic view on communities that is especially useful if parallel evolutions of communities are of interest for the observer. To observe the internal evolution of a community, we present in the following Chapter 4, a procedure that takes a microscopic view on communities.

---

<sup>3</sup><http://www.netminer.com/>

<sup>4</sup><http://pajek.imfm.si/>

<sup>5</sup><http://www.sonivis.org/>

<sup>6</sup><http://www.analytictech.com/>

<sup>7</sup><http://visone.info/>

**Complexity of Modularity** The complexity of the chosen edge betweenness clustering procedure makes an application of this methods to larger data sets (several thousand nodes for example) not feasible. Finding the shortest paths to calculate the edge betweenness centrality as well as optimizing the modularity are expensive procedures. The betweenness centrality is traditionally determined in two steps: (i) compute the length and number of shortest paths between all pairs of vertices and (ii) sum all pair-dependencies.

The complexity of determining betweenness centrality is dominated by step (ii), i.e. the  $(n^3)$  time for the summation of pair-dependencies. Using Brandes' algorithm for betweenness centrality reduces time complexity for weighted graphs to  $\mathcal{O}(nm + n^2 \log(n))$ , while  $n$  denotes the number of vertices and  $m$  the number of edges [22]. Maximizing modularity is  $\mathcal{NP}$ -hard and therefore the runtime of exact algorithms is prohibitive for large networks (see, e.g., [23]).

**Limitations of Modularity** There are also questions about the limits of the modularity as a measure for good community structures. Fortunato and Barthélemy reported in [63] that even weakly interconnected graphs, which have the highest possible density of internal edges (complete subgraphs), would be merged if the network is sufficiently large.<sup>8</sup> For this reason, even well defined clusters (in our case community instances) might not be identifiable when optimizing modularity in large networks.

**Limitations of Hierarchical Clustering** The algorithm is not appropriate for data sets with a high number of noise objects as it is not able to remove them efficiently. In a hierarchical clustering, all noise objects are clustered as singletons. We therefore propose in the next chapter a density-based approach which is capable to detect noise in linear time.

**Utility of Similarity Function** If we define a community as a set of actors that interacts closely over time, the chosen matching function is reasonable. However, a community might not necessarily be defined by a certain set of individuals but by a set of individuals with a certain characteristic or similarity. In one application, which we present in the next Chapter (see Section 4.5, the aim is to track user groups with a similar music listening behavior. Here it is not required that these members are always identical; a community 'survives' to the next timestep if a closely connected subgraph exhibiting users with the

---

<sup>8</sup>As discussed in Section 2.4.2, modularity compares the number of edges inside a cluster with the expected number of edges that one would find in the cluster if the network was randomly built. A random network has the same number of nodes and each node has a stable degree, but edges are randomly attached. This model implicitly assumes that the probability that a node gets attached to any other node of the network is the same for all nodes. However, taking for example large social networks, this assumption is not reasonable as a person's range only includes a small part of the network. The model implies moreover that the expected number of edges between two clusters decreases if the size of the network increases. If a network gets large enough, the expected number of edges between two clusters in the random network may be smaller than one. In this case, a single edge between the two clusters would be interpreted as a sign of a strong connection between the two clusters. This would lead to a merge of both, independently of the clusters' structures.

same music preferences is existent. The utility of the chosen matching function should be assessed regarding its alignment with the observation objective.

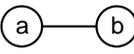
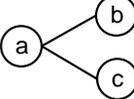
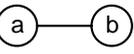
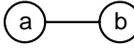
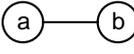
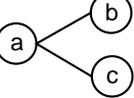
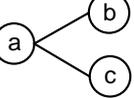
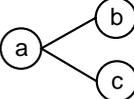
**Stability of Clustering** We observed that the clustering method is sensitive to perturbations of the input data; in particular using time windows poses problems. A varying number of vertices leads to significantly different clusterings. For example, if the number of actors is low, the cluster algorithm might return communities even though only few actors are loosely connected because the data set in this time window is too small. For static analysis, changes are acceptable, however since we rely on tracking changes of cluster over time we must be able to re-identify clusters. If the clusterings are sensitive to changes for example in the number of nodes, it is difficult to distinguish between real changes versus accidental changes. We reduce this risk by using overlapping time windows and looking out for a similar degree-distribution over time.

The same observation has been reported by Hopcroft et al. who use an agglomerative clustering for community detection [89]. The authors looked for what they call ‘natural communities’; relative robust clusters in clustering trees which are obtained under minor perturbations of the input data. In [90], the author compared the community structure of two snapshots and determined changes such as the emergence of new communities.

**Snapshot-Approach** By taking snapshots of the graph instead of aggregating all data, a study of the temporal dynamics becomes possible. In Table 3.2, an example for the differences of both approaches is shown. The first two rows display the evolution of two graphs over four periods. In each period, a snapshot illustrates the current situation, e.g., the exchange of messages between actors. By looking at each snapshot separately, no information about the evolution of the graph can be obtained. However, it is possible to compare both graphs with each other. If the observed interactions are aggregated over all periods, the aggregate graph would look the same for both evolutions as shown in the third row. The information about the evolution of the graph is lost. The aggregate would suggest, that actor  $c$  belongs to a group with actors  $a$  and  $b$ . However, if we compare this with the two evolutions, the sequence  $E_1$  shown in the first row indicates that  $c$  was a member of the group in period 2 but left the group and the sequence  $E_2$  depicts that  $c$  joined the group in period 3 and is still a member of this group. On the other hand, the aggregates of these evolutions do not allow for an assessment of the dynamics of the graphs over time.

Even though, the snapshot approach is superior to the aggregates, the ongoing change is not captured and can only be determined by comparing clusterings obtained in different snapshots. However, this procedure brings about some problems which have been discussed in the last paragraph. It is therefore desirable to overcome this shortcoming by considering the dynamic aspects of social networks by adapting the clustering to temporal changes.

Table 3.2: Temporal scenarios and the implications for temporal analysis depending on the observation technique.

	Period 1	Period 2	Period 3	Period 4
Snapshots of graph evolution $E_1$				
Snapshots of graph evolution $E_2$				
Aggregates for both evolutions $E_1$ and $E_2$				

### Summary

The observations from our analysis of community dynamics applying the edge betweenness clustering algorithm revealed some drawbacks:

- Due to its high time complexity it is unfeasible to use the algorithm for large networks with several thousand nodes.
- A measure is needed to determine the cut-off level in the dendrogram.
- Temporal evolution can only be observed by aggregating data in time windows; this makes it difficult to compare detected subgroups in different time windows over time.

Furthermore, since social networks usually have a high number of actors that do not contribute (*noise objects*), for the analysis of large dynamic networks a fast algorithm that efficiently deals with noise objects is required. In the following Chapter 4, we present DENGGRAPH-IO, a density-based incremental clustering algorithm that is designed to deal with large dynamic data sets with noise.

# 4 Community as a Subgraph that Evolves Over Time

In this chapter, we define a community as a group of individuals that evolves over time. We propose a clustering technique to detect communities and an incremental procedure to update the clusterings based on temporal changes, allowing for a microscopic view on the evolution of subgroups in networks. We first present the density-based community discovery algorithm DENGGRAPH (Section 4.1) and its incremental extension to efficiently handle incoming graph updates (Section 4.2). Furthermore, cluster evaluation measures presented in Chapter 2 are extended to evaluate the obtained graph clusters regarding their quality and stability (Section 4.3). Afterwards, results of experiments with the *Enron* data set are presented (Section 4.4) and an application to data obtained from the music community platform *Last.fm* is discussed (Section 4.5). Thereafter, we discuss the observed characteristics of the DENGGRAPH clustering algorithm (Section 4.6). The chapter closes with a discussion of the related work (Section 4.7) and concluding remarks (Section 4.8).

## 4.1 DENGGRAPH: Density-based Graph Clustering

It is not clear that for all networks a single optimal partition regarding modularity or any other clustering metric can be found. That is, in many instances the same network can have multiple partitions that fit equally well. Second, while most recent work has focused on mappings that are exhaustive and mutually exclusive, i.e every actor is assigned to one group, there is often good reason to assume that

- (some) people are not members of any community, so-called noise objects
- (some) people belong to multiple communities resulting in overlapping groups.

Inspired by the algorithm DBSCAN for spatial data proposed by Ester et al. [50], the intention of DENGGRAPH is to find clusters of similar nodes in graphs with many noise objects. Furthermore, we present an extension of the core algorithm to allow for overlapping clusters. Before introducing the algorithm, we present the clustering preliminaries (cf. also Section 2.4).

**Preliminaries** Let  $G = (V, E)$  be a weighted, undirected graph. A *cluster*  $C \subseteq V$  is a subset of vertices. Each cluster represents a collection of vertices that are similar to one another and are thus regarded as a group.  $G[C]$  is the graph induced by  $C$ . A *clustering*

$\zeta = \{C_1, \dots, C_k\}$  of  $G$  is a set of clusters. The cluster procedure proposed in the following is a non-partitional clustering, i.e. the clustering does not necessarily contain all vertices. Rather it is designed to efficiently remove vertices that are not closely connected to other elements in the graph. These objects are not clustered but labeled as so-called *noise objects*.

### 4.1.1 Density-Reachability in a Graph

As we have discussed in Section 2.3.3, density-based clustering applies a local optimization criterion. Clusters are regarded as regions in the graph in which the nodes are dense and which are separated by regions of low node density. To detect regions of higher density, DENGGRAPH computes *neighborhoods* which have a given radius ( $\epsilon$ ) and must contain a minimum number of nodes ( $\eta$ ) to ensure that the neighborhood is dense. A node having such a neighborhood is termed a *core node*. Nodes that have no such neighborhood are either *border nodes* if they are in the neighborhood of a core node or *noise nodes*.

To build a cluster, DENGGRAPH traverses the graph by randomly picking nodes and places all *density connected* (cf. Figure 4.1) nodes it encounters to the same cluster. If a node is not density connected to the nodes seen thus far, it is assigned to the next cluster candidate. Not each node becomes member of a cluster: If a node does not have an adequately dense neighborhood with respect to  $\epsilon$  and  $\eta$  and is not density connected to any other node, then it is termed a *noise node* and its cluster candidate is dropped [57].

To cluster similar data vertices into groups a measure of *similarity* or *dissimilarity* is needed. The similarity between two vertices is a numerical measure of the degree to which both objects are alike. Accordingly, dissimilarity is a numerical measure of the degree to which both objects are different. The more alike the objects are, the higher is the similarity and the less similar the vertices are the higher the dissimilarity. Usually, when speaking of dissimilarity, the term *distance* is used. Frequently used distance functions are the Euclidean distance and the cosine distance.

To group actors according to their closeness in graph structures, we need to define a function that determines the distance between two actors. Similarity or closeness in social networks depends on the relationship between actors. We therefore define the distance based on the semantics of the relationship. In the case of the *Enron* data set, we assume that the closeness of actors is reflected in the number of interactions between them and define a distance function based in the frequency of interaction (cf. Section 4.4.2). For the *Last.fm* data we define a distance function based on the similarity of user profiles (cf. Section 4.5.2).

The approach discussed in the following works with any distance function which can be chosen depending on the data set and the goal of the analysis, the  $\epsilon$ -*neighborhood* of a vertex  $u$  is defined as follows:

**Definition 4.1** *The  $\epsilon$ -neighborhood of a vertex  $u$  denoted by  $N_\epsilon(u)$  is defined by*

$$N_\epsilon(u) = \{v \in V \mid \exists(u, v) \in E \wedge \text{dist}(u, v) \leq \epsilon\} \quad (4.1)$$

*and produces a set set of vertices in the  $\epsilon$ -neighborhood of  $u$ .*

The definition of a  $\epsilon$ -neighborhood leads to the definition of two types of vertices in a cluster: *core vertices* and *border vertices*. A vertex that does not belong to any cluster is called *noise vertex*.

**Definition 4.2**  $u \in V$  is a “core vertex” if and only if  $|N_\epsilon(u)| \geq \eta$ , where  $\eta$  denotes the minimal number of neighbors required. If  $|N_\epsilon(u)| < \eta$ , then  $u$  is a “noise vertex”, unless there is a core vertex  $v$  such that  $u \in N_\epsilon(v)$ . Then,  $u$  is a “border vertex”.

We use the notion of core vertices to define *reachability* among vertices. We define *directly density-reachable*, *density-reachable* and *density-connected* vertices. However, we do so on the basis of  $\epsilon$ -neighborhoods rather than using a distance function over the whole set of vertices. The concepts of direct density reachability, density reachability and direct connectivity are illustrated in Figure 4.1.

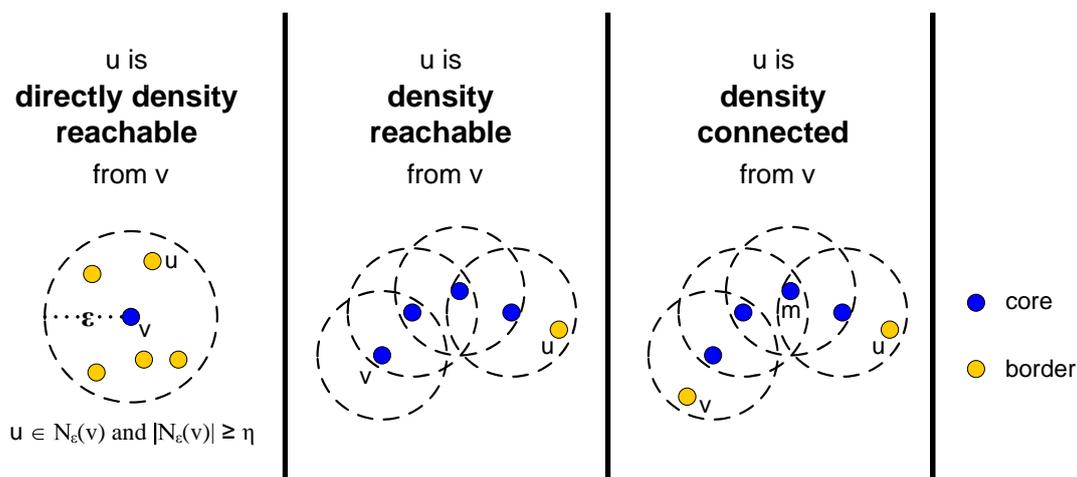


Figure 4.1: The concepts *directly density reachability*, *density reachability* and *density connectedness* to determine whether nodes are density connected.

**Definition 4.3** Let  $u, v \in V$  be two vertices.  $u$  is “directly density-reachable” from  $v$  within  $V$  with respect to  $\epsilon$  and  $\eta$  if and only if  $v$  is a core vertex and  $u$  is in its neighborhood, i.e.  $u \in N_\epsilon(v)$ .

Directly density-reachability is a symmetric relation for pairs of core vertices. If a core and a border vertex is involved, it is not symmetric (cf. Figure 4.1 (left)).

**Definition 4.4** Let  $u, v \in V$  be two vertices.  $u$  is “density-reachable” from  $v$  within  $V$  with respect to  $\epsilon$  and  $\eta$  if there is a chain of vertices  $p_1, \dots, p_n$  such that  $p_1 = v, p_n = u$  and for each  $i = 2, \dots, n$  it holds that  $p_i$  is directly density-reachable from  $p_{i-1}$  within  $V$  with respect to  $\epsilon$  and  $\eta$ .

The density-reachability relation is transitive but only for core vertices symmetric. Figure 4.1(middle) shows an example where vertex  $u$  is density-reachable from  $v$  via three other core vertices. By this definition, a vertex  $u$  cannot be density-reachable from a vertex  $v$  unless  $v$  is a core vertex. This restriction is removed by introducing the notion of *density-connectivity* between vertices, none of which needs to be a core vertex.

**Definition 4.5** *Let  $u, v \in V$  be two vertices.  $u$  is “density-connected” to  $v$  within  $V$  with respect to  $\epsilon$  and  $\eta$  if and only if there is a vertex  $m \in V$  such that  $u$  is density reachable from  $m$  and  $v$  is density reachable from  $m$ .*

Density-connectivity is a symmetric relation (cf. Figure 4.1 (right)). Based on the connectivity of vertices, we can now define a cluster in graph as a “community” composed of all vertices that are density-connected within  $V$  with respect to  $\epsilon$  and  $\eta$ .

**Definition 4.6** *Let  $G(V, E)$  be an undirected, weighted graph. A non-empty set  $C \subseteq V$  is a “community” with respect to  $\epsilon$  and  $\eta$  if and only if:*

- *For all  $u, v \in V$  it holds that if  $u \in C$  and  $v$  is density reachable from  $u$ , then  $v \in C$  (Maximality condition).*
- *For all  $u, v \in C$  it holds that  $u$  is density-connected to  $v$  within  $V$  with respect to  $\epsilon$  and  $\eta$  (Connectivity condition).*

**Definition 4.7** *Let  $C_1, \dots, C_k$  be the communities with respect to  $\epsilon$  and  $\eta$ . We define “noise” as the set of vertices in the graph not belonging to any community, i.e. noise =  $\{u \in V \mid u \notin C_i \text{ for } i = 1, \dots, k\}$ .*

### 4.1.2 Graph Clustering for Overlapping Communities

As discussed in the introduction, in social networking sites it is often observable that members belong to more than one community. So far, if a member  $u$  is close to more than one community, it is assigned to the cluster which is discovered first. In this case, the clustering result is not deterministic but depends on the order in which the vertices are visited. By changing the order of the seed vertices, the assignment of members who are actually close to two or more communities might change as well. To overcome this problem we extend the algorithm to handle overlapping clusters. By this, we also achieve a more realistic clustering as individuals can be members in different communities now.

To allow for cluster overlaps we introduce for each actor a list of cluster-ids. All cluster-ids of the  $\epsilon$ -neighborhoods a vertex belongs to are added to this list.

Vertices with more than one cluster-id are in either case border vertices. Noise vertices do not belong to clusters and core vertices can not belong to multiple clusters. The number of clusters a vertex is assigned to is limited by the parameter  $\eta$ : A vertex  $u$  can belong to up to  $\eta - 1$  clusters. Values greater than  $\eta$  are not possible because the vertex would have an  $\epsilon$ -neighborhood with more than  $\eta$  neighbors which makes  $u$  a core vertex yielding a cluster on its own.

The main procedure of DENGGRAPH to detect overlapping clusters works as follows (cf. Algorithm 4.1):

- Select an object  $u$  from  $V$  and determine all vertices that are density-reachable from  $u$  with respect to  $\epsilon$ .
- If  $u$  is a core vertex, i.e. if the number of vertices in  $u$ 's  $\epsilon$ -neighborhood reaches the threshold  $\eta$ , a new cluster is yielded and  $u$  is labeled as a core vertex.
- All vertices in the  $\epsilon$ -neighborhood are marked with the current cluster- $id$
- All vertices from  $N_\epsilon$  are put on a stack. This stack contains vertices that are density-connected to each other. They are then assigned to the same cluster, marked by the cluster  $id$  and labeled as border vertices.
- If the number of vertices in the neighborhood of the current vertex exceeds  $\eta$ , all vertices that are not yet classified are marked as noise are pushed on the stack and marked with the current  $id$ .
- The expansion continues until the stack is empty, i.e. until no further density-connected vertices can be found.
- If  $u$  is not a core vertex, i.e.  $|N_\epsilon(u)| < \eta$  and  $u$  has not been assigned to a cluster yet,  $u$  is marked as noise.
- Then, DENGGRAPH continues with the next non-labeled vertex until all vertices are labeled as either core, border or noise.

A more detailed description in pseudocode of the DENGGRAPH core for community discovery in the static graphs is given in Algorithm A.3.

### 4.1.3 Complexity and Computation Time

The time complexity for determining the number of neighbors (the degree) of vertex  $u$  is  $\mathcal{O}(1)$  because the value of the degree is stored in each vertex. If the edge information is stored in an adjacency list for each vertex, the run time complexity of DENGGRAPH is  $\mathcal{O}(|V| + |E|)$ , where  $|V|$  is the number of vertices and  $|E|$  the number of edges. All vertices are visited once and each edge is visited up to two times, once from both end-vertices.

In the following section, the extensions of DENGGRAPH to an incremental version to handle large dynamic data sets is discussed.

## 4.2 DENGGRAPH-IO: Incremental Graph Clustering

To allow for tracking and analyzing the temporal dynamics of clusters, DENGGRAPH is extended to an incremental procedure: The clustering is updated incrementally based on the edge-changes that are observed in the graph structure from one interval to another. These changes may evoke one of the following clustering updates: (1) *creation* of a new cluster, (2) *removal of a cluster*, (3) *absorption* of a new cluster member, (4) *reduction* of a cluster member, (5) *merge* of two or more clusters and (6) *split* of a cluster into two or more clusters (cf. Figure 4.2).

**Algorithm 4.1:** DENGGRAPH: Core

---

```
1 Algorithm:DENGGRAPH(Graph)
  input : Graph,  $\epsilon$ ,  $\eta$ 
  output: Set  $V$  with each vertex labeled with its status
2 begin
3   repeat
4     Select a  $u \in V$  that is not yet labeled
5     Compute  $N_\epsilon(u)$ 
6     if  $u$  is core vertex then
7       A new cluster  $id$  is generated
8        $u$  is assigned to the cluster and labeled as “core vertex”
9       All  $v \in N_\epsilon(u)$  are labeled as “border vertex” and the new  $id$  is added to their
       list of cluster-ids
10      All  $v$  are pushed on a stack
11      repeat
12        Pop the top vertex  $v$  of the stack
13        Compute the  $\epsilon$ -neighborhood of  $v$ 
14        if  $v$  is core vertex then
15          Label  $v$  as “core vertex”
16          foreach  $n \in N_\epsilon(v)$  do
17            Add new cluster-id to list of  $n$ 
18            Label  $n$  as “border vertex”
19            Push  $n$  on the stack
20          end
21        end
22      until (the stack is empty)
23    end
24    if  $u$  is not labeled then
25      Label  $u$  as “noise vertex”
26    end
27  until (all vertices in  $V$  are labeled)
28 end
```

---

### 4.2.1 Cluster Changes

So far we have assumed that the observed graph is static. In the dynamic scenario, the graph changes over time: new edges are added, old edges are deleted, distances between actors have changed, e.g., due to more frequent interactions or an increased similarity. The DENGGRAPH core of Algorithm 4.1 is extended to adapt the clusters incrementally. In the following, we present the encountered types of change and describe the actions taken for each [56].

**Types of Changes in the Graph** In comparison to an incremental clustering algorithm upon conventional data, DENGGRAPH-IO has to cope with a larger set of changes, namely changes in the *set of actors*  $V$  and changes in the *relationships among actors*  $E$ :

1. Two actors start interacting with each other, i.e. a relationship emerges. This includes the case that at least one of the actors is new in the network.
2. The intensity of the relationship between two actors increases when the actors interact more often with each other.
3. Conversely, the intensity of the relationship decreases when the actors interact less often or are less similar.
4. Finally, the relationship may disappear. These changes influence the distance between vertices and thus the contents of  $\epsilon$ -neighborhoods.

The changes to be dealt with by DENGGRAPH-IO can be summarized as changes in the *relationships among actors*, namely changes in the intensity of interaction, including the emerging of new interactions (and new actors) and the decay of old ones. Intensification of an interaction may make two vertices more proximal to the effect of one entering the neighborhood of the other and making it a core vertex. Then, this neighborhood can result in the expansion of the cluster it belongs to or even become a bridge between two formerly distinct clusters.

For example, consider two interacting actors  $u, v$ , whose distance at time  $t$  was  $dist_t(u, v) > \epsilon$ . If at time  $t + 1$  the intensity of their interactions has increased, then  $dist_{t+1}(u, v) < dist_t(u, v)$ . If  $dist_{t+1}(u, v) \leq \epsilon$ , then one (or both) of  $u, v$  may become a core vertex. The corresponding neighborhood may (i) extend an existing dense subgroup, may (ii) become a cluster itself or may (iii) form a bridge between two formerly distinct dense subgroups. Conversely, if the two actors interact less often than before, then  $dist_{t+1}(u, v) > dist_t(u, v)$ . If  $dist_{t+1}(u, v) > \epsilon$ , then the  $\epsilon$ -neighborhoods of the two vertices shrink and formerly density-connected vertices may become disconnected. This may lead to a split or a removal of communities.

In [49], Ester et al. propose an incremental variation of their DBSCAN algorithm which handles insertions and deletions of objects. In graph structures, adding and deleting vertices are comparatively rare updates, whereas changes in the distance between vertices occur very often. All changes such as new vertices, new edges and lost edges are observed on the edge-level, i.e. all relevant changes are edge-changes: A new vertex is only relevant if also an edge between another actor is established, a lost edge results in an edge-change such that a weight decreases to zero. Therefore, we capture all changes in the graph structure and update the clusters accordingly.

DENGGRAPH distinguishes between *positive changes* that may lead to community expansion, fusion or to new communities, and *negative changes* that may result to community split or decay. Since neighborhoods, distances and sets of vertices/edges are time-dependent, we mark each variable and function with the timepoint  $t$  to which it refers, e.g.  $dist_t(u, v)$  is the distance between  $u, v$  at time  $t$ .

In general, we distinguish six cases of cluster changes that might be observed after an edge-change which can be divided into two categories:

- Updates due to “positive changes”, i.e. the distance between two actors  $u$  and  $v$  in period  $t + 1$  reduces compared to period  $t$  such that they are in  $t + 1$  in each other’s  $\epsilon$ -neighborhood. We say that a new close edge has been established.
- Updates due to “negative changes”, i.e. the distance between two actors  $u$  and  $v$  in period  $t + 1$  increases compared to period  $t$  such that they are no longer in each other’s  $\epsilon$ -neighborhood. We say that a close edge is dissolved.

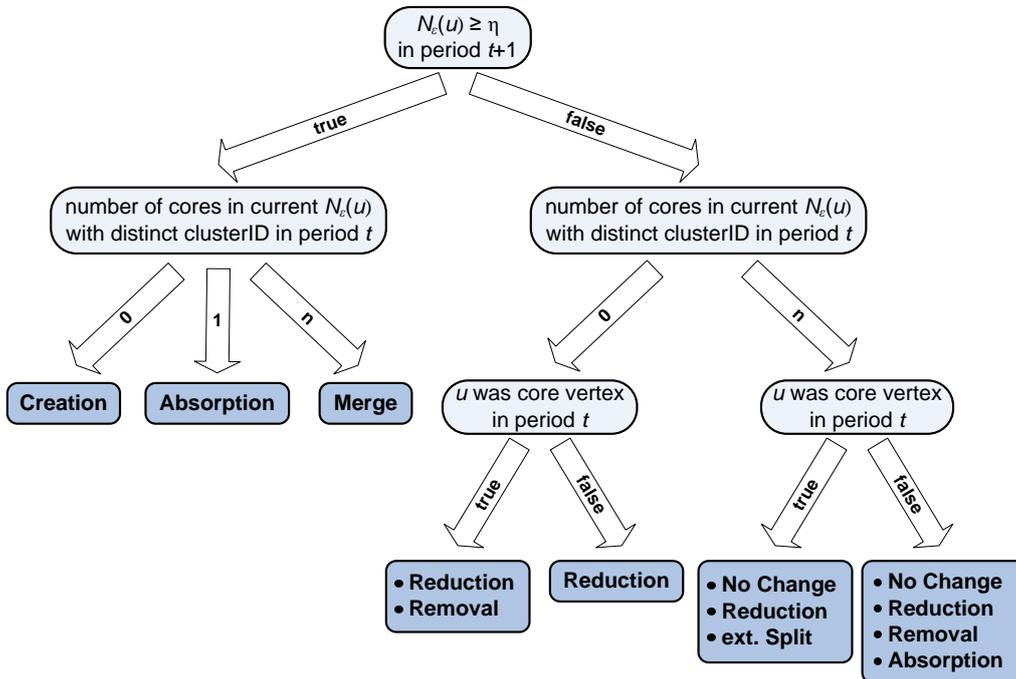


Figure 4.2: The *update* Method to detect cluster transitions. (A potential *split* is handled beforehand.)

### Adaptations to Positive Changes

Let  $t, t + 1$  be two adjacent timepoints and let  $u, v$  be two actors. Further, assume that  $dist_t(u, v)$  was undefined or larger than  $\epsilon$  and that  $dist_{t+1}(u, v) \leq \epsilon$ . Thus, a new close edge between  $u$  and  $v$  has been established which might lead to the following cases (cf. also Figure 4.3 (a)):

- *New core vertex*: At least one of  $u, v$  becomes a core vertex. DENGGRAPH-IO processes the new core vertex  $u$  and its neighbors including  $v$  and forms a neighborhood. It checks whether  $u, v$  were already associated with cluster identifiers. If not, they form a new cluster, otherwise their neighborhood becomes part of an existing cluster

or causes the fusion of clusters. If  $v$  is also a core vertex, DENGGRAPH-IO processes it with the neighbors of  $u$ .

- *New border vertex:* A former noise vertex  $u$  becomes (directly) density-reachable from a core vertex  $v$ . DENGGRAPH labels  $u$  with the  $id$  of the cluster containing  $v$ .
- *New noise vertex:* A new vertex  $u$  is added but none of the above cases holds. Then no clustering update is invoked.

The newly established edge might result in new density-connections. This can lead to the following adaptations of clusters:

**Creation** A new cluster emerges to accommodate vertices that have become core vertices and have not yet been assigned to a cluster. These vertices appear in the stack of DENGGRAPH-IO without a cluster  $id$  assigned to them; DENGGRAPH-IO generates a new cluster  $id$  and assigns it to them. DENGGRAPH-IO processes the new core vertex  $u$  and its neighbors including  $v$  and forms a cluster with all neighbors of  $u$  becoming border vertices.

**Absorption** If  $u$  has an  $\epsilon$ -neighborhood in  $t + 1$  and was associated with a cluster-id in  $t$  then  $u$  becomes absorbed by an existing cluster. The same holds if  $u$  has no cluster-id yet, but its neighborhood in  $t + 1$  consists of one core with exactly one cluster-id. Then  $u$  and its neighborhood become part of an existing cluster.

An absorption can also happen if a new close edge between a former noise vertex  $u$  and a core vertex  $v$  is established. Then, vertex  $u$  becomes a border vertex in the cluster built by  $v$  (see right branch in Figure 4.2).

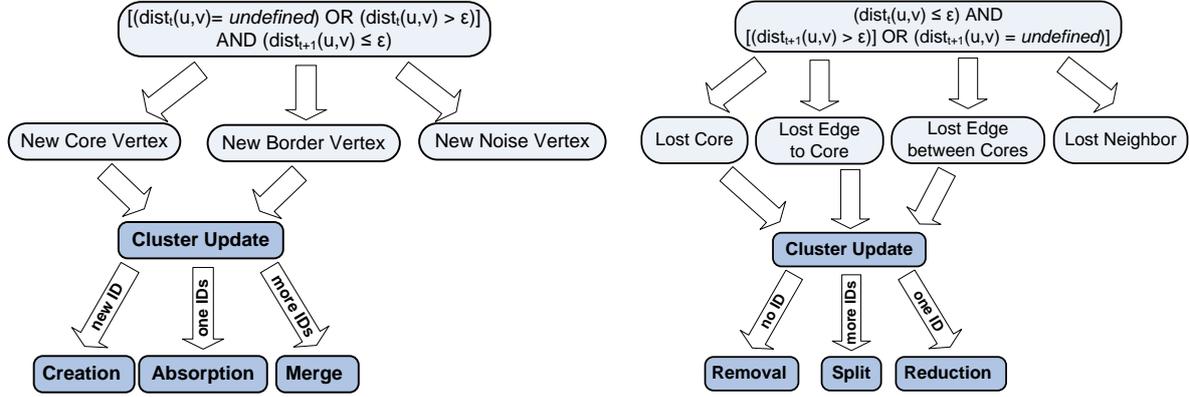
**Merge** If the neighborhood of a new core vertex  $u$  contains core objects which are members of different clusters. These vertices appear in the stack of DENGGRAPH-IO associated with different cluster identifiers. DENGGRAPH-IO generates a new cluster  $id$ , initiates a merge of these clusters and replaces the old identifiers with it.

The insertion of a new core can lead to a merge of up to  $\eta$  clusters. A merge a more than  $\eta$  clusters is not possible as  $u$  would have been already a core object before the insertion of the new close edge. This would lead to a merge before as all vertices would have been already density-reachable.

In general, an absorption is observed if a cluster gains new members. A new member or a border member of another cluster becomes a participant of the observed cluster. We define a community as growing if in the period under observation its number of members increases. This evolution is considered more evident if at the same time the number of edges in relation to the number of vertices in the community increases. To identify a significant development this observation should be made over a certain number of periods. The actual number of observed time steps depends on the length of the intervals. If the number of edges in relation to the number of vertices inside a community increases in several consecutive periods we observe that the community evolves. Parallel the average

shortest path between nodes might decrease. If the number of vertices remains constant at a high level but the number of edges increases, we observe a community that matures.

If  $u$  and  $v$  do not belong to a cluster and do not become a core or border vertex, they are marked as *noise*. If both vertices are already core vertices in  $t$ , no changes in the cluster structure are invoked.



(a) The effects of increased proximity among actors upon a cluster (b) The effects of decreased proximity or actor/edge removal upon a cluster

Figure 4.3: DENGGRAPH-IO: New close Relation (a) and lost close relation (b).

### Adaptations to Negative Changes

Let  $t, t + 1$  be two adjacent time points and let  $u, v$  be two actors. Further, assume that  $\text{dist}_t(u, v) \leq \epsilon$  and that  $\text{dist}_{t+1}(u, v) > \epsilon$  or  $\text{dist}_{t+1}(u, v)$  is undefined, i.e. the most recent interactions between  $u, v$  are so old that they are forgotten or  $u$  and  $v$  are no longer similar to each other. Then,  $v \in N_\epsilon^t(u)$  and  $v \notin N_\epsilon^{t+1}(u)$ . For the loss of a close edge the following cases can be distinguished (cf. Figure 4.3 (b)).

- *Lost core vertex:*  $u$  was a core vertex but lost its  $\epsilon$ -neighborhood due to the removal of a close edge to vertex  $v$ , i.e.  $|N_\epsilon^{t+1}(u)| < \eta$ . The *update* procedure checks whether the cluster is removed, reduced or split.
- *Lost edge to core vertex:* The border vertex  $u$  loses its directly-reachability to  $v$ . It must be checked, whether  $v$  is still a core vertex and the cluster still exists. Otherwise the cluster is split, reduced or removed
- *Lost edge between cores:* Both  $u, v$  were and are still core vertices, but they are no more directly density-reachable from each other. Hence, the cluster containing the two vertices may split.
- *Lost neighbor:* None of  $u, v$  was a core vertex. If a close edge between two border or noise vertices  $u$  and  $v$  is deleted, the clusters are not affected because noise vertices

do not belong to a cluster and changes between border vertices do not affect the border status. Furthermore, the core vertices that are connected with  $u$  and  $v$  are not affected by the lost edge between  $u$  and  $v$ .

For each of the above cases except the last, DENGGRAPH-IO recomputes the neighborhoods of the affected core vertices. It then recomputes the contents of the clusters to which these neighborhoods belonged and performs one of the following types of cluster adaptation:

**Removal** If after the deletion of a core vertex  $u$  there are no more core vertices in the neighborhood  $N_\epsilon(u)$ , the cluster is deleted. All former cluster members with only one cluster-id become noise and all border members of more than one cluster are removed from this cluster.

**Reduction** A reduction is observed if the cluster still survives, but the vertex  $u$  is removed from the cluster. Some vertices in  $N_\epsilon(u)$  may become noise if  $u$  was core vertex.

**Split** A cluster potentially splits if a close edge between two core nodes has been removed. If  $u$  and  $v$  belonged to exactly one cluster before their close edge was removed, it has to be checked whether these vertices are still density-connected via other core vertices in the cluster.

The function  $DensityConnectedNodes(u,v)$  is used to determine whether two former directly connected vertices are still density-connected after the edge between them is deleted or their distance is increased such that they are no longer in each other's  $\epsilon$ -neighborhood. The function gets as input the vertices  $u$  and  $v$ , starts from  $u$  and puts all directly-connected vertices from  $u$  to the queue and processes the next vertex in the queue. The process stops, if the vertex  $v$  is found and the function returns the set of density-connected vertices from  $u$  to  $v$ . By this, we can assess whether the edge deletion results in a split: if the two vertices are still density-connected, no split occurs, if they are no longer density-connected, a split is observed (see Algorithm A.1). The process to detect a potential split is displayed in Figure 4.4.

The pseudocode of the incremental version of DENGGRAPH is shown in Algorithm 4.2 and the update function in Algorithm 4.3. More detailed pseudocode of the incremental DENGGRAPH algorithm and the respective update functions is given in Appendix A.

### 4.2.2 Correctness of the Incremental Updates

To assess the correctness of the incremental updates we show that a clustering update with DENGGRAPH-IO yields the same results as a clustering with DENGGRAPH. For a given clustering we show that after deletions and insertions of edges the clustering obtained with the update function of DENGGRAPH-IO is the same as the non-incremental DENGGRAPH. For this, we show that for a given update (i) a set can be defined that consists of all vertices that can be affected by the update and (ii) that the update function of DENGGRAPH-IO

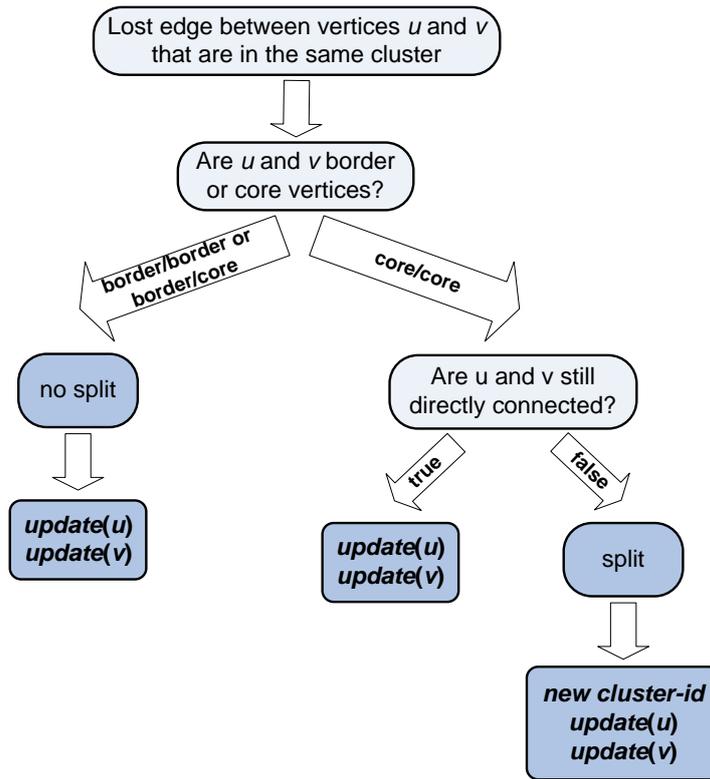


Figure 4.4: A cluster split occurs if core vertices  $u$  and  $v$  are no longer density connected after the removal of a close edge.

handles all vertices that belong to this set. Furthermore, we conduct empirical tests and compare the obtained clusterings.

A deletion or insertion of a close edge  $(u, v)$  between vertices  $u$  and  $v$  can result in a changed core status of one or both vertices. In the following, we discuss only the affected area based on one of the end-vertices of the lost or inserted close edge. All statements also hold for the other end-vertex and the incremental update procedure handles both vertices separately.

If a new close edge is established, former border vertices can become core vertices. The same holds, if a close edge is lost:  $u$  might lose its core or border property. Furthermore, vertices in the neighborhood of the neighbors of  $u$  are affected: The vertices in  $N_{2\epsilon}(u) \setminus N_{\epsilon}(u)$  keep their core status but a border vertex  $o$  in the neighborhood of  $u$  might become a noise vertex and vice versa. The first case happens, if  $u$  loses its core status and in the neighborhood of  $o$ , no other core vertex is present. The second case is observable, if  $u$  becomes core vertex and  $o$  becomes border. However, vertices in the neighborhood of  $o$  are not affected, as  $o$  can not become a core vertex. Thus, for all vertices outside  $N_{2\epsilon}(u)$  it holds that their core status is not affected by the changed core status of  $u$ . The same holds for their neighbors. Therefore, the cluster membership of vertices outside  $N_{2\epsilon}(u)$  is not affected.

**Algorithm 4.2:** DENGRAPH-IO: Incremental Clustering (overview)

---

```

1 Algorithm:DENGRAPH-IO(ClusterModel, LastGraph, NewGraph)
   input : Old Clustering, Old Graph, New Graph
   output: New Clustering
2 begin
3   foreach Changed Edge do
4     if Positive Update (new close edge) then
5       //Check for creation, absorption or merge
6       Update Clustering
7     end
8     else if Negative Update (lost close edge) then
9       if Edge was between border and core vertex then
10        Update Clustering
11        else if Edge was between core vertices then
12          //Check for split
13          if Both vertices are still density connected then
14            //No split
15            Update Clustering
16          end
17          else
18            //Split
19            Create new cluster
20            Update Clustering
21          end
22        end
23      end
24    end
25  end
26  return New Clustering
27 end

```

---

Because of the insertion of a new close relation, a border or noise vertex  $u$  might become core. This could lead to new density-connections between vertices  $p_1$  and  $p_n$  that were not density-reachable before the insertion, while one core vertex  $p_i$  for  $i < n$  on the chain  $p_1, \dots, p_n$  must be contained in  $N_\epsilon(u)$ . A deletion of a close relation may imply that a core vertex loses its core status resulting in a loss of density-connections. Thus, a chain  $p_1, \dots, p_n$  of density-reachable vertices may be separated and one of the  $p_i$  for  $i < n$  must be contained in the  $\epsilon$ -neighborhood of  $u$ .

**Definition 4.8** *The set  $affected(u, v)$  of vertices affected by the change of  $u, v$  is defined as*

$$\begin{aligned}
affected(u, v) = & (N_\epsilon(u) \cup p \mid \exists o \in N_\epsilon(u) \wedge p \text{ is density-reachable from } o) \\
& \cup (N_\epsilon(v) \cup p \mid \exists o \in N_\epsilon(v) \wedge p \text{ is density-reachable from } o)
\end{aligned} \tag{4.2}$$

---

**Algorithm 4.3:** DENGGRAPH-IO: *UpdateClustering*( $u$ ) Function (overview)

---

```
1 Algorithm:Update Clustering
  input : Vertex
2 begin
3   if Vertex is core then
4     if Neighbors have no cluster id then
5       | Creation of a new cluster
6     end
7     if Vertex was not core in last interval and in the neighborhood is one cluster-id
      then
8       | Absorption of the vertex to a cluster
9     end
10    if Neighborhood has cores with different cluster-ids then
11      | Merge of clusters
12    end
13  end
14  else
15    | Vertex has no epsilon neighborhood
16    if Vertex was core in the last interval then
17      | Check for extended split
18    end
19    if Vertex has no core neighbors then
20      | Vertex becomes noise object
21    end
22    else
23      | Vertex has core neighbors and becomes border object
24    end
25  end
26 end
```

---

In summary, the insertion of a new close edge  $(u, v)$  or the deletion of a former close edge  $(u, v)$  can only affect the vertices in the  $\epsilon$ -neighborhood of the respective end-vertices  $u$  and  $v$  plus all vertices that are density-reachable from one of these vertices. For all other vertices that are not in the set, the cluster membership will not change. Therefore, after deleting or inserting a close edge, all vertices in the set of affected vertices must be updated regarding their cluster membership by applying the update procedure.

Additionally, we empirically test whether the incremental updates are correct by comparing the clustering results obtained with DENGGRAPH-IO with those of DENGGRAPH. For this, we cluster the data sets, which are presented in the following Section 4.4 with the static community clustering procedure and with the incremental clustering and compare the results using the Jaccard distance (cf. Equation 2.10). For all data sets the Jaccard distance between clusters is 0, indicating that the clusters obtained with DENGGRAPH are identical with the clusters obtained with the incremental DENGGRAPH-IO. Both cluster procedures produce the same clusterings.

### 4.2.3 Complexity and Computation Time

In the following, the complexity and computation time of the DENGGRAPH-IO procedures are discussed.

**DENGGRAPH-IO** The complexity of the *DENGGRAPH-IO* (cf. Algorithm A.4) depends on the number of edge-updates that need to be performed and the complexity of the cluster re-calculation. When updating in  $t + 1$  a graph clustering built in  $t$ , in worst case, all edges in  $t$  are deleted leading to  $|E_t|$  updates due to lost close edges and  $|E_{t+1}|$  updates caused by new close edges, where  $|E_t|$  is the number of edges in the graph  $G_t$  and  $|E_{t+1}|$  the number of edges in  $G_{t+1}$ . In a fully meshed graph,  $|E|$  can be approximated with  $|V|^2$ . However, the social networks we use in the experiments have very low densities (see Table 4.2 and Table 4.11), thus  $\max_{t=1,\dots,n} |E_t| \ll |V|^2$ .

The procedure *DENGGRAPH-IO* calls the *UpdateClustering(u)* procedure for both end-vertices of a changed edge. Therefore, the running time is linearly dependent on the number of *UpdateClustering(u)* procedure calls which is again linearly dependent on the number of changed edges.

A deletion of a close edge between two core vertices might result in a split of a cluster. To check whether a split occurs, the procedure *DensityConnectedNodes(u,v)* is called. Therefore, the number of calls of the *DensityConnectedNodes(u,v)* procedure depends on the number of negative edge changes between two core vertices that are members of the same cluster. The realization of a split (update of cluster-ids which are stored in the vertices) needs in the worst-case  $\mathcal{O}(|V|)$  if all vertices need to be updated.

**UpdateClustering(u)** The running time of the *UpdateClustering(u)* procedure (cf. Algorithm A.5) depends on the size of the  $\epsilon$ -neighborhood of  $u$ . Besides, in case of a cluster merge due to a positive edge change, the running time is dominated by the complexity of the merge operation. Here, the complexity depends on the number of cluster members that are merged, irrespective of the number of clusters merged; in worst case  $\mathcal{O}(|V|)$  in case all vertices are merged.

In case of a lost close relation due to which a vertex  $u$  loses its core status, the cluster of  $u$  might collapse or split. To check for an extended cluster split the procedure *DensityConnectedNodes(u,v)* is used. If a split is observed no additional computational costs result because the necessary information for separating the clusters is already collected during the check for a split.

The total running time of the *UpdateClustering(u)* procedure is basically determined by the running time of the merge-operation or the split-checking, i.e. in worst-case  $\mathcal{O}(|V| + |E|)$ .

**DensityConnectedNodes(u,v)** Given vertices  $u$  and  $v$ , the *DensityConnectedNodes(u,v)* procedure (cf. Algorithm A.1) starts from  $u$  and traverses the graph in search for  $v$ . It stops if  $v$  is reached and returns the set of all density-connected nodes which were encountered during the search.

While the static DENGGRAPH algorithm implements a stack to process the vertices in the  $\epsilon$ -neighborhood of a given vertex (see Algorithm A.3), the function *DensityConnectedNodes*( $u, v$ ) to determine whether  $v$  is density-connected from  $u$  implements a queue. Basically, the procedure is a breadth-first-search which uses core vertices for exploitation and border vertices for exploration. The complexity of the procedure is  $\mathcal{O}(|V| + |E|)$ .

**DensityConnectedNodes( $u, old-clid, new-clid$ )** The running time of the *DensityConnectedNodes*( $u, old-clid, new-clid$ ) procedure (cf. Algorithm A.2) is the same as for the procedure *DensityConnectedNodes*( $u, v$ ). Additionally, cluster-ids are updated in constant time:  $\mathcal{O}(1)$ .

## Summary

In worst-case, the DENGGRAPH-IO has the same complexity as DENGGRAPH, viz.  $\mathcal{O}(|V| + |E|)$ , i.e. each incremental update takes at most the same time as a reclustering with DENGGRAPH. However, for social graphs, the expected running time is much lower than the upper bound, because a single edge update does not result in changes that affect the entire graph, but is restricted to the number of vertices in the set “affected( $u, v$ )” (cf. Definition 4.8). Given that  $x$ -updates need to be performed, we compare the actual running time of DENGGRAPH-IO with the running time of DENGGRAPH. If we assume the worst-case, the incremental algorithm is  $x$ -times slower than a re-clustering with DENGGRAPH, i.e. we estimated that  $\text{CPU-time}(\text{DENGGRAPH-IO}) \approx x * \text{CPU-time}(\text{DENGGRAPH})$ . However, the experiments have shown that in fact DENGGRAPH-IO is faster by at least a factor of 400 compared to DENGGRAPH. For example, in an interval with 2,500 updates, DENGGRAPH-IO was only 6 times slower than DENGGRAPH, i.e. if we break this down to just one update, the incremental update is about 400-times faster than the re-clustering with DENGGRAPH (cf., Figure 4.19 (week 144)). Furthermore, by incrementally updating, we obtain all information about the temporal evolution of the graph. Please note that we chose an interval where the discrepancy in CPU run time was maximal: On average the speed-up factor is much higher. In summary, given one update, DENGGRAPH-IO has a much lower time complexity than DENGGRAPH.

## 4.3 Evaluating Graph Clusters

As discussed in Section 2.4, the evaluation of the obtained graph clusters concerns two aspects: (i) the stability of the clusters to perturbations in the input graph and (ii) the quality of the detected clusters considering the requirement that a cluster should consist of densely connected nodes that are only sparsely connected to nodes outside the cluster. In the following, we extend the measures presented in Section 2.4 to evaluate the DENGGRAPH clusters regarding these aspects.

### 4.3.1 Graph Cluster Stability

A measure for network resilience which is based on the idea of connectivity has been introduced in Section 2.4.1 (cf. Definition 2.6). Based on this idea, we introduce a measure to assess the stability of a graph cluster  $C_i$  to perturbations of the input graph  $G[C_i]$ . Let  $G[C_i]$  be an undirected subgraph induced by the cluster  $C_i$  with  $n$  edges.

**Definition 4.9** *The probability of a graph cluster to be split or removed is defined as*

$$P(G[C_i], j) = \Pr[G[C_i] \text{ is split or removed exactly after removal of the } j\text{-th edge}], \quad (4.3)$$

where  $G[C_i]$  is the induced subgraph of the cluster  $C_i$ .

**Definition 4.10** *Let  $G[C_i]$  be an undirected subgraph induced by the cluster  $C_i$  with  $n$  edges. The DENGGRAPH-stability  $\text{stability}(G[C_i], p)$  is the largest number  $j$  of edges removed such that  $G[C_i]$  is still a DENGGRAPH cluster with probability  $1 - p$ .*

$$\text{stability}(G[C_i], p) = \max \left\{ J \mid \sum_{j=1}^J P(G[C_i], j) \leq p \right\} \quad (4.4)$$

The relative graph cluster stability relates  $\text{stability}(G[C_i], p)$  to the size of  $G[C_i]$ :

$$\overline{\text{stability}}(G[C_i], p) = \frac{\text{stability}(G[C_i], p)}{n} \quad (4.5)$$

We assess the quality and stability of the detected DENGGRAPH-clusters by calculating the cluster resilience (cf. Definition 2.6) and the stability of the obtained clusters regarding edge removals (cf. Definition 4.10) for different graph clusterings. First, we apply DENGGRAPH to extract clusters from graphs. Then, we test for each cluster the disconnection probability by randomly removing edges and checking whether all vertices in the cluster are still connected, i.e. whether all vertices are reachable from each other.

Furthermore, to compute the cluster stability, we re-cluster the graph from which we removed the edge with DENGGRAPH to see whether the clustering is still the same or if the edge removal results in a split or removal of a cluster. The experiments are repeated and in the end we calculate the average number of removed edges until the cluster is split or removed.

In summary, in our experiments we calculate for the obtained clusters

- the *cluster resilience*
- the probability  $P(G[C_i], j)$  for a cluster to be split or removed after the deletion of the  $j$ -th vertex and
- the *DENGGRAPH-stability* for  $p=1$ .

The pseudocode for the calculation of the DENGGRAPH-stability for  $p=1$  is shown in Algorithm 4.4. The results of the experiments are discussed in the following sections on applications (cf. Sections 4.4.3 and 4.5.3).

---

**Algorithm 4.4:** DENGGRAPH: Stability

---

```

1 Algorithm:DENGGRAPH-stability(m)
   input :  $G[C_i], m$ 
   output:  $j$ =average number of removed edges until  $C_i$  is split or removed
2 begin
3    $j=0$ 
4   for  $j=1$  to  $|E(C_i)|$  do  $count[j] = 0$ 
5   for  $n=1$  to  $m$  do
6      $j=0$ 
7     repeat
8       remove random edge from  $G[C_i]$ 
9        $j = j + 1$ 
10    until ( $C_i$  is removed or split)
11     $count[j] = count[j] + 1$ 
12  end
13   $j=0$ 
14   $p=0$ 
15  repeat
16     $j=j+1$ 
17     $p = p + \frac{count[j]}{m}$ 
18  until ( $p = 1$ )
19 end
20 return  $j$ 

```

---

### 4.3.2 Graph Cluster Quality

In the following, we propose an adaptation of the modularity measure presented in Section 2.4 for the assessment of the cluster quality in weighted networks and for overlapping clusters.

#### Weighted Modularity

Based on the definition for the modularity (cf. Definition 2.22), we define the modularity for weighted graphs. Instead of the number of edges, the definition of modularity employs the sum of the edge weights for edges within clusters, between clusters and in the total graph:

**Definition 4.11** *Given a graph  $G = (V, E)$  with edge weights  $\omega(e)$ , the modularity is defined as*

$$Q(\zeta) = \sum_{C \in \zeta} \left[ \frac{\omega(E(C))}{\omega(E)} - \left( \frac{\omega(E(C)) + \sum_{C' \in \zeta} \omega(E(C, C'))}{2 \cdot \omega(E)} \right)^2 \right], \quad (4.6)$$

where  $\omega(E)$  is the sum of all edge weights,  $\omega(E(C))$  is the sum of the edge weights in cluster  $C$  and  $E(C, C')$  is the set of edges that have at least one end-node in cluster  $C$ .

As discussed in Section 2.4,  $Q$  is in the interval  $[-\frac{1}{2}, 1]$  for undirected and unweighted graphs. The same holds for weighted graphs, since the minimal value of the first term

is zero if the clusters contain no edges and the second term is maximized if all edges are between clusters; then the value of  $Q$  is  $-\frac{1}{2}$ . A value of 1 is reached only if the graph does not contain any edges because the first term (the coverage) is in this case defined to be 1. In case of a “good clustering”, the weighted modularity is greater than the unweighted, i.e. if intra-cluster edges have a higher weight than edges between clusters.

### Modularity for Overlapping Clusters

Since the modularity was not yet defined for overlapping clusters we have to decide how to apply the measure in case of cluster overlaps. As shown in Figure 4.5, basically four different options are possible to deal with vertices that belong to more than one cluster: all vertices that belong to more than one cluster are (i) removed from the graph as well as the respective edges, (ii) handled as vertices that do not belong to any of the clusters, (iii) handled as vertices in all clusters and (iv) handled as in (iii) but furthermore inter-cluster edges are added.

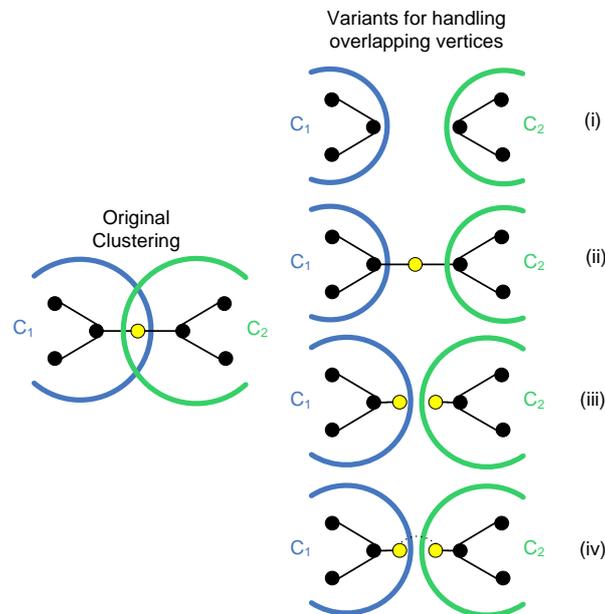


Figure 4.5: Options for Modularity Calculation with Overlapping Clusters.

Left side: Original clustering with two clusters  $C_1$  and  $C_2$ . Right side: four options for handling overlapping vertices.

The consequences for the modularity depends on whether the graph is weighted or not or rather whether the clusters have symmetric edge values or not: Cases (i) and (ii) are equivalent in terms of modularity as vertices that do not belong to a cluster are not considered in the calculation. Therefore, edges are not considered as well. This holds for weighted and unweighted graphs. If we take the modularity that we would obtain applying option (i) or (ii) as the baseline, the last option (iv) would reduce the modularity as inter-cluster edges are established that increase the second term of the formula. Furthermore, when applying this mode, one would have to decide which weight the edge should obtain.

In case (iii), the modularity is maximal if both edges between the vertex that overlaps and the two vertices in the clusters have the same weight. In this case, the modularity equals the value reached in case (i) or (ii). The modularity decreases if the edge weights are not equal: The higher the difference between the two edge weights, the lower the modularity. This is because, the coverage of the cluster with the higher edge weight is larger as the numerator and even though the second term in the formula that is subtracted from the first increases as well, it is too low to balance the different coverage values. Therefore, comparing cases (i) and (ii) with case (iii), the modularity values are only the same if the edge weights are equal.

For the applications we chose, this behavior is not desirable since differences in edge weights should not be punished. We therefore apply case (i) for calculating the modularity in weighted graphs with overlaps and omit all vertices which are members in more than one cluster.

## 4.4 Application: Graph over Interactions

To assess the applicability and to determine the characteristics of the proposed clustering algorithm we applied DENGGRAPH-IO on the *Enron* email data set provided by the University of Southern California.<sup>1</sup> The data set consists of e-mail messages sent between individuals over a period of several years. First, we introduce a distance function on the *Enron* graph and analyze the data set regarding its social network characteristic. Furthermore, the influence of the parameters  $\epsilon$  and  $\eta$  on the clusterings obtained by applying DENGGRAPH-IO are analyzed and the stability of the clusters regarding perturbations of the input graph is studied. Finally, observations which were made by studying the temporal aspects of the graph are discussed.

### 4.4.1 Enron Data Set

The collapse of *Enron*, a U.S. company honored in six consecutive years by “Fortune” as “America’s Most Innovative Company”, caused one of the biggest bankruptcy cases in US-history. To investigate the case, a data set of approximately 1.5 million e-mails sent or received by *Enron* employees was published by FERC, the Federal Energy Regulatory Commission.<sup>2</sup>

Shetty and Adibi published a subset of the original data containing approximately 250,000 e-mails from/to 151 *Enron* employees which were sent during 1998 and 2002 [151]. We use this data set and a subset consisting only of messages sent from *Enron* employees to *Enron* employees for the experiments with the DENGGRAPH-IO clustering algorithm.

### Understanding the Enron Data Set

From the data set we dropped the first three months because there was only a very low amount of interactions observable. Thus, we took all messages (roughly 245,000) from January 2000 to March 2002 for our analysis which were sent from approximately 17,000 distinct senders to about 66,000 distinct recipients. As expected, the interaction graph which represents the e-mail exchange between individuals shows a low density, a right-skewed degree distribution and a short average distance between vertices (small-world effect). These measures indicate that the graph has a clustered structure. Furthermore, since the data set encompasses e-mail interactions over a period of approximately three years, it is particularly suitable for the analysis of subgraph evolutions. The general data set information are given in Table 4.1.

Based on the total graph, the average shortest paths length and the diameter are shown over all periods in Figure 4.6. The average shortest path length varies over the observation period. In the beginning, the shortest paths length is low (about two) because the graph was very small. It increases in the first four months strongly and remains almost constant

<sup>1</sup><http://www.isi.edu/~adibi/Enron/Enron.htm>, as of March 13, 2009.

<sup>2</sup>The original data set is available at <http://www.ferc.gov/industries/electric/indus-act/wec/enron/info-release.asp>, as of March 13, 2009.

Table 4.1: *Enron*: Data Set Information

<b>Period</b>	from 01/01/2000 to 03/25/2002
<b>Number of Messages</b>	245,503
<b>Number of Senders</b>	17,308
<b>Number of Recipients</b>	66,879

around 3.4 before it decreases in the last periods because of a strong reduction in the size of the graph. The diameter varies accordingly and the average value is around eight. Only in the beginning and at the end, the values are significantly lower. The highest value (thirteen) is observed in July 2001.

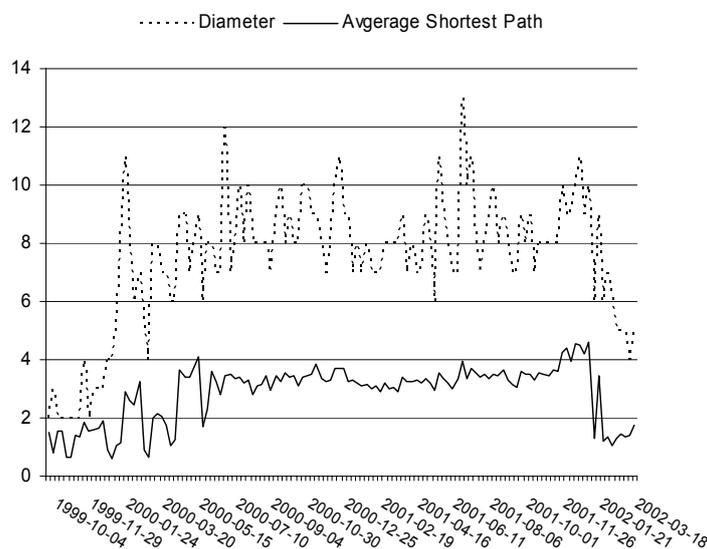


Figure 4.6: Average shortest path length and diameter of the *Enron* data set over the observation period ( $Enron_{total}$ ).

From the original data set which included messages from *Enron* employees and external partners, named  $Enron_{total}$ , a subset is taken which consists only of messages between *Enron* employees, named  $Enron_{intern}$ . For the temporal analysis we choose two different time window lengths: weekly and monthly. In the first case, the messages are aggregate over one week ( $Enron_{total}^{week}$  and  $Enron_{intern}^{week}$ , respectively) in the latter case interactions are cumulated over a period of one month ( $Enron_{total}^{month}$  and  $Enron_{intern}^{month}$ , respectively).

For each scenario, graph statistics are calculated over all intervals and the averages are displayed in Table 4.2. With regard to the number of vertices, the  $Enron_{intern}^{week}$  data set is the smallest with on average 60 actors. Compared to  $Enron_{intern}^{month}$ , the average cluster coefficient, which is in both cases very high and the average degree are similar. This indicates that the aggregation of interactions over a period of one month does not alter significantly the characteristic of the social graph. When comparing the statistics

of  $Enron_{total}^{week}$  and  $Enron_{total}^{month}$  a similar observation can be made: Both graphs have the same density and almost the same clustering coefficient. The average number of vertices triples from a weekly to a monthly perspective as well as the average number of edges. Furthermore, the  $Enron_{intern}^{month}$  and  $Enron_{total}^{week}$  have similar values for the average degree and the clustering coefficient.

Table 4.2: *Enron*: Graph statistics averaged over all intervals.

	$Enron_{intern}^{week}$	$Enron_{intern}^{month}$	$Enron_{total}^{week}$	$Enron_{total}^{month}$
<b>Min. Number of Vertices</b>	15	25	422	1103
<b>Avg. Number of Vertices</b>	59.4	88.3	1729.6	3132.3
<b>Max. Number of Vertices</b>	118	133	3706	5539
<b>Min. Number of Edges</b>	15	40	549	2254
<b>Avg. Number of Edges</b>	82.8	198.7	4064.2	11799.4
<b>Max. Number of Edges</b>	262	531	12156	27677
<b>Avg. Degree</b>	2.59	4.15	4.36	7.06
<b>Avg. Clustering Coefficient</b>	0.44	0.48	0.43	0.45
<b>Avg. Density</b>	0.056	0.054	0.003	0.003

#### 4.4.2 Defining Proximity in the Enron Graph

To group actors according to their closeness in graph structures, we need to define a function that determines the distance between them. In social networks we can assume that the closeness of actors is reflected in the number of interactions between them. Therefore, we define a distance function based on the frequency of interactions.

The set of vertices  $V$  contains the *actors* involved in the social network. The set of edges  $E$  captures their interactions. An edge between two actors  $u, v$  exists only if they interacted at least once during the period of observation. It is undirected but is associated with two values that capture directional semantics:  $(u \rightarrow v)$  is the number of interactions that actor  $u$  performed towards  $v$ , while  $(v \rightarrow u)$  is the number of interactions from  $v$  to  $u$ . If either  $(u \rightarrow v)$  or  $(v \rightarrow u)$  is zero, i.e. only one actor sent messages to the other, no edge between  $u$  and  $v$  is established. It holds that  $(u \rightarrow v + v \rightarrow u) > 1$ , otherwise the edge  $(u, v)$  would not exist.

The intention of DENGGRAPH is to cluster actors into communities. Traditionally, clustering is based on *proximity* of the objects to be clustered. On a graph of interactions, we model proximity between two actors based on the number of their interactions. For each graph ( $Enron_{intern}^{week}$ ,  $Enron_{intern}^{month}$ ,  $Enron_{total}^{week}$ ,  $Enron_{total}^{month}$ ) we chose a value  $z$  so that about 1.5 percent of all edges have an edge weight larger than  $z$ . The edges with weights higher than  $z$  are mapped to  $z$ . The value  $z$  ensures that the weight of edges with comparably very high values are bounded to  $z$ .

**Definition 4.12** *The proximity between two actors  $u$  and  $v$  is defined as*

$$proximity(u, v) = \frac{\min \{u \rightarrow v, v \rightarrow u, z\} - 1}{z - 1} \quad (4.7)$$

where  $u \rightarrow v$  is the number of messages sent from  $u$  to  $v$ ,  $v \rightarrow u$  the number of messages sent from  $v$  to  $u$  and  $z$  is a value specified for each graph.

The proximity function ranges in  $[0, 1]$  and is symmetric. If only one reciprocated interaction exists between  $u$  and  $v$ , then their proximity is zero. The proximity value increases asymptotically towards 1 as the number of interactions increases *reciprocally*.

DENGRAPH is based on the notion of distance rather than proximity. So, we define the following distance function:

**Definition 4.13** *Let  $G(V, E)$  be the graph of interactions and let  $u, v \in V$  be two interacting actors, i.e.  $\exists(u, v) \in E$ . Their “distance” is*

$$dist(u, v) := 1 - proximity(u, v). \quad (4.8)$$

Since proximity is only defined for pairs of actors that are connected with an edge, its 1-complement does not satisfy the triangle inequality and is thus not a distance metric. However, this partial definition of proximity is adequate for computing neighborhoods and density-reachable areas within the graph.

This function is, as discussed, defined only for interacting actors. It is still adequate for our graph clustering task, though, since DENGRAPH does not consider the proximity between arbitrary vertices but rather computes the “ $\epsilon$ -neighborhood” of each vertex:

**Definition 4.14** *Let  $G(V, E)$  be the graph of interactions and let  $u \in V$  be a vertex. The “ $\epsilon$ -neighborhood” of  $u$  is the set of vertices that are more proximal to  $u$  than  $\epsilon$ , i.e.*

$$N_\epsilon(u) = \{v \in V \mid \exists(u, v) \in E \wedge dist(u, v) \leq \epsilon\}. \quad (4.9)$$

### 4.4.3 Experiments and Results

In this subsection the influence of the parameters  $\epsilon$  and  $\eta$  on the clusterings of the *Enron* graph is presented. Furthermore, we discuss the cluster resilience and the stability of the obtained clusters to assess the sensitivity of the clusters to changes of the input graph.

#### Influence of Parameters $\epsilon$ and $\eta$

In the following, we present a study of the influence of the parameters  $\epsilon$  and  $\eta$  on the clustering. For this, we performed experiments with different parameter combinations on all four graphs and observe a number of measures to assess the impact of the parameters on the clustering. In the interval  $[0.05, 1]$  we chose 20 values equidistantly distributed for  $\epsilon$  and 19 values for  $\eta$  equidistantly distributed in the interval  $[2, 20]$ . Due to the size of the graph  $Enron_{total}^{month}$  we extended the range of  $\eta$  to  $[2, 40]$ . We tested each parameter combination and calculated the following measures to assess the impact on the clustering:

- Number of clusters
- Cluster performance
- Modularity (unweighted)
- Noise ratio

Figures 4.7, 4.8, 4.9 and 4.10 show the results of the experiments for each graph for one interval. In the upper left corner, the number of clusters that are yielded with the given parameter combination are shown. The cluster performance values calculated as given in Definition 2.9 are depicted in the upper right corner. The plot in the lower left corner shows the unweighted modularity and in the lower right, the ratio of noise vertices to the number of all vertices in the graph is given. Please note that the x- and y-axis are rotated in the plot showing the noise ratio to enhance visibility.

*Enron*<sub>intern</sub><sup>week</sup>: **Influence of parameters  $\epsilon$  and  $\eta$**  Due to the small number of actors and the low interaction strength between the actors only small clusters are observable which are detected by choosing a high value for  $\epsilon$  (between 0.9 and 1) and a low value for  $\eta$  (between 3 to 5). Over all intervals and parameter combinations, maximal five clusters are detected with about 50 vertices in total. This clustering is reached choosing  $\epsilon = 1$  and  $\eta = 2$ . The highest cluster performance observed is 0.6 and the maximal modularity is 0.8. The noise ratio obviously depends on the number of vertices which are members of clusters. It decreases strongly as soon as a cluster is built because the graph is small and if a cluster is yielded a comparably high number of vertices become cluster members.

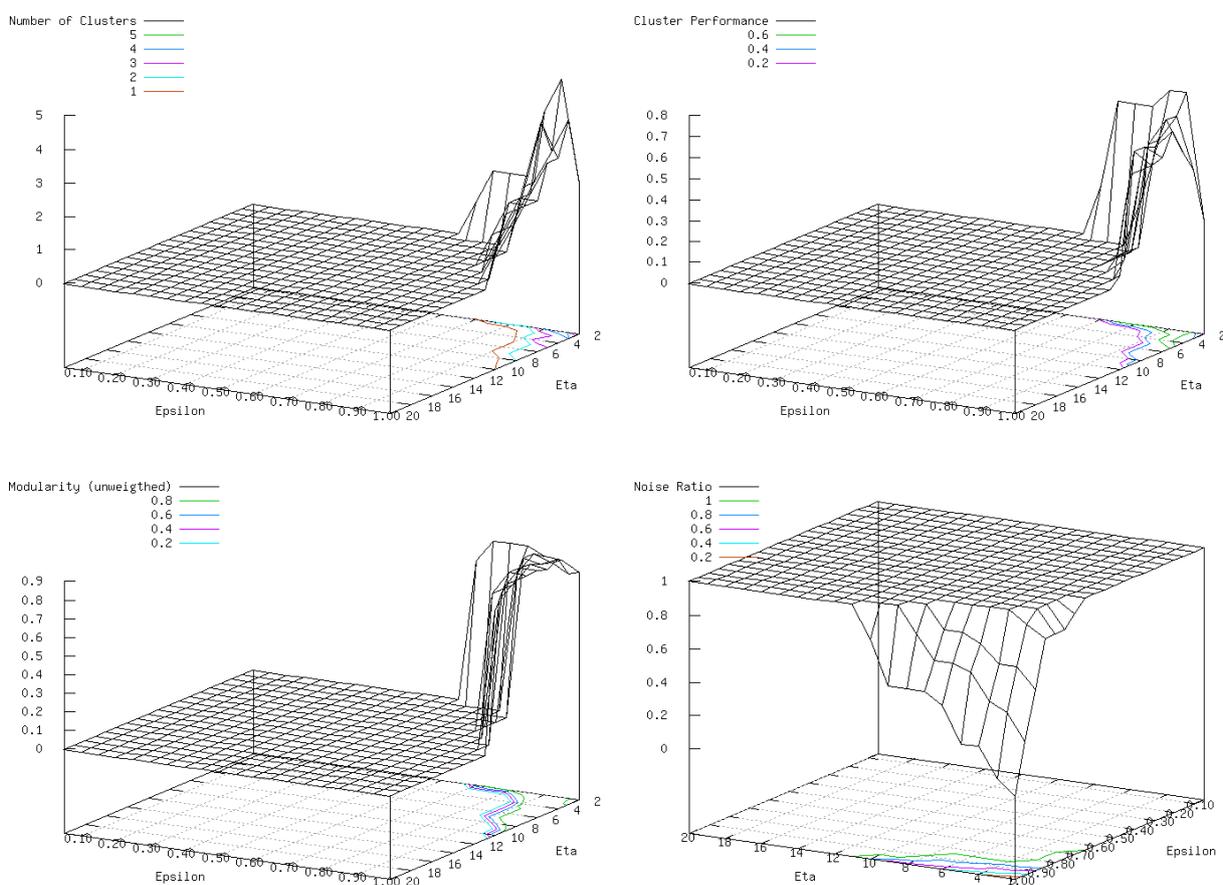


Figure 4.7: The influence of the parameters  $\epsilon$  and  $\eta$  on the clustering results of the data set *Enron*<sub>intern</sub><sup>week</sup> in week 49/2000.

For further information on the *Enron*<sub>intern</sub><sup>week</sup> graph such as the number of core, border and noise vertices, the clustering coefficient and average degree over all periods, see Figure B.5.

*Enron<sub>intern</sub><sup>month</sup>*: **Influence of parameters  $\epsilon$  and  $\eta$**  Similar to the graph *Enron<sub>intern</sub><sup>week</sup>*, the number of clusters ranges in  $[1, 5]$ . The maximal value of five is reached for  $\epsilon = 0.95$  and  $\eta = 7$ . In this area, also high cluster performance and modularity values are reached, while the modularity is equally sensitive to changes of the parameters. Even though the range for high performance and modularity values is quite big, the noise ratio differs considerable; values range between 20% and 60%. This indicates that the noise ratio is an important factor that should be considered when determining the parameters for the clustering.

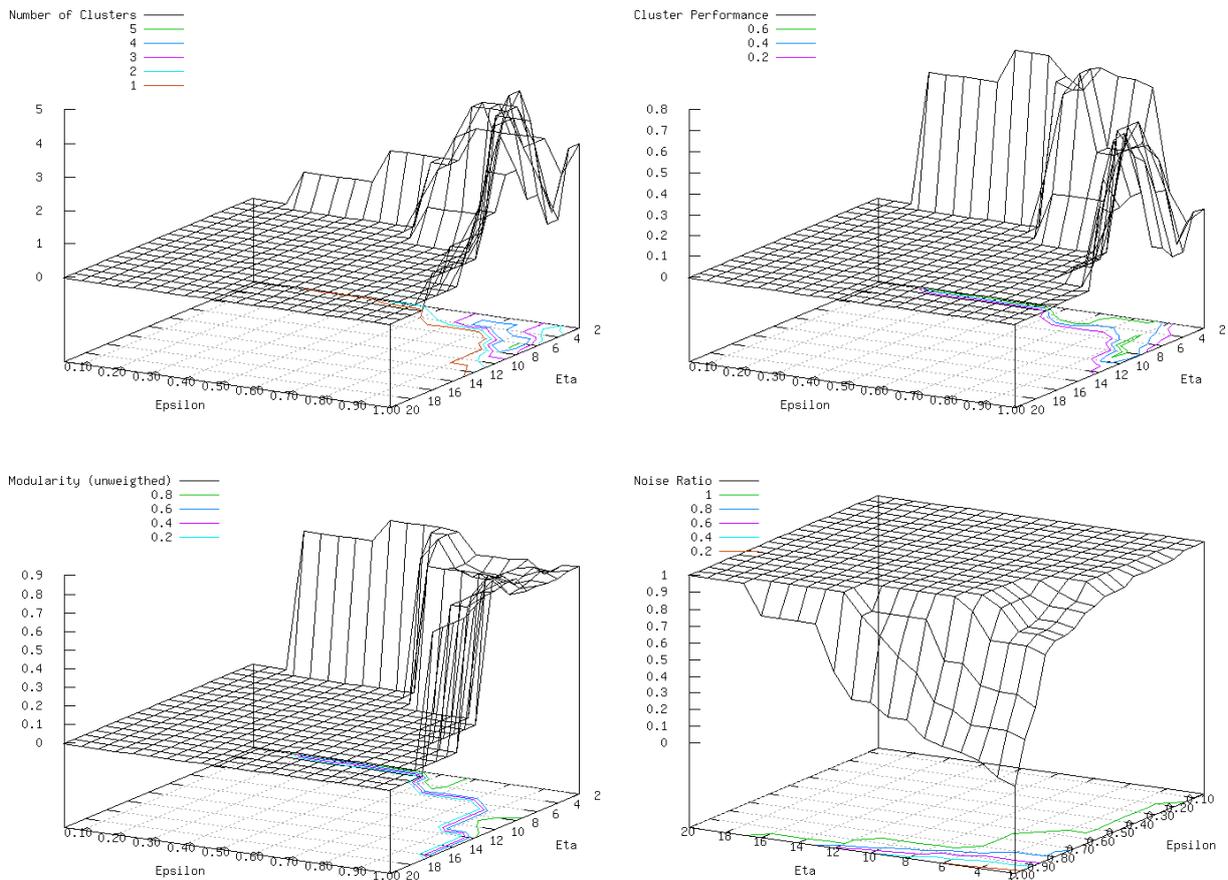


Figure 4.8: The influence of the parameters  $\epsilon$  and  $\eta$  on the clustering results of the data set *Enron<sub>intern</sub><sup>month</sup>* in month 1/2001.

For further information on the *Enron<sub>intern</sub><sup>month</sup>* graph such as the number of core, border and noise vertices, the clustering coefficient and average degree over all periods, see Figure B.6.

*Enron<sup>week</sup><sub>total</sub>*: **Influence of parameters  $\epsilon$  and  $\eta$**  As can be seen in all four plots in Figure 4.9, the parameters  $\epsilon$  and  $\eta$  have a strong influence on the outcome of the clustering. The experiments show that other than for the previously presented graphs, here the range of  $\epsilon$  and  $\eta$  which results in highest performance and modularity do not exactly match. Highest performance values are obtained by fixing  $\epsilon$  between 0.4 and 0.6 and  $\eta$  between 6 and 10. The highest modularity values are obtained with  $\epsilon$  between 0.3 and 0.6 and  $\eta$  from 12 to 16. In this case, modularity would reach very high values around 0.9, because only very few clusters are yielded which have only few inter-cluster edges resulting in high modularity values. Therefore, at the same time, the noise ratio is rather high. Please note that for a parameter combination which maximize the cluster performance, the modularity is also above 0.8. Thus, high performance and high modularity values can be reached at the same time. The noise ratio is above 80% as long as  $\epsilon$  is lower than 0.8. This observation is independent of the value for  $\eta$ . Thus, most vertices become cluster members only for large  $\epsilon$  values.

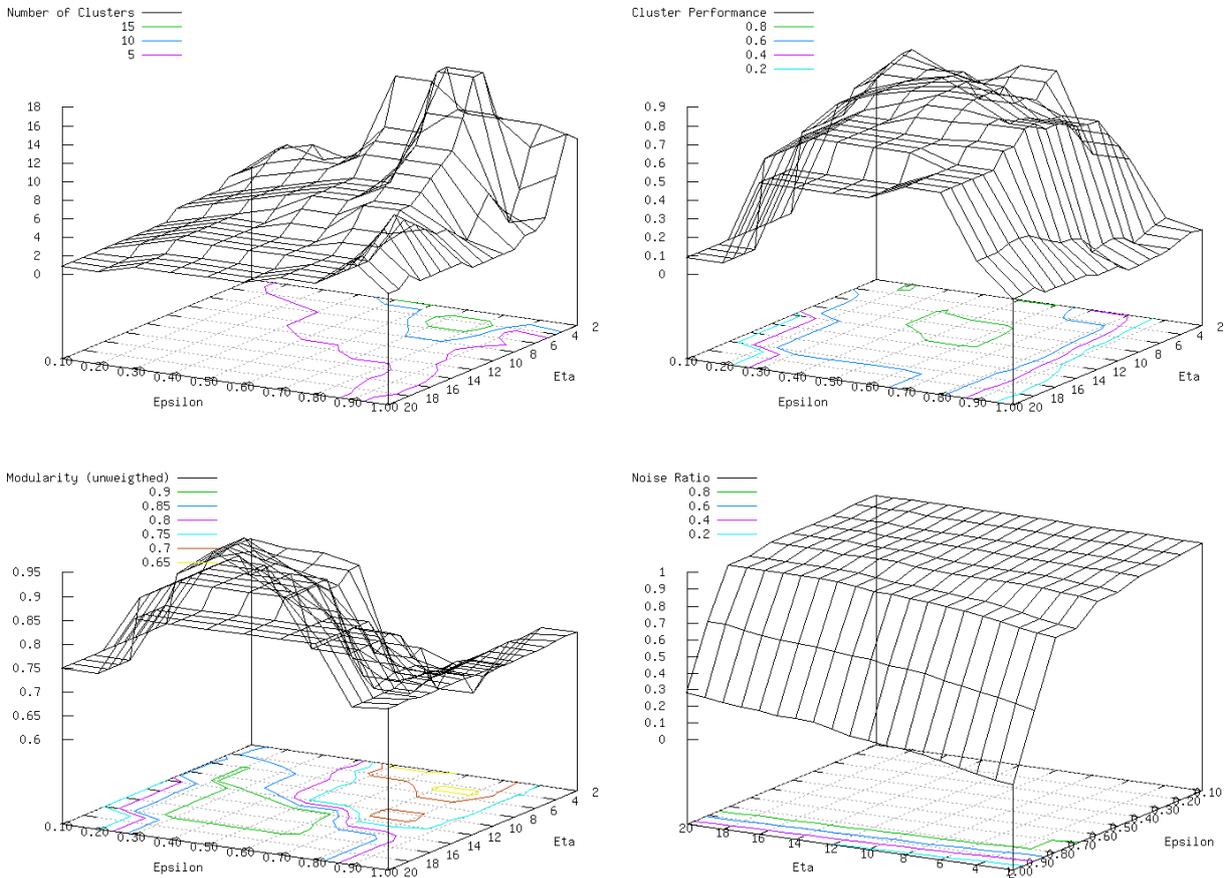


Figure 4.9: The influence of the parameters  $\epsilon$  and  $\eta$  on the clustering results of the data set *Enron<sup>week</sup><sub>total</sub>* in week 50/2001.

For further information on the *Enron<sup>week</sup><sub>total</sub>* graph such as the number of core, border and noise vertices, the clustering coefficient and average degree over all periods, see Figure B.7.

**$Enron_{total}^{month}$ : Influence of parameters  $\epsilon$  and  $\eta$**  The main observation for the  $Enron_{total}^{month}$  graph is that for high  $\epsilon$  values,  $\eta$  has almost no influence on the cluster performance, the modularity or the noise ratio. An influence is still observable on the number of clusters: for high  $\eta$  values, less clusters are detected. For small values of  $\epsilon$ ,  $\eta$  influences the measures considerably: the higher  $\eta$  the smaller the cluster performance, the modularity and the noise ratio. This can be explained by the large number of vertices in the graph which is the main difference of  $Enron_{total}^{month}$  to the other three graphs. If  $\epsilon$  is high, even edges with few interactions are close, resulting in many vertices with an own  $\epsilon$ -neighborhood with more than  $\eta$  neighbors. However, this results in very few clusters and a high noise ratio.

The noise ratio is 1 if no cluster is detected and ranges between 0.8 and 1 for  $\epsilon$  values between 0.6 and 0.85. The higher  $\epsilon$ , the lower the proportion of noise vertices to the total number of vertices. Highest performance values are yielded with  $\epsilon$  up to 0.7 and  $\eta$  between 5 and 10.

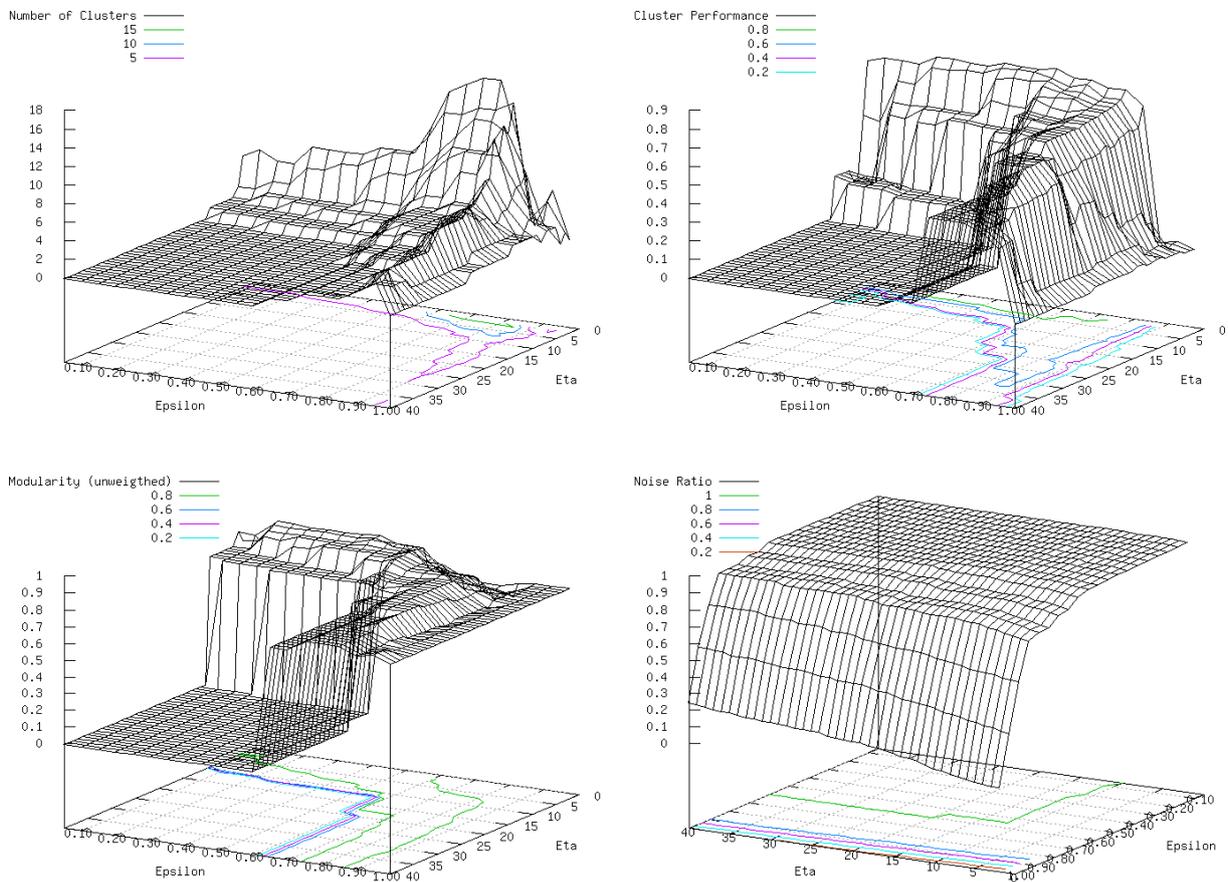


Figure 4.10: The influence of the parameters  $\epsilon$  and  $\eta$  on the clustering results of the data set  $Enron_{total}^{month}$  in month 4/2001.

For further information on the  $Enron_{total}^{month}$  graph such as the number of core, border and noise vertices, the clustering coefficient and average degree over all periods, see Figure B.8.

**Remarks** As expected, the parameters  $\epsilon$  and  $\eta$  have a high influence on the outcome of the DENGGRAPH-clustering. As discussed in Section 2.4.2, performance and modularity are quality measures for clusters and we showed that both measures are not contradictory and could be used for parameter determination. Thus, one could say that the best parameter combination is yielded when cluster performance measure and/or the modularity are optimal. However, the noise ratio and the number of clusters are also important factors that should not be neglected. Ester et al. propose for example to determine the best parameters by specifying a fixed percentage of noise [50]. Besides the noise ratio we decided to consider also the number of clusters because the modularity value can easily be maximized by reducing the number of clusters. We therefore first determine the parameter range with best modularity and performance values and chose than a combination which yield an average noise ratio and average number of clusters.

The chosen parameters for the experiments are shown in Table 4.3.

### Resilience and Stability of DENGGRAPH-Clusters in the Enron Graph

In the following, the cluster resilience (cf. Definition 2.6) and the stability of the obtained clusters regarding edge removals (cf. Definition 4.10) are presented. For the experiments, we chose one interval for each graph clustered the graphs with DENGGRAPH-IO. In Table 4.3, information about the graphs and the obtained clusterings that were used as input for the analysis are given:

Table 4.3: *Enron*: Clustering statistics for a chosen interval and parameter combination.

	$Enron_{intern}^{week}$	$Enron_{intern}^{month}$	$Enron_{total}^{week}$	$Enron_{total}^{month}$
$\epsilon$	1	0.95	0.6	0.6
$\eta$	3	7	7	10
<b>Interval-id</b>	101	42	155	45
<b>Number of Clusters</b>	4	5	6	8
Number of Vertices	Core	24	9	28
	Border	57	77	190
	Noise	28	46	100
<b>Number of Edges</b>	81	112	179	1278
<b>Density</b>	0.05	0.04	0.01	0.003
<b>Avg. Degree</b>	2.8	2.9	1.9	2.9

The number of clusters varies between 4 and 8 and the density values are comparable to the averages over all intervals while the average degree in our example is lower than the average over all periods (cf. Table 4.2). In Figures 4.11, 4.13, 4.15 and 4.17, visualizations of the obtained clusterings are shown. Since most graphs consist of many noise vertices,

especially  $Enron_{total}^{month}$ , the graph visualizations only include close edges, i.e. the distance between the end-vertices  $(u, v)$  is smaller than  $\epsilon$ . All edges  $dist(u, v) > \epsilon$ , are removed from the graph to allow for a better presentation of the clusters.

The cluster resilience measure shown in Tables 4.4, 4.5, 4.6 and 4.7 specifies the number  $j$  of edges that need to be removed such that the probability of disconnection is 1. The DENGGRAPH-stability shows, after how many edge removals the probability that the cluster is removed or split is 1.

For this, each graph is clustered and the disconnection probability and the probability of a DENGGRAPH-cluster to be split or removed are determined for each obtained cluster. Furthermore, the observations for two clusters are shown exemplary for each graph. The results of the analysis are discussed in the following.

*Enron*<sub>intern</sub><sup>week</sup>: **Cluster Resilience and DENGGRAPH-Stability** The graph shown in Figure 4.11 consists of four clusters with 4 to 39 vertices and 3 to 63 edges. The cluster resilience for the largest cluster (Cluster 1) which has 39 vertices and 63 edges is 11, i.e. the probability for a cluster disconnection is 1 after the removal of 11 edges. However, the probability that the cluster is still existent after the removal of 11 edges regarding a reclustering with DENGGRAPH is approx. 0.4. The cluster collapses or splits with probability 1 after the deletion of 37 edges (cf. Table 4.4 and Figure 4.12).

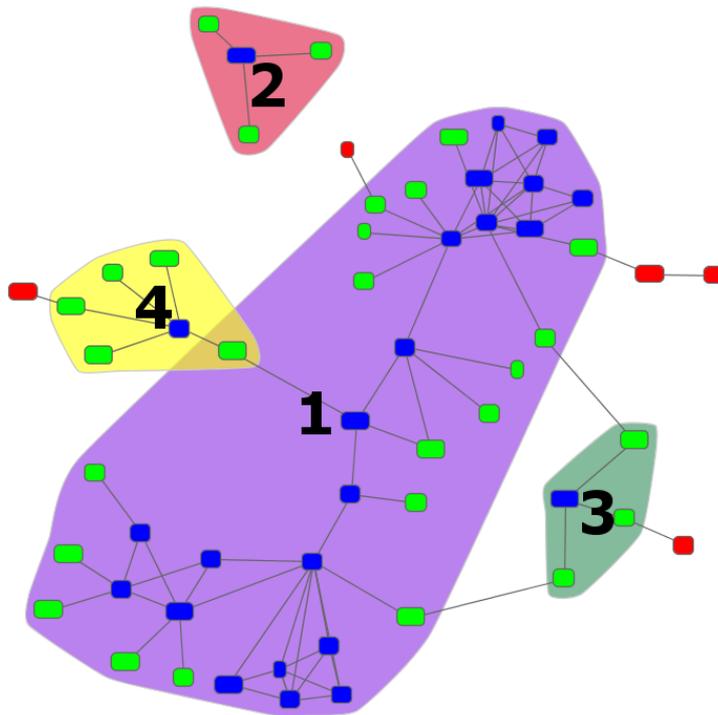
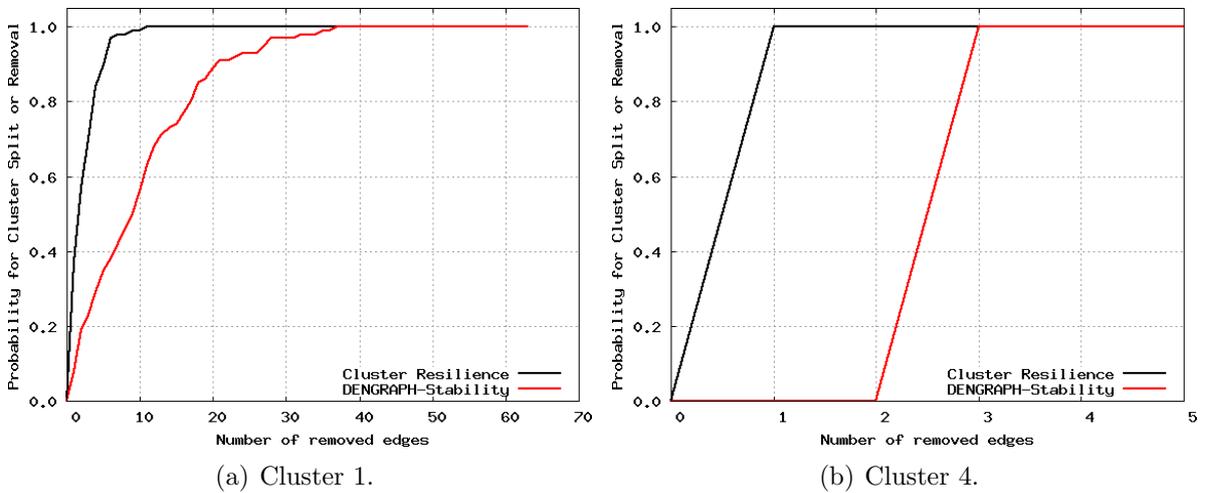


Figure 4.11: A graph visualization of the *Enron*<sub>intern</sub><sup>week</sup> graph captured in week 49/2000 clustered with DENGGRAPH-IO ( $\epsilon = 1$  and  $\eta = 3$ ).

In Table 4.4 the cluster resilience and DENGGRAPH-stability values for each cluster are shown. The second observed cluster (Cluster 4) is a star, i.e. it consists of one core vertex and five border vertices which are connected with the graph. Since all border vertices are only connected to the core, a removal of one edge results in a disconnection of the graph, i.e. the disconnection probability reaches one exactly after the removal of one edge. The DENGGRAPH-stability depends on the value of  $\eta$ . Since  $\eta$  is set to three in this experiment, the DENGGRAPH-stability is three: the cluster is removed if three edges are deleted as the core vertex has no longer an  $\epsilon$ -neighborhood with more than three neighbors. If  $\eta$  were five, the DENGGRAPH-stability would be one as the core vertex would lose its status after the deletion of one edge to a close neighbor. The clusters 3 and 4 are both stars each with one core vertex and three border neighbors. Both, cluster resilience and DENGGRAPH-stability are one for the two clusters.

Table 4.4: *Enron*: Cluster Resilience and DENGGRAPH-Stability (*Enron-intern*, *week*)

	Number of Vertices	Number of Edges	Cluster Resilience (j)	DENGGRAPH-Stability (j)
<b>Cluster 1</b>	39	63	11	37
<b>Cluster 2</b>	4	3	1	1
<b>Cluster 3</b>	4	3	1	1
<b>Cluster 4</b>	6	5	1	3

Figure 4.12: The cluster resilience (black line) and DENGGRAPH-stability (red line) of two clusters taken from the clustering of the *Enron*<sub>*intern*</sub><sup>*week*</sup> graph captured in week 49/2000 and clustered with DENGGRAPH-IO ( $\epsilon = 1$  and  $\eta = 3$ ).

*Enron*<sub>intern</sub><sup>month</sup>: **Cluster Resilience and DENGGRAPH-Stability** The clustering of the graph *Enron*<sub>intern</sub><sup>month</sup> with the parameters  $\epsilon = 0.95$  and  $\eta = 7$  reveals five clusters of different sizes and densities as shown in Figure 4.13. The number of vertices ranges from 8 to 25 and the number of edges from 7 to 45 (cf. Table 4.5). The largest cluster (Cluster 1) is disconnected with probability one after 17 edges have been removed. A removal or split of the clusters regarding DENGGRAPH-criteria is observable after the deletion of 27 edges. As shown in Figure 4.14 (a), the probability that the DENGGRAPH-cluster is removed is lower than the disconnection probability.

Clusters 2 and 4 are stars, both with one core vertex and eight and seven border neighbors, respectively. Both clusters are disconnected after the removal of one edge and since  $\eta$  is set to seven, Cluster 4 is also removed after the deletion of one edge (cf. Figure 4.14 (b)). Cluster resilience and DENGGRAPH-stability are in this case both one. Cluster 2 has a DENGGRAPH-stability of two because it is removed after the deletion of two edges.

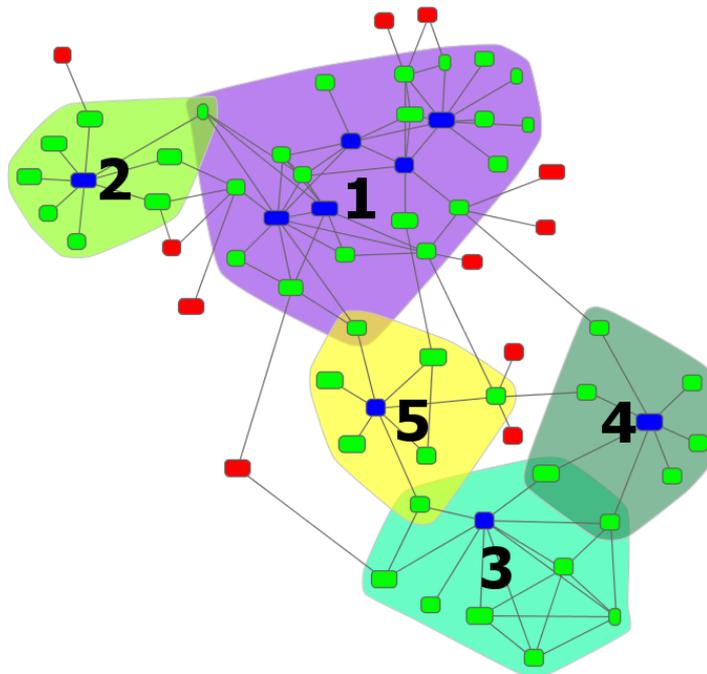
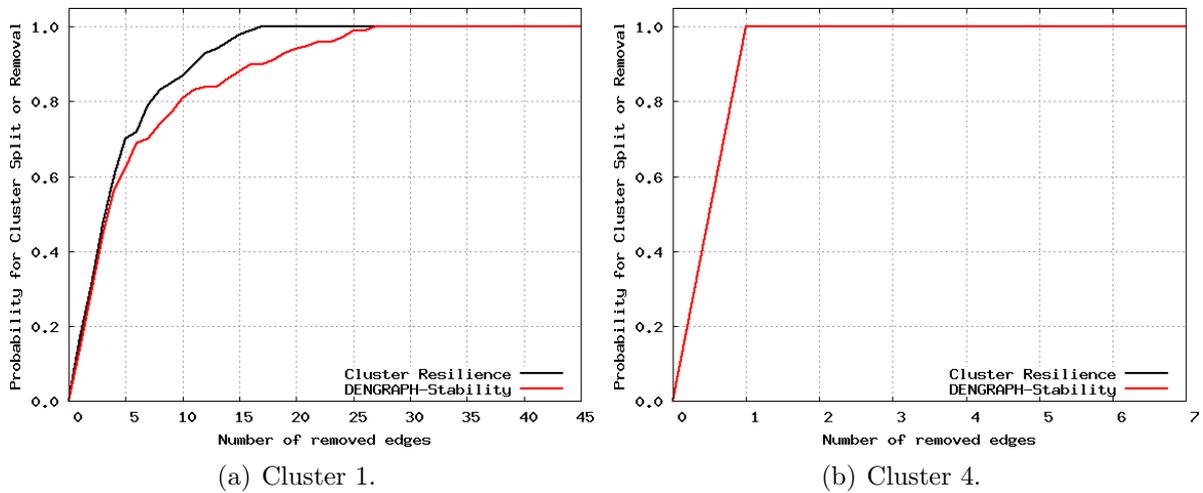


Figure 4.13: A graph visualization of the *Enron*<sub>intern</sub><sup>month</sup> graph captured in month 1/2001 clustered with DENGGRAPH-IO ( $\epsilon = 0.95$  and  $\eta = 7$ ).

Table 4.5: *Enron*: Cluster Resilience and DENGGRAPH-Stability (*Enron-intern*, month)

	Number of Vertices	Number of Edges	Cluster Resilience (j)	DENGGRAPH-Stability (j)
<b>Cluster 1</b>	25	45	17	27
<b>Cluster 2</b>	9	8	1	2
<b>Cluster 3</b>	10	18	10	11
<b>Cluster 4</b>	8	7	1	1
<b>Cluster 5</b>	8	8	2	2

Figure 4.14: The cluster resilience (black line) and DENGGRAPH-stability (red line) of two clusters taken from the clustering of the *Enron*<sup>month</sup><sub>intern</sub> graph captured in month 1/2001 and clustered with DENGGRAPH-IO ( $\epsilon = 0.95$  and  $\eta = 7$ ).

*Enron*<sub>total</sub><sup>week</sup>: **Cluster Resilience and DENGGRAPH-Stability** The graph *Enron*<sub>total</sub><sup>week</sup> clustered with parameters  $\epsilon = 0.6$  and  $\eta = 7$  is shown in Figure 4.15. Six clusters are detected with sizes varying from 11 to 25. Since all clusters are star-like structures, the disconnection probability is very high. Cluster 3, 4 and 5 are disconnected after the removal of one edge, Cluster 6 after two edge losses and Cluster 1 and 2 after three edge removals (cf. Table 4.6). The DENGGRAPH-stability is higher, depending on the difference between the number of neighbors and  $\eta$ . The higher the number of vertices, the higher the DENGGRAPH-stability.

The cluster resilience and DENGGRAPH-stability for Cluster 1 and 2 are depicted in Figure 4.16(a) and (b). The curves show that Cluster 2 is rather stable regarding a reclustering with DENGGRAPH since two core vertices with many close neighbors are present.

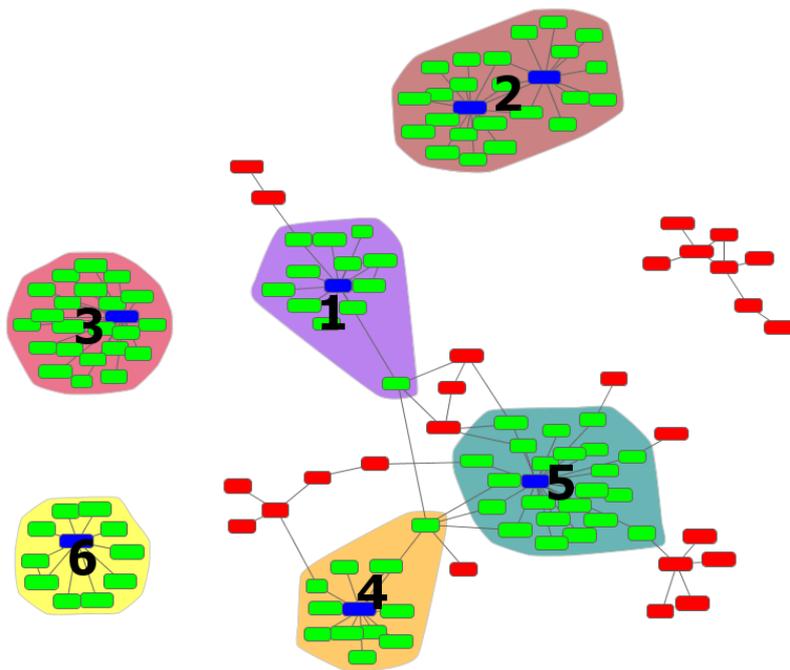


Figure 4.15: A graph visualization of the *Enron*<sub>total</sub><sup>week</sup> graph captured in week 50/2001 clustered with DENGGRAPH-IO ( $\epsilon = 0.6$  and  $\eta = 7$ ).

Table 4.6: *Enron*: Cluster Resilience and DENGGRAPH-Stability (*Enron-total*, *week*)

	Number of Vertices	Number of Edges	Cluster Resilience (j)	DENGGRAPH-Stability (j)
<b>Cluster 1</b>	13	14	3	8
<b>Cluster 2</b>	25	27	3	20
<b>Cluster 3</b>	23	22	1	16
<b>Cluster 4</b>	12	11	1	5
<b>Cluster 5</b>	23	22	1	16
<b>Cluster 6</b>	11	11	2	5

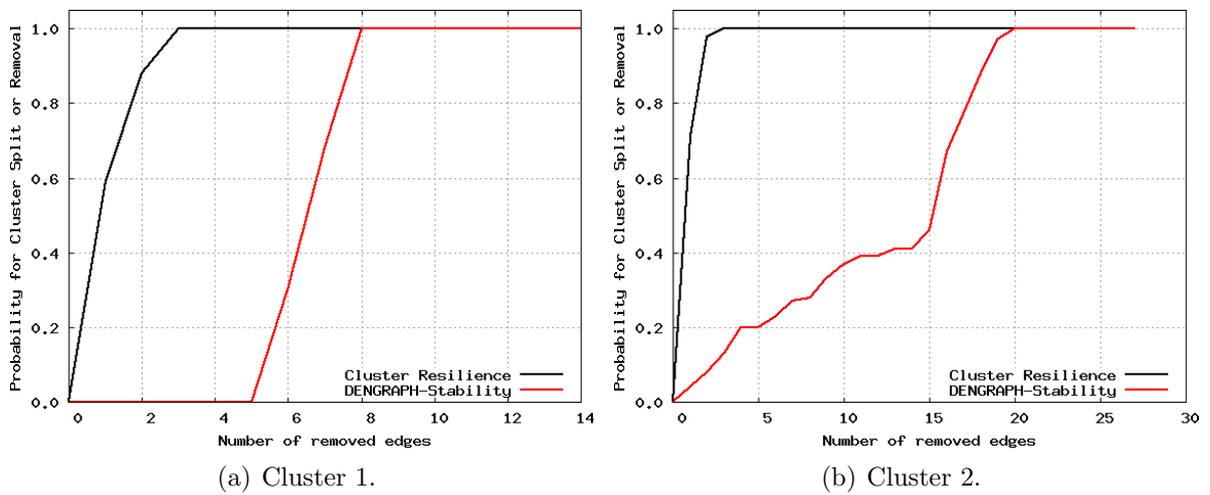


Figure 4.16: The cluster resilience (black line) and DENGGRAPH-stability (red line) of two clusters taken from the clustering of the  $Enron_{total}^{week}$  graph captured in week 50/2001 and clustered with DENGGRAPH-IO ( $\epsilon = 0.6$  and  $\eta = 7$ ).

*Enron*<sub>total</sub><sup>month</sup>: **Cluster Resilience and DENGGRAPH-Stability** The clustered graph displayed in Figure 4.17 is obtained after clustering the *Enron*<sub>total</sub><sup>month</sup> graph with parameters  $\epsilon = 0.6$  and  $\eta = 10$ . The clustering consists of one big component (Cluster 1) and seven clusters of size 10 to 20. Considering the large number of edges in Cluster 1, the disconnection resilience is very low. The cluster is already disconnected after the removal of 13 edges (cf. Table 4.7 and Figure 4.18(a)). This can be explained by the very low density of the cluster which results in a higher likelihood of disconnections if edges are removed. The stability regarding DENGGRAPH is higher and the probability of a removal or split reaches 1 after the deletion of 231 edges.

Cluster 6, as shown in Figure 4.18(b), is disconnected after the removal of three edges and the cluster is no longer a DENGGRAPH-cluster after seven edges are deleted.

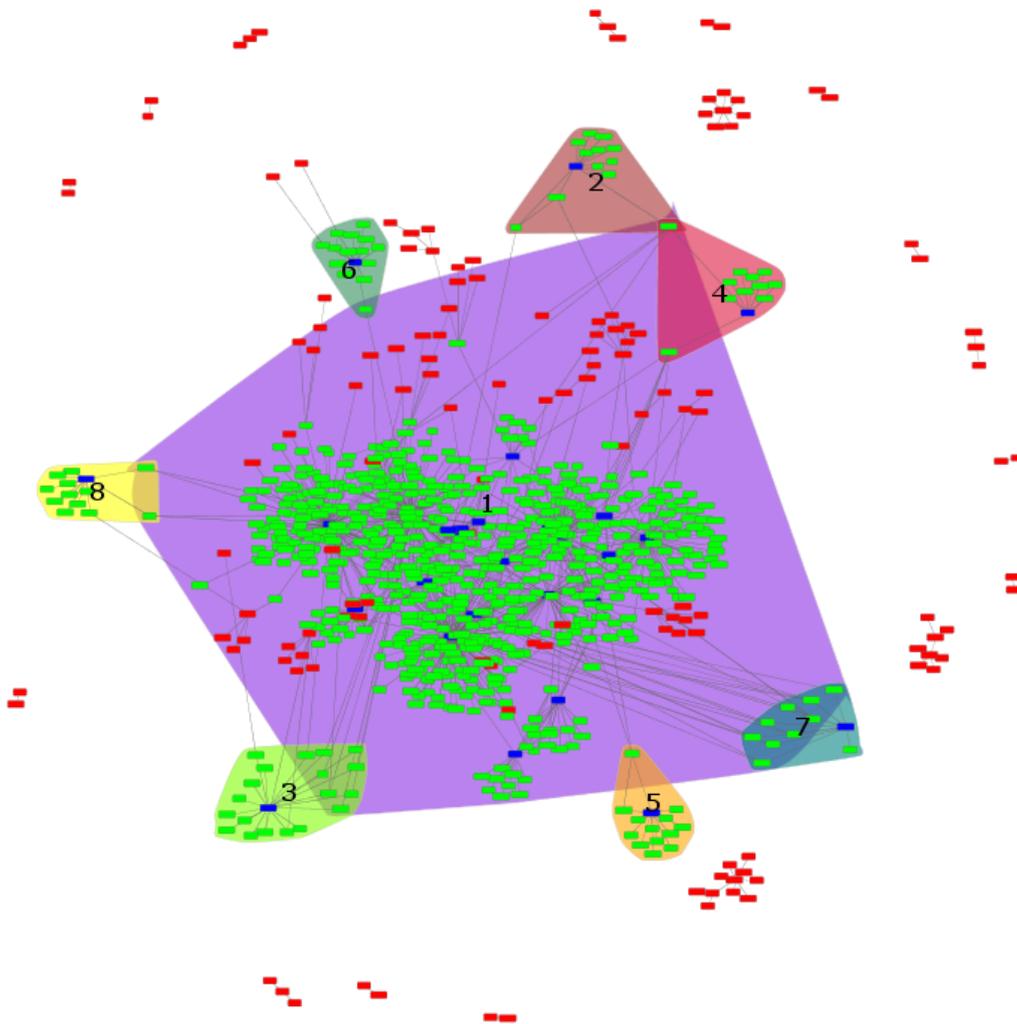
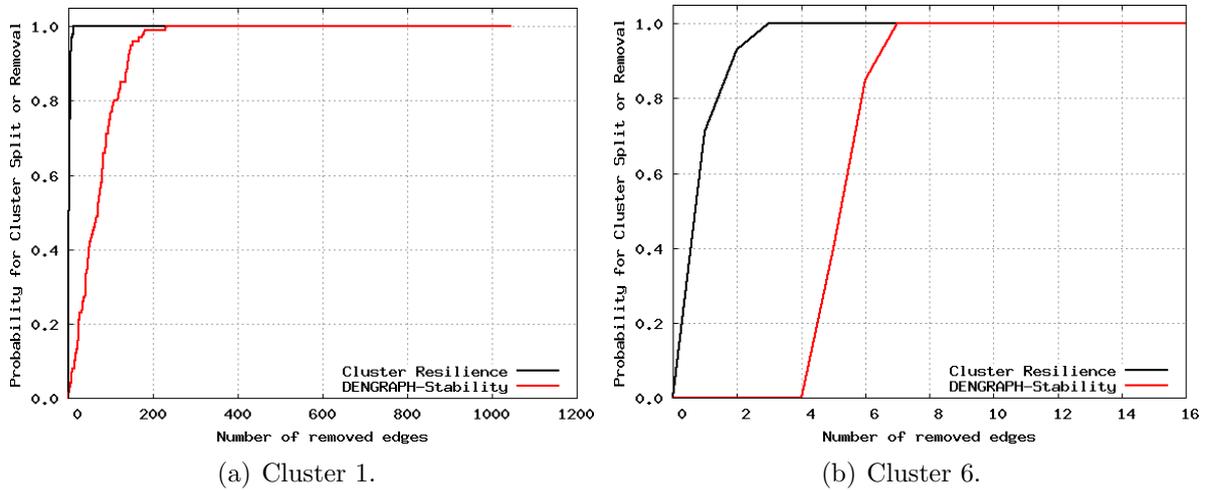


Figure 4.17: A graph visualization of the *Enron*<sub>total</sub><sup>month</sup> graph captured in month 4/2001 clustered with DENGGRAPH-IO ( $\epsilon = 0.6$  and  $\eta = 10$ ).

Table 4.7: *Enron*: Cluster Resilience and DENGGRAPH-Stability (*Enron-week, month*)

	Number of Vertices	Number of Edges	Cluster Resilience (j)	DENGGRAPH-Stability (j)
<b>Cluster 1</b>	668	1044	13	231
<b>Cluster 2</b>	13	13	2	4
<b>Cluster 3</b>	20	19	1	10
<b>Cluster 4</b>	12	11	1	2
<b>Cluster 5</b>	14	14	2	5
<b>Cluster 6</b>	15	16	3	7
<b>Cluster 7</b>	11	10	1	1
<b>Cluster 8</b>	13	12	1	3

Figure 4.18: The cluster resilience (black line) and DENGGRAPH-stability (red line) of two clusters taken from the clustering of the  $Enron_{total}^{week}$  graph captured in month 4/2001 and clustered with DENGGRAPH-IO ( $\epsilon = 0.6$  and  $\eta = 10$ ).

**Remarks** Experiments have shown that the disconnection probability strongly depends on the structure of the cluster. Stars or star-like structures are particularly vulnerable to edge deletions as border vertices are only connected to one core vertex. Besides, if border members have no other neighbor, a deletion of one edge already results in a disconnection of the cluster. On the other hand, DENGGRAPH-stability is in general higher, indicating that the obtained clusters are not particularly sensitive to edge removals. Furthermore, depending on the number of vertices and edges in the cluster, the experiments indicate that DENGGRAPH-clusters are stable to minor perturbations of the input data. This is an important characteristic to ensure that small changes in the graph structure do not strongly influence the clustering.

### Temporal Dynamics in the Enron Graph

In the following, an analysis of the temporal evolution of the Enron graph is presented. We discuss the changes that can be observed based on graph and cluster statistics. Over the observation period for each interval the following measures are charted:

- Number of edges and nodes (border, core and noise)
- Clustering coefficient and average degree
- Number of clusters
- Cluster performance and weighted and unweighted modularity
- Number of cluster creations, merges and splits
- Number of positive and negative updates
- CPU-time for DENGGRAPH and DENGGRAPH-IO

Exemplary the plots for  $Enron_{total}^{week}$  are shown in Figure 4.19. However, the results presented in the following are observable for all four graphs and the diagrams of all graphs, which depict the presented information, are shown in Figures B.5, B.6, B.7 and B.8.

The number of clusters varies for all graphs. In the  $Enron_{intern}^{week}$  and  $Enron_{intern}^{month}$  graph, only a small number of clusters are detected because the graphs are very small. We often encounter periods, where no cluster is detected. For  $Enron_{total}^{week}$  the number of clusters varies from one to fourteen and for  $Enron_{total}^{month}$  from one to eight. We observe that the values for weighted and unweighted modularity are in general comparable. As expected, the unweighted modularity is usually lower than the weighted and only in some cases it is the other way around (cf. the discussion about the differences between weighted and unweighted similarity in Section 2.4).

The clustering coefficient fluctuates slightly around an average value of 0.4. Only the curve for  $Enron_{intern}^{week}$  shows a stronger oscillation. The number of vertices does not change considerable, however, the number of edges increases over time and also shows periods with sudden increases followed by periods with strong reductions. This corresponds with

the observation that the average degree increases over the observation period and has higher fluctuations when the number of edges increases or decreases.

Furthermore, a correlation between the number of edges and the number of updates can be observed: When the number of edges increases, the number of positive updates increases as well, usually followed by a period with a higher number of negative updates. Therefore, fluctuations of the number of edges result in fluctuations in the number of updates. In some intervals a correlation between the number of positive updates and the number of splits can be seen (cf. for example Figure B.8, interval 51 or Figure B.7, interval 102). The same holds for the number of negative updates and the number of splits: The number of splits increases when many edges are lost (cf. for example Figure B.7, interval 103 or Figure B.6, interval 52).

The CPU-time for the static DENGGRAPH procedure shows only slight fluctuations, depending on the size of the graph. An obvious observation is that the number of updates has a strong impact on the computation time of DENGGRAPH-IO: The more updates the procedure has to process, the higher the required CPU-time. However, it is noteworthy that the two types of updates have a different affect on the computation time: The CPU-time for positive updates which result in more merges is higher than for negative updates which implicate more splits. Therefore, in general a correlation between the number of updates and the CPU-time exists but it is higher the larger the fraction of positive updates is on the number of all updates. We will discuss this observation in more detail in Section 4.6.

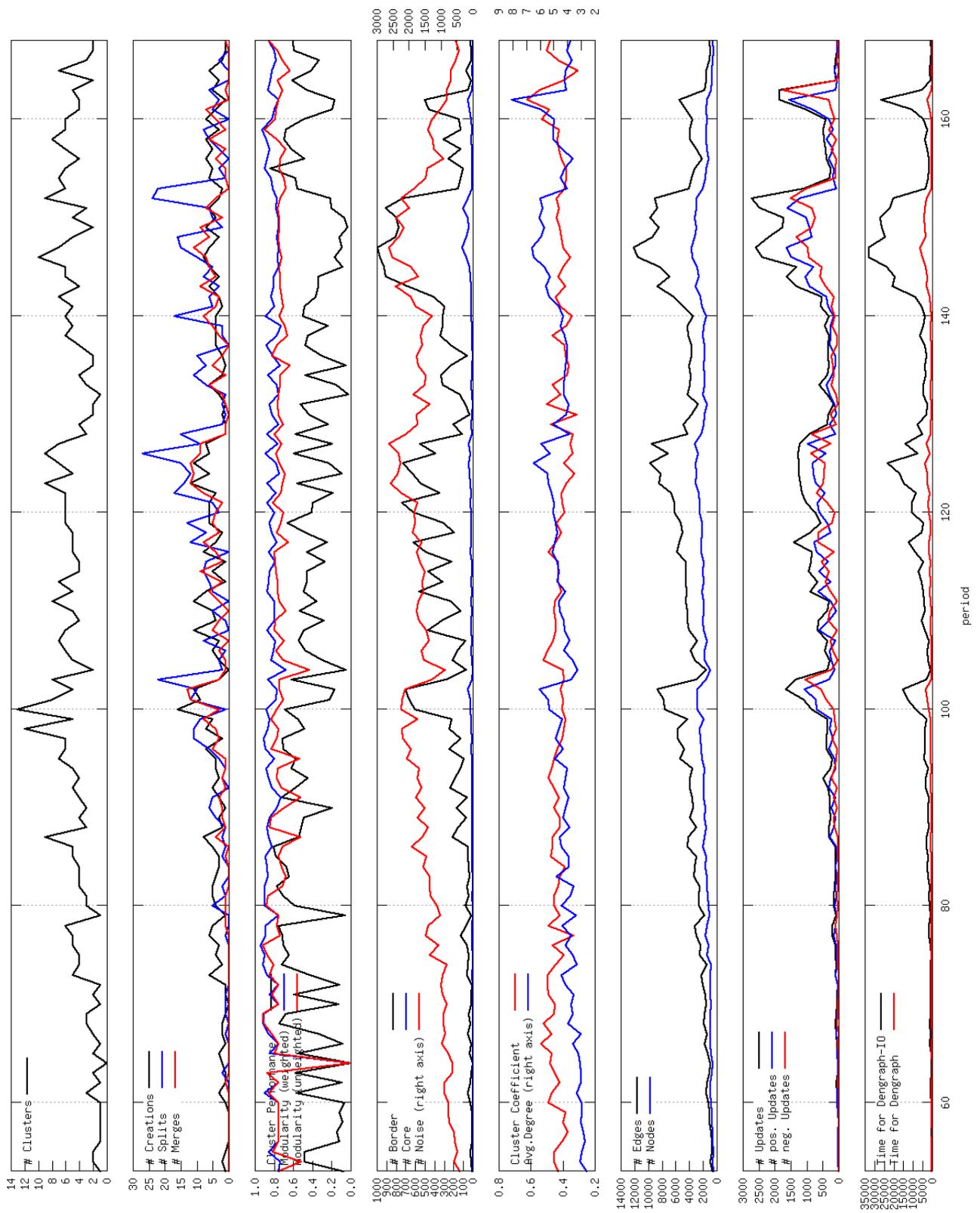


Figure 4.19:  $Enron_{total}^{week}$ ;  $\epsilon = 0.6$ ,  $\eta = 7$ .

## 4.5 Application: Graph over Similarity

In this section, we discuss results of an application of DENGGRAPH-IO on temporal data we collected which represents the music listening preferences of individuals over time. Based on data obtained from a social networking sites, a graph of users is constructed in which individuals are connected regarding the similarity of their user profiles. For this, at first, user profiles are built and a distance function is selected to determine the distance between individuals.

The influence of the parameters  $\epsilon$  and  $\eta$  on the clusterings is analyzed and the stability of the clusters regarding perturbations of the input graph are discussed. Furthermore, a temporal analysis of the graph evolution is conducted and the resulting changes are discussed. To assess whether the obtained clusterings and the detected temporal changes are meaningful, the results of a semantic evaluation are presented. This evaluation is based on a comparison of labels which are assigned to clusters in each time step.

### Why Using Data about Music Preferences?

Music is a constant companion for most people. In 2000, Eurostat, the statistical office of the European Communities, carried out a survey on Europeans' participation in cultural activities, one part of which concerned music. The results show, for example, that over 60% of Europeans listen to music every day [51]. Music is known to induce emotions such as "happiness", "sadness" and "fear" [108], but the meaning of music in the lives of people might change over time. In parallel the music preferences of people might change as well. Younger people tend to link music preferences to a variety of psychological characteristics such as personality, personal qualities and values. Since, individuals have clearly defined stereotypes about the fans of various music genres, music is used to convey information about oneself to observers [138]. On the other hand, for older people, music often contributes to positive aging by providing ways to maintain positive self-esteem, feel competent, independent and avoid feelings of isolation or loneliness [85].

Why is all this of interest here?: A problem of every community detection technique is to show that the subgraphs that are extracted are actually "structures" that are considered to be meaningful communities in the context of the platform, i.e., a definition of a community is always context dependent. In a network of interactions we define a community as a subgroup of actors who interact more often inside the community than to outsiders (cf. Section 2.2.2). For example, in the *Enron* application, we determined groups of people by counting the numbers of interactions between pairs of users without considering the content of the communication. We did not take into account the semantics of the interactions: A spam email is also considered as an interaction as well as a personal discussion. Thus, we built communities based on the number of interactions while neglecting the nature of the interaction, i.e. we cannot distinguish between a constructive discussion, flame messages or a heated debate.

We therefore chose to build a data set where

- the similarity of users can be determined based on an objective criteria, here, the similarity of their music preferences and

- it can be assumed that the similarity changes over time. The aim is then to determine the music preferences of people and to observe how the taste in music changes over time.

### 4.5.1 Last.fm Data Set

*Last.fm*<sup>3</sup> is a social networking platform established in 2002. According to *Last.fm*, the platform has over 20 million users on the site every month, which are based in more than 200 countries. After a user signs up with *Last.fm*, a plugin is installed and all music tracks he listens to are submitted to a database. This process is called “scrobbling”. *Last.fm* provides plugins for different devices, besides music players on PCs also other portable devices such as mobile phones and mp3 players are supported. The submitted information about the tracks listened to are used to provide statistics about a user’s listening behavior that are displayed on a profile page. The site also recommends similar artists based on the user’s favorites and offers several social networking features.

*Last.fm* records - among others - all artists a user listens to, aggregates this information over seven days and provides lists of the most listened artists for each week over the lifetime of a user. We use this information to build a user’s profile by extracting the genres of the most listened artists. The artist’s genre is determined by the tags that the community members use to characterize the artist. The profile of a user is then represented by a vector of the weighted tags of his favorite artists. We represent the users as vertices in a graph and connect users with an edge, if their profile similarity reaches a predefined threshold. The similarity is determined by calculating the distance between pairs of genre vectors using the cosine similarity measure.

From the 20 million users we randomly chose approx. 600,000 and obtained over a period of 167 weeks (September 2005 to November 2008) their weekly artists charts. Since many users are not active on a regular basis, we chose randomly 2,000 users from this set who were active in at least 80% of all periods.

### 4.5.2 Defining Similarity in the Last.fm Graph

To study music listening behaviors and their temporal evolution, we need a representation for a user’s taste in music. Therefore, we describe in this section how we built the user profiles. Furthermore, to be able to find groups of similar users, we discuss how a graph of similar users is constructed.

#### Building User Profiles

In affiliation networks, users are connected based on a certain property that they share (cf. Section 2.1). This property can be for example a similar taste in music. We represent a user by a profile that describes his music preferences and determine communities of users by clustering similar profiles using DENGRAPH-IO. The profile that we build is a combination of the genres assigned to the artists a user listens to.

---

<sup>3</sup><http://www.last.fm/>

Genre labels are often used as ground-truth for describing song or artist similarity in music information retrieval (MIR) systems. The goal is to identify the genre of a given artist or song, e.g. for indexing purposes or to recommend similar artists. A genre is a rather high level label for an artist and depends on a variety of aspects such as the rhythm, melody or used instruments. Genres are usually predefined by producers or the artist and the resulting classes are highly overlapping.<sup>4</sup>

In contrast to using a single label, McKay and Fujinaga [118] suggest to use a ranked list of multiple genres to label one recording. These labels that are assigned to an artist are often called *tags*. However, the question remains, how to obtain tags that are deemed objective to characterize the artists appropriately. For this purpose, Geleijnse, Schedl and Knees [71] propose to use tags that result from community Web sites where users collectively attribute artists with labels. Contrary to expert-defined genre classification, a community-based tagging results in a ranked list of terms that is richer than a single genre. The authors discuss in [71] that *Last.fm* tag data is composed by a large community of users and that the tags present a valuable characterization of artists and their music. Experiments have shown that the tags applied to artists in *Last.fm* are descriptive and consistent with respect to their similarity and that good results were achieved in a genre classification task [71]. Therefore, we used the most often used genre tags for artists which are provided by *Last.fm*. We merged some tags that were semantically identical (such as “hip hop” and “hip-hop”) and obtained a set  $W$  of 222 genre tags  $w_i$ .

For each artist, *Last.fm* provides the tags that are used by users to describe the artist. For example, the band “ABBA” has 41 tags. The most often used tags are “pop”, “disco”, “swedish” and “70s”. The singer “Amy Winehouse” has 34 tags, the most often used tags are “soul”, “jazz”, “female vocalists” and “british”. We use all provided tags that are in the set  $W$  to describe the artists in a genre vector as follows:

**Definition 4.15**  $A = \{\vec{a}_1, \dots, \vec{a}_n\}$  is a set of artists. An artist  $a_i$  is defined as a genre vector  $\vec{a}_i = (w_1, w_2, \dots, w_k)$  of the  $k$ -most used genre tags  $w_i \in W = \{w_1, \dots, w_k\}$ , where  $W$  is the set of genres.

For each user  $u$  we obtain for each one-week interval  $t$  a list of the most listened to artists, including the number of times the respective artist was heard by the user (*playcount*). Thus, we retrieve from *Last.fm* for each interval  $t$  and user  $u$  a set of artists and their playcount:  $u_t = \{(\vec{a}_i, \text{playcount}_i)\}$ . The user profile in interval  $t$  is defined as follows:

**Definition 4.16** For each user  $u$  in interval  $t$ , a user profile is defined as

$$\vec{u}_t = \sum_{i=1}^m \vec{a}_i \cdot \text{playcount}_i, \quad (4.10)$$

while  $m$  is the number of artists listened to in the interval.

Thus, each genre element is weighted by the number of times played. The weighted vectors are then added and the user profile is a vector representing the user’s music preferences.

<sup>4</sup>Developing automatic genre classification systems is a controversial (see, e.g., [8]) but still popular topic in music information retrieval and remains a challenging task (cf., e.g, [147], [12], [115]).

### Building a Graph of Similar Users

To find groups of users with similar music preferences, we first build a similarity graph. An edge in the similarity graph indicates that two users are similar to a certain extent. We build the graph in each iteration based on the similarity of the user profiles. We determine the similarity of two profiles by using the *cosine similarity* which is often applied in information retrieval to measure the similarity of documents which are represented as term vectors (see. e.g. [145]). The cosine of the angle  $\theta$  between two vectors  $\vec{a}$ ,  $\vec{b}$  can be interpreted as a measure of similarity and is calculated as follows:  $\cos(\theta) = (\vec{a} \bullet \vec{b}) / (|\vec{a}| |\vec{b}|)$ . If the vectors are pre-normalized, then the dot product between unit vectors equals the cosine angle since the vector length will be unit length. A cosine value of zero indicates that the two profile vectors are orthogonal, a value of one indicates an exact match. The similarity between the user  $u$  and  $v$  is then defined as follows:

**Definition 4.17** *The similarity between two users  $u$  and  $v$  is defined as*

$$\text{sim}(u, v) = \frac{\sum_{i=1}^m u_i \cdot v_i}{\sqrt{\sum_{i=1}^m u_i^2 \cdot \sum_{i=1}^m v_i^2}} \quad (4.11)$$

Using this formula, we calculate the pairwise similarity of user profiles for each interval  $t$ . Since DENGGRAPH-IO requires the distance between actors, we defined the distance as follows:

**Definition 4.18** *The distance between two users  $u$  and  $v$  is defined as*

$$\text{dist}(u, v) = 1 - \text{sim}(u, v) \quad (4.12)$$

The extracted graph consists of edges  $(u, v)$  where  $\text{dist}(u, v) \leq \epsilon$ . Thus, two nodes are only connected with an edge, if their distance is lower than  $\epsilon$ . By this we achieve, that only users with similar music preferences are in each other's  $\epsilon$ -neighborhood. Finally, we obtain an weighted, undirected graph of similar users.

**Definition 4.19** *The similarity graph  $G$  is represented by a list of nodes  $V$  and a list of edges  $E$ , where  $\text{dist}(u, v) \leq \epsilon \forall (u, v) \in E$ .*

### Cluster Labeling

After determining groups of users with similar music preferences in each interval by applying DENGGRAPH-IO on the graph, we label each cluster with the most often used genre tags to characterize the cluster. For this, we combine the genre vectors of the user profiles of all core nodes in the respective cluster. The cluster is thus labeled with a genre vector just as a user profile.

**Definition 4.20** *A cluster label  $\vec{l}(C_i)$  of cluster  $C_i$  is defined as*

$$\vec{l}(C_i) = \frac{\sum_{m=1}^n \vec{u}_m}{n}, \quad (4.13)$$

where  $u$  is core vertex in cluster  $C_i$  and  $n$  is the number of core vertices in cluster  $C_i$ .

### 4.5.3 Experiments and Results

We discuss the impact of the parameters  $\epsilon$  and  $\eta$  on the clustering of the *Last.fm* graph and present results of an experiment for a chosen parameter combination. Furthermore, transitions of genre clusters which were detected during the analysis are discussed.

#### Influence of Parameters $\epsilon$ and $\eta$

To determine the impact of  $\epsilon$  and  $\eta$  we conducted experiments with different parameter combinations. We tested  $\epsilon$  values in the range from 0.005 to 0.2. Starting from 0.005 we increased at each step by 0.005 until  $\epsilon$  reached 0.02. Afterwards, we increased the value by 0.01, resulting in 22 different values for  $\epsilon$ . For  $\eta$ , 14 equidistantly distributed values in the range from 2 to 15 were tested. With each step, we increased  $\eta$  by one. These combinations were used to determine the number of clusters, the cluster performance, the modularity and the noise ratio for each interval of one week. Exemplary, the results for each combination of  $\epsilon$  and  $\eta$  for a chosen interval are shown in Figure 4.20.

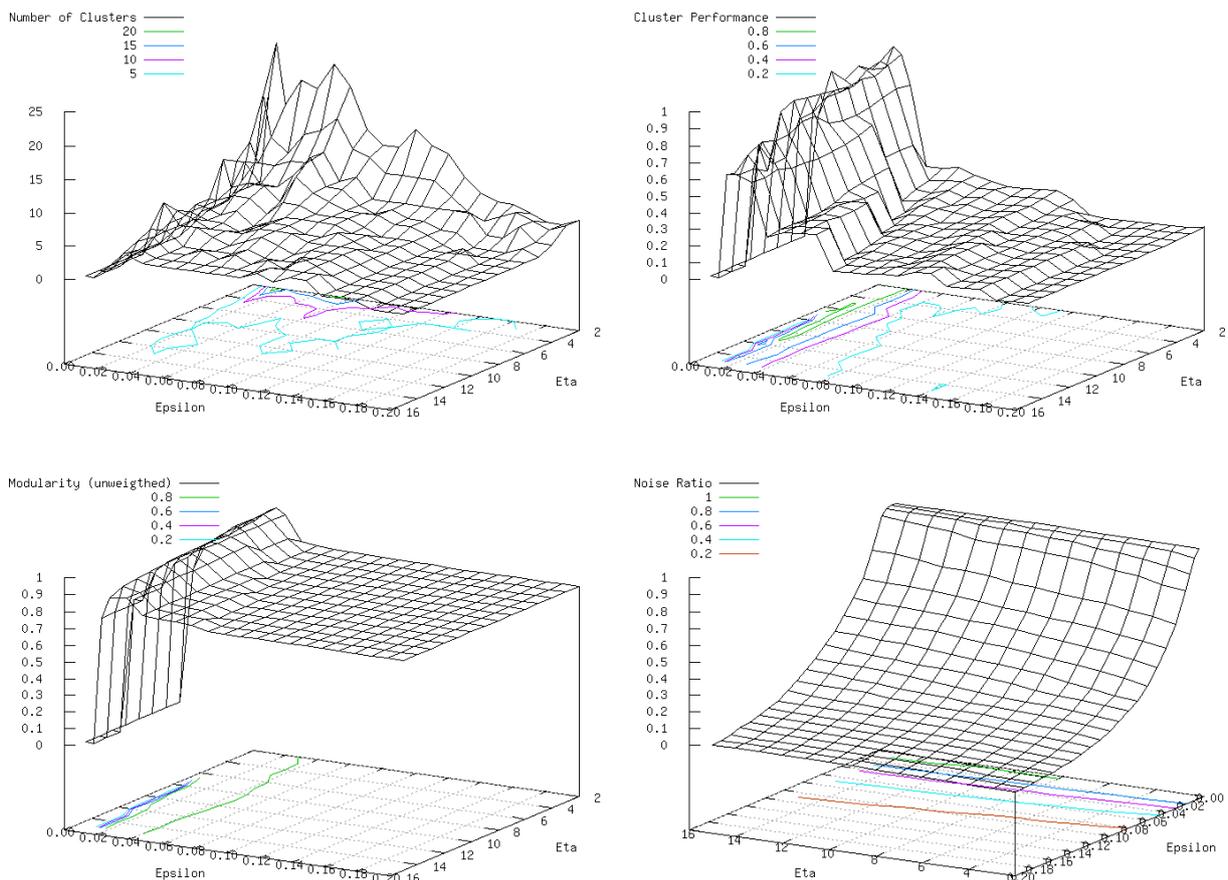


Figure 4.20: The influence of the parameters  $\epsilon$  and  $\eta$  on the clustering in one interval.)

The quality of clusters, regarding the cluster performance measure, mainly depends on the choice of  $\epsilon$ ;  $\eta$  has only a minor impact on the performance. High cluster performance

values, as depicted in the upper right plot in Figure 4.20, can be reached for  $\epsilon$  values between  $[0.01, 0.04]$ . The maximal value is reached for  $\epsilon = 0.015$ . The influence of  $\eta$  is less strong: from  $[2, 12]$ , performance values do not change considerably. The same holds for the modularity, however, this measure is less sensitive to changes of  $\epsilon$ . Highest values are also obtained by choosing  $\epsilon$  in the interval  $[0.01, 0.04]$  (cf. lower left plot in Figure 4.20). As soon as clusters are established, modularity increases strongly, because only few inter-cluster edges are observable in the clusterings which results in high modularity values.

Both quality measure reach very high values for this data set, because almost no inter-cluster edges are observable. Compared to the modularity, the cluster performance value fluctuates stronger because the measure also considers the number of possible edges in a cluster and since the density of clusters changes, mainly depending on  $\epsilon$ , the performance values differ stronger than the modularity values.

As discussed in the previous application, for the choice of parameters it is important to consider also the number of clusters and the noise ratio. The number of clusters increases with decreasing  $\eta$  since the probability for core nodes increases (cf. upper left plot in Figure 4.20). A high value for  $\epsilon$  does not necessarily result in a higher cluster number. This can be explained by low similarity values between profiles, due to a high dimensionality of the profile vectors. The noise ratio decreases with increasing  $\epsilon$  as more actors become members of each other's  $\epsilon$ -neighborhood. For  $\epsilon$  between 0.04 and 0.08, about 20-60% of all actors are noise vertices. When choosing an  $\epsilon$  that maximizes the cluster performance, the noise ratio is about 60-80%.

For the experiments presented in the following we chose  $\epsilon = 0.04$  and  $\eta = 5$ . This combination yields medium to high performance and modularity values and an average number of clusters.

### Resilience and Stability of DENGGRAPH-Clusters in the Last.fm Graph

In this section, the cluster resilience (cf. Definition 2.6) and the stability of the obtained clusters regarding edge removals (cf. Definition 4.10) are discussed. The statistics of the graphs in the respective intervals are presented in Table 4.8.

Table 4.8: *Last.fm*: Clustering statistics for two chosen weeks.

		$Last.fm_{\epsilon=0.04}^{\eta=5}(week10/2007)$	$Last.fm_{\epsilon=0.04}^{\eta=5}(week14/2007)$
Number of Vertices	Core	645	585
	Border	1016	205
	Noise	166	203
<b>Number of Edges</b>		7072	6911
<b>Density</b>		0.014	0.014
<b>Avg. Degree</b>		13.9	13.7

*Last.fm* $_{\epsilon=0.04}^{\eta=5}$  (**week 10/2007**) In week 10/2007, seven clusters are detected (cf. Figure 4.26). The cluster labels as well as the cluster resilience and the cluster stability regarding DENGGRAPH (DENGGRAPH-stability) are shown in Table 4.9. The values vary depending on the structure of the cluster. In most cases, the cluster resilience is lower than the DENGGRAPH-stability, an observation that was also made with the *Enron* data.

Table 4.9: *Last.fm*: Cluster Resilience and DENGGRAPH-Stability ( $\epsilon = 0.04$ ,  $\eta = 5$ , week 10/2007)

	Number of Vertices	Fraction of Core/Border	Cluster Density	Cluster Resilience	DENGGRAPH- Stability
<b>Cluster 1:</b> <b>indie</b>	643	0.77/0.23	0.03	496	2676
<b>Cluster 2:</b> <b>metal</b>	136	0.74/0.26	0.08	217	376
<b>Cluster 3:</b> <b>hip-hop</b>	32	0.91/0.09	0.66	273	315
<b>Cluster 4:</b> <b>j-rock,rock</b>	7	0.14/0.86	0.43	4	5
<b>Cluster 5:</b> <b>j-rock,j-pop</b>	7	0.43/0.57	0.57	7	7
<b>Cluster 6:</b> <b>j-pop</b>	21	0.76/0.24	0.41	66	81
<b>Cluster 7:</b> <b>industrial</b>	6	0.17/0.83	0.47	3	3

In Figure 4.21 the cluster resilience and stability measures for four clusters are displayed. A graph visualization of the clusters is shown in Figure 4.26. The difference between the cluster resilience and the DENGGRAPH-stability depends, as mentioned, on the cluster structure.

The “indie” cluster shown in Figure 4.21 (a), is a very large cluster with more than 600 vertices and a very low density of 0.03. Most vertices in this cluster are core vertices (approx. 75%). The cluster resilience is lower compared to the DENGGRAPH-stability. An explanation for the higher DENGGRAPH-stability could be that the high percentage of core vertices compensates the very low density. Therefore, disconnections of single vertices due to the removal of edges do not immediately result in a split or removal as many core vertices are still connected.

Compared to the previously discussed clusters, the “hip-hop” and “j-pop” clusters are much denser (density=0.66 and 0.41, respectively) resulting in much higher cluster resilience and DENGGRAPH-stability values (cf. Figures 4.21 (b) and (c)). The cluster resilience of the “hip-hop” cluster is lower than of the “j-pop” cluster, because the “hip-

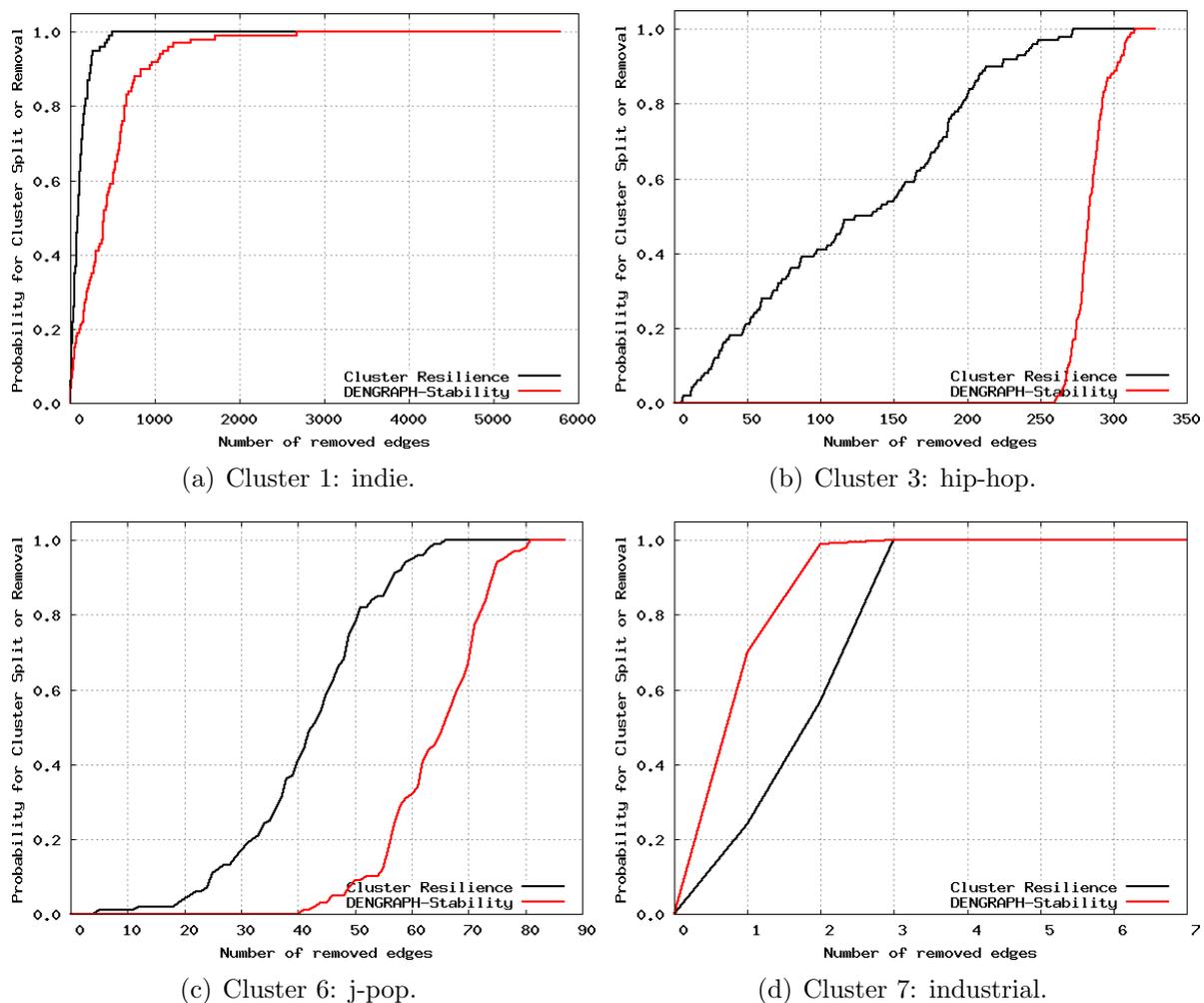


Figure 4.21: The cluster resilience (black line) and DENGGRAPH-stability (red line) of four clusters taken from the clustering of the *Last.fm* graph with  $\epsilon = 0.04$ ,  $\eta = 5$  captured in week 10/2007 and clustered with DENGGRAPH-IO.

hop” cluster contains more star-like structures and disconnections of single vertices are therefore more likely.

In contrast to the other three clusters, in the “industrial” cluster the resilience is higher than the DENGGRAPH-stability (cf. Figure 4.21 (d)). The reason is that this cluster has only one core vertex with five members in its  $\epsilon$ -neighborhood. Since  $\eta$  is also five, the removal of one edge to a neighbor would result in a lost core status of this vertex. The probability for a cluster removal is therefore very high, because five out of seven edges are necessary to ensure the core status of the vertex.

*Last.fm* $_{\epsilon=0.04}^{\eta=5}$  (**week 14/2007**) In week 14/2007, DENGGRAPH detects ten clusters as shown in the graph visualization in Figure 4.28. The cluster labels as well as the cluster resilience and the DENGGRAPH-stability are shown in Table 4.10. As already observed in week 10/2007, the progression of the curves varies depending on the structure of the clusters.

Table 4.10: *Last.fm*: Cluster Resilience and DENGGRAPH-Stability ( $\epsilon = 0.04$ ,  $\eta = 5$ , week 14/2007)

	Number of Vertices	Fraction of Core/Border	Cluster Density	Cluster Resilience	DENGGRAPH- Stability
<b>Cluster 1:</b> <b>indie</b>	591	0.75/0.25	0.03	599	4425
<b>Cluster 2:</b> <b>death metal</b>	104	0.79/0.21	0.11	234	463
<b>Cluster 3:</b> <b>hip-hop</b>	31	0.84/0.16	0.5	156	219
<b>Cluster 4:</b> <b>industrial</b>	18	0.56/0.44	0.16	41	34
<b>Cluster 5:</b> <b>j-pop</b>	13	0.54/0.46	0.39	19	23
<b>Cluster 6:</b> <b>rock</b>	13	0.38/0.62	0.32	14	20
<b>Cluster 7:</b> <b>metal, heavy</b>	12	0.33/0.67	0.32	10	16
<b>Cluster 8:</b> <b>metal, punk</b>	7	0.14/0.86	0.57	7	7
<b>Cluster 9:</b> <b>pop</b>	7	0.29/0.71	0.57	7	6
<b>Cluster 10:</b> <b>rock, metal</b>	6	0.17/0.83	0.6	5	5

In Figure 4.22 plots of the cluster resilience and the DENGGRAPH-stability are shown for four different clusters. The “death metal” cluster has, compared to the other three cluster, a low density of 0.11 but with 79% a high percentage of core vertices. The probability of disconnections is comparably high resulting in a lower cluster resilience. The DENGGRAPH-stability on the other hand is higher, mainly because of the large number of core vertices (cf. Figure 4.22 (a)).

For the “industrial” cluster, even though the cluster resilience is again lower than the DENGGRAPH-stability, both curves do not differ much (cf. Figure 4.22 (c)). The density lies in between the values of other clusters and almost 50% of the vertices are cores,

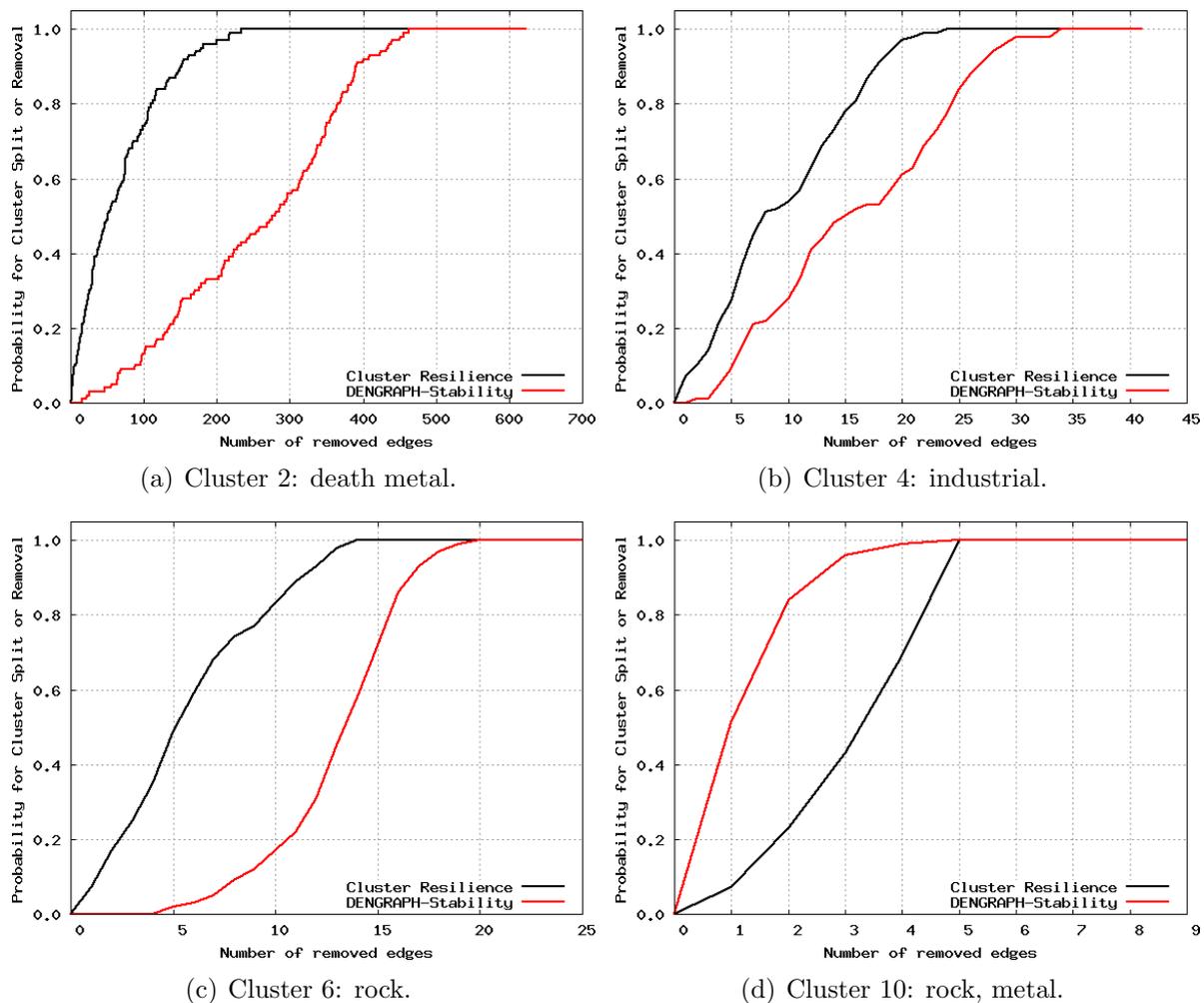


Figure 4.22: The cluster resilience (black line) and DENGGRAPH-stability (red line) of four clusters taken from the clustering of the *Last.fm* graph with  $\epsilon = 0.04$ ,  $\eta = 5$  captured in week 14/2007 and clustered with DENGGRAPH-IO.

the other borders. The “rock” cluster consists of only one core vertex with five neighbors. Since  $\eta$  is five, the cluster is removed if an edge to one of these members is lost. The cluster resilience is higher, because also connections between border members exists. These edges are not relevant for the DENGGRAPH-cluster, but they result in a higher connectivity in the cluster and vertices are still connected even though a border loses an edge to the core (cf. Figure 4.22 (d)).

The cluster resilience and DENGGRAPH-stability plots of all clusters in week 10/2007 and 14/2007 are displayed in Appendix B.

### Temporal Dynamics in the Last.fm Graph

We applied DENGGRAPH on the data set to detect and observe the evolution of clusters during the observation period of 115 weeks. The aim was to see, whether the proposed clustering technique detects meaningful communities and their evolutions.

The data set characteristics in five weeks for the *Last.fm* graph obtained with the parameters  $\epsilon = 0.04$  and  $\eta = 5$  are shown in Table 4.11.

Table 4.11: *Last.fm*: Statistics; Exemplary for Five Weeks ( $\epsilon = 0.04$ ,  $\eta = 5$ )

	<i>Last.fm</i> (08/07)	<i>Last.fm</i> (09/07)	<i>Last.fm</i> (10/07)	<i>Last.fm</i> (13/07)	<i>Last.fm</i> (14/07)
<b>Number of Clusters</b>	9	8	7	10	10
<b>Number of Vertices</b>	1029	1032	1016	1026	1006
<b>Number of Edges</b>	6765	7128	7072	6439	6911
<b>Density</b>	0.013	0.013	0.014	0.012	0.014
<b>Avg. Degree</b>	13	14	14	13	14
<b>Clustering Coefficient</b>	0.55	0.56	0.55	0.56	0.57

Based on a comparison of the number of vertices and edges, the density, average degree and clustering coefficient, it can be observed that the five graphs have similar graph characteristics. Per week, the list of vertices consists of all users that have a profile, i.e. that listened to any artist. The edge list consists of all vertices that have a distance lower than  $\epsilon$ . The clustering coefficient varies at a high level around 0.55 and all graphs show a low density with values around 0.013. The percentage of noise nodes is constantly high and varies between 0.63 and 0.75. The number of detected clusters lies between seven and ten. Due to the labeling of the clusters we observe the following six main clusters in the data set labeled with the genre tags: “indie”, “metal”, “hip-hop”, “j-pop/j-rock”<sup>5</sup>, “industrial” and “rock”.

All five clusterings consist of a giant component and six to nine clusters of different sizes. Therefore, the average cluster size is very high (126 members). The large component can be explained by the user structure of the data set: We observe a large amount of very similar user profiles. These users have a profile vector with a large weight in the genre “indie”, “indie rock” and “alternative”. This cluster is very strong over all intervals and the number of members varies only slightly over time. Besides this cluster, smaller clusters consisting of users with a different listening behavior are detected. The graph visualizations of the five clusterings are shown in Figures 4.24, 4.25, 4.26, 4.27 and 4.28.

In the experiments, all cluster transitions depicted in Figure 4.2 can be observed. We focus on this example to check the validity of the proposed method for detecting evolution. The observed cluster transitions are shown schematically in Figure 4.23.

<sup>5</sup>“j-pop” and “j-rock” are short forms of “japanese pop” and “japanese rock”, respectively.

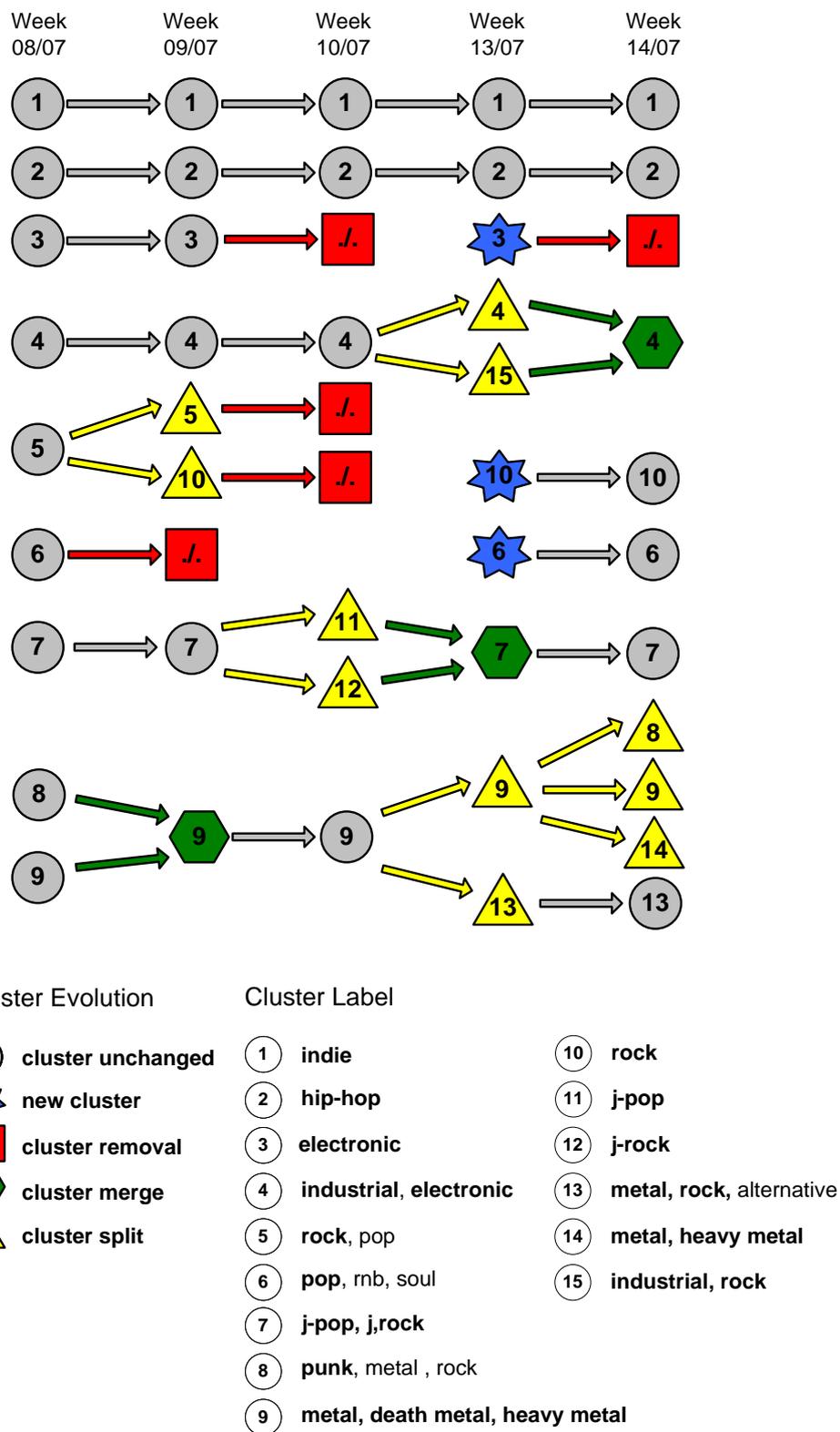


Figure 4.23: Exemplary cluster evolution

**Week 8/2007** In the first week of the chosen period, nine clusters are detected. The labels of these clusters can be seen in Figure 4.24. The biggest cluster is, as discussed before, the “indie” group. In this period, all six main genre clusters are detected: the “indie”, the “hip-hop”, the “metal”, the “industrial”, the “j-pop/j-rock” and the “rock” cluster. Besides, a small cluster is detected which is labeled with tags that are also used for labels of three other clusters (“punk, metal, rock”). With one of these cluster (“indie”) exists an overlap and edges exit also to the other two clusters (“rock, pop” and “metal”). This indicates a closeness of the cluster to three other clusters and, in fact, in the next interval a merge of this cluster with one of the other clusters can be observed.

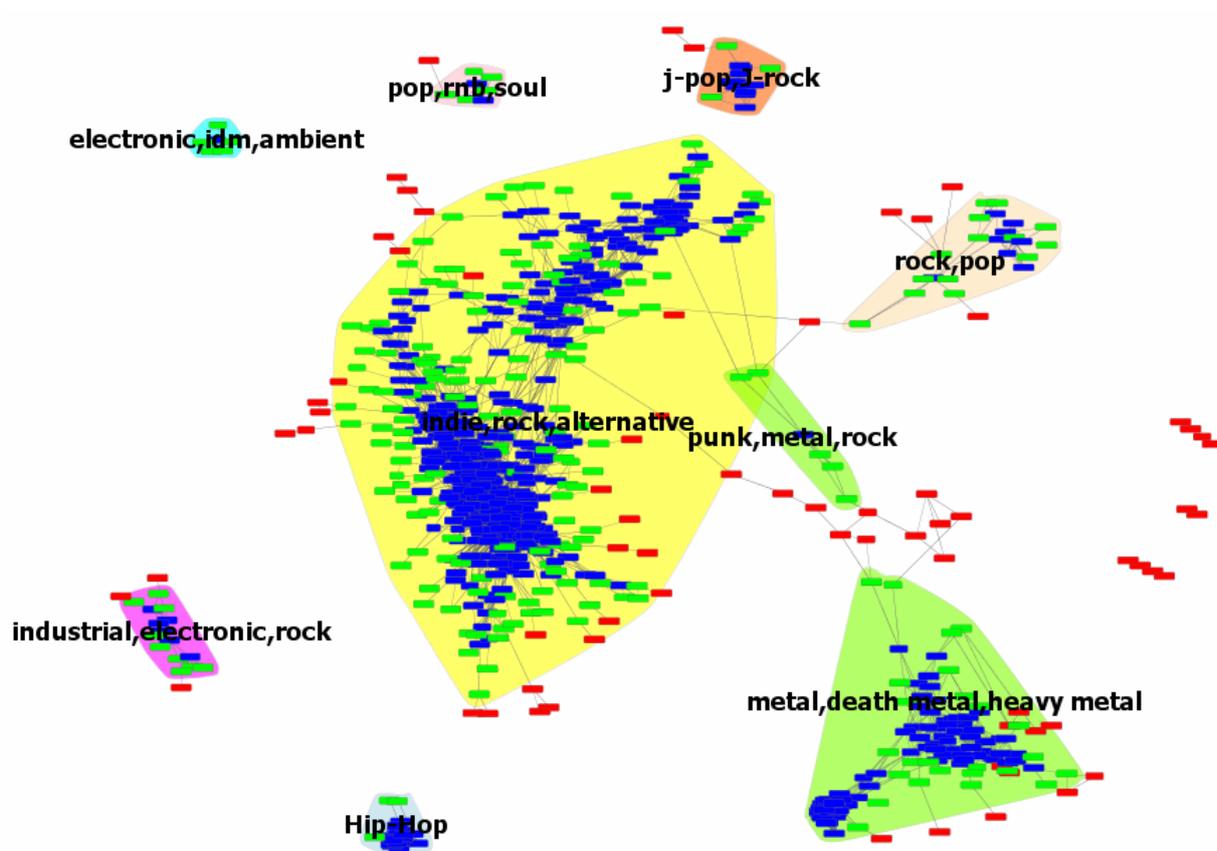


Figure 4.24: Clustered Graph in Week 8/2007.

**Week 9/2007** In week 9/2007, see Figure 4.25, a merge of the “punk, metal, rock” cluster with the larger “metal” cluster is observable. One small cluster (“pop, rnb, soul”) has been removed. Some members of the “rock, pop” cluster split into a “rock” cluster which overlaps with the “indie” cluster. The other five clusters (“indie”, “industrial”, “hip-hop”, “j-pop”, and “electronic”) are unchanged.

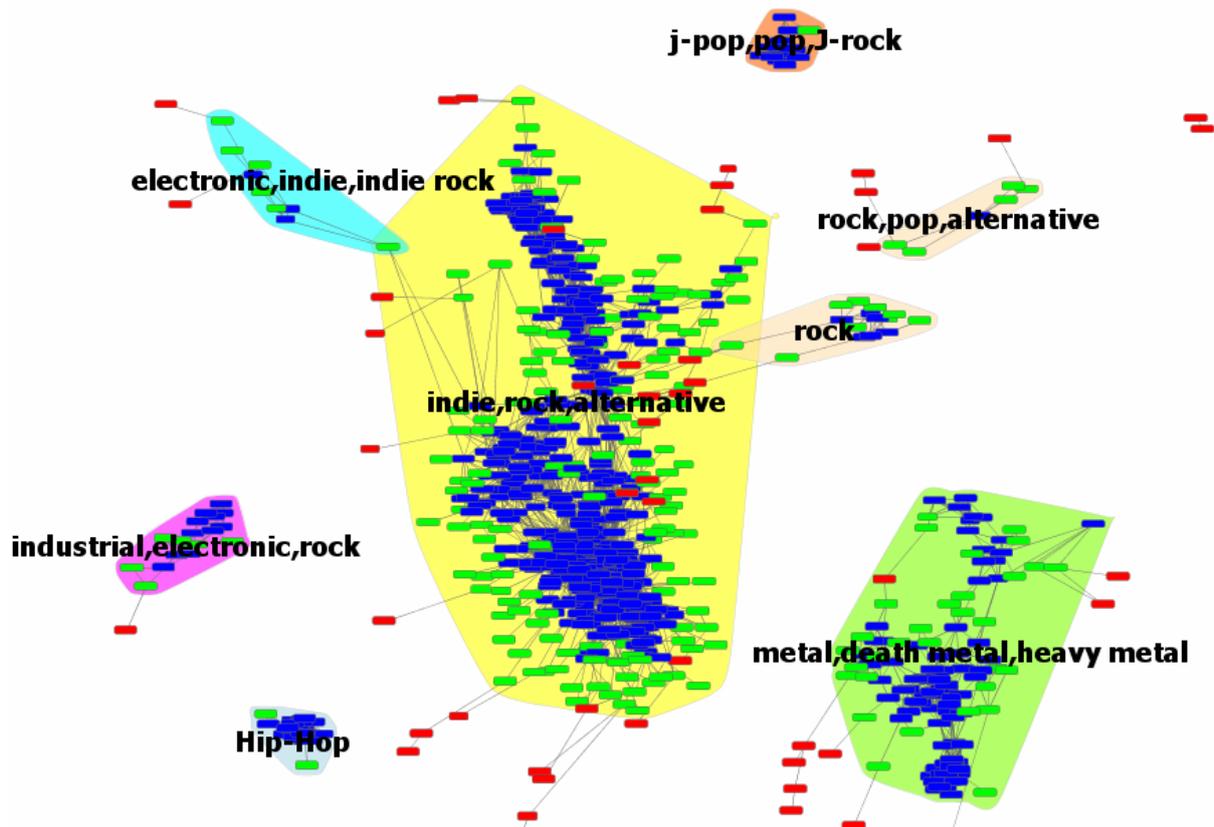


Figure 4.25: Clustered Graph in Week 9/2007.

**Week 10/2007** From week 9/2007 to 10/2007 three cluster removals are observable: Two of the removed clusters will show up again in the next interval (“electronic” and “rock”) and the cluster which was labeled with the tags “rock” and “pop” is not detected in later periods. Furthermore, a split of the “j-pop, j-rock” cluster can be observed. The cluster splits into three smaller clusters: one “j-pop” cluster and two overlapping clusters labeled with “j-rock”. The two clusters will be merged again in the next interval.

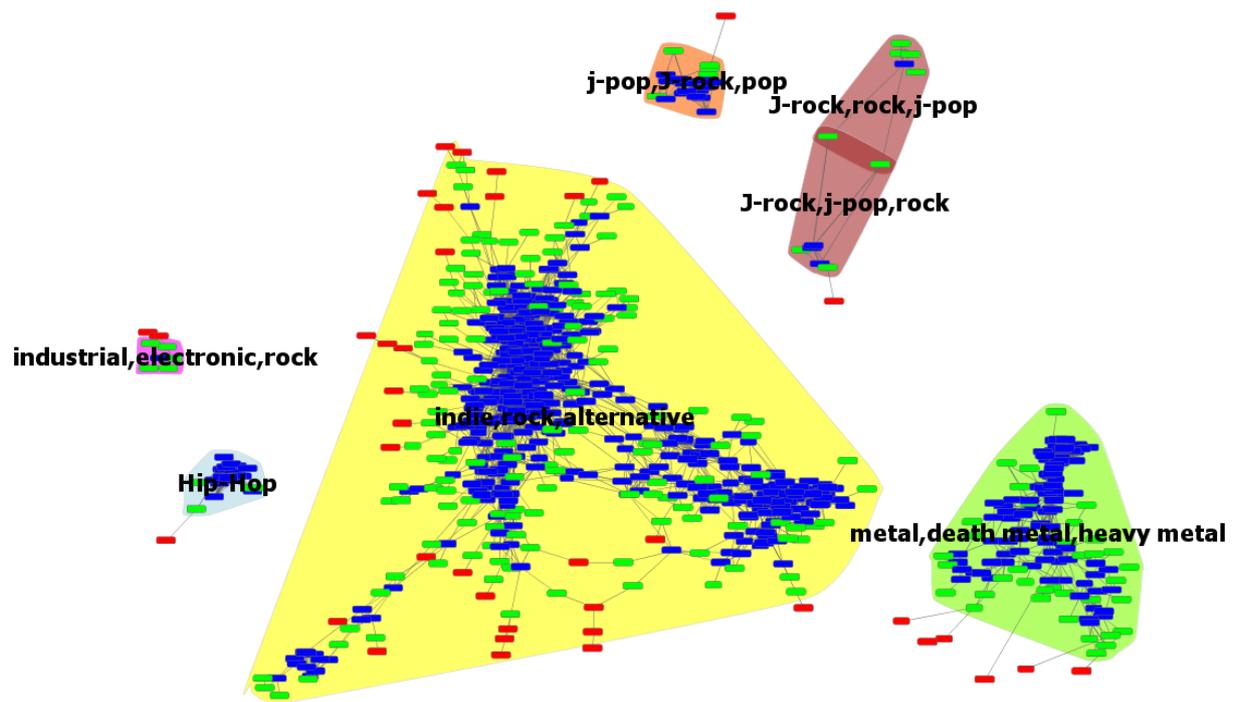


Figure 4.26: Clustered Graph in Week 10/2007.

**Week 13/2007** In week 13/2007 ten clusters are detected. Eight of them have been discovered in earlier periods and two are new. Both two new clusters result from a split. The “industrial, electronic” cluster splits into an “industrial, rock” and an “industrial, pop” cluster. The second split concerns the “metal” cluster: a smaller “metal, rock” cluster is separated from the larger “metal” cluster.

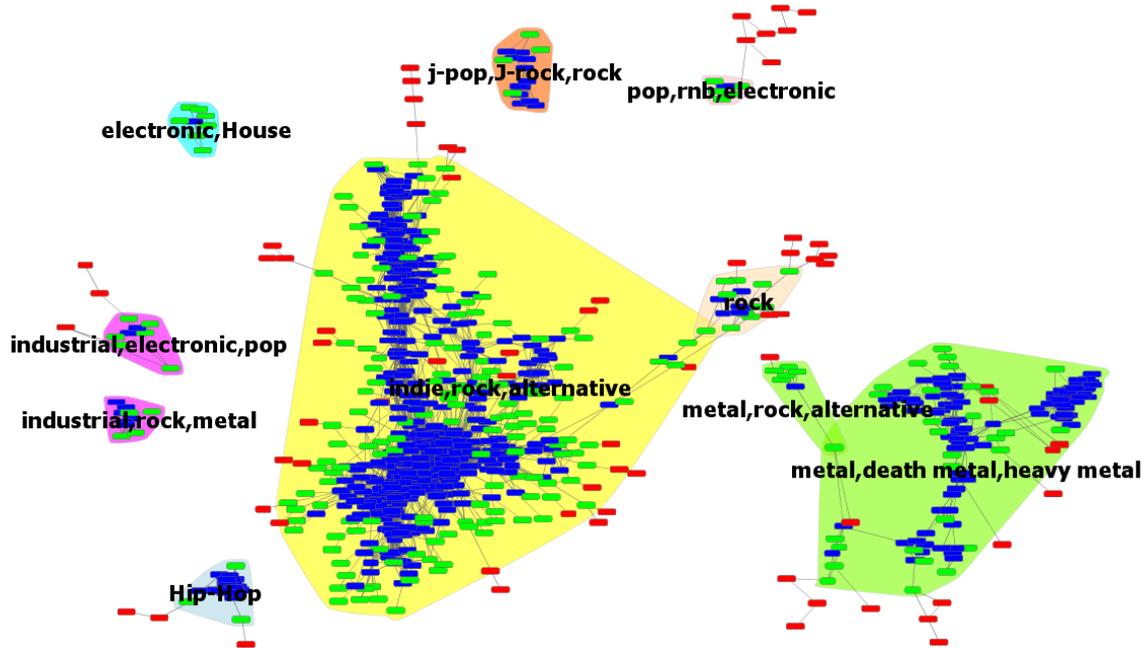


Figure 4.27: Clustered Graph in Week 13/2007.

**Week 14/2007** From week 13/2007 to 14/2007 one split, one merge and a removal take place. The two “industrial” clusters are merged to one cluster and the “electronic” cluster is removed. From the large “metal” cluster two clusters are separated. One cluster is labeled “metal, heavy metal, rock” and the second cluster is labeled with the tags “punk, metal”, a cluster we already observed in week 8/2007 which merged in week 9/2007 with the larger “metal” cluster. Furthermore, the cluster “metal, rock, alternative” has no longer an overlap with the “metal” cluster.

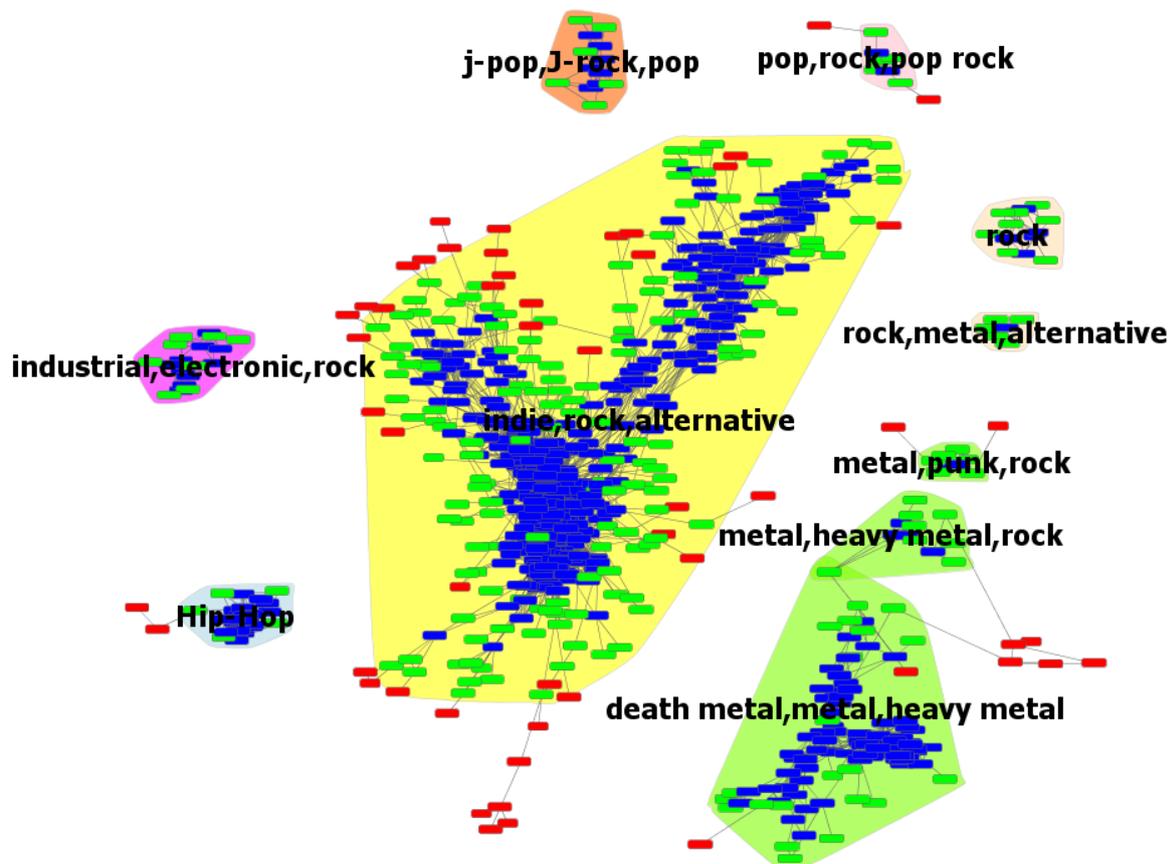


Figure 4.28: Clustered Graph in Week 14/2007.

## Summary

Over a period of 115 weeks, we determined for each week user profiles of 2,000 users that represent their music listening behavior. By building and incrementally clustering a graph of similar users, we obtained groups of people with similar music preferences. We labeled these clusters with genre tags according to the user profiles of the cluster members. Some of these clusters overlap with each other and others are separated over the entire period of observation. Groups that overlap are more similar, because they have tags in common. On the other hand, we detect groups that are always separated, because they have no similarity with others, such as the “hip-hop” cluster.

Furthermore, we could show, that the incremental procedure allows to track and observe temporal dynamics of user clusters. We observe the growth and decline of clusters as well as the creation of new clusters and the removal of formerly existing clusters. Using DENGRAPH-IO we can also observe the evolution of genre clusters: Transitions such as merges and splits are detected and due to the cluster labeling we are able to verify that these transitions are semantically reasonable. In particular we observed that for similar clusters, especially those which overlap, the likelihood of splits and merges is higher.

**Remark** We observed in the analysis that the similarity due to the high-dimensionality of user profiles was very low. This problem could be overcome by using another representation of the user profile. Profiles with too many genres are poorly represented because they have low similarity values (a small scalar product due to a large dimensionality). Thus, distances in a high-dimensional vector space are very low as well. A reduction of the number of genres in the user profiles could mitigate this problem. Levy and Sandler [115] showed that a genre space with a dimensionality around 100 is a good representation and an effective feature space for music retrieval purposes, i.e. for music similarity. Thus, by building, e.g., a genre hierarchy, a reduction of the number of dimensions by merging genres that belong to the same superordinate concept (e.g., merging “classic rock” and “rock”) could be reached.

## 4.6 Characteristics of the DENGRAPH-IO Algorithm

In this subsection, we summarize the DENGRAPH-IO cluster characteristics which we obtained by analyzing the algorithm and observed during the experiments.

### 4.6.1 The Impact of the Parameters on the Clustering

The outcome of the cluster procedure depends on the choice of the parameters  $\epsilon$  and  $\eta$ . Both parameters influence the clustering, however, to different degrees. Ester et al. propose for example to determine the  $\epsilon$  by (i) plotting the sorted distance of each point to the  $k$ -th nearest neighbor and (ii) specifying the threshold point by fixing the percentage of noise [50]. For this approach two values, namely  $k$  and the noise ratio, need to be given to decide on the parameters  $\epsilon$  and  $\eta$ . Ester et al. propose that the user visually inspects the curve and decides whether the proposed parameters are acceptable or not. For  $\eta$  the

authors claim that four is an appropriate value and basically eliminate the parameter by setting it to this value.

Since it is hard to decide on a desirable noise ratio, we propose to base the parameter decision mainly on the quality of the clusterings using the quality measures *cluster performance* and *modularity* as discussed in Section 2.4.2. The noise ratio and the number of clusters are used as additional indicators for the influence of the parameters.

The impact of the parameters on the cluster performance, the modularity, the number of clusters and the noise ratio were discussed for the graphs obtained from the *Enron* data in Section 4.4.3 and for the *Last.fm* data in Section 4.5.3. The results can be summed up by the following observations:

- Overall, it can be observed that the number of clusters increases with increasing  $\epsilon$  and with decreasing  $\eta$ , however, the maximal number of clusters is not reached with the highest  $\epsilon$  and lowest  $\eta$  values, but local maxima are observable.
- Comparing the influence of  $\epsilon$  and  $\eta$ , the quality of the clustering given by the cluster performance and the modularity is more sensitive to changes of  $\epsilon$ .
- In particular for large graphs, the value of  $\eta$  has a lower influence on the cluster quality.
- The noise ratio increases strongly with decreasing  $\epsilon$ . The effect of  $\epsilon$  is stronger for larger graphs. For smaller graphs, the influence of  $\eta$  is bigger than for larger graphs.
- A correlation between the cluster performance measure and the modularity is observable.

### 4.6.2 Characteristics of Core and Border Vertices

For the vertices obtained with DENGRAPH(-IO) as cluster members, the following properties can be observed which result from the clustering procedure:

- All core vertices  $v$  in the  $\epsilon$ -neighborhood of a core vertex  $u$  have the same cluster-id as  $u$ . Otherwise clusters would be merged.
- For all border vertices  $v$  in the  $\epsilon$ -neighborhood of a core vertex  $u$  holds that the list of cluster-ids of  $v$  contains at least the cluster-id of  $u$ .
- Only border vertices can be member of more than one cluster.
- A border vertex  $u$  can only be a member of  $\eta - 1$  different clusters, i.e. the number of distinct cluster-ids of vertices in the  $\epsilon$ -neighborhood of a border vertex  $u$  is smaller than  $\eta - 1$ . Otherwise,  $u$  itself would be a core vertex resulting in a merge of all clusters.

### 4.6.3 Relevant Edge-Changes for the Clustering

Relevant edge-changes are those changes that might influence the clustering of the graph. Insertions of close edges that might change the clustering are insertions between (i) a border and a core vertex if both have different cluster ids or if one vertex becomes core, (ii) two cores if both end-vertices have different cluster-ids (since this would result in a cluster merge) or (iii) noise and border or noise and noise or border and border vertices if at least one vertex becomes core.

Edge removals between (i) border vertices, (ii) noise vertices and (iii) noise and border vertices are in general irrelevant because they do not affect the clustering of the graph. Removals of close edges between (i) cores and border and (ii) two core vertices are relevant as a vertex might lose its core status which can result in a reduction, removal or split of a cluster.

Please note that even though some insertions and deletions have no effect on the clustering in the sense that the cluster-membership of vertices might be changed, the update always affects the graph structure and if a cluster member is involved also on the cluster structure and in this case also on cluster quality measures. Therefore, the graph and the cluster structure are in either case updated accordingly.

### 4.6.4 Impact of Positive and Negative Changes on Running Time

The number of updates equals the number of edge-changes, i.e. it equals the sum of new close edges and lost close edges. For each edge-update, both end-vertices need to be updated. As we can see, for example, in the graph  $Enron_{total}^{week}$  in Figure B.7, the number of weekly updates varies strongly over the observation period. In the chosen example, at the beginning the number of updates ranges between 0 and 500 and in some intervals the number increases to 2,500.

All experiments have shown that the CPU-runtime correlates with the number of updates and that the detection and processing of positive updates are computationally more costly than negative updates. If we look, for example, at the results of the graph  $Enron_{total}^{month}$  shown in Figure B.8: The number of positive and negative updates is depicted in the chart before the last and the CPU-time is shown in the last chart. It can be observed that in interval 51 and 52 the total number of updates is similar but the CPU-runtime differs significantly. In interval 51, most updates are positive updates, i.e. many merges must be handled and in interval 52, most updates are negative updates resulting in many splits (cf. same figure, second diagram). The last curve shows that the CPU-runtime for interval 51, the period with many merges, is much higher than for interval 52 where many splits occur.

Therefore, we observe that CPU-time is dominated in case of positive updates by merges and in case of negative updates by splits, while merges are more expensive than splits. One explanation for this is that merges are in general more costly than splits because at the time a split occurs, clusters are often already rather small whereas a merge also happens between bigger clusters. Thus, before a cluster finally splits, many reductions can be observed (which are less costly) and the cluster shrinks, whereas merges also occur

between rather big clusters. This is likely because, for example, if a border vertex that is member of more than one cluster becomes core vertex due to a new close edge, it leads to a merge of clusters. Whereas, if this border loses an edge, only a reduction occurs, not necessarily a split. Hence, the detection of splits as well as the necessary updates of the data structure is usually less expensive than a merge because less vertices are affected that need to be processed.

The observation that merges are more complex than splits is interesting as this might also be an indication for the characteristics of the network or the type of evolution that the network could undergo in the future. It could be of interest to see, whether a correlation exists for example between the number of splits and merges, the time it takes to detect the event and to perform the necessary updates on the clustering and the clustering coefficient of the graph. A hypothesis could be that clusters in a graph with a high clustering coefficient are more likely to merge and a low clustering coefficient indicates a higher probability for splits. Furthermore, splits in graphs with a low clustering coefficient are most likely not so costly than splits in a highly clustered graph, because rather few vertices are involved in a split.

The corresponding results for all data sets are shown in Figures B.5, B.6, B.7, B.8 and B.9.

### 4.6.5 Cluster Resilience and DENGGRAPH-Stability

In Sections 4.4.3 and 4.5.3 we studied the disconnection probability and the DENGGRAPH-stability of the obtained clusters for the four graphs extracted from the *Enron* data and the *Last.fm* graph, respectively. In summary, it can be observed that the network resilience is relatively low, if the cluster resembles a star, because each lost edge to a border vertex which was, for example, only connected to the cluster via one edge, results in a cluster disconnection. To assess the stability of a cluster, the introduced DENGGRAPH-stability measure indicates how sensitive the clustering procedure is to minor perturbations of the input graph. Our experiments suggest that the stability of DENGGRAPH-clusters depends (i) on the density of the cluster and (ii) on the proportion of core to border vertices. Star-like structures, i.e. clusters with a lower density and a small fraction of core vertices compared to cliquish structures, are more easily disconnected and are only stable regarding DENGGRAPH, if the number of border vertices is considerably higher than  $\eta$ , because then core vertices do not immediately lose their status. Furthermore, clusters which are more dense have in general a higher cluster resilience and also a higher DENGGRAPH-stability compared to clusters which are less dense, because a higher number of edges need to be removed before the cluster is disconnected or before a vertex loses its core status. The proportion of core to border vertices is also an important indicator: The higher the proportion of core vertices the higher the cluster resilience and the DENGGRAPH-stability.

## 4.7 Related Work

**Density-based Community Detection** Recently Xu et al. proposed a framework for community detection based on dense structures [169]. Their approach resembles - as DENGGRAPH - the DBSCAN algorithm. The method reveals clusters as well as noise objects. The method does not provide an incremental update of the clustering and is therefore only applicable for static data sets. Furthermore, the detection of overlapping clusters is not possible.

**Overlapping Communities** Palla et al. view communities as a union of adjacent  $k$ -cliques where adjacency means sharing  $k - 1$  nodes [133]. In their studies of scientific collaboration, word-association and protein interaction graphs they focus on the extent to which  $k$ -cliques are nested and overlap. This approach requires the input of parameter  $k$  is relies on  $k$ -cliques instead of arbitrary dense structures.

Pinney and Westhead [136] propose to extend the edge betweenness algorithm of Girvan and Newman algorithm with the ability to split vertices between clusters. At each iteration, the decision to split a vertex or to remove an edge is based on the values for edge and vertex betweenness. The edge with the highest betweenness is removed only if its two end-vertices have similar betweenness. Otherwise the vertex with highest betweenness is temporarily removed. This decision requires a user input parameter to assess the similarity between pairs of vertices.

Another extension of the edge betweenness algorithm is proposed by Gregory [79]. The author presents CONGA, an algorithm to extract overlapping community structure in networks. He performs a hierarchical clustering based on betweenness centrality and allows the clusters to overlap. The model is based on a defined “split betweenness” which is used to decide whether a vertex is split, what vertex to split and how to split. The procedure has a high complexity of  $\mathcal{O}(m^3)$  and inherits the drawbacks of edge betweenness clustering as discussed in Section 3.6.

Zhang et al. present an algorithm to identify overlapping communities in networks by the combination of a the modularity function, an approximation mapping of network nodes into Euclidean space and fuzzy c-means clustering which assigns a membership degree to every vertex. The procedure requires a user input parameter to indicate an upper bound of the community’s number, which is often hard to determine a-priori in real networks [171].

**Studying Changing Customer Behavior** To identify for example customer segments and to track their change over time could be an interesting application for organizations who need to understand what their customers expect from them now and in the future. This is of particular importance for businesses which operate in dynamic markets and whose customers have highly changing demands. In a recent study in the area of telecommunications, Böttcher et al. propose an approach to detect changing customer segments [29]. Segmentation is typically done by clustering similar customer segments to which future customers are assigned to. For this, the authors discover frequent itemsets and analyze their change over time which results in a changing “interestingness” of segments.

Interesting segments are defined as segments that display some temporal change reflected in the data. The approach is illustrated with two data sets:

- Customer surveys are used to investigate what is likely to drive customer satisfaction in the future
- Network usage data is used to understand the drivers of change in customer behavior when they are using services

The interpretation for an organization could be that growing segments can be seen as opportunities and shrinking segments as threats and in both cases the organization can initiate appropriate actions. This stresses the need for data mining methods that are capable of finding the most relevant and interesting changes in a data set.

## 4.8 Concluding Remarks

We proposed in this chapter a graph clustering procedure DENGGRAPH-IO to detect communities in large noisy data sets and introduced incremental update functions to efficiently handle changes in the cluster structure induced by changes in the graph structure. Furthermore, we designed the procedure to allow for overlapping communities, i.e. actors can be members in several communities. By this, on the one hand, a more realistic clustering is achieved, since users might be members of several clusters, but we also achieve on the other hand that users are not randomly assigned to one cluster if they would actually be a member of two or more clusters. By this we ensure a deterministic behavior of the clustering procedure.

To determine the characteristics of the clusterings and to assess the applicability of the proposed clustering algorithm we applied DENGGRAPH-IO to different types of social networks. For this, we defined for each application a distance function to determine the proximity of actors in the graph. DENGGRAPH-IO was applied on the graphs and based on the observations we determined the characteristics of the obtained clusters.

A semantic evaluation using data which represents music preferences revealed that the detected clusters and the observed cluster transitions are meaningful in the sense that detected merges and splits of clusters are explainable since in both cases clusters with similar labels were involved. Overall the application on the *Last.fm* data suggests that the detected clusters as well as the cluster transitions are semantically reasonable observations.

Furthermore, the theoretical analysis of the computational complexity and the actual CPU-runtime measurements indicate a high efficiency of the algorithm which leads, combined with the observation that the clusters and their transitions are well-founded, to the conclusion that DENGGRAPH-IO is an efficient clustering procedure for large networks which exhibit a high percentage of noise vertices and numerous updates over time.



# 5 Conclusion

In this thesis, two frameworks for the analysis of community dynamics in large, noisy social networks have been developed. Since the concept of a community differs depending on the context, we proposed two definitions for a community over time and introduced techniques to study their temporal dynamics. For both definitions, we proposed a method to capture the dynamics of communities and developed algorithms to trace evolving communities. Furthermore, visualization techniques were introduced to explore the temporal dynamics of communities in social networks. We conducted experiments with real data sets to assess the applicability of the proposed methods. For this purpose, we defined distance measures on these graphs that represent the strength of an edge between pairs of connected actors. The experiments have shown that the algorithms achieve the goals they are designed for.

## 5.1 Main Contributions

The main contributions of this thesis are briefly summarized in the following.

**A Macroscopic View on Community Dynamics** In Chapter 3, we defined a community as a cluster of similar community instances that are observable over time. The communities are detected by moving a time window over the interactions of individuals and by extracting community instances using a hierarchical edge betweenness clustering algorithm for each aggregated graph. By modeling a community as a persistent structure over time, we are able to detect community dynamics even if parts of the membership fluctuate. To further analyze the temporal dynamics we introduced a visualization technique that allows to detect structural changes in the evolution of communities. This approach was applied to a data set from the online student community *IKUS* and in two applications we have shown that significant insights into the evolution of communities could be gained.

In the first application, we detected structural breaks in the evolution of communities with fluctuating members that occurred at points in time when the interaction behavior changed due to “global events”, i.e. in our case, semester breaks. This knowledge can be valuable for an observer to understand the community evolution and to draw conclusions from it (cf. Section 3.3). Furthermore, in the second application, we developed a method that associates a user with the most fitting community, even considering passive community members. The obtained knowledge about the membership of an individual to a community is of importance, for example, for personalized services and recommendation purposes (cf. Section 3.4).

This macroscopic view on a community is valuable if the parallel evolution of communities is of interest, as the visualization technique allows for an exploration of the

temporal evolution of different communities. For the analysis of the internal evolution of communities, we proposed in Chapter 4 a microscopic view.

**A Microscopic View on Community Dynamics** In Chapter 4, we defined a community as a dense substructure that evolves over time. In our definition of a community we have eliminated the restrictive assumption pertinent in earlier literature that an individual belongs to only one community: We detect and trace overlapping clusters, thus capturing simultaneous membership of individuals to multiple communities - a usual phenomenon.

For the detection of this type of community we proposed DENGGRAPH, a density-based graph clustering algorithm and extended the static version to an incremental graph clustering algorithm (DENGGRAPH-IO). Furthermore, we introduced measures to assess the quality and stability of the DENGGRAPH-IO clusters and the results indicate that the obtained clusters are meaningful considering the application and reasonably stable regarding minor changes in the graph structure.

We presented applications of the DENGGRAPH-IO clustering algorithm on two types of real graph data: (i) A graph representing interactions between individuals built from *Enron* data and (ii) a graph representing similarity of users regarding their music preferences constructed of data from the *Last.fm* platform.

As shown in the experiments, an important advantage of DENGGRAPH-IO is its efficiency in detecting and updating clusters in highly dynamic networks. DENGGRAPH-IO is therefore a well suited clustering algorithm to detect and analyze the temporal development of clusters in large, noisy networks that exhibit a high number of changes over time.

## 5.2 Future Work

Encouraged by our results which indicate that the chosen approaches are appropriate to efficiently analyze the dynamics of social network, future work could be for example (i) to study more application-oriented aspects to figure out how the obtained insights into the evolution of communities could be used to assist decision makers or (ii) to study the algorithmic behavior of the algorithms for example in very large networks in more depth. In the following, we briefly discuss examples for future work in the two areas.

**User Recommendations** The experiments with the *Last.fm* showed that we are able to observe and track the evolution of genre clusters over time. Changes in music preferences are often triggered by the environment a person lives in, such as media or friends. Many people appreciate of being introduced to works that they do not know of yet. This is a reason why *recommendation systems* for music that use, e.g., *collaborative filtering* techniques became more and more popular. Collaborative filtering is a procedure that detects the preferences of users for items such as books, DVDs or music. A recommendation system based on collaborative filtering uses, e.g., the extracted preferences of a user to provide recommendations for an item(see, e.g., [146]).

The results of our study with the *Last.fm* data indicate that DENGGRAPH-IO is capable to efficiently extract communities of users with a similar taste in music and that it furthermore allows us to monitor and analyze the changes in the listening behavior of individuals and the resulting shifts in the membership to a certain community. The goal of future work could be to detect patterns of temporal dynamics, for example, to predict changing music preferences or to recommend music genres a user might enjoy in the future.

However, besides music recommendation also any other area where users receive recommendations could be studied.

**Studying Large-Scale Networks** The efficiency of the incremental DENGGRAPH allows to analyze much larger networks than those we have studied yet. It would be of interest to apply our algorithm on large-scale networks to analyze, on the one hand, the clustering behavior of our algorithm on very large data sets and, on the other hand, to see whether for example massive graphs show global phenomena that are genuine but invisible in graphs with a smaller scale. However, when studying very large graph data the meaning of a single vertex or edge becomes hard to ascertain. It might be rather easy to calculate structural or statistical measures, but it becomes hard to interpret the data. This brings us to our next option for future work: Interdisciplinary research.

**Interdisciplinary Research** The analysis of the temporal evolution of (online) communities or customer behavior in general gives raise to a huge amount of interesting research issues. Most of them require the cooperation between rather diverse research areas such as computer science, sociology, psychology and business administration. In future work, the aim could be to achieve synergies between the different fields to obtain even more valuable results. Examples for this could be to incorporate knowledge from the field of linguistics to consider also the content or nature of the observed interactions or to involve specialists from psychology or sociology to improve the interpretation of the results and the implications for decision makers.

**Determining Parameters** An aspect that concerns further studies of the algorithmic behavior is the determination of the required parameters. So far, we have assumed that the parameters needed for the DENGGRAPH-IO clustering algorithm do not change over time and fixed them over the observation period. This assumption might not be realistic for certain applications as the data set characteristic might change over time demanding for different parameter settings to obtain meaningful clusters. It could be therefore of interest to study this aspect more closely in future work.

**Privacy Protection** An observation, which does not only concern our work on community analysis, but also the social network analysis community in general, is related to privacy aspects: So far it is not well understood how privacy concerns influence interactions within social networking sites. A study conducted by EDUCAUSE, a non-profit organization whose mission is to promote information technology to advance higher education, revealed that users of social networking sites are not overly concerned about

privacy issues. Only about fifty percent are at least “moderately” concerned. Females are more concerned than males and older students are more concerned than younger students [144]. Even more interesting, results reported in [47] indicate that online relationships can also develop on sites where the perceived privacy safeguards are rather weak. While any post on the Web is essentially “public”, any automated collection and combination of network information raises privacy issues. Therefore, users should realize that the data they provide is a public indicator of social interaction. Kleinberg et al. [9] demonstrated how easy it is to obtain private information in a social network by creating a subgraph that is then linked via relations to the rest of the graph. An outcome of their work is that the knowledge about the created edges and nodes can result in a privacy breach. They showed that the inserted subgraph is easily findable, even in anonymized data, because it is likely to be unique and efficiently traceable. By this the privacy of the neighbors of the graphs, the “friends of a friend”, can be disclosed.

It is just in the beginning that users as well as providers discuss privacy issues in the context of social networks more intensively and first proposals have been made that specifically aim at protecting privacy in social networks [84, 172]. For future social network research relying on real data, it is of great importance that users have reason to remain confident regarding the measures taken to protect their interests.

# A Pseudocode

---

**Algorithm A.1:** DENGRAPH-IO: *DensityConnectedNodes*(u,v) Function

---

```
1 Algorithm:DensityConnectedNodes(u,v)
  input : Nodes u and v
  output: Set of Nodes density connected from u when searching for v
2 begin
3   NodeSet = {u}
4   NodeQueue.push(u)
5   repeat
6     r=NodeQueue.pop()
7     foreach  $n \in N_\epsilon(r)$  | ( $n \notin \text{NodeSet}$ ) and ( $n.state \neq \text{noise}$ ) do
8       if  $n \neq v$  then
9         NodeSet=NodeSet  $\cup$  n
10        if  $n.state = \text{core}$  then
11          NodeQueue.push(n)
12        end
13      end
14      else
15        NodeSet=NodeSet  $\cup$  v
16        Break
17      end
18    end
19  until (Queue is empty )
20 end
21 return NodeSet
```

---

---

**Algorithm A.2:** DENGRAPH-IO: *DensityConnectedNodes*(u) Function

---

```
1 Algorithm:DensityConnectedNodes(u,newCIID,oldCIID)
  input : Node u, newCIID and oldCIID
  output: Updated Cluster IDs of all Nodes density connected from u
2 begin
3   NodeSet = {u}
4   NodeQueue.push(u)
5   repeat
6     r=NodeQueue.pop()
7     foreach  $n \in N_\epsilon(r)$  | ( $n \notin \text{NodeSet}$ ) and ( $n.state \neq \text{noise}$ ) do
8       NodeSet=NodeSet  $\cup$  n
9       n.CIID=n.CIID  $\cup$  newCIID  $\setminus$  oldCIID
10      if  $n.state = \text{core}$  then NodeQueue.push(n)
11    end
12  until (Queue is empty )
13 end
14 return NodeSet
```

---

**Algorithm A.3:** DENGGRAPH: Non-incremental Clustering

---

```

1 Algorithm:DENGGRAPH(Graph)
  input : Graph, ParentClusterID
  output: ClusterModel
2 begin
3   foreach  $r \in V$  do
4      $r.state = noise$ 
5      $r.CIID = \{\}$ 
6   end
7   foreach ( $u \in V | u.state = noise$ ) do
8     Compute the set  $N_\epsilon(u)$  of all nodes in the  $\epsilon$ -neighborhood of  $u$ 
9     if ( $|N_\epsilon(u)| \geq \eta$ ) then
10       $u.CIID = CreateNewCluster()$ 
11       $u.state = core$ 
12      foreach  $n \in N_\epsilon(u)$  do
13         $n.CIID = n.CIID \cup u.CIID$ 
14         $n.state = border$ 
15        if  $n.stack = false$  then
16           $stack.push(n)$ 
17           $n.stack = true$ 
18        end
19      end
20      //Stack for all nodes that are density reachable from u
21      repeat
22         $v = stack.pop()$ 
23        if ( $|N_\epsilon(v)| \geq \eta$ ) then
24           $v.state = core$ 
25          foreach  $n \in N_\epsilon(v) | n.state \neq core$  do
26             $n.CIID = n.CIID \cup v.CIID$ 
27             $n.state = border$ 
28            if ( $n.stack = false$ ) then
29               $stack.push(n)$ 
30               $n.stack = true$ 
31            end
32          end
33        end
34      until ( $stack$  is empty)
35    end
36  end
37  return ClusterModel
38 end

```

---

---

**Algorithm A.4:** DENGRAPH-IO: Incremental Clustering

---

1 **Algorithm:**DENGRAPH-IO(ClusterModel, LastGraph, NewGraph)

input : ClusterModel, LastGraph, NewGraph

output: ClusterModel

2 **begin**

3 CurrentGraph:=LastGraph

4 Create List of Changed Edges

5 **foreach** *Changed Edge*  $E(t, u, v)$  **do**

6 Update CurrentGraph( $E(t, u, v)$ )

7 **if** ( $(E(t-1, u, v)$  is undefined or  $w(E(t-1, u, v)) > \epsilon$ ) and  $w(E(t, u, v)) \leq \epsilon$ ) **then**

8 UpdateClustering(u)

9 UpdateClustering(v)

10 **end**

11 **else if** ( $(E(t, u, v)$  is undefined or  $w(E(t-1, u, v)) \leq \epsilon$ ) and  $w(E(t, u, v)) > \epsilon$ ) **then**

12 **if** ( $|u.CIID \cap v.CIID| > 0$ ) **then**

13 **if** ( $(u.state=border)$  and  $(v.state=core)$ ) or  $(u.state=core)$  and  $(v.state=border)$ ) **then**

14 UpdateClustering(u)

15 UpdateClustering(v)

16 **end**

17 **else if**  $(u.state=core)$  and  $(v.state=core)$  **then**

18 //Check for split

19 DCNodesu=DensityConnectedNodes(u,v)

20 DCNodesv=DensityConnectedNodes(v,u)

21 **if**  $v \in DCNodesu$  **then**

22 UpdateClustering(u)

23 UpdateClustering(v)

24 **end**

25 **else**

26 //Split

27  $v\_oldCIID=v.CIID$

28  $u.CIID=CreateNewCluster(u.CIID)$

29  $v.CIID=CreateNewCluster(v.CIID)$

30 **foreach**  $r \in DCNodesu$  **do**  $r.CIID=(r.CIID \cup u.CIID) \setminus v\_oldCIID$

31 **foreach**  $r \in DCNodesv$  **do**  $r.CIID=(r.CIID \cup v.CIID)$

32 UpdateClustering(u)

33 UpdateClustering(v)

34 **end**

35 **end**

36 **end**

37 **end**

38 **end**

39 **return** ClusterModel

40 **end**

---

**Algorithm A.5:** DENGRAPH-IO: *UpdateClustering(u)* Function

---

```

1 Algorithm:Update Clustering(u)
   input : Node u
2 begin
3   if ( $|N_\epsilon(u)| \geq \eta$ ) then
4     foreach  $n \in N_\epsilon(u) | n.state = core$  do SetOfDistinctClusterIDs.insert(n.CIID)
5     if  $u.state = core$  then
6       SetOfDistinctClusterIDs.insert(u.CIID)
7     end
8     if ( $|SetOfDistinctClusterIDs| = 0$ ) then
9       //Creation of a new cluster
10      u.CIID=CreateNewClusterIDs(null)
11      foreach  $n \in N_\epsilon(u)$  do
12        n.state=border
13        n.CIID=n.CIID  $\cup$  u.CIID
14      end
15    end
16    if ( $(|SetOfDistinctClusterIDs| = 1) \text{ and } (u.state \neq core)$ ) then
17      //Absorption of a node to a cluster
18      u.CIID=SetOfDistinctClusterIDs
19      foreach  $n \in N_\epsilon(u) | n.state \neq core$  do
20        n.state=border
21        n.CIID=n.CIID  $\cup$  u.CIID
22      end
23    end
24    if ( $|SetOfDistinctClusterIDs| > 1$ ) then
25      //Merge of clusters
26      u.CIID=CreateNewCluster(SetOfDistinctClusterIDs)
27      foreach  $r \in V | (r.CIID \cap SetOfDistinctClusterIDs \neq \emptyset)$  do
28        r.CIID=(r.CIID  $\cup$  u.CIID)  $\setminus$  SetOfDistinctClusterIDs
29      end
30      foreach  $n \in N_\epsilon(u) | n.state \neq core$  do
31        n.state=border
32        n.CIID=(n.CIID  $\cup$  u.CIID)
33      end
34    end
35    u.state=core
36  end
37  else
38    //u has no epsilon neighborhood
39    if ( $u.state = core$ ) then
40      u.state = border
41      //Check for extended split
42      foreach  $n \in N_\epsilon(u) | n.state = core$  do
43        if  $n \notin E$  then
44          newCIID = CreateNewClusterID(u.CIID)
45           $E = E \cup DensityConnectedNodes(n, newCIID, u.CIID)$ 
46        end
47      end
48      foreach  $n \in N_\epsilon(u) | ((n.state = border) \text{ and } (\nexists m \in N_\epsilon(n) | m.state = core \text{ and } m.CIID = u.CIID))$  do
49        n.CIID=n.CIID  $\setminus$  u.CIID
50        if ( $|n.CIID| = 0$ ) then n.state=noise
51      end
52    foreach  $n \in N_\epsilon(u) | n.state = core$  do SetOfDistinctClusterIDs.insert(n.CIID)
53    if ( $|SetOfDistinctClusterIDs| = 0$ ) then
54      u.CIID={}
55      u.state=noise
56    end
57    else
58      u.CIID=SetOfDistinctClusterIDs
59      u.state=border
60    end
61  end
62 end

```

---

## B Experimental Results

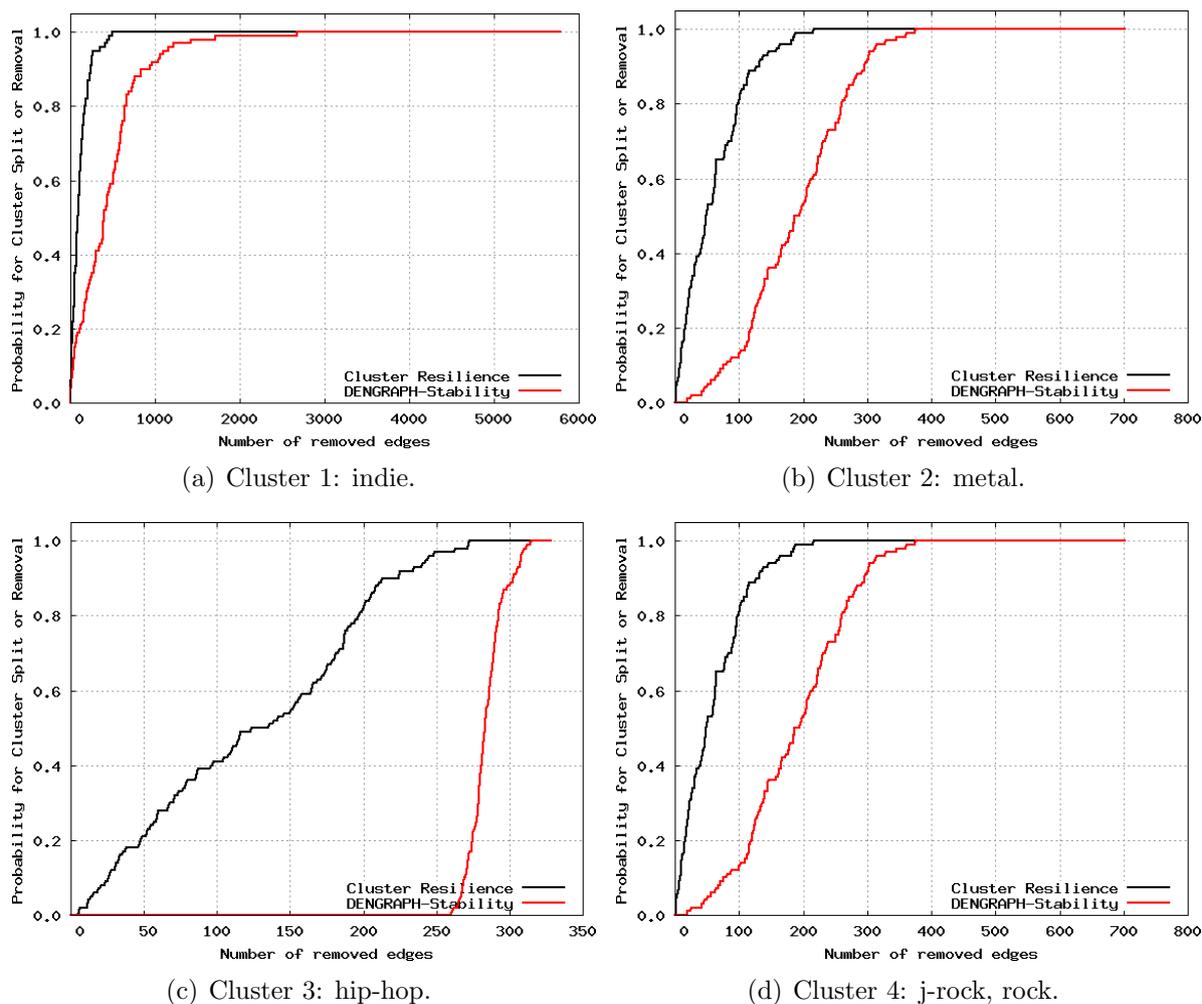
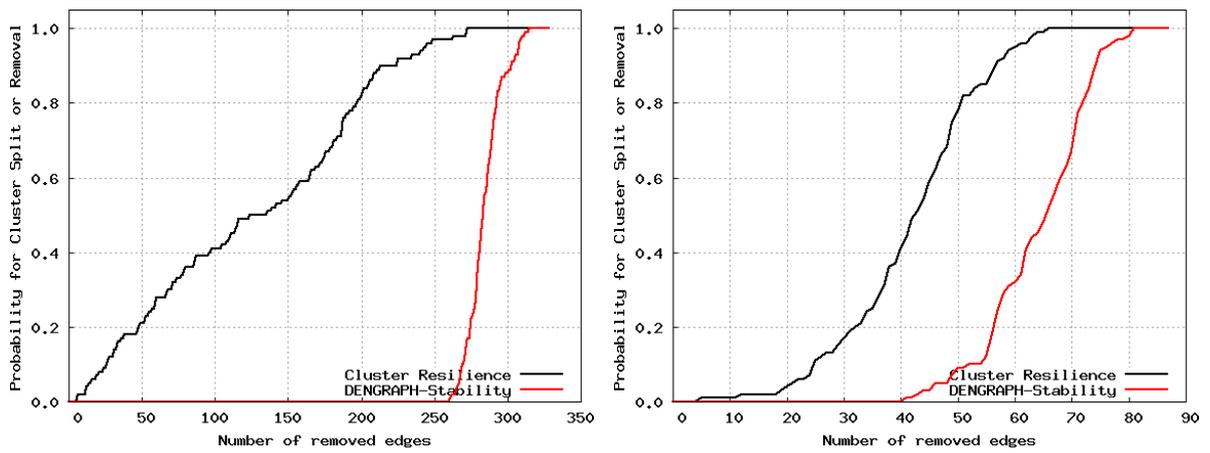
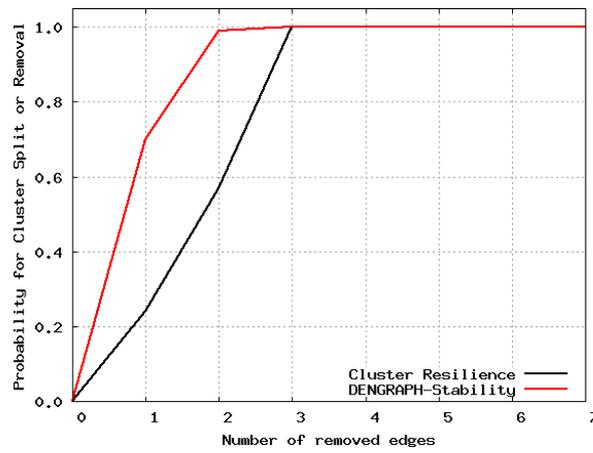


Figure B.1: The cluster resilience (black line) and DENGGRAPH-stability (red line) of clusters 1-4 taken from the clustering of the *Last.fm* graph with  $\epsilon = 0.04$ ,  $\eta = 5$  captured in week 10/2007 and clustered with DENGGRAPH-IO.



(a) Cluster 5: j-rock, j-pop.

(b) Cluster 6: j-pop.



(c) Cluster 7: industrial.

Figure B.2: The cluster resilience (black line) and DENGGRAPH-stability (red line) of clusters 5-7 taken from the clustering of the *Last.fm* graph with  $\epsilon = 0.04$ ,  $\eta = 5$  captured in week 10/2007 and clustered with DENGGRAPH-IO.

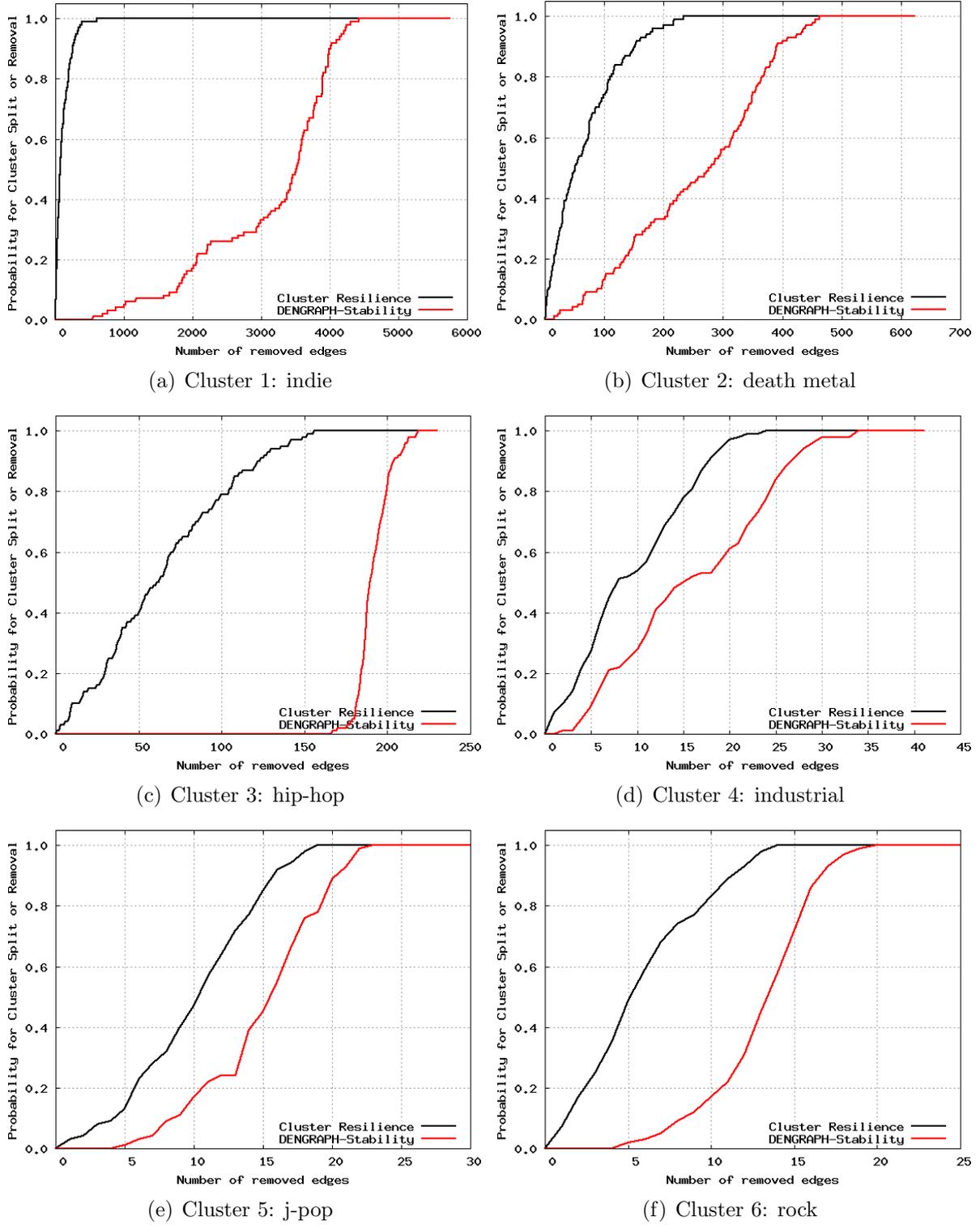


Figure B.3: The cluster resilience (black line) and DENGGRAPH-stability (red line) of clusters 1-6 taken from the clustering of the *Last.fm* graph with  $\epsilon = 0.04$ ,  $\eta = 5$  captured in week 14/2007 and clustered with DENGGRAPH-IO.

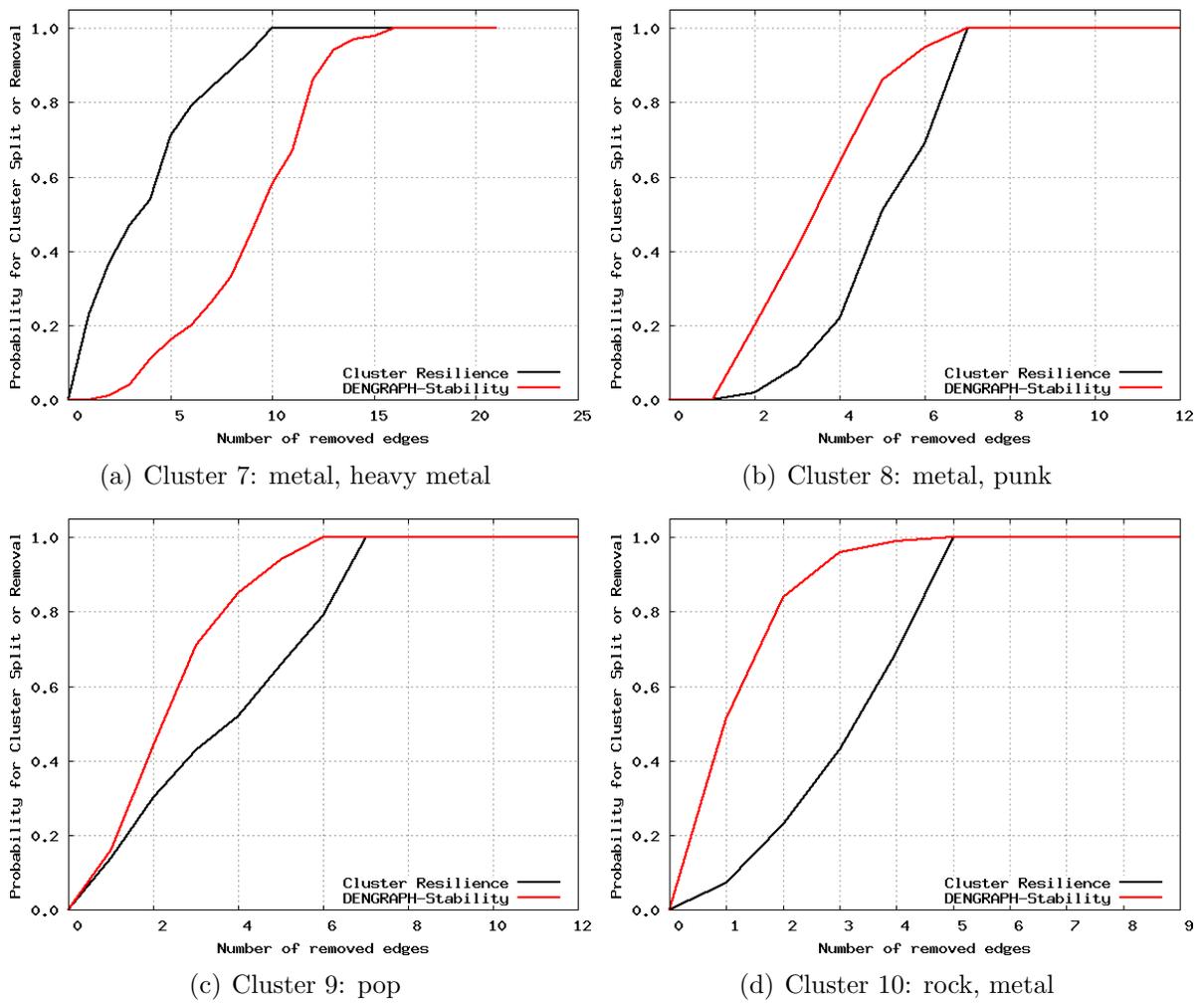


Figure B.4: The cluster resilience (black line) and DENGGRAPH-stability (red line) of clusters 7-10 taken from the clustering of the *Last.fm* graph with  $\epsilon = 0.04$ ,  $\eta = 5$  captured in week 14/2007 and clustered with DENGGRAPH-IO.

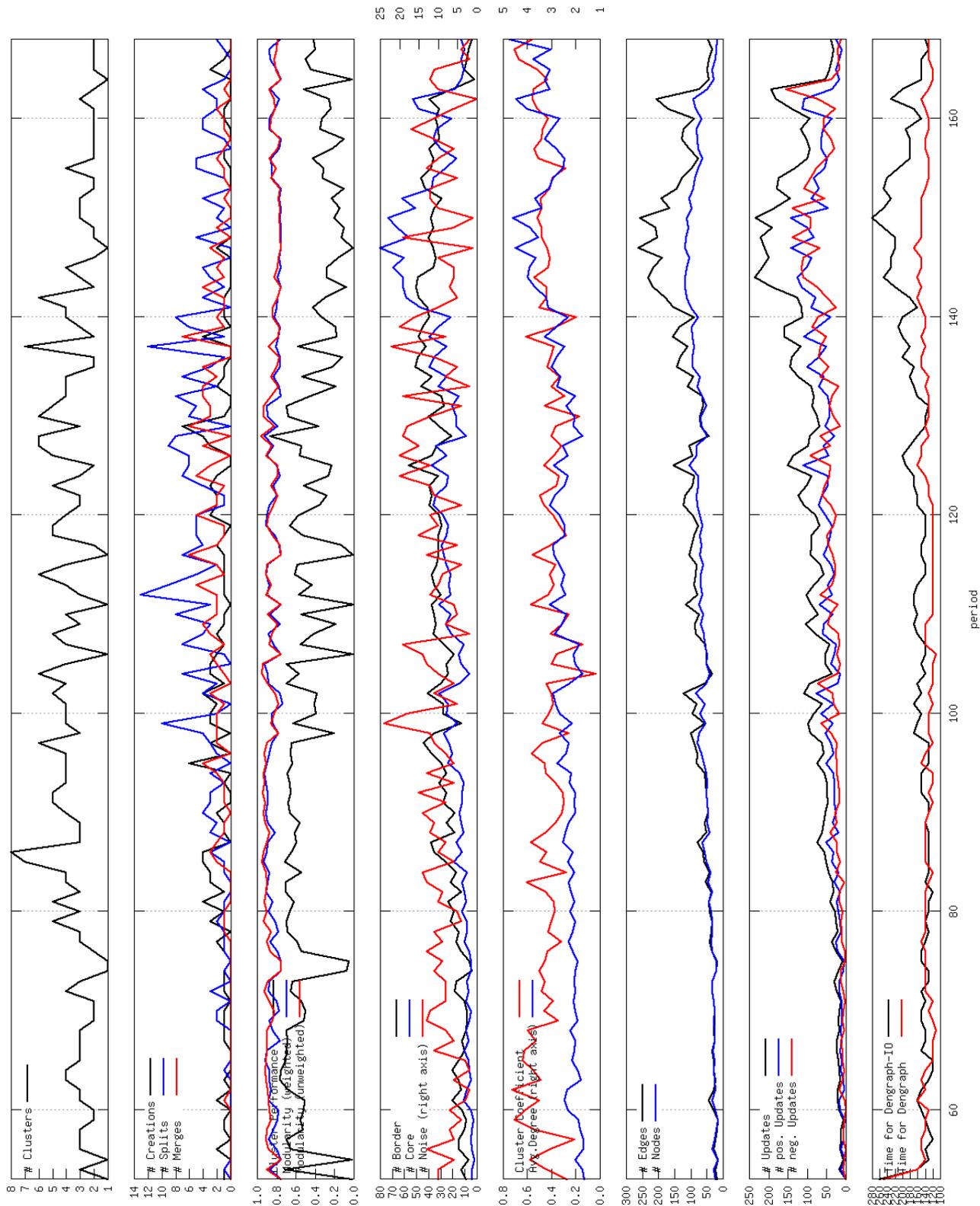


Figure B.5:  $Enron_{intern}^{week}$ ;  $\epsilon = 1$ ,  $\eta = 3$ .

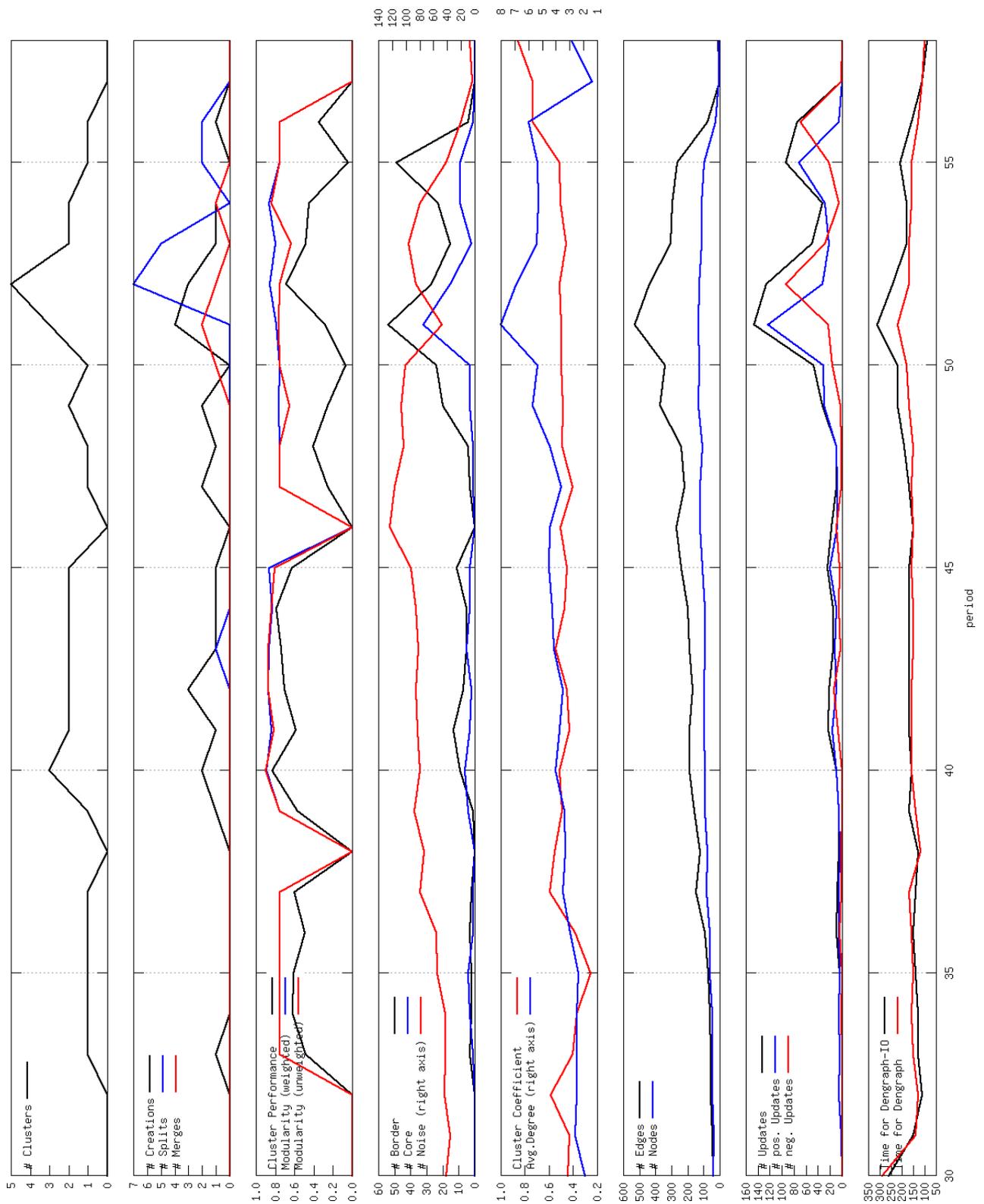


Figure B.6:  $Enron_{intern}^{month}$ ;  $\epsilon = 0.6$ ,  $\eta = 3$ .

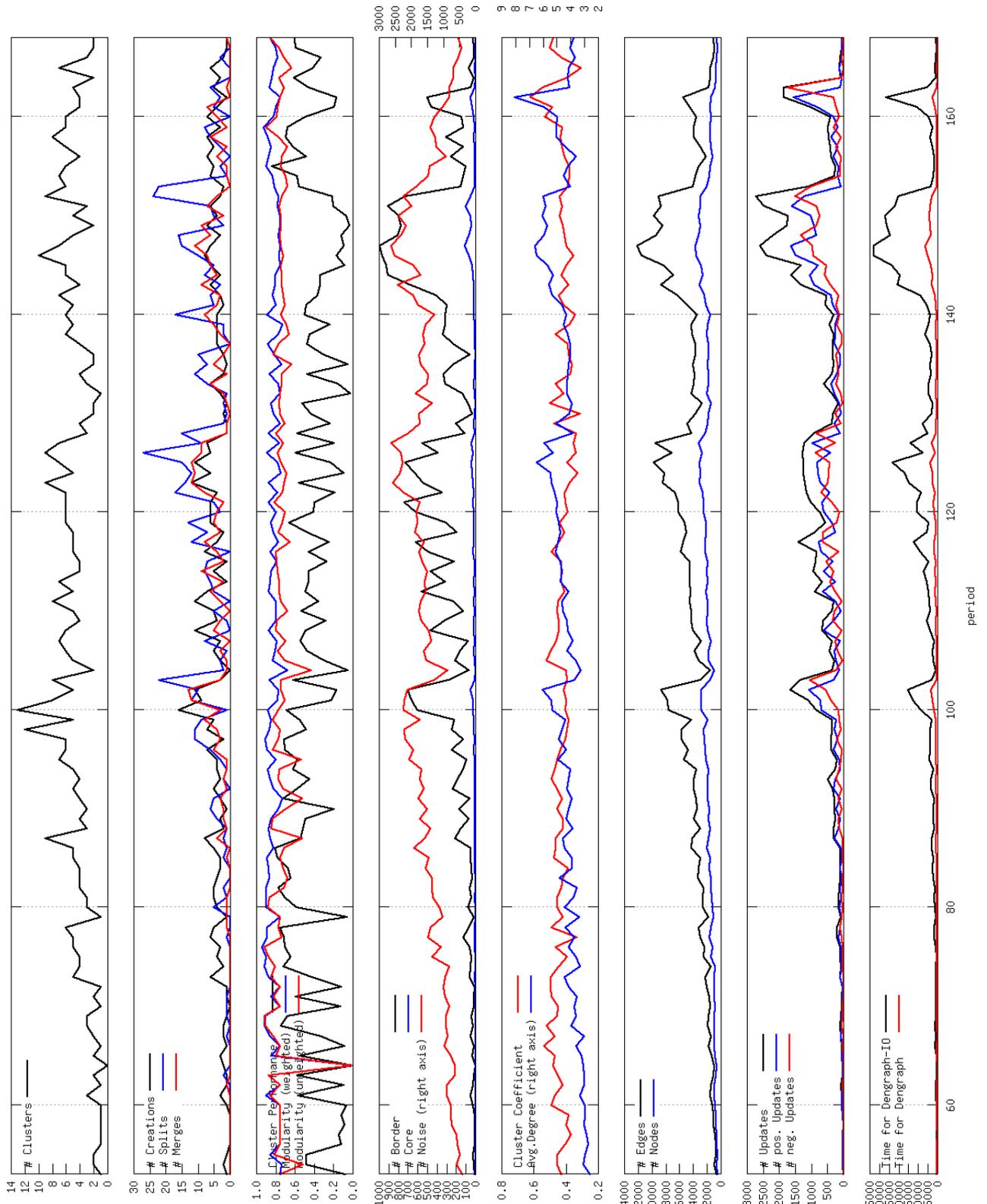


Figure B.7:  $Enron_{total}^{week}$ ;  $\epsilon = 0.6$ ,  $\eta = 7$ .

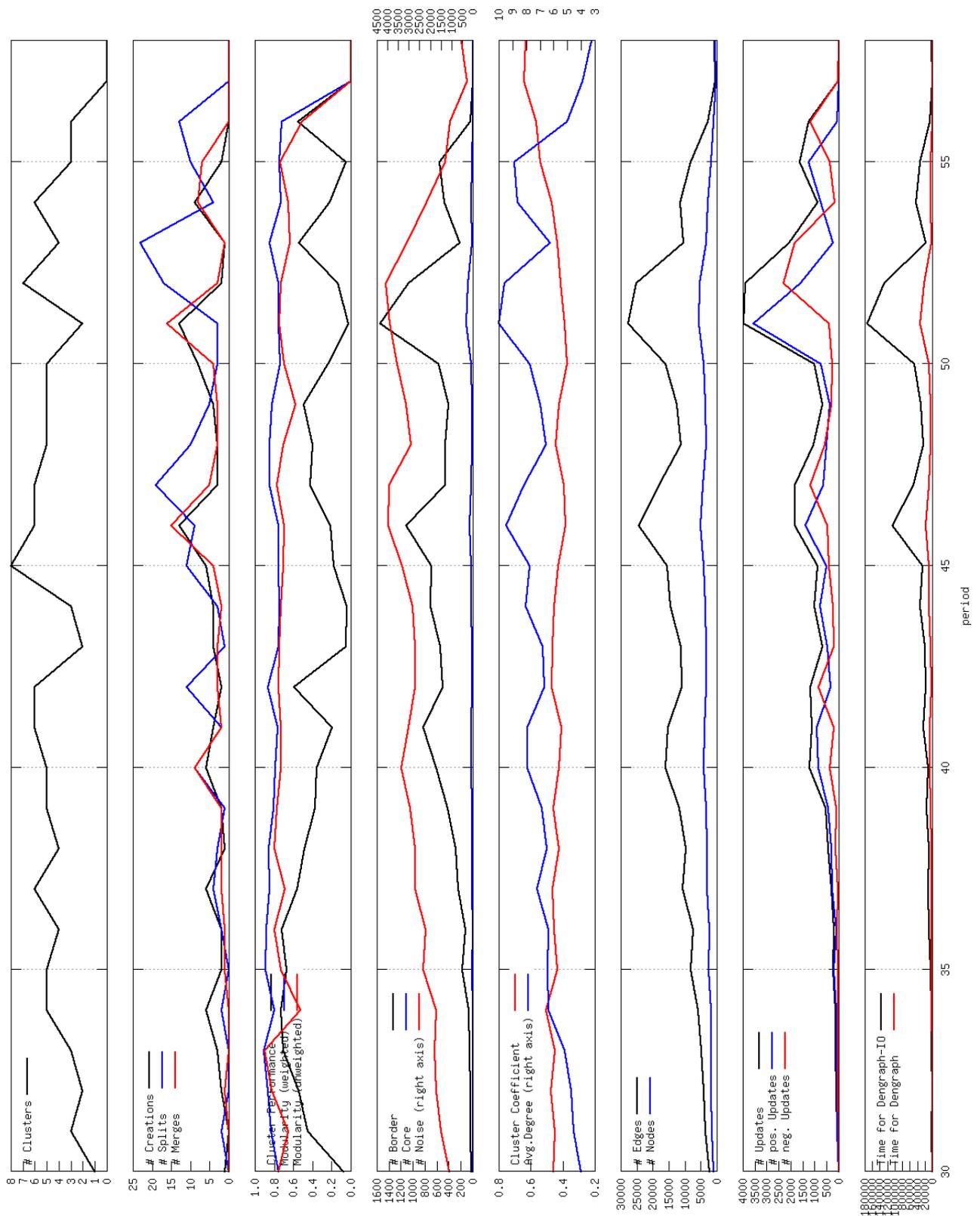


Figure B.8:  $Enron_{total}^{month}$ ;  $\epsilon = 0.6$ ,  $\eta = 10$ .

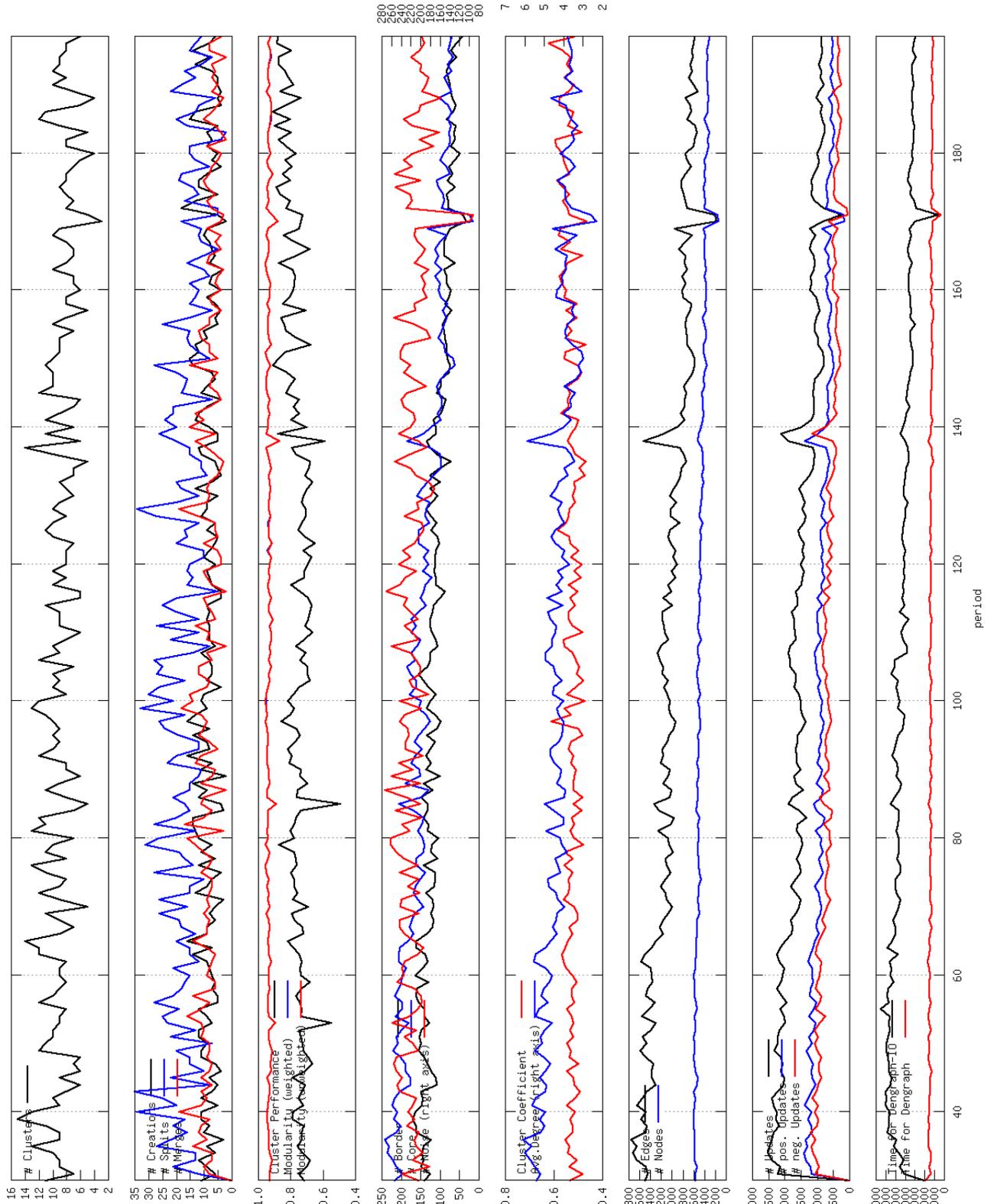


Figure B.9: Last.fm data set with  $\epsilon = 0.02$  and  $\eta = 5$ .



# Bibliography

- [1] ADAMIC, L. A. The small world web. In *Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries* (1999), Springer, pp. 443–452.
- [2] ADAMIC, L. A., AND ADAR, E. Friends and neighbors on the web. *Social Networks* 25, 3 (2003), 211–230.
- [3] AGGARWAL, C. C., AND YU, P. S. Online analysis of community evolution in data streams. In *Proceedings of SIAM International Data Mining Conference* (2005).
- [4] ALBA, R. D. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology* 3 (1973), 113–126.
- [5] AMABILE, T. M. *Creativity in Context*. Westview Press, 1996.
- [6] ANDREWS, D., NONNECKE, B., AND PREECE, J. Electronic survey methodology: A case study in reaching hard to involve internet users. *International Journal of Human-Computer Interaction* 16, 2 (2003), 185–210.
- [7] ANKERST, M., BREUNIG, M. M., KRIEGEL, H.-P., AND SANDER, J. Optics: Ordering points to identify the clustering structure. *ACM SIGMOD Record* 28, 2 (1999), 49–60.
- [8] AUCOUTURIER, J.-J., AND PACHET, F. Representing musical genre: A state of the art. *Journal of New Music Research* 32 (2003), 83–93.
- [9] BACKSTROM, L., DWORK, C., AND KLEINBERG, J. M. Wherefore art thou r3579x?: Anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th International Conference on World Wide Web (WWW'07)* (2007), pp. 181–190.
- [10] BACKSTROM, L., HUTTENLOCHER, D., KLEINBERG, J., AND LAN, X. Group formation in large social networks: Membership, growth, and evolution. In *Proc. of KDD'06* (2006).
- [11] BARABÁSI, A.-L., AND ALBERT, R. Emergence of scaling in random networks. *Science* 286 (1999), 509–512.
- [12] BASILI, R., SERAFINI, A., AND STELLATO, A. Classification of musical genre: a machine learning approach. In *ISMIR* (2004).

- [13] BATTISTA, G. D., EADES, P., TAMASSIA, R., AND TOLLIS, I. G. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1998.
- [14] BELL, G. G. Clusters, networks, and firm innovativeness. *Strategic Management Journal* 26, 3 (2005), 287–295.
- [15] BERGER-WOLF, T. Y., AND SAIA, J. A framework for analysis of dynamic social networks. In *Proceedings of the 12 th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2006), pp. 523–528.
- [16] BERRY, M. J. A., AND LINOFF, G. S. *Data Mining Techniques. For Marketing, Sales, and Customer Support*. Wiley, 1997.
- [17] BORGATTI, S., EVERETT, M., AND SHIREY, P. Ls sets, lambda sets and other cohesive subsets. *Social Networks* 12 (1990), 337–357.
- [18] BORGATTI, S. P., AND FOSTER, P. C. The network paradigm in organizational research: A review and typology. *Journal of Management* 29, 6 (December 2003), 991–1013.
- [19] BORGELT, C., BERTHOLD, M. R., AND PATTERSON, D. E. Molecular fragment mining for drug discovery. In *ECSQARU* (2005), L. Godo, Ed., vol. 3571 of *Lecture Notes in Computer Science*, Springer, pp. 1002–1013.
- [20] BOYD, D. M., AND ELLISON, N. B. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication* 13, 1 (2007). <http://jcmc.indiana.edu/vol13/issue1/boyd.ellison.html>.
- [21] BOZEMAN, A. B. *Politics and Culture in International History: From the Ancient Near East to the Opening of the Modern Age*, 2 sub ed. Transaction Publishers, 1994.
- [22] BRANDES, U. Drawing on physical analogies. In *Drawing Graphs: Methods and Models - LNCS 2025*, M. Kaufmann and D. Wagner, Eds. Springer, 2001, pp. 71–86.
- [23] BRANDES, U., DELLING, D., GAERTLER, M., GOERKE, R., HOEFER, M., NIKOLOSKI, Z., AND WAGNER, D. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering* 20, 2 (2008), 172–188.
- [24] BRANDES, U., DELLING, D., GAERTLER, M., GÖRKE, R., HOEFER, M., NIKOLOSKI, Z., AND WAGNER, D. On finding graph clusterings with maximum modularity. In *WG 2007, LNCS 4769* (2007), A. Brandstädt, D. Kratsch, and H. Müller, Eds., Springer, pp. 121–132.
- [25] BRANDES, U., AND ERLEBACH, T. *Network Analysis: Methodological Foundations*. Springer, 2005.

- [26] BRANDES, U., AND WAGNER, D. visone-analysis and visualization of social networks. In *Graph Drawing Software*, M. Jünger and P. Mutzel, Eds. Springer-Verlag, 2004, pp. 321–340.
- [27] BRIN, S., AND PAGE, L. The anatomy of a large-scale hypertextual web search engine. In *Proc. of 7th International World Wide Web Conference* (1998).
- [28] BRINKMEIER, M., AND SCHANK, T. Network statistics. In *Network Analysis*, U. Brandes and T. Erlebach, Eds. Springer, 2005, pp. 293–317.
- [29] BÖTTCHER, M., SPOTT, M., NAUCK, D., AND KRUSE, R. Mining changing customer segments in dynamic markets. *Expert Systems with Applications* 36, 1 (2009), 155–164.
- [30] BURT, R. S. Models of network structure. *Annual Review of Sociology* 6 (1980), 79–141.
- [31] BURT, R. S., AND MINOR, M. J., Eds. *Applied Network Analysis - A Methodological Introduction*. Sage Publications, 1983.
- [32] BUTLER, B. S. Membership size, communication activity, and sustainability: A resource-based model of online social structures. *ISR - Information Systems Research* 12, 4 (2001), 346 – 362.
- [33] CHAKRABARTI, D., KUMAR, R., AND TOMKINS, A. Evolutionary clustering. In *The Twelfth Annual SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (2006), pp. 554–560.
- [34] CHAVIS, D. M., AND WANDERSMAN, A. E. Sense of community in the urban environment: A catalyst for participation and community development. *American Journal of Community Psychology* 18, 1 (February 1990), 55–81.
- [35] CHI, Y., SONG, X., ZHOU, D., HINO, K., AND TSENG, B. Evolutionary spectral clustering by incorporating temporal smoothness. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2007), pp. 153–162.
- [36] CLARK, H., AND BRENNAN, S. Grounding in communication. In *Perspectives on Socially Shared Cognition*, L. Resnick, J. Levine, and S. Teasley, Eds. American Psychological Association, 1991, pp. 127–149.
- [37] CLARY, E., SNYDER, M., RIDGE, R., COPELAND, J., STUKAS, A., HAUGEN, J., AND MIENE, P. Understanding and assessing the motivations of volunteers: A functional approach, 1998.
- [38] CORTES, C., PREGIBON, D., AND VOLINSKY, C. Communities of interest. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis* (2001), pp. 105–114.

- [39] CROSS, R., BORGATTI, S. P., AND PARKER, A. Beyond answers: dimensions of the advice network. *Social Networks* 23 (2001), 215–235.
- [40] CROSS, R., NOHRIA, N., AND PARKER, A. Six myths about informal networks - and how to overcome them. *MIT Sloan Management Review* (2002), 67–75.
- [41] CROSS, R., PARKER, A., AND BORGATTI, S. P. A bird’s-eye view: Using social network analysis to improve knowledge creation and sharing. Tech. rep., IBM, 2002.
- [42] CROSS, R. L., AND PARKER, A. *The Hidden Power of Social Networks: Understanding how work really gets done in organizations*. Harvard Business School Press, Cambridge, 2004.
- [43] DAVENPORT, T. H., AND PRUSAK, L. *Working Knowledge: How Organizations Manage What They Know*. Harvard Business School Press, Boston, 1998.
- [44] DE NOOY, W., MRVAR, A., AND BATAGELJ, V. *Exploratory Social Network Analysis with Pajek (Structural Analysis in the Social Sciences)*. Cambridge University Press, 2005.
- [45] DESHPANDE, M., KURAMOCHI, M., WALE, N., AND KARYPIS, G. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. Knowl. Data Eng.* 17, 8 (2005), 1036–1050.
- [46] DUNNE, J. A., WILLIAMS, R. J., AND MARTINEZ, N. D. Food-web structure and network theory: The role of connectance and size. *PNAS* 99, 20 (2002), 12917–12922.
- [47] DWYER, C., HILTZ, S. R., AND PASSERINI, K. Trust and privacy concern within social networking sites: A comparison of facebook and myspace. In *Proceedings of the Thirteenth Americas Conference on Information Systems (AMCIS 2007)* (2007). Paper 339.
- [48] EALES, R. T. J. Support informal communities of practice within organizations. In *Sharing Expertise: Beyond Knowledge Management*, M. S. Ackerman, V. Pipek, and V. Wulf, Eds. The MIT Press, 2003.
- [49] ESTER, M., KRIEGEL, H.-P., SANDER, J., WIMMER, M., AND XU, X. Incremental clustering for mining in a data warehouse environment. In *Proc. of 24th VLDB Conference* (1998).
- [50] ESTER, M., KRIEGEL, H.-P., SANDER, J., AND XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)* (1996), pp. 226–231.
- [51] EUROSTAT. Cultural statistics, <http://epp.eurostat.ec.europa.eu/>, (retrieved 23/08/2008), 2007.

- [52] EVERITT, B. *Cluster Analysis*. Heineman Educational Books Ltd., 1980.
- [53] FALKOWSKI, T., AND BARTELHEIMER, J. Applying social network analysis methods to explore community dynamics. In *Applications of Social Network Analysis 2005*, U. Serdült and V. Täube, Eds. Wissenschaftlicher Verlag Berlin, 2008, pp. 189–212.
- [54] FALKOWSKI, T., BARTELHEIMER, J., AND SPILIOPOULOU, M. Community dynamics mining. In *Proc. of 14th European Conference on Information Systems (ECIS 2006)* (Göteborg, Sweden, 2006).
- [55] FALKOWSKI, T., BARTELHEIMER, J., AND SPILIOPOULOU, M. Mining and visualizing the evolution of subgroups in social networks. In *Web Intelligence (2006)*, IEEE Computer Society, pp. 52–58.
- [56] FALKOWSKI, T., BARTH, A., AND SPILIOPOULOU, M. Dengraph: A density-based community detection algorithm. In *In Proc. of the 2007 IEEE / WIC / ACM International Conference on Web Intelligence*, (2007), pp. 112–115.
- [57] FALKOWSKI, T., BARTH, A., AND SPILIOPOULOU, M. Studying community dynamics with an incremental graph mining algorithm. In *Proc. of the 14 th Americas Conference on Information Systems (AMCIS 2008)* (2008).
- [58] FALKOWSKI, T., AND SPILIOPOULOU, M. Observing dynamics in community structures. In *Proc. of Adaptation in Artificial and Biological Systems (AISB'06)* (2006), vol. 3, pp. 102–105.
- [59] FALKOWSKI, T., AND SPILIOPOULOU, M. Data mining for community dynamics. *Künstliche Intelligenz 3 / 2007* (2007), 23–29.
- [60] FALKOWSKI, T., AND SPILIOPOULOU, M. Users in volatile communities: Studying active participation and community evolution. In *In Proceedings of User Modeling 2007 - LNAI 4511* (2007), C. Conati, K. McCoy, and G. Paliouras, Eds., Springer, pp. 57–66.
- [61] FLAKE, G. W., LAWRENCE, S., AND GILES, C. L. Efficient identification of web communities. In *Sixth International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD 2000)*. ACM, 2000, pp. 150–160.
- [62] FORD, L. R., AND FULKERSON, D. R. *Flows in Networks*. Princeton University Press, 1962.
- [63] FORTUNATO, S., AND BARTHÉLEMY, M. Resolution limit in community detection. *Proceedings of the National Academy of Sciences of the United States of America (PNAS) 104*, 1 (2007), 36–41.
- [64] FREEMAN, L. C. A set of measures of centrality based on betweenness. *Sociometry 40* (1977), 35–41.

- [65] FREEMAN, L. C. The sociological concept of "group": An empirical test of two models. *The American Journal of Sociology* 98, 1 (1992), 152–166.
- [66] FREEMAN, L. C. *The Development Of Social Network Analysis: A Study in the Sociology of Science*. Booksurge, 2004.
- [67] FREEMAN, L. C. *The Development of Social Network Analysis: A Study in the Sociology of Science*. BookSurge Publishing, 2004.
- [68] FRUCHTERMAN, T., AND REINGOLD, E. Graph drawing by force-directed placement. *Software-Practice & Experience* 21, 11 (1991), 1129–1164.
- [69] FUKUYAMA, F., AND SHULSKY, A. N. *The "virtual corporation" and army organization*. RAND, 1997.
- [70] GAERTLER, M. Clustering. In *Network Analysis* (2004), U. Brandes and T. Erlebach, Eds., vol. 3418 of *Lecture Notes in Computer Science*, Springer, pp. 178–215.
- [71] GELEIJNSE, G., SCHEDL, M., AND KNEES, P. The quest for ground truth in musical artist tagging in the social web era. In *Proceedings of the Eighth International Conference on Music Information Retrieval (ISMIR'07)* (Vienna, Austria, September 2007), S. Dixon, D. Bainbridge, and R. Typke, Eds., pp. 525 – 530.
- [72] GIBSON, D., KLEINBERG, J., AND RAGHAVAN, P. Inferring web communities from link topology. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia* (1998), pp. 225–234.
- [73] GIRVAN, M., AND NEWMAN, M. E. J. Community structure in social and biological networks. *PNAS* 99, 12 (June 2002), 7821–7826.
- [74] GLOOR, P. A., AND ZHAO, Y. Tecflow - a temporal communication flow visualizer for social networks analysis. In *CSCW'04 Workshop on Social Networks* (2004), ACM.
- [75] GOLDMAN, S. L., PREISS, K., AND NAGEL, R. N. *Agile Competitors and Virtual Organizations*. 1995.
- [76] GONGLA, P., AND RIZZUTO, C. R. Evolving communities of practice: Ibm global services experience. *IBM Systems Journal* 40, 4 (2001), 842–862.
- [77] GONGLA, P., AND RIZZUTO, C. R. Where did that community go? - communities of practice that disappear. In *Knowledge Networks: Innovation through Communities of Practice*, P. Hildreth and C. Kimble, Eds. Idea Group Publishing, Hershey, PA, USA, 2004.
- [78] GRANOVETTER, M. The strength of weak ties. *American Journal of Sociology* 78, 6 (1973), 1360–1380.

- [79] GREGORY, S. An algorithm to find overlapping community structure in networks. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007)* (2007), Springer-Verlag, pp. 91–102.
- [80] GUIMERÁ, R., UZZI, B., SPIRO, J., AND AMARAL, L. A. N. Team assembly mechanisms determine collaboration network structure and team performance. *Science* 308, 5722 (April 2005), 697–702.
- [81] HAN, J., AND KAMBER, M. *Data Mining. Concepts and Techniques*, 2nd ed. ed. Morgan Kaufmann, 2006.
- [82] HAND, D., MANNILA, H., AND SMYTH, P. *Principles of Data Mining*. MIT Press, 2001.
- [83] HANSEN, M. T. Knowledge networks: Explaining effective knowledge sharing in multiunit companies. *Organization Science* 13, 3 (2002), 232–248.
- [84] HAY, M., MIKLAU, G., JENSEN, D., WEIS, P., AND SRIVASTAVA, S. Anonymizing social networks. Tech. rep., University of Massachusetts Amherst, 2007.
- [85] HAYS, T., AND MINICHELLO, V. The meaning of music in the lives of older people: a qualitative study. *Psychology of Music* 33, 4 (2005), 437–451.
- [86] HIGHTOWER, R. T., WARKENTIN, M. E., SAYEED, L., AND MCHANEY, R. Information exchange in virtual work groups. In *The virtual workplace*, M. Igarria and M. Tan, Eds. IGI Publishing, Hershey, PA, USA, 1998, pp. 199–216.
- [87] HILLERY, G. Definitions of community: Areas of agreement. *Rural Sociology* 20, 2 (1955), 111–123.
- [88] HINNEBURG, A., AND KEIM, D. A. An efficient approach to clustering in large multimedia databased with noise. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)* (1998), pp. 58–65.
- [89] HOPCROFT, J., KHAN, O., KULIS, B., AND SELMAN, B. Natural communities in large linked networks. In *Proc. of KDD'03* (2003).
- [90] HOPCROFT, J., KHAN, O., KULIS, B., AND SELMAN, B. Tracking evolving communities in large linked networks. *PNAS* (2004).
- [91] HUAN, J., BANDYOPADHYAY, D., WANG, W., SNOEYINK, J., PRINS, J., AND TROPSHA, A. Comparing graph representations of protein structure for mining family-specific residue-based packing motifs. *Journal of Computational Biology* 12, 6 (2005), 657–671.

- [92] HUISMAN, M., AND VAN DUIJN, M. A. Software for social network analysis. In *Models and Methods in Social Network Analysis*, P. J. Carrington, J. Scott, and S. Wasserman, Eds. Cambridge University Press, 2005, pp. 270–316.
- [93] IGBARIA, M., AND TAN, M., Eds. *The Virtual Workplace*. Idea Group, 1998.
- [94] III, J. H., AND ARMSTRONG, A. G. *Net Gain: Expanding Markets Through Virtual Communities*. Harvard Business School Press, 1997.
- [95] JACKSON, M. H. Assessing the structure of communication on the world wide web. *Journal of Computer-Mediated Communication* 3, 1 (1997).
- [96] JACKSON, P., Ed. *Virtual Working: Social and Organizational Dynamics*. 1999.
- [97] JAIN, A. K., AND DUBES., R. *Algorithms for Clustering Data*. Prentice Hall, 1988.
- [98] JARVENPAA, S. L., AND LEIDNER, D. E. Communication and trust in global virtual teams. *ORGANIZATION SCIENCE* 10, 6 (1999), 791–815.
- [99] JEONG, H., NÉDA, Z., AND BARABÁSI, A.-L. Measuring preferential attachment in evolving networks. *Europhysics Letters* 61, 4 (2003), 567–572.
- [100] KAMADA, T., AND KAWAI, S. An algorithm for drawing general undirected graphs. *Information Processing Letters* 31, 1 (1989), 7–15.
- [101] KARYPIS, G., HAN, E.-H., AND KUMAR, V. Chameleon: Hierarchical clustering using dynamic modeling. *Computer* 32, 8 (1999), 68–75.
- [102] KAUFMANN, M., AND WAGNER, D., Eds. *Drawing Graphs: Methods and Models (Lecture Notes in Computer Science)*, 1 ed. Springer, Berlin, 2001.
- [103] KLEINBERG, J. Authoritative sources in a hyperlinked environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms* (1998).
- [104] KOGUT, B., AND WALKER, G. The small world of germany and the durability of national networks. *American Sociological Review* 66 (2001), 317–335.
- [105] KOSCHÜTZKI, D., LEHMANN, K. A., PEETERS, L., RICHTER, S., TENFELDE-PODEHL, AND ZLOTOWSKI, O. Centrality indices. In *Network Analysis*, U. Brandes and T. Erlebach, Eds. Springer, 2005.
- [106] KOSUB, S. Local density. In *Network Analysis - Methodological Foundations*, U. Brandes and T. Erlebach, Eds., vol. 3418 of *LNCS*. Springer, 2005, ch. 6, pp. 112–142.
- [107] KRAUSE, J., AND RUXTON, G. *Living in Groups*. Oxford University Press, USA, 2002.

- [108] KREUTZ, G., OTT, U., TEICHMANN, D., OSAWA, P., AND VAITL, D. Using music to induce emotions: Influences of musical preference and absorption. *Psychology of Music* 36, 1 (2008), 101–126.
- [109] KUMAR, R., NOVAK, J., AND TOMKINS, A. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2006), ACM, pp. 611–617.
- [110] KUMAR, R., RAGHAVAN, P., RAJAGOPALAN, S., AND TOMKINS, A. Trawling the web for emerging cyber-communities. In *Computer Networks* (1999), pp. 1481–1493.
- [111] LANPHEAR, B. P., BYRD, R. S., AUINGER, P., AND SCHAFFER, S. J. Community characteristics associated with elevated blood lead levels in children. *Pediatrics* 101, 2 (1998), 264–271.
- [112] LATANÉ, B. Dynamic social impact: The creation of culture by communication. *Journal of Communication* 46, 4 (1996), 13–25.
- [113] LESKOVEC, J., KLEINBERG, J., AND FALOUTSOS, C. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proc. of KDD'05* (2005).
- [114] LESSER, E. L., AND STORCK, J. Communities of practice and organizational performance. *IBM Systems Journal* 40, 4 (2001), 831–841.
- [115] LEVY, M., AND SANDLER, M. A semantic space for music derived from social tags. In *8th International Conference on Music Information Retrieval (ISMIR 2007)* (2007).
- [116] MACQUEEN, J. B. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability* (1967), pp. 281–297.
- [117] MALHOTRA, Y., Ed. *Knowledge Management and Virtual Organizations*, 1 ed. IGI Global, 2000.
- [118] MCKAY, C., AND FUJINAGA, I. Musical genre classification: Is it worth pursuing and how can it be improved? In *ISMIR* (2006), pp. 101–106.
- [119] MCKENNA, K. Y. A., AND GREEN, A. S. Virtual group dynamics. *Group Dynamics: Theory, Research, and Practice* 6, 1 (2002), 116–127.
- [120] MEI, Q., AND ZHAI, C. Discovering evolutionary theme patterns from text: An exploration of temporal text mining. In *Proc. of KDD'05* (2005), pp. 198–207.
- [121] MILGRAM, S. The small world problem. *Psychology Today* 22 (1967), 61–67.

- [122] MILO, R., SHEN-ORR, S., ITZKOVITZ, S., KASHTAN, N., CHKLOVSKII, D., AND ALON, U. Network motifs: Simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.
- [123] MOODY, J., MCFARLAND, D., AND BENDER-DEMOLL, S. Dynamic network visualization. *American Journal of Sociology* 110, 4 (2005), 1206–1241.
- [124] MORAN, P. Structural vs. relational embeddedness: social capital and managerial performance. *Strategic Management Journal* 26, 12 (2005), 1129–1151.
- [125] NAJJAR, W., AND GAUDIOT, J.-L. Network resilience: A measure of network fault tolerance. *IEEE Transactions on Computers* 39, 2 (1990), 174–181.
- [126] NEWMAN, M. E. J. The structure of scientific collaboration networks. *PNAS* 98, 2 (2001), 404–409.
- [127] NEWMAN, M. E. J. Fast algorithm for detecting community structure in networks. *Physical Review E* 69, 066133 (2004).
- [128] NEWMAN, M. E. J., AND GIRVAN, M. Finding and evaluating community structure in networks. *Physical Review E* 69, 026113 (2004).
- [129] NG, R. T., AND HAN, J. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94:)* (San Francisco, CA, USA, 1994), Morgan Kaufmann Publishers Inc., pp. 144–155.
- [130] NG, R. T., AND HAN, J. Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering* 14, 5 (2002), 1003–1016.
- [131] NOV, O. What motivates wikipedians? *Communications of the ACM* 50, 11 (2007), 60–64.
- [132] ORAVEC, J. A. *Virtual individuals, virtual groups: human dimensions of groupware and computer networking*. Cambridge University Press, New York, NY, USA, 1996.
- [133] PALLA, G., DERÉNYI, I., FARKAS, I., AND VICSEK, T. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435, 7043 (June 2005), 814–818.
- [134] PARK, H. W. Hyperlink network analysis: A new method for the study of social structure on the web. *Connections* 25, 1 (2003), 49–61.
- [135] PAULEEN, D., Ed. *Virtual teams: Projects, protocols and processes*. Idea Group Publishing, 2004.

- [136] PINNEY, J. W., AND WESTHEAD, D. R. Betweenness-based decomposition methods for social and biological networks. In *Interdisciplinary Statistics and Bioinformatics* (2007), S. Barber, P. Baxter, K. Mardia, and R. Walls, Eds., Leeds University Press, pp. 87–90.
- [137] PREECE, J. *Online Communities: Designing Usability, Supporting Sociability*. John Wiley & Sons, Chichester, UK, 2000.
- [138] RENTFROW, P. J., AND GOSLING, S. D. The content and validity of music-genre stereotypes among college students. *Psychology of Music* 35, 2 (2007), 306–326.
- [139] RHEINGOLD, H. *The Virtual Community*. Addison-Wesley, Reading, MA, USA, 1993.
- [140] RODAN, S., AND GALUNIC, C. More than network structure: how knowledge heterogeneity influences managerial performance and innovativeness. *Strategic Management Journal* 25, 6 (2004), 541–562.
- [141] RUUSKA, I., AND VARTIAINEN, M. Communities and other social structures for knowledge sharing: A case study in an internet consultancy company. In *Communities and Technologies*, M. Huysman, E. Wenger, and V. Wulf, Eds. Kluwer, 2003, pp. 163–183.
- [142] SAILER, L. D., AND GAULIN, S. J. C. Proximity, sociality, and observation: The definition of social groups. *American Anthropologist* 86, 1 (1984), 91–98.
- [143] SAINT-ONGE, H., AND WALLACE, D. *Leveraging Communities of Practice for Strategic Advantage*. Butterworth-Heinemann, 2003.
- [144] SALAWAY, G., CARUSO, J. B., NELSON, M. R., AND ELLISON, N. The ecar study of undergraduate students and information technology. Tech. rep., 2008.
- [145] SALTON, G. *Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1988.
- [146] SARWAR, B., KARYPIS, G., KONSTAN, J., AND RIEDL, J. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web* (New York, NY, USA, 2001), ACM, pp. 285–295.
- [147] SCHEDL, M., POHLE, T., KNEES, P., AND WIDMER, G. Assigning and visualizing music genres by web-based co-occurrence analysis. In *Proceedings of the Seventh International Conference on Music Information Retrieval (ISMIR'06)* (Victoria, Canada, October 2006).
- [148] SCHOBERTH, T., PREECE, J., AND HEINZL, A. Online communities: A longitudinal analysis of communication activities. In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)* (2003).

- [149] SCOTT, J. *Social Network Analysis: A Handbook*. Sage, 2000.
- [150] SEIDMAN, S. Internal cohesion of ls sets in graphs. *Social Networks* 5 (1983), 97–107.
- [151] SHETTY, J., AND ADIBI, J. Enron email dataset. Tech. rep., 2004.
- [152] SINGH, J., HANSEN, M. T., AND PODOLNY, J. M. The world is not small for everyone: Pathways of discrimination in searching for information in organizations. Tech. rep., INSEAD Working Paper, 2008.
- [153] SKVORETZ, J., AND FAUST, K. Logit models for affiliation networks. *Sociological Methodology* 29, 28 (1999), 253–280.
- [154] SNIJDERS, T. Models for longitudinal network data. In *Models and Methods in Social Network Analysis*, P. Carrington, J. Scott, and S. Wasserman, Eds. Cambridge University Press, 2005, pp. 215–247.
- [155] SPILIOPOULOU, M., NTOUTSI, I., THEODORIDIS, Y., AND SCHULT, R. Monic - modeling and monitoring cluster transitions. In *Proc. of KDD'06* (2006), pp. 706–711.
- [156] TAN, P.-N., STEINBACH, M., AND KUMAR, V. *Introduction to Data Mining*. Pearson, 2006.
- [157] TANG, L., LIU, H., ZHANG, J., AND NAZERI, Z. Community evolution in dynamic multi-mode networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2008), pp. 677–685.
- [158] TANTIPATHANANANDTH, C., BERGER-WOLF, T., AND KEMPE, D. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2007).
- [159] TOENNIES, F. *Community and Society*. Transaction Publishers, 1988.
- [160] WARREN, R. L. *The Community in America*, vol. 3 rd. Rand McNally College Publishing Co., 1987.
- [161] WASSERMAN, S., AND FAUST, K. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.
- [162] WATTS, D. J., AND STROGATZ, S. H. Collective dynamics of small-world networks. *Nature* 393 (June 1998), 440–442.
- [163] WELLMAN, B. *Networks In The Global Village: Life In Contemporary Communities*. Westview Press, 1998.

- [164] WELLMAN, B., AND BERKOWITZ, S. *Social Structures: A Network Approach*. Cambridge University Press, 1988.
- [165] WELLMAN, B., AND GULIA, M. Virtual communities as communities: Net surfers don't ride alone. In *Communities in Cyberspace*, M. A. Smith and P. Kollock, Eds. Routledge, 1999, pp. 167–192.
- [166] WENGER, E., MCDERMOTT, R., AND SNYDER, W. M. *Cultivating Communities of Practice*. Harvard Business School Press, Cambridge, 2002.
- [167] WENGER, E., AND SNYDER, W. Communities of practice: The organizational frontier. *Harvard Business Review* 78, 1 (January 2000), 139–145.
- [168] WITTEN, I. H., AND FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.
- [169] XU, X., YURUK, N., FENG, Z., AND SCHWEIGER, T. A. J. Scan: A structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2007)*, pp. 824–833.
- [170] YOOK, S., JEONG, H., AND BARABÁSI, A.-L. Modeling the internet's large-scale topology. *PNAS* 99 (2002), 13382–13386.
- [171] ZHANG, S., WANG, R.-S., AND ZHANG, X.-S. Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *Physica A: Statistical Mechanics and its Applications* 374, 1 (2007), 483–490.
- [172] ZHELEVA, E., AND GETOOR, L. Preserving the privacy of sensitive relationships in graph data. In *Proceedings of the First SIGKDD International Workshop on Privacy, Security, and Trust in KDD (PinKDD 2007)* (March 2007), vol. 4890 of *Lecture Notes in Computer Science*, Springer, pp. 153–171.