Local Properties of Social Networks

PHD THESIS

2018 Šárka Zehnalová
I hereby declare that this thesis was written by myself. I have quoted all the references I have drawn upon.

24.8.2018
ACKNOWLEDGMENTS

I wish to thank all those who helped me with my work, especially prof. RNDr. Václav Snášel CSc. for his advice, ideas and inspiration; Miloš Kudělka for his guidance and support throughout the years. Moreover, I have had a wonderful group of coauthors and collaborators who were a pleasure to work with, my thanks to them all. I also want to thank my friend Zdeněk Horák for his encouragement and helpful advice.
ABSTRACT

Due to the development of information technology, we have a large amount of data available in recent years, perhaps from all areas of research. Many real world data and processes have a network structure and can usefully be represented as graphs. The network representation of complex systems provides a useful model for studying many processes, including biological, technological and social networks. Network analysis focuses on the relations among the nodes exploring the properties of each network. Owing to enormous and sustained growth of real world networks, the current trend in analyzing networks is to focus on local methods.

This thesis is focused on investigating local characteristics of social networks and application of local methods on large datasets. Two novel local measures for node importance in the network are presented. The first one can be used for ranking of nodes or as an approach to transforming an unweighted network to weighted one, or to assist community detection. The second one can be utilized for network sampling or graph construction. Another topic of this thesis is also the analysis of large co-authorship datasets in order to develop a model capable of generating realistic graphs.

Keywords: social network analysis, social networks, complex networks, graphs, graph reduction, ranking, centrality measure, collaboration networks, nearest neighbor, role identification
# CONTENTS

1 INTRODUCTION  
1.1 History of SNA ................................. 1  
1.2 Topics of research ............................. 2  
1.3 Contents of this thesis ......................... 3  

2 THEORETICAL BACKGROUND  
2.1 Basic concepts and definitions ................ 4  
2.1.1 Centrality and ranking ....................... 4  
2.2 Statistical properties of networks .......... 5  
2.2.1 Community structure ....................... 6  
2.3 Other methods for analyzing large-scale data 6  
2.3.1 Sampling ................................. 6  

3 LOCAL DEPENDENCY  
3.1 Introduction .................................. 7  
3.2 Related work .................................. 7  
3.3 Dependency measure ............................ 8  
3.4 Dependency in a weighted network .......... 9  
3.5 Dependency in an unweighted network ....... 10  
3.6 Network edge weighting ....................... 11  
3.6.1 Transforming unweighted networks ........ 12  
3.7 Dependency degree ............................. 13  
3.7.1 PageRank .................................. 13  
3.8 Experiment 1: Ranking of nodes using PageRank and LocalDependency 17  
3.8.1 Distributed PageRank and LocalDependency computation ........ 18  
3.8.2 Results .................................. 18  
3.8.3 A note on performance ...................... 20  
3.9 Community detection based on dependency 22  
3.9.1 Community detection ....................... 22  
3.9.2 Local methods ............................. 23  
3.9.3 Problem definition ......................... 23  
3.9.4 Dependency of a node on a set of network nodes . . . . . 24  
3.9.5 Community detection based on node dependency .... 24  
3.9.6 Detection of communities around one node .. 25  
3.10 Experiment 2: Comparing two local methods for community detection 27  
3.10.1 Local Algorithm 2 - Iterative local expansion .......... 27
Contents

3.10.2 Settings ....................................................... 28
3.10.3 Dataset ....................................................... 28
3.10.4 Results ....................................................... 29
3.11 Discussion ..................................................... 29

4 LOCAL REPRESENTATIVENESS 31
4.1 Introduction ................................................... 31
4.2 Preliminaries ................................................... 31
4.3 Deterministic sampling based on local representativeness .... 33
  4.3.1 Motivation .................................................. 33
  4.3.2 Related work ............................................... 33
  4.3.3 Sampling algorithm: x-RSampling ...................... 34
4.4 Local representativeness in vector data ....................... 35
  4.4.1 Properties of x-RSampling algorithm .................. 35
  4.4.2 Experimental evaluation ................................ 36
4.5 Local representatives in weighted networks .................. 43
  4.5.1 Experimental evaluation ................................ 45
4.6 Graph construction based on local representativeness ....... 51
  4.6.1 Related work ............................................... 53
  4.6.2 Proposed LRNet algorithm .............................. 54
  4.6.3 Experimental evaluation ................................ 55
  4.6.4 Analysis using classes .................................... 57
4.7 Discussion ..................................................... 60

5 CO-AUTHORSHIP NETWORKS - EXPERIMENTS 62
5.1 Using self-organizing maps for identification of roles ........ 62
  5.1.1 Introduction ............................................... 62
  5.1.2 Related work ............................................... 63
  5.1.3 The self-organizing map .................................. 63
  5.1.4 Methodology ............................................... 64
  5.1.5 Experiment ................................................ 67
5.2 Evolution of author’s topic .................................... 72
  5.2.1 Introduction ............................................... 72
  5.2.2 Related work ............................................... 73
  5.2.3 Forgetting function ....................................... 74
  5.2.4 Topic evolution visualization ............................ 78
5.3 Discussion ..................................................... 81

6 CO-AUTHORSHIP NETWORK MODEL 82
6.1 Introduction ................................................... 82
6.2 Related work .................................................. 83
## Contents

6.3 Analysis of co-authorship datasets ........................................... 84
   6.3.1 Co-author roles ......................................................... 85
   6.3.2 Poisson distribution ................................................... 87
6.4 3-lambda model ............................................................... 89
   6.4.1 Network generator .................................................... 92
6.5 Experimental evaluation ...................................................... 94
   6.5.1 Properties of generated network ................................... 94
   6.5.2 Growth of generated network ....................................... 96
   6.5.3 Model limitations ..................................................... 97
6.6 Discussion ................................................................. 98

7 Email network analysis ....................................................... 99
   7.1 Introduction ............................................................. 99
   7.2 Related Work ............................................................ 100
      7.2.1 Email network analysis .......................................... 100
      7.2.2 Email visualization ............................................... 101
   7.3 Background ............................................................. 101
      7.3.1 Conversations ....................................................... 101
      7.3.2 Roles ............................................................... 102
      7.3.3 Context and community ......................................... 103
   7.4 System Overview ....................................................... 105
      7.4.1 Pre-processing ...................................................... 105
      7.4.2 Data import ......................................................... 106
      7.4.3 Analysis method ................................................... 106
      7.4.4 User interaction ................................................... 106
   7.5 Case study - Inflex LLC ................................................ 107
      7.5.1 Inflex LLC dataset ................................................ 107
      7.5.2 Project example .................................................... 107
      7.5.3 Project example with limited time interval .................... 109
      7.5.4 Accuracy ............................................................ 112
      7.5.5 Evaluation .......................................................... 112
   7.6 Discussion ............................................................. 113

8 Conclusions ................................................................. 114
   8.1 Future work ............................................................. 115
   8.2 Author’s bibliography .................................................. 116

Bibliography ................................................................. 118
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Examples of dependency between two nodes.</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>Example of weight derived from a path.</td>
<td>10</td>
</tr>
<tr>
<td>3.3</td>
<td>Dependency weights</td>
<td>12</td>
</tr>
<tr>
<td>3.4</td>
<td>Reduced network</td>
<td>12</td>
</tr>
<tr>
<td>3.5</td>
<td>Inverted dependency</td>
<td>12</td>
</tr>
<tr>
<td>3.6</td>
<td>Reduced network</td>
<td>12</td>
</tr>
<tr>
<td>3.7</td>
<td>Dependency weights</td>
<td>13</td>
</tr>
<tr>
<td>3.8</td>
<td>Reduced network</td>
<td>13</td>
</tr>
<tr>
<td>3.9</td>
<td>Inverted dependency</td>
<td>13</td>
</tr>
<tr>
<td>3.10</td>
<td>Reduced network</td>
<td>13</td>
</tr>
<tr>
<td>3.11</td>
<td>Dependency Degree</td>
<td>15</td>
</tr>
<tr>
<td>3.12</td>
<td>PageRank</td>
<td>15</td>
</tr>
<tr>
<td>3.13</td>
<td>Cumulative distributions (normalized).</td>
<td>19</td>
</tr>
<tr>
<td>3.14</td>
<td>Difference between PageRank and LocalDependency.</td>
<td>20</td>
</tr>
<tr>
<td>3.15</td>
<td>Execution time on Anselm</td>
<td>22</td>
</tr>
<tr>
<td>3.16</td>
<td>Local community illustration - following terms are used: community base (a starting set of nodes), community core C, community boundary B, community shell S.</td>
<td>24</td>
</tr>
<tr>
<td>3.17</td>
<td>Examples of dependency between two nodes.</td>
<td>25</td>
</tr>
<tr>
<td>3.18</td>
<td>Detected community with Algorithm 1 (left) and Algorithm 2 (right) for 'Vaclav Snasel'.</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Two-cluster dataset - x-RSampling algorithm</td>
<td>36</td>
</tr>
<tr>
<td>4.2</td>
<td>Two-cluster dataset - Random samples</td>
<td>36</td>
</tr>
<tr>
<td>4.3</td>
<td>Two-cluster dataset - Different parameters of x-RSampling algorithm</td>
<td>37</td>
</tr>
<tr>
<td>4.4</td>
<td>Birch3 dataset</td>
<td>39</td>
</tr>
<tr>
<td>4.5</td>
<td>Birch3 Representativeness bases</td>
<td>39</td>
</tr>
<tr>
<td>4.6</td>
<td>Birch3 - x-RSamples comparison</td>
<td>40</td>
</tr>
<tr>
<td>4.7</td>
<td>Birch3 - log.base 1.31</td>
<td>41</td>
</tr>
<tr>
<td>4.8</td>
<td>Birch3 - log.base 1.41</td>
<td>41</td>
</tr>
<tr>
<td>4.9</td>
<td>Birch3 - log.base 1.44</td>
<td>41</td>
</tr>
<tr>
<td>4.10</td>
<td>Birch3 - log.base 651</td>
<td>41</td>
</tr>
<tr>
<td>4.11</td>
<td>Czech Map dataset</td>
<td>41</td>
</tr>
<tr>
<td>4.12</td>
<td>Czech Map Representativeness bases</td>
<td>41</td>
</tr>
<tr>
<td>4.13</td>
<td>Czech Map - x-RSamples comparison</td>
<td>42</td>
</tr>
<tr>
<td>4.14</td>
<td>Czech Map - log.base 1.30</td>
<td>43</td>
</tr>
<tr>
<td>4.15</td>
<td>Czech Map - log.base 1.46</td>
<td>43</td>
</tr>
<tr>
<td>Figure 4.16</td>
<td>Czech Map - log.base 1.99</td>
<td>43</td>
</tr>
<tr>
<td>Figure 4.17</td>
<td>Czech Map - log.base 90</td>
<td>43</td>
</tr>
<tr>
<td>Figure 4.18</td>
<td>Example network with five x-RSamples</td>
<td>45</td>
</tr>
<tr>
<td>Figure 4.19</td>
<td>Dataset log-base distributions</td>
<td>46</td>
</tr>
<tr>
<td>Figure 4.20</td>
<td>DBLP x-RSamples metrics</td>
<td>50</td>
</tr>
<tr>
<td>Figure 4.21</td>
<td>DBLP Faloutos</td>
<td>51</td>
</tr>
<tr>
<td>Figure 4.22</td>
<td>2.5-RSample</td>
<td>51</td>
</tr>
<tr>
<td>Figure 4.23</td>
<td>1.8-RSample</td>
<td>51</td>
</tr>
<tr>
<td>Figure 4.24</td>
<td>1.5-RSample</td>
<td>51</td>
</tr>
<tr>
<td>Figure 4.25</td>
<td>Degree distribution, LRNet</td>
<td>56</td>
</tr>
<tr>
<td>Figure 4.26</td>
<td>Degree distribution, SqrtNN</td>
<td>57</td>
</tr>
<tr>
<td>Figure 4.27</td>
<td>Constructed graphs: Ecoli dataset</td>
<td>58</td>
</tr>
<tr>
<td>Figure 4.28</td>
<td>Constructed graphs: Seeds dataset</td>
<td>59</td>
</tr>
<tr>
<td>Figure 4.29</td>
<td>Receiver Operating Characteristic (ROC) curves, LRNet</td>
<td>60</td>
</tr>
<tr>
<td>Figure 4.30</td>
<td>Receiver Operating Characteristic (ROC) curves, SqrtNN</td>
<td>60</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Role extraction process</td>
<td>64</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Original network of dolphins</td>
<td>67</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>Network of dolphins clustered by self-organizing map (5x5)</td>
<td>67</td>
</tr>
<tr>
<td>Figure 5.4</td>
<td>Particular dolphins, nodes of the trained SOM and roles extracted from the dataset</td>
<td>68</td>
</tr>
<tr>
<td>Figure 5.5</td>
<td>Self-organizing map of authors from the DBLP dataset (8x8)</td>
<td>70</td>
</tr>
<tr>
<td>Figure 5.6</td>
<td>Self-organizing map of authors from the DBLP dataset (5x5)</td>
<td>70</td>
</tr>
<tr>
<td>Figure 5.7</td>
<td>Sample authors with role nr. 1</td>
<td>71</td>
</tr>
<tr>
<td>Figure 5.8</td>
<td>Sample authors with role nr. 2</td>
<td>71</td>
</tr>
<tr>
<td>Figure 5.9</td>
<td>Sample authors with role nr. 3</td>
<td>71</td>
</tr>
<tr>
<td>Figure 5.10</td>
<td>Sample authors with role nr. 4</td>
<td>71</td>
</tr>
<tr>
<td>Figure 5.11</td>
<td>Sample authors with role nr. 5</td>
<td>71</td>
</tr>
<tr>
<td>Figure 5.12</td>
<td>Sample authors with role nr. 6</td>
<td>71</td>
</tr>
<tr>
<td>Figure 5.13</td>
<td>Sample authors with role nr. 7</td>
<td>72</td>
</tr>
<tr>
<td>Figure 5.14</td>
<td>Sample authors with role nr. 8</td>
<td>72</td>
</tr>
<tr>
<td>Figure 5.15</td>
<td>Sample authors with role nr. 9</td>
<td>72</td>
</tr>
<tr>
<td>Figure 5.16</td>
<td>Sample authors with role nr. 10</td>
<td>72</td>
</tr>
<tr>
<td>Figure 5.17</td>
<td>Forgetting Curve</td>
<td>75</td>
</tr>
<tr>
<td>Figure 5.18</td>
<td>Calculation of S in time t</td>
<td>76</td>
</tr>
<tr>
<td>Figure 5.19</td>
<td>Example of calculation of stability for a keyword over a three year period.</td>
<td>78</td>
</tr>
<tr>
<td>Figure 5.20</td>
<td>Example of visualization of three keywords over a three year period.</td>
<td>78</td>
</tr>
<tr>
<td>Figure 5.21</td>
<td>Screenshot of Forcoa.NET page with ‘topic history’ window for Philip S. Yu.</td>
<td>79</td>
</tr>
</tbody>
</table>
List of Figures

Figure 5.22  Two Snapshots from Forcoa.NET 'topic history' window for Philip S. Yu - A) from month 1/2005, B) from month 12/2011.  .......... 80
Figure 5.23  The evolution of the top 7 keywords from the profile of Philip S. Yu.  81
Figure 6.1   DBLP: Poisson distribution  ......................... 86
Figure 6.2   Co-authors histograms for top 10 authors (by number of publications) in DBLP  .......................... 87
Figure 6.3   Distribution of the co-authors of proactive nodes overall and in individual roles - DBLP  .......................... 88
Figure 6.4   Distribution of the co-authors of proactive nodes overall and in individual roles - PubMed  .......................... 88
Figure 6.5   $\lambda_1 = 0, \lambda_2 = 3, \lambda_3 = 0$  .......................... 89
Figure 6.6   $\lambda_1 = 2, \lambda_2 = 3, \lambda_3 = 0$  .......................... 89
Figure 6.7   $\lambda_1 = 2, \lambda_2 = 3, \lambda_3 = 0.5$  .......................... 89
Figure 6.8  3-lambda model: generated networks with 200 nodes  .......................... 89
Figure 6.9   $\lambda_1$  .......................... 91
Figure 6.10  $\lambda_2$  .......................... 91
Figure 6.11  $\lambda_3$  .......................... 91
Figure 6.12  Growing lambdas: CC, modularity, assortativity  .......................... 91
Figure 6.13  $\lambda_1$  .......................... 91
Figure 6.14  $\lambda_2$  .......................... 91
Figure 6.15  $\lambda_3$  .......................... 91
Figure 6.16  Growing lambdas: average degree, average shortest path, diameter  .......................... 91
Figure 6.17  Time complexity: changing lambdas  .......................... 93
Figure 6.18  Setting$_1$  .......................... 94
Figure 6.19  Setting$_2$  .......................... 94
Figure 6.20  Setting$_3$  .......................... 94
Figure 6.21  Degree distribution (power law)  .......................... 94
Figure 6.22  Network: 1000 nodes, Setting$_1$  .......................... 95
Figure 6.23  Degree  .......................... 96
Figure 6.24  Clustering coeff.  .......................... 96
Figure 6.25  Assortativity  .......................... 96
Figure 6.26  Global properties influenced by changing lambdas  .......................... 96
Figure 6.27  DBLP: average number of co-authors of proactive author in total and in individual roles over time (1936 - 2016)  .......................... 98
Figure 7.1   System overview.  .......................... 105
Figure 7.2   TeamNet usage after importing email accounts.  .......................... 106
Figure 7.3   TeamNet chart - major team members for IntraDoc  .......................... 109
Figure 7.4   TeamNet network - users relations in IntraDoc  .......................... 109
Figure 7.5   TeamNet network - detected subgroup - Inflex employees  .......................... 110
Figure 7.6   TeamNet roles - team members distribution of roles in IntraDoc  110
| Figure 7.7 | TeamNet network evolution                      | 111 |
| Figure 7.8 | TeamNet organizations evolution                | 111 |
| Figure 7.9 | TeamNet roles evolution                        | 112 |
## List of Tables

<table>
<thead>
<tr>
<th>Table 3.1</th>
<th>Centrality measures for the Zachary’s Karate Club Network</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3.2</td>
<td>Kendall’s τ of two orderings</td>
<td>18</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>Execution time on Anselm</td>
<td>21</td>
</tr>
<tr>
<td>Table 3.4</td>
<td>Sub-communities detected around ‘Vaclav Snasel’</td>
<td>30</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Two cluster Dataset Sampling</td>
<td>35</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Dataset properties</td>
<td>39</td>
</tr>
<tr>
<td>Table 4.3</td>
<td>Birch3 - Representativeness bases</td>
<td>40</td>
</tr>
<tr>
<td>Table 4.4</td>
<td>Czech Map - Representativeness bases</td>
<td>42</td>
</tr>
<tr>
<td>Table 4.5</td>
<td>Parameters of nodes from the example network</td>
<td>44</td>
</tr>
<tr>
<td>Table 4.6</td>
<td>x-RSamples for the example network</td>
<td>45</td>
</tr>
<tr>
<td>Table 4.7</td>
<td>D-statistic for Astrocollab</td>
<td>47</td>
</tr>
<tr>
<td>Table 4.8</td>
<td>D-statistic for Condmatcollab</td>
<td>48</td>
</tr>
<tr>
<td>Table 4.9</td>
<td>D-statistic for Conmatcollab</td>
<td>48</td>
</tr>
<tr>
<td>Table 4.10</td>
<td>D-statistic for DBLP</td>
<td>49</td>
</tr>
<tr>
<td>Table 4.11</td>
<td>D-statistic for DBLP and PubMed sample datasets</td>
<td>50</td>
</tr>
<tr>
<td>Table 4.12</td>
<td>Global properties of constructed graphs</td>
<td>56</td>
</tr>
<tr>
<td>Table 4.13</td>
<td>Classification accuracy of constructed graphs</td>
<td>57</td>
</tr>
<tr>
<td>Table 5.1</td>
<td>Roles extracted from the DBLP dataset (using 8x8 SOM)</td>
<td>69</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>Roles extracted from the DBLP dataset (using 5x5 SOM)</td>
<td>70</td>
</tr>
<tr>
<td>Table 6.1</td>
<td>Global properties of generated networks</td>
<td>85</td>
</tr>
<tr>
<td>Table 6.2</td>
<td>Evolution of network properties</td>
<td>97</td>
</tr>
<tr>
<td>Table 6.3</td>
<td>Inflex LLC - Dataset Overview</td>
<td>108</td>
</tr>
<tr>
<td>Table 7.1</td>
<td>Inflex LLC - Dataset Details</td>
<td>108</td>
</tr>
<tr>
<td>Table 7.2</td>
<td>Precision and Recall (the first n-selected persons)</td>
<td>113</td>
</tr>
</tbody>
</table>
INTRODUCTION

Due to the development of information technology, we have a large amount of data available in recent years from all areas of research. Still a common problem is the analysis of these data. Classical statistical analysis is in many respects useful, researched and sufficient. However, if we are faced with very large data and our goal is not only to confirm a predetermined hypothesis, but get a general insight into the overall structure of the data, statistics is not ideal.

Network analysis focuses on the relations among the nodes exploring the properties of each network. Due to the enormous and sustained growth of real world networks, the current trend in analyzing networks is to focus on local methods and also on weighted networks. This is related to the fact that some tasks as originally formulated are difficult to solve in very large networks. Other tasks were formulated only for unweighted networks, which, however, does not quite correspond to the development of real networks in which, for example, the relationship between the nodes changes over time. Nevertheless, it appears that the same or modified tasks become significant when looking at a relatively small part of the network.

1.1 HISTORY OF SNA

Social network analysis has a history of almost a century, and has applications in many different areas e.g. social groups, communication of information, social influence, etc. In many application domains we can identify some sort of networks. Social scientists were the first academics to study real-world networks. It started in the 1930s with the works of sociologists Jacob Moreno and Helen Jennings introducing basic analytical methods [1, 2]. Moreno devised the ‘sociogram’ to represent formal properties of social configurations. The general method to gather data at that time was by directly querying participants and by the use of questionnaires. Early studies usually focused on centrality (in the sense of who has the most connections or influence in the group) or connectivity (how are individuals connected or not to others).

As a theoretical framework for social analysis mathematical and computational models were developed. Frank Harary used methods from the graph theory to represent social structures [3, 4]. The mathematical psychologist Anatol Rapoport studied processes underlying the diffusion of information [5]. Along with R. Solomonoff, they proposed a simple model for a network, known as the random graph [6], which was independently studied by Paul Erdos and Alfred Renyi [7].
1.2 TOPICS OF RESEARCH

Many advances to the field were later done by physicists. Derek de Solla Price was studying citation networks in which nodes represent academic publications, and if a particular paper cites another paper, there is an edge between them. Price’s mathematical model for the growth of citation networks aims at explaining the origin of networks with strongly skewed degree distributions. Price was continuing already existing work on publication patterns, starting from the work of Alfred Lotka who in 1926 formulated his “Law of Scientific Productivity”, which is an inverse power function describing the rate of published scientific papers over time and states that the distribution of scientific publications follows a power-law [8]. Price was the first to study this in network form and he found that both the number of papers that receive citations, as well as the number of papers a particular paper cites also follow power-laws [9].

A large number of papers have been published proposing new network models, after the random graph model most notable works are the family of models based on preferential attachment [10, 11]. Preferential attachment is a universal principle, during network growth, there is a preference for selecting high-degree nodes as neighbors for new nodes. Another family of models generates small diameters, e.g. the ‘small-world model’ of Watts and Strogatz [12].

Later, with the growth of social networks over the internet, their analysis has been at the forefront of importance for researchers in computer science. It has been accelerated by the increasing popularity of social networking sites such as LinkedIn, Twitter, etc.

In this thesis I focus on practical applications of Social Network Analysis (SNA) using existing or novel methods for data analysis. In practical application, it is always necessary to solve the following tasks:

- converting input data (in not already a network) to a form that can be processed (selecting the most appropriate method, e.g. for weighting network nodes)
- interpretation of the results of analysis

This thesis covers several areas, each of them is focused on studying network properties using local approach. The main topics are:

LOCAL MEASURES FOR NODE IMPORTANCE IN THE NETWORK. We introduce a method for measuring the dependency between the nodes of a network, that is based on a structure in the local surroundings of the node. The approach extracts relations between the network’s nodes and from either unweighted or already weighted network we get a weighted network where the assigned edge weights reflect the dependency between the nodes. Additionally, from dependency between the nodes, we derive a novel degree centrality measure which provides an interesting view
on the importance of the node in a network. Next we introduce a measure to find representative objects, which may be used for reducing the amount of data or to aid graph construction.

**Analysis of Large Co-Authorship Datasets** Publication activities make up a large, complex and steadily growing social network. In such a network, the evolution and dynamics contain much hidden information and implicit knowledge. In co-authorship networks ties among authors indicate common professional interests. Role detection (role extraction) is one of the tools that help us to understand the characteristics of the network. In one experiment we use SOM to extract roles. Another experiment focuses on evolution of topics of an individual author. As a topic we understand a set of keywords used in paper titles. If an author uses a keyword (or a phrase) in paper titles frequently and periodically, we can trace a history of this usage.

**Generating Realistic Graphs** We study one characteristic that significantly affects the development of a co-authorship network structure. Based on our observation and analysis of co-authorship networks, it is a Poisson distribution of the number of co-authors of publications in four different roles. To assess the relevance of this characteristic, we define a network growth model which uses this characteristic and four roles as its principle. Our experiments show that networks generated by this model have properties known from the environment of real-world networks.

**Email Network Analysis** Email communication is a source of important information, much of which is at first sight hidden. We created an analytical tool to analyze the deeper relationships in the email data. Those include relationships based on an interaction of multiple users in a team. We analyze the conversations that take place and get results that are in several different forms presented to the users. We present methods for weighting conversations, users and relationships, as well as methods for finding communities associated with the specified context.

### 1.3 Contents of This Thesis

The content of this thesis is divided in the following way. Chapter 2 presents some of the techniques currently used by researchers in the area of network analysis. Chapter 3 introduces local dependency measure which is followed by Chapter 4 introducing representativeness measure. Both chapters include experiments and possible applications. Special class of networks - co-authorship networks are analyzed in Chapter 5 and 6 where a network growth model is proposed. Chapter 7 is focused on email networks and team performance. The final chapter, Chapter 8, contains a summary of this thesis and a conclusion.
Theoretical Background

The field of social network analysis contains methodologies for analyzing network data using theoretical and mathematical methods for investigating network phenomena. (For general overview see [13, 14, 15, 16].)

2.1 Basic Concepts and Definitions

This section will explain some basic terms and definitions needed for understanding the following chapters.

As a network, we understand an ordered pair $G = (V, E)$ (undirected unweighted graph) of a set $V$ of nodes and a set $E$ of edges which are unordered pairs of nodes from $G$. Terms vertex or node are used interchangeably to refer to elements of the vertex set $V$, and similarly edge, link or connection to refer to elements of the edge set $E$.

Definition 1. (Directed and undirected graph): A graph is undirected if $(i, j) \in E \iff (j, i) \in E$, i.e., edges are unordered pairs of nodes. If pairs of nodes are ordered, i.e., edges have direction, then the graph is directed.

Definition 2. (Connected component): A connected component or just a component is a maximal set of nodes where any two nodes are connected to each other by a path.

Definition 3. (Node degree): a node has degree $d$ where $d$ is the number of connections it has to other nodes. If a network is directed, then nodes have two types of degrees, the in-degree (the number of incoming edges), and the out-degree (the number of outgoing edges). The graph’s average degree $\bar{d} = \frac{2E}{N}$.

Definition 4. (Triad): A triad (or a triangle) is a triple of connected nodes $(u, v, w)$, i.e., $(u, v), (v, w), (w, u) \in E$.

2.1.1 Centrality and ranking

The centrality of nodes, or the identification of which nodes are more ‘important’ than others, has been a key issue in network analysis. For various purposes one can use a variety of methods, the most known are the centrality measures, such as degree, closeness, betweenness (Freeman [17]), but also PageRank [18]. These methods provide an evaluation of the importance of nodes from the viewpoint of the whole network. Global ranking has been gradually losing its original meaning due to the enormous sizes of...
existing real-world networks. The methods can be adapted also for local tasks, i.e. tasks where for the calculation of the node’s importance it is not necessary to know the entire network.

**Definition 5. (Degree centrality):** Degree centrality is the number of nodes that a selected node is connected to. Often, in directed network we recognize in-degree and out-degree centrality.

**Definition 6. (Closeness centrality):** Closeness is a measure of the degree to which an individual is near all other individuals in a network. More formally, closeness centrality $C_C$ is defined as the total graph-theoretic distance to all other nodes in the network.

**Definition 7. (Betweenness Centrality):** Betweenness is a measure of the extent to which a node is connected to other nodes that are not connected to each other. It is a measure of the degree to which a node serves as a bridge. The betweenness centrality for each node is the number of shortest paths that pass through this node.

### 2.2 Statistical Properties of Networks

Research over the past few years has identified classes of properties that can be found in many real-world networks from various domains. While many patterns have been discovered, two of the principal ones are heavy-tailed degree distributions and small diameters.

**Definition 8. (Degree distributions):** The degree-distribution of a graph is a power law if the number of nodes $N_d$ of degree $d$ is given by $N_d \propto d^{-\gamma}$ ($\gamma > 1$) where $\gamma$ is called the power law degree exponent.

Further details on mathematics of power laws can be found in Newman [19].

**Definition 9. (Graph diameter):** A graph has diameter $d$ if every pair of nodes can be connected by a path of length at most $d$. Many real-world graphs display a relatively small diameter, which is also known as the “small-world” phenomenon.

Duncan J. Watts and Steven H. Strogatz, together published an article on the small world phenomenon [12]. Small-world properties are found in many real-world networks, social networks especially, but also in biological networks, technological networks, networks of brain neurons etc.

**Definition 10. (Clustering coefficient):** Clustering coefficient is a measure of transitivity in networks. The local clustering coefficient of a node in a graph quantifies how close its neighbors are to being a clique (complete graph). $C_v$ is the fraction of triangles (triads) centered at node $v$ among the $d(d-1)/2$ triangles that could possibly exist. Then $C_d$ is defined as the average $C_v$ over all nodes $v$ of degree $d$, and the global clustering coefficient $C$ is the average $C_v$ over all nodes $v$. 


2.3 Other Methods for Analyzing Large-Scale Data

2.2.1 Community structure

The concept of community is rather intuitive, so the definition is quite vague and still a subject of debate [20]. In reality, community-like sets of nodes usually correspond to organizational units in social networks [21], functional modules in biological networks, and scientific disciplines in collaboration networks between scientists [20].

Traditional techniques for detecting communities are graph partitioning and hierarchical clustering based on similarity measures [22]. The knowledge of the whole network or even the total number of communities is required. When the task is just to identify a community to which a particular node belongs, global algorithms are impractical, so it makes sense to consider local algorithms [23].

2.3 Other Methods for Analyzing Large-Scale Data

2.3.1 Sampling

Sampling [24] has already been applied to various types of data in many areas. The goal is to reduce the original dataset to a more manageable size. For large n-dimensional datasets, sampling methods have been developed in order to optimize data mining tasks, such as clustering or outlier detection. In the area of networks, sampling goals range from speeding up simulations to refined visualization.

The approaches can be divided into two groups: unbiased and biased. Each is suitable in different applications.

Uniform random sampling In uniform random sampling, every data point has the same probability of being selected for the sample. This approach has been used extensively in database and data mining tasks [25]. In the BIRCH [26] clustering algorithm, a random sample is selected as the initial step.

Biased sampling In biased sampling, every data point has a different probability of being selected for the sample. This approach is suitable for clustering tasks where the dataset includes clusters of different sizes. With uniform random sampling, small clusters are likely to be omitted. In this case, density-biased sampling [27] gives better results since sparse groups (areas) are given a higher probability of being included in the sample.
LOCAL DEPENDENCY

This chapter comes from my work previously published in journal [J1] and presents local approach to studying social network properties. The experiments were published in conference proceedings [P1, P2, P3].

3.1 INTRODUCTION

We introduce a method for measuring the strength of the relationship between two nodes of a network and for their ranking. This method is applicable to all kinds of networks, including directed and weighted networks. The approach extracts dependency relations among the network’s nodes from the structure in the local surroundings of the node. For the tasks we deal with in this chapter, the key technical parameter is locality. Since only the surroundings of the examined nodes are used in computations, there is no need to analyze the entire network. This allows the application of our approach to the area of large-scale networks. We present several experiments using small networks as well as large-scale artificial and real world networks. The results of the experiments show high effectiveness due to the locality of our approach and also the high quality node ranking comparable to PageRank.

3.2 RELATED WORK

Weighted networks have previously been studied in many papers and applications. Many of them emphasize the advantages of weighted networks over classical - binary - ones [28, 29]. Edge weighting can be based on different approaches, which are always related to the type of network. Node weighting (ranking) is usually understood as directly related to the edges that nodes have with their neighbors and the weight of those edges.

There are many approaches to assigning weights to edges in networks. The most natural ones usually arise from reality. For example, for a social network of telephone company customers we may assign weight to an edge based on the total length of conversation between the nodes [30]. Edge weights in a network of air transport between cities can be assigned on the basis of the total number of passenger (or seats) on particular flights [31]. The genes network may also benefit from assigning weights to edges based on a similar function of particular genes [28]. It’s quite common that more than one weighting approach based on reality exists.
3.3 Dependency Measure

A weighted network might be also considered a result of network evolution [32]. An interesting area of research with many potential applications is the field of link prediction, particularly link weight prediction [33]. Zhang et al. (2005) [29] described an approach similar to that presented in this article. A method of assigning weight to an edge as a measure of topological overlap between two nodes was used. Han et al. (2009) [34] introduced the concept of supportiveness, which captures co-authorship relations in a non-symmetric way and derived a supportiveness-based author ranking scheme.

It makes sense to assign weights not only to edges, but also to the nodes [35]. Freeman (1979) [17] formalized three different measures of node centrality: degree, closeness and betweenness. Degree is the number of nodes that a selected node is connected to, and measures the involvement of the node in the network. Based on our dependency measure we propose a novel degree centrality measure, which provides a ranking of nodes in the network from the most independent to the most dependent ones.

Community extraction is another area that has been studied extensively [22, 36], because analyzing interconnected groups provides important information about how they function [16]. One of the first works on this topic [37] illustrates the problem of community detection within a group of monkeys. In this particular case, the unweighted network was unable to separate the monkeys based on their grooming habits. When considering the amount of grooming as a weight of an edge within the network, the problem became solvable.

The term ‘dependency’ used in this chapter has already been used in conjunction with social networks, but this approach is based on probability and is suited to collaborative filtering [38] or modeling influence [39]. Also when constructing partial correlation networks [40], the ‘dependency’ of one node on another node is calculated for the entire network.

3.3 Dependency Measure

We understand dependency as a generally asymmetric measure describing a relationship between two nodes of a network. In a network or graph \( G = (V, E) \), \( V \) is a set of nodes and \( E \) is a set of edges.

The computation of the dependency \( D(x, y) \) of node \( x \) on node \( y \) is done locally, only in the immediate surroundings of the two nodes. From the surroundings of node \( x \) only edges that lead to the neighbors of \( y \) are taken into account. This means that only relations between a ‘friend of a friend’ have some significance.

We assume that the dependency between two nodes is influenced not only by the relation between them, but also by the relationships in their surroundings. Let us show an example with two adjacent nodes, \( x \) and \( y \), in an unweighted undirected network. Consider the situation shown in Figures 3.1a and 3.1b, where nodes \( x \) and \( y \) share an edge. The nodes in the first figure have no additional neighbors. In the second figure,
node \( y \) is adjacent to three additional nodes. The intuition behind the term *dependency* says that the relation between nodes in Fig. 3.1a is balanced, while the situation in Fig. 3.1b is different - node \( y \) is less dependent on node \( x \), than vice versa. Fig. 3.1c contains two additional edges between node \( x \) and two different nodes. In this situation, node \( x \) is no longer so highly dependent on node \( y \) because of the two new edges. When thinking about dependency in Fig. 3.1c, we should also consider that the new edges include common neighbors of nodes \( x \) and \( y \) (which transmit some part of the dependency on node \( y \)). It is evident that dependency is an asymmetric measure since \( D(x, y) = D(y, x) \) may not always hold.

![Figure 3.1: Examples of dependency between two nodes.](image)

### 3.4 Dependency in a Weighted Network

In a weighted network, the weight of edges usually reflects the similarity between the nodes, the strength of an edge describes the level of cooperation or traffic between the nodes. These weights must be taken into consideration when calculating dependency. To illustrate how ‘friend of a friend’ relations affect the dependency between two nodes, look at the situation in Fig. 7.2, where we want to assess the weight from node \( V_i \) to node \( V_j \) across node \( V_x \). When \( w_{xj} \) is weak, \( w_{ij} \) must also be weak, e.g. \( \lim_{w_{xj} \to 0} (V_iV_j)V_x = 0 \); alternatively, when \( w_{xj} \) is very strong (in terms of similarity), meaning that nodes \( V_x \) and \( V_j \) are almost identical, then \( w_{ij} \) is identical to \( w_{ix} \), e.g. \( \lim_{w_{xj} \to \infty} (V_iV_j)V_x = w_{ix} \).

We define dependency as follows:

Let \( E(x) \) be the set of all non-zero weight edges adjacent to node \( x \). Let \( \text{Adj}(x, y) \) be the set of all edges between node \( x \) and any of the neighbors of node \( y \). Clearly, \( \text{Adj}(x, y) \subseteq E(x) \) and \( \text{Adj}(x, y) \) does not contain an edge between node \( x \) and node \( y \). Let \( W(e) \) be the weight of edge \( e \) and \( W(v_1, v_2) \) the weight of an edge between nodes \( v_1 \) and \( v_2 \) (\( W(v_1, v_2) = 0 \), if there is no such edge).
Let $x$ be a non-isolated node of the network. The dependency $D(x, y)$ of node $x$ on node $y$ is defined as:

$$D(x, y) = \frac{W(x, y) + \sum_{e_i \in \text{Adj}(x, y)} W(e_i) \cdot R(e_i)}{\sum_{e_i \in E(x)} W(e_i)}$$

(3.1)

$$R(e_i) = \frac{W(y, v_i)}{W(e_i) + W(y, v_i)}$$

(3.2)

where $R(e_i)$ is the coefficient of the dependency of node $x$ on node $y$ via the common neighbor $v_i$, therefore, $v_i \in e_i$.

This dependency describes the relation of one node to another node from the point of view of their surroundings. Using this concept, the dependency of one node on another node is calculated for the entire network (see Algorithm 6). We obtain a directed weighted adjacency matrix representing the fully connected network, which is capable of uncovering hidden relationships between the nodes. Once the adjacency matrix has been constructed, it is possible to reconstruct the network; several algorithms may be used, such as minimum spanning tree (MST), or just some sort of threshold.

The presented equations infer $D(x, y) \in \langle 0; 1 \rangle$. The dependency being equal to zero means that vertices $x$ and $y$ have no common edge or neighbor. Full dependency (dependency equal to one) describes the situation where node $x$ has only one common edge with node $y$.

**Remark 1.** The dependency is non-zero, if at least one of the following conditions holds:

1. There exists an edge between vertices $x$ and $y$.
2. Vertices $x$ and $y$ share at least one common neighbor.

### 3.5 Dependency in an Unweighted Network

There are many ways of weighting the relations [31], it could be based on the traffic between the nodes, similarity or the amount of collaboration. We may use the dependency
3.6 NETWORK EDGE WEIGHTING

Algorithm 2: Returns directed weighted adjacency matrix

```
input : a graph G = (V, E), w : E → R
output: D = |V| × |V|
1 foreach node v ∈ V do
  2 foreach node x from Adj(v, y) do
    3 if x is adjacent to v then
      4 sum_nom += w_v,x
    5 end
    6 sum_nom+= = w_x,y · R(x, y) sum_den+= = w_v,y D(v, y) = \frac{w_v,y + sum_nom}{sum_den}
  7 end
8 end
```

approach to unweighted networks and obtain a weighted version of the network, again a directed adjacency matrix is calculated, which is capable of uncovering hidden relationships between the nodes. Dependency degree centrality may be used for assigning weights to the nodes.

Using the formula in Equation 3.1 and 3.2 for an unweighted network, i.e. all weights are equal 1, and when measuring the dependency only between the neighbors, we get:

Let x not be a non-isolated node of the network. The dependency \( D(x, y) \) of node x on node y is:

\[
D(x, y) = \frac{1 + \sum_{i \in \text{Adj}(x, y)} \frac{1}{\deg(x)}}{\deg(x)}
\]  (3.3)

For situations depicted in Fig. 3.1 it holds:

3.1A: \( D(x, y) = D(y, x) = 1 \),
3.1B: \( D(x, y) = 1, D(y, x) = \frac{1}{4} \),
3.1C: \( D(x, y) = \frac{2}{3}, D(y, x) = \frac{1}{2} \).

3.6 NETWORK EDGE WEIGHTING

In this section we focus on applications of our method on real-world network data and we analyze two well-known networks in the literature, Zachary’s karate club network [41] and a social network of a community of bottlenose dolphins [42]. We took the original unweighted data and using our method from the previous section transformed those networks into weighted ones. From the obtained directed adjacency matrix we retained only the edges between the nodes that were adjacent in the original networks. Then from a pair of dependency edges between two nodes we took the value of a max-
3.6 Network Edge Weighting

imum of them. We assigned weights to the nodes according to the dependency degree centrality. The resulting networks are in Figures 3.3 and 3.7.

3.6.1 Transforming unweighted networks

We may perform a simple reduction of this network and filter some of the weakest edges. We define a binary dependency $D(V_i, V_j) \in \{0, 1\}$ and say that nodes are dependent when $D(V_i, V_j) = 1 : d_w(V_i, V_j) \geq 0.5$, they are not dependent when $D(V_i, V_j) = 0 : d_w(V_i, V_j) < 0.5$. Using this simple threshold we reduced our weighted networks, see the emerging community structures in Figures 3.4 and 3.8. Also, the remaining parts of the network suggest a strong dependency of nodes with small dependency degree on the nodes with high dependency degree. In a co-authorship network this is usually a type of relationship doctoral students have to their supervisors.

To obtain yet another view of the network, this time from the perspective of independency, we made an inversion of dependency weights $I(V_i, V_j) = 1 - D(V_i, V_j)$, see Figures 3.5 and 3.9 for the results. Here, the strong edges are between the nodes that are not dependent on each other; they represent independent relationships and their removal would affect the network connectivity.

And after another reduction based on binary division with independent nodes defined as $I(V_i, V_j) \in \{0, 1\}$ where $I(V_i, V_j) = 1 : i_w(V_i, V_j) > 0.5$ and $I(V_i, V_j) = 0 : i_w(V_i, V_j) \leq 0.5$, we get the part of the network where the remaining nodes are independent, see Figures 3.6 and 3.10 for the results. In a co-authorship network this is a peer-to-peer type of relationship that professors may have between each other. Removal of any of the nodes that are connected in this reduced network would considerably affect this node’s surroundings, conversely removal of any of the isolated nodes from the network would not affect the network or the surroundings of that node.

![Figures 3.3: Dependency weights](image1)
![Figures 3.4: Reduced network](image2)
![Figures 3.5: Inverted dependency](image3)
![Figures 3.6: Reduced network](image4)
3.7 Dependency Degree

We define the dependency degree of a node as the sum of all calculated dependency weights of incoming edges from its neighbors. Let $N(x)$ be the set of all nodes adjacent to node $x$. Let $d_{ij}$ be the dependency weight of a connection from $i$ to $j$.

$$\text{Deg}_x = \sum_{y \in N(x)} d_{yx}, \quad (3.4)$$

Since dependency between the nodes $D(x, y) \in (0; 1)$, the sum of dependencies will be smaller than the classical degree centrality of the node. The higher the dependency degree is, the more nodes in the surroundings of a focal node are dependent on it. In this manner we are able to create a natural ranking of nodes based on their importance in the network. For reference, we calculated some of the known centrality measures for the unweighted karate club network of Zachary [41]. In Table 3.1, each measure has a different scale of values, but it is possible to cross-reference them and find differences in proportionality. In Table 3.1, only half of the nodes (those with the highest node degree) were listed, see Fig. 3.11 and Fig. 3.12, with the whole network, where the size of the nodes corresponds to the dependency degree and PageRank respectively (the values here have been scaled to a proper interval). Nodes with a noticeable visible difference were highlighted.

**Remark 2.** The dependency degree centrality is by definition suitable for weighted networks.

3.7.1 PageRank

The PageRank [google, 18] algorithm was conceived in order to rank linked documents and is inseparably linked to the area of Web searches. However, from a more gen-
Table 3.1: Centrality measures for the Zachary’s Karate Club Network

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Degree</th>
<th>Dependency</th>
<th>Betweenness</th>
<th>Closeness</th>
<th>PageRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>17</td>
<td>9,267</td>
<td>160,552</td>
<td>0,017</td>
<td>3.431</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>9,108</td>
<td>231,071</td>
<td>0,017</td>
<td>3.298</td>
</tr>
<tr>
<td>33</td>
<td>12</td>
<td>6,403</td>
<td>76,690</td>
<td>0,016</td>
<td>2.438</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>3,785</td>
<td>75,851</td>
<td>0,017</td>
<td>1.941</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>4,456</td>
<td>28,479</td>
<td>0,015</td>
<td>1.798</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>2,727</td>
<td>6,288</td>
<td>0,014</td>
<td>1.219</td>
</tr>
<tr>
<td>32</td>
<td>6</td>
<td>1,805</td>
<td>73,010</td>
<td>0,016</td>
<td>1.263</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>1,120</td>
<td>29,529</td>
<td>0,016</td>
<td>1.012</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>1,160</td>
<td>24,216</td>
<td>0,016</td>
<td>1.004</td>
</tr>
<tr>
<td>24</td>
<td>5</td>
<td>1,522</td>
<td>9,300</td>
<td>0,012</td>
<td>1.072</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>1,875</td>
<td>15,833</td>
<td>0,012</td>
<td>0,990</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>1,875</td>
<td>15,833</td>
<td>0,012</td>
<td>0,990</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>1,101</td>
<td>0,000</td>
<td>0,013</td>
<td>0,833</td>
</tr>
<tr>
<td>28</td>
<td>4</td>
<td>0,822</td>
<td>11,792</td>
<td>0,014</td>
<td>0,872</td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>1,464</td>
<td>1,543</td>
<td>0,012</td>
<td>0,894</td>
</tr>
<tr>
<td>31</td>
<td>4</td>
<td>0,795</td>
<td>7,610</td>
<td>0,014</td>
<td>0,836</td>
</tr>
</tbody>
</table>


eral perspective, it can be seen as a weighting algorithm that assigns numerical scores (weights) to the nodes of a network, similarly as LocalDependency does. In contrast to LocalDependency, which utilizes local information to infer node weights, PageRank is a global measure that uses structural information about the whole network to weight nodes. Some local approximations were considered, Fortunato et al. (2008) [43] estimated PageRank from local knowledge of in-degree (in the specific case of the Web network) and Bar-Yossef and Mashiach (2008) [44] studied the effectiveness of Reverse PageRank. The basics of PageRank are outlined in this section in order to clarify the similarities and differences with LocalDependency.

The PageRank algorithm, originally inspired by bibliometric measures used for ranking scientific papers, is an objective estimation of the relevance of hypertext documents on the basis of their position within a hypertext network. The main idea behind PageRank is very simple. It assumes that documents that are linked from other trusted pages (i.e. pages with a high PageRank score) are trusted. The contribution of an incoming link is proportional to the trustworthiness of its origin. The highest contribution comes from links that are incoming from prestigious documents. In Google’s search algorithm, PageRank is utilized for the prioritization of documents in the result set retrieved by a keyword-based search. The PageRank of document $A$, denoted $PR(A)$, can be evaluated using [google, 18]:

$$PR(A) = \frac{1-d}{N} + d \sum_{B \in \text{links} to A} \frac{PR(B)}{L(B)}$$
The variable $d$ in Equation 3.5 represents a dumping factor [18, 45] (or teleportation probability [46]) from the interval $[0,1]$. The function $C(A)$ evaluates the number of hyperlinks leading from document $A$. PageRank is based on a so-called random surfer model. The random surfer starts her session at a random Web document. In every document, the surfer can either follow any of the hyperlinks linking the page to the rest of the World Wide Web, or with probability $d$, teleport to another randomly chosen Web document (including documents not linked from the current one). The concept of teleportation allows documents to be reached that are not linked from any other document on the Web [46].

Practical computation of the page rank vector is realized by applying power iteration to the sparse stochastic primitive matrix $G$ which is a twice-modified row-normalized hyperlink matrix $H$. The relationship between $G$ and $H$ is illustrated in Equation 3.6. Google matrix $G$ depends on the parameter $\alpha$, stochastic matrix $S$ and teleportation matrix $E$.

$$G = \alpha S + (1 - \alpha)E$$  \hspace{1cm} (3.6)

Stochastic matrix $S$ is used to deal with sinks in the hyperlink graph. Once the user enters a node (opens a document) that has no outlinks, she or he is offered the possibility of jumping to any other document in the collection with uniform probability. In other words, stochastic matrix $S$ replaces all $0$ rows of $H$ with rows containing $\frac{1}{n}$.

$$S = H + \frac{1}{n}ae^T$$  \hspace{1cm} (3.7)
where vector $a$ is a dangling vector having $a_i = 1$ if page $i$ is a dangling node (has no outlinks); otherwise $a_i = 0$. Matrix $E$ represents a teleportation matrix. It gives the surfer the possibility to teleport to any document in the collection at any time.

$$E = \frac{1}{n}ee^T$$  \hspace{1cm} (3.8)

In Equation 3.8, vector $e^T$ is the row vector of all 1s. The elements of $H$ are set according to Equation 3.9. The non-zero elements of row $i$ represent outlinks from $i$; non-zero elements of column $j$ represent inlinks to $j$.

$$H_{ij} = \begin{cases} \frac{1}{|P_i|} & \text{if } i \text{ links to } j \\ 0 & \text{otherwise} \end{cases}$$  \hspace{1cm} (3.9)

The parameter $\alpha \in [0, 1]$ in Equation 3.6 is a scaling parameter controlling the priority between link-following and random teleportation.

The random surfer is bound to the assumptions and simplifications made by Page and Brin in order to be able to compute PageRank. A real Web surfer, however, does not seem to follow all outlinks from a page with uniform probability or teleport to any document in the collection randomly. One approach to the improvement of PageRank is the personalization of hyperlink matrix $H$ and teleportation matrix $E$.

**Distributed PageRank computation**

Since PageRank processes primarily the Web matrix and other very large networks, distribution is necessary to obtain results in a reasonable time. Google uses MapReduce [47] based distributed implementation of the power iteration method. However, many other studies dealt with different types of distributed PageRank computations in the past.

The design of the fully distributed PageRank for P2P networks is due to Sankaralingam et al. (2003) [48]. The authors based their solution on the chaotic iterative solution of linear systems and developed a complex model of the computation. The method allowed incremental PageRank updates as documents were added and was intended for a ranking and search application designed for P2P networks. The evaluation of the proposed algorithm was only theoretical (i.e. no implementation details were considered). It was, however, cited as an inspiration for another work dealing with distributed PageRank computation [49]. The study proposed three distributed algorithms based on the original PageRank, which incrementally mitigated distribution- and communication-related issues of the distributed PageRank computation.

A practical approach to the distributed PageRank computation in the environment of a real-word cluster was presented by Rungsawang and Manaskasemsak [50]. The proposal included a design of data structures and used the Message Passing Interface
(MPI), namely the MPICH library, for communication. The authors performed computational experiments and showed that their method scales. Practical issues related to data distribution were discussed as well.

The communication overhead of the previous method was addressed in [51] and an optimized algorithm was proposed. Another improvement involving a hybrid MPI–multi-threaded implementation which very well fits the architecture of compute nodes of real-world clusters was presented and evaluated in [52].

The communication overhead which exists when executing the traditional power iteration method in a distributed way was also the focus of the work of Zhu et al. (2005) [53]. The authors proposed a divide-and-conquer algorithm for iterative aggregation and disaggregation utilizing the Block Jacobi smoothing method to reduce the communication.

An MPI-based library for MapReduce [47] was presented by Plimpton and Devine (2011) [54] and PageRank was used as one of the examples of graph algorithms implemented efficiently and in a portable way using their framework. The framework isolated application developers from distribution and communication details, but remained transparent, i.e. allowed the client code to call MPI directly. The work pointed out and solved a number of issues found during the implementation of MapReduce in MPI but, in contrast to other MapReduce libraries, did not provide fault-tolerance and redundancy.

A distributed Monte Carlo-based approach to the PageRank computation was presented by Das Sarma et al. (2013) [55]. The work emphasized the suitability of random-walk based Monte Carlo methods for scalable distributed PageRank computation (especially in contrast to power iteration, which is hard to perform in a distributed context and is sensitive to the volume of communication involved). The study proposed two PageRank algorithms with good time complexity $O\left(\log\frac{n}{\epsilon}\right)$ and $O\left(\sqrt{\log\frac{n}{\epsilon}}\right)$ for directed and undirected graphs, respectively. Moreover, proof of the probability of convergence of the proposed algorithms was presented as well.

3.8 Experiment 1: Ranking of Nodes using PageRank and LocalDependency

For this experiment, we implemented a simple MPI-based distributed implementation of the iterative version of PageRank as defined by eq. (3.5) for execution on the Anselm Cluster. The Anselm cluster represents the first phase ("small cluster") of Czech National Supercomputing Centre IT4Innovations. The cluster consists of 209 compute nodes, totaling 3344 compute cores with 15TB RAM and giving over 94 Tflop/s theoretical peak performance. Each node is an x86-64 computer with 16 cores and 64+GB RAM. The nodes are interconnected by fully non-blocking fat-tree Infiniband networks. A few nodes are also equipped with NVIDIA Kepler GPU or Intel Xeon Phi MIC accelerators. However,
3.8 Experiment 1: Ranking of nodes using PageRank and LocalDependency

GPUs and MIC accelerators were not used in our computations. Anselm uses the PBS Pro [56] resource allocation manager in version 12.

3.8.1 Distributed PageRank and LocalDependency computation

The distributed implementation was meant to be simple, intuitive, well portable (e.g. to clusters of commodity PCs), and last but not least, easy to extend by other graph computations that cannot be expressed in a vector\times matrix form like PageRank such as LocalDependency. The main aim was to enable computation of iterative graph algorithms in a distributed and memory-efficient way. The MPI code quite literally corresponds to PageRank as defined by eq. (3.5) and LocalDependency. Each MPI process in this approach computes the PageRank (or LocalDependency) score for a portion of nodes of the original graph. It knows the local topology (i.e. the edges among its assigned nodes) and neighborhood connections (i.e. the edges connecting local nodes to remote nodes), and no other information is stored in the MPI process. Each PageRank update between local nodes is handled locally with no communication overhead, and each PageRank update between remote nodes causes a message to be sent and received. The communication is handled in an asynchronous manner using the MPI_Isend and MPI_Irecv MPI functions for message passing and MPI_Waitall for synchronization. The MPI library used for our computations was OpenMPI 1.6.5.

<table>
<thead>
<tr>
<th>Graph</th>
<th>No. of Nodes</th>
<th>(\tau)</th>
</tr>
</thead>
<tbody>
<tr>
<td>artificial 100k</td>
<td>100,000</td>
<td>0.841</td>
</tr>
<tr>
<td>artificial 1M</td>
<td>1,000,000</td>
<td>0.774</td>
</tr>
<tr>
<td>DBLP unweighted</td>
<td>1,216,515</td>
<td>0.822</td>
</tr>
<tr>
<td>DBLP reduced</td>
<td>318,971</td>
<td>0.969</td>
</tr>
</tbody>
</table>

3.8.2 Results

We computed PageRank and LocalDependency (as defined in Section 3.7) on four different large datasets, two artificial networks with 100,000 and 1,000,000 nodes, and on two co-authorship networks constructed from a DBLP database\(^1\). We chose the PageRank measure to compare with LocalDependency, because the results on small networks (as seen in Table 3.1) suggested some similarity between them. Different scales of values of PageRank and LocalDependency were normalized for the following comparison. Fig. 3.13 (a) and (b) depict cumulative distributions of PageRank and LocalDependency.

\(^1\) \url{http://www.informatik.uni-trier.de/~ley/db/}
values for artificial networks with 100,000 and 1,000,000 nodes, respectively. Likewise Fig. 3.13 (c) and (d) show those distributions for unweighted and reduced DBLP, respectively. The reduced DBLP network was constructed using the forgetting function, which takes into account the frequency and regularity of publishing, for details see Kudelka et al. (2012) [57], so it became a weighted network.

It is clear that the distribution of LocalDependency values is similar to PageRank. For the difference in actual values, see Fig. 3.14, where in the case of the DBLP networks more than 95% of values fall in the 0 - 5% range. Since both PageRank and LocalDependency provide orderings of the data based on the importance of the nodes, we compare orders given by PageRank and LocalDependency. We use Kendall’s τ [58] as a measure of concordance between the two ranked lists. Kendall’s τ is a number in the interval [−1, 1], where two orders induced by the ranks that are the same have the value of 1, conversely, two orders that are opposite of each other have the value of −1, while value

\[ \tau_b \]

There are several definitions of τ based on how ties should be treated, we used the one known as τb.
3.8 EXPERIMENT 1: RANKING OF NODES USING PAGERANK AND LOCALDEPENDENCY

Figure 3.14: Difference between PageRank and LocalDependency.

0 can be interpreted as a lack of correlation. High values of τ measured on our four datasets, see Table 3.2, suggest that most pairs of values are in the same order in both lists.

3.8.3 A note on performance

The performance of the used implementation on two artificial data sets and on two variants of the DBLP is presented in this section. However, we note that the experiments were executed in a shared production environment which was utilized up to 90% before and during our experiments (i.e. only limited resources were available). The results of the experiments in terms of LocalDependency (LD) and PageRank (PR) execution times are shown for all data sets in Table 3.3 and illustrated in Fig. 3.15. Table 3.3 also shows the speedup (or slowdown) in execution times for 40, 64, 80, 100, and 120 MPI processes when compared with execution using 32 MPI processes.

The results of the performance measurements clearly confirm the expected: a naïve parallel implementation of PageRank and LocalDependency does not yield a significant performance boost and scales poorly. The computation of LocalDependency, which in contrast to PageRank requires only a single iteration, even slows down due to the growing communication overhead, especially for smaller graphs. The speedup on graphs with a larger number of nodes and edges is superlinear, but the execution time of the
3.8 Experiment 1: Ranking of Nodes Using PageRank and LocalDependency

Table 3.3: Execution time on Anselm.

<table>
<thead>
<tr>
<th>Processes</th>
<th>Time [s]</th>
<th>Speedup</th>
<th>Processes</th>
<th>Time [s]</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>100k nodes</td>
<td></td>
<td></td>
<td>DBLP unweighted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LD PR LD PR LD PR LD PR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>1134.57</td>
<td>6512.81</td>
<td>1.00</td>
<td>1.00</td>
<td>32</td>
</tr>
<tr>
<td>40</td>
<td>984.367</td>
<td>3650.96</td>
<td>1.15</td>
<td>1.78</td>
<td>40</td>
</tr>
<tr>
<td>64</td>
<td>1396.18</td>
<td>2987.83</td>
<td>0.81</td>
<td>2.18</td>
<td>64</td>
</tr>
<tr>
<td>80</td>
<td>1984.11</td>
<td>2783.59</td>
<td>0.57</td>
<td>2.34</td>
<td>80</td>
</tr>
<tr>
<td>100</td>
<td>2143.56</td>
<td>2972.12</td>
<td>0.53</td>
<td>2.19</td>
<td>100</td>
</tr>
<tr>
<td>120</td>
<td>3057.69</td>
<td>3112.97</td>
<td>0.37</td>
<td>2.09</td>
<td>120</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1M nodes</th>
<th></th>
<th></th>
<th>DBLP reduced</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LD PR LD PR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>21515.3</td>
<td>211747.00</td>
<td>1.00</td>
<td>1.00</td>
<td>32</td>
</tr>
<tr>
<td>40</td>
<td>14681</td>
<td>111701.00</td>
<td>1.47</td>
<td>1.90</td>
<td>40</td>
</tr>
<tr>
<td>64</td>
<td>5316.6</td>
<td>29276.20</td>
<td>4.05</td>
<td>7.23</td>
<td>64</td>
</tr>
<tr>
<td>80</td>
<td>6431.91</td>
<td>16912.90</td>
<td>3.35</td>
<td>12.52</td>
<td>80</td>
</tr>
<tr>
<td>100</td>
<td>3260.74</td>
<td>11623.50</td>
<td>6.60</td>
<td>18.22</td>
<td>100</td>
</tr>
<tr>
<td>120</td>
<td>3917.37</td>
<td>9323.83</td>
<td>5.49</td>
<td>22.71</td>
<td>120</td>
</tr>
</tbody>
</table>

Application using 32 MPI processes was actually slower than a fine tuned single-process implementation for small graphs. It can also be seen that the performance is data bound and depends on the structure of the graph because the execution times for 1M nodes, DBLP unweighted, and DBLP reduced are different despite the similar number of nodes.
3.9 Community detection based on dependency

One of the most obvious features of social networks is their community structure. Several types of methods were developed for discovering communities in the networks, either from the global perspective or based on local information only. Local methods are appropriate when working with large and dynamic networks or when real-time results are expected.

3.9.1 Community detection

There has been a considerable interest recently in the idea of communities and algorithms for their detection. The concept of community is rather intuitive, so the definition is quite vague and still a subject of debate [59, 60, 61]. Universally accepted qualitative definition states that community is a collection of nodes with dense internal connections, but sparser connections to other communities. These communities are also referred to as

Figure 3.15: Execution time on Anselm.
modules or clusters [20, 22]. Alternatively, one might define communities as the output of a community detection procedure [61, 62], which means that different techniques for detecting communities may lead to slightly different yet equally valid results [63].

Recently, many algorithms for detecting communities have been proposed. Traditional techniques are graph partitioning and hierarchical clustering based on similarity measures [64, 65]. Such methods usually identify all communities in an unweighted, undirected network, assigning each node to one community. The knowledge of the context of a whole network and even the total number of communities is required. The fixed number and size of communities are the constraints on community detection, as well as non-overlapping communities. Various metric functions have been proposed to help solve these problems. Many community-finding algorithms are based on maximizing the quantity known as modularity [62, 66, 67, 68], but any algorithm using modularity requires complete knowledge of the entire network.

When the task is just to identify a community to which a particular node belongs, global algorithms are impractical. Additional motivation for local methods comes from networks that require a rather demanding generation or exploration with a crawler [65]. Also, because the knowledge of the structure of the whole network may not be available, local algorithms were considered [23].

3.9.2 Local methods

Several local methods exist [63, 69, 70, 71, 72, 73], they start the search from a random node, and then gradually merge neighboring nodes one-at-a-time by optimizing a measure metric. In the paper [70] they tried to avoid the need of working with an entire social network during the community search.

Clauset [71] proposed a fast agglomerative algorithm that maximizes a measure called local modularity in a greedy fashion. Bagrow et al. [63] proposed an alternative method to detect local communities, which consists of an l-shell spreading outward from a starting node. The result depends on a starting node and a predefined threshold l, where l is the distance from the starting node to all shell nodes.

3.9.3 Problem definition

As a network we understand an undirected weighted graph G with N nodes and E edges. Local network community is a set of densely connected nodes from this network.

Local methods usually work with the terms community core - C, community boundary - B and community shell - S similarly as is shown in Fig 3.16 and communities are usually generated from a random starting node or a set of nodes (a community base). A local community expansion is an iterative process, in which only base nodes are considered to
be a community, while the other network nodes are gradually examined and the nodes that follow certain criteria then progressively expand the community.

Figure 3.16: Local community illustration - following terms are used: community base (a starting set of nodes), community core \( C \), community boundary \( B \), community shell \( S \).

3.9.4 **Dependency of a node on a set of network nodes**

The relation of dependency of one node on another could be generalized as a relation of dependency of node \( x \) on a set of \( n \) nodes \( Y = \{y_1, y_2, \ldots, y_n\} \).

Let \( x \) not be an isolated network node. Then the dependency \( D(x, Y) \) of node \( x \) on set \( Y \) of nodes is defined as:

\[
D(x, y) = \frac{\sum_{y_i \in N(x, Y)} W(x, y_i) + \sum_{e_i \in \text{Adj}(x, Y)} W(e_i) \cdot R(e_i)}{\sum_{e_i \in E(x)} W(e_i)} \quad (3.10)
\]

\[
R(e_i) = \max_{y_j \in Y(y_i)} \left( \frac{W(y_j, v_i)}{W(e_i) + W(y_j, v_i)} \right) \quad (3.11)
\]

3.9.5 **Community detection based on node dependency**

Every node in the community (excluding max. one base node) is dependent on the rest of community nodes.

**Definition 11.** (Community base) A community base is a starting set of \( n \) nodes appropriately chosen in advance, which by definition belongs to the community and which meets the following criteria:

1. It is a biconnected subgraph.
2. At least \((n - 1)\) nodes have to be dependent on the other base nodes.
Definition 12. (Recognition of a network node) An unrecognized network node becomes a recognized one, if during the process of local expansion, it meets the following criteria:

1. It is adjacent to at least two different community nodes.
2. It is dependent on the other community nodes.

![Figure 3.17: Examples of dependency between two nodes.](image)

By using dependency to indicate affiliation with a community, it can occur that a community node can be dependent on a node that does not belong to the community. For example, Figure 3.17 shows node $x$ dependent on each of its two neighbours (it shares a weight 2 edge with both of them and does not have any other edges). Thus node $x$ creates a basis with each of these neighbours. If we detect a community above this basis, only the basis will constitute the community (as there is no other node that meets the requirements for affiliation to the community). Therefore the node becomes a part of two communities. However, it would be more natural if each of these two bases generated an identical community with three nodes. Therefore we establish the term community closure, which solves this situation.

Definition 13. (Community Closure) Community closure is a set of community nodes, each of which qualifies:

1. At least one of the community nodes is dependent on it.
2. It is a neighbour of at least one basis or closure node.

A community is further understood as a local community, including its closure.

3.9.6 Detection of communities around one node

Unless it is said otherwise, a community base is considered to be two nodes connected by an edge, while at least one of the nodes is dependent on the other one. This couple should be called the edge base. The purpose of implementing the edge base is an effective detection of more communities.
The two detected local communities $L_1$ and $L_2$ can have nodes in four different set relations:

1. $L_1 = L_2$,
2. $L_1 \cap L_2$,
3. $L_1 \subset L_2$ resp. $L_2 \subset L_1$,
4. $L_1 \cap L_2 = X, X \neq \emptyset, X \subset L_1, X \subset L_2$

Detected communities can be nested or overlapping.

- If a starting base of a community is an edge base, the result of the detection algorithm is a biconnected subgraph.
3.10  EXPERIMENT 2: COMPARING TWO LOCAL METHODS FOR COMMUNITY DETECTION

3.10  EXPERIMENT 2: COMPARING TWO LOCAL METHODS FOR COMMUNITY DETECTION

In this experiment, we review two different algorithms, which apply the known approaches of local community detection. The aim is to compare the results they give on the co-authorship network from the DBLP dataset. For the purpose of our experiment, we have used a weighted network implemented in our Forcoa.NET system, where also one of the algorithms is used for community detection.

3.10.1  Local Algorithm 2 - Iterative local expansion

This algorithm searches communities according to a definition of a community. The way the community is defined is the most important thing for their search. In the article [70], authors evaluate quality of the community by the sharpness of its boundary. In other words, for a group of nodes to be a community, its boundary must be somewhat sharp. To measure sharpness we first have to define what a boundary is. The boundary node is such a node that is connected by an edge with at least one node both from outside and from inside of the community. And all the boundary nodes constitute the boundary. Then the sharpness R of the boundary is measured as a ratio of edges that lead from the boundary into the community and edges that lead both in and out of it.

Apart from the boundary, there are two other important sets. The first one is called D (discovered) with nodes that belong to the community. The second one is S (shell) which contains all the nodes that are connected with at least one boundary node. Algorithm starts from a single node, so at the beginning the boundary and discovered sets contain only the first node and the shell contains all of its neighbors. In each step, the algorithm counts hypothetical sharpness R for each node in shell. Then the node with the highest value of R (if the new R is higher than the current R) is added to the community and the three sets are updated. Algorithm repeats this process until the current R cannot be raised anymore. Further description of this algorithm is beyond the scope of this paper, for the details see [70].

In the article, authors work only with unweighted graph, but a lot of today’s social networks can be easily transformed to weighted. The algorithm can be quite easily adapted to weighted networks. Just instead of the number of edges will algorithm work with the sum of their weights.

In the article [75] T. Opsahl presented the new weighted variants of algorithms for node centrality measurement. He came with an idea to combine unweighted and weighted types of degree of a node and of shortest paths in a single calculation.

3  http://www.informatik.uni-trier.de/ley/db/
4  http://www.forcoa.net
Let us define $C$ as number of edges and $W$ as sum of edges either in a shortest path or in a degree of a node. Then his combined formula looks like:

$$O = C^{(1-\alpha)} \cdot W^\alpha$$  \hspace{1cm} (3.12)

where $\alpha$ is the number that affects the importance of both parts of the formula. When $\alpha$ is 1, this formula simulates common weighted variant and when $\alpha$ is 0, the common unweighted variant is simulated. Values between 0 and 1 changes the ratio of how both parts affect the result. Values lesser than 0 or greater than 1 make the particular part affect the result in a negative way, e.g. for values higher than 1 the node (path) with less edges is favored over the one with more edges even if the sums of weights of their edges are the same.

For this experiment, we have made a modification of the iterative local expansion algorithm described in [70] using the Opsahl’s idea. Instead of measuring just number of edges or just sum of their weights, we used Opsahl’s formula to combine them during the community search.

### 3.10.2 Settings

In this section the two algorithms described in subsection 3.9.5 and subsection 3.10.1 are used for community detection on a sample of weighted network. Both algorithms started with a preselected node, for Algorithm 1 (Dependency based) we have chosen the threshold of 0.5, which corresponds with natural intuition, for measuring the affiliation of a node to a community. If we further consider that node $x$ is dependent on node $y$, respectively on the set of nodes $Y$, we have to assume that it holds: $D(x,y) \geq 0.5$, resp. $D(x,Y) \geq 0.5$.

For Algorithm 2 (Iterative Local Expansion) we have selected $\alpha$ as 0.5, so both the weights and the connectivity between the nodes were equally taken into consideration.

### 3.10.3 Dataset

For our experiment, we have used a part of a data collection of a weighted network from our Forcoa.NET system. This weighted network is based on the DBLP dataset of publications from the field of computer science. These data contain highly relevant information about publication activity from the period of nearly fifty years and are freely available. Total number of authors was 1,060,175 with 6,450,138 edges. After we have performed a network denoising based on forgetting concept [76], the set used in experi-

---

5 http://www.forcoa.net
6 http://www.informatik.uni-trier.de/ ley/db/
7 http://dblp.uni-trier.de/xml/
ment contained 96,172 authors and 67,854 edges in total, however for local methods only the immediate surroundings of a selected author was necessary to examine.

3.10.4 Results

When starting from a particular single node, communities detected by each of the algorithms were always slightly different, for example see community around ‘Vaclav Snasel’ in Figure 3.18. Both algorithms may start also from an edge or a set of nodes. In this case, the main difference between the algorithms was that when starting from an edge or set of nodes that belonged to previously detected community, Algorithm 2 detected the same community, but Algorithm 1 detected a set of overlapping sub-communities and a nested set of sub-communities. We have listed those sub-communities in Table 3.4. From our experience we may corroborate, that in case of author ‘Vaclav Snasel’, the detected sub-communities correspond to research sub-teams.

3.11 DISCUSSION

In this chapter we described the dependency relationship between two nodes of a network. We understand dependency as a local non-symmetrical property of a pair of nodes. We used this dependency measure when proposing a new degree centrality measure that is well suited for weighted networks. Dependency centrality is computationally similar to classical degree centrality, however, it is able to capture more precisely the importance of a node in a network. The advantage of both degree and dependency centralities is that knowledge of the entire graph structure is not required, nevertheless,
### 3.11 Discussion

Table 3.4: Sub-communities detected around ‘Vaclav Snasel’.

<table>
<thead>
<tr>
<th>Sub-community</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1$</td>
<td>Vaclav Snasel, Pavel Kromer, Jiri Dvorsky, Katerina Slaninova, Pavla Drazdilova, Lukas Vojacek, Milos Kudelka, Zdenek Horak, Jan Martinovic, Ajith Abraham, Jan Platos, Eliska Ochodkova</td>
</tr>
<tr>
<td>$L_2$</td>
<td>Vaclav Snasel, Pavel Kromer, Jiri Dvorsky, Katerina Slaninova, Pavla Drazdilova, Lukas Vojacek, Jan Martinovic, Ajith Abraham, Jan Platos, Eliska Ochodkova</td>
</tr>
<tr>
<td>$L_3$</td>
<td>Vaclav Snasel, Jiri Dvorsky, Katerina Slaninova, Pavla Drazdilova, Lukas Vojacek, Jan Martinovic, Ajith Abraham, Eliska Ochodkova</td>
</tr>
<tr>
<td>$L_4$</td>
<td>Vaclav Snasel, Jiri Dvorsky, Jan Martinovic, Ajith Abraham, Eliska Ochodkova</td>
</tr>
<tr>
<td>$L_5$</td>
<td>Vaclav Snasel, Pavel Kromer, Ajith Abraham, Jan Platos</td>
</tr>
<tr>
<td>$L_6$</td>
<td>Vaclav Snasel, Milos Kudelka, Zdenek Horak</td>
</tr>
<tr>
<td>$L_7$, $L_8$, $L_9$</td>
<td>Vaclav Snasel, Hana Rezankova</td>
</tr>
</tbody>
</table>

Set relations:
- $L_4 \subseteq L_3 \subseteq L_2 \subseteq L_1$,
- $L_5 \subseteq L_2$, $L_6 \subseteq L_1$,
- $L_4 \cap L_5 = L_3 \cap L_5 = \{\text{Vaclav Snasel, Ajith Abraham}\}$

Dependency centrality naturally evaluates the local surroundings of a node. Thanks to this, dependency centrality is applicable in the analysis of large-scale weighted or unweighted complex networks.

Furthermore, we used the dependency measure for the transformation of an originally unweighted network to a weighted network. In the resulting network we interpret the edges as a dependency of a pair of nodes on each other. We may understand dependency also as a binary property. In such a case it is possible to detect strongly dependent or independent subgroups in a network.
This chapter comes from my work previously published in conference proceedings [P4, P5, P6, P7, P8]. The method for graph construction was used in article [J2].

4.1 INTRODUCTION

The main features of current real-world networks are their large sizes and structures, which show varying degrees of importance of the nodes in their surroundings. Local representativeness is a novel, simple and straightforward approach for the evaluation of the network’s nodes with a focus on local properties in their surroundings. The representatives are more important than others considering their neighborhoods.

4.2 PRELIMINARIES

In this section, we define the basic ideas behind the proposed representativeness measure. This measure should reflect the importance of objects according their local features. The locality is very important aspect of our approach and it is realized as a computation on neighborhood of each object in n-dimensional space.

In the context of this chapter, we assume that $D = (O, \delta, \varepsilon)$ is a dataset, where $O$ is a set of objects, $\delta: O \times O \rightarrow \mathbb{R}^+_0$ is a function that defines the distance between two objects, and $\varepsilon \in \mathbb{R}, \varepsilon > 0$ is a constant defining neighborhood of object $O_i \in O$. Objects $O_j \in O$ for which $\delta(O_i, O_j) \leq \varepsilon$ are called neighbors.

In the following text we use vector datasets, so the object is an n-dimensional vector. We use the standard Euclidean distance as a distance measure. The neighborhood of the object $O_i$ are all of its neighbors that lie in an n-dimensional sphere of radius $\varepsilon$ with the center being the object $O_i$.

**Definition 14.** (d-centrality): Let $d(O_i)$ be the number of neighbors of $O_i$. Then $d(O_i)$ is d-centrality of object $O_i$ in dataset $D$.

Objects for which $d(O_i) = 0$ are outliers. Unless otherwise stated, we assume that in the datasets we work with, are no outliers (or have been removed). The nearest neighbor of object $O_i \in O$ we call an object $O_j \in O$ which is closest to the object $O_i$. In case there are more such objects, object $O_j$ has more nearest neighbors. Now we define the function of local significance, which returns the number of neighbors for which the examined object is the nearest neighbor.
4.2 Preliminaries

Definition 15. (Local significance): Let \( s(O_i) \) be the number of neighbors of \( O_i \), for which object \( O_i \) is the nearest neighbor. Then \( s(O_i) \) is local significance of object \( O_i \) in dataset \( D \).

An object \( O_i \) is locally significant if \( s(O_i) > 0 \), otherwise it is locally insignificant. Next, we define a function of x-representativeness which for every object returns its importance based on its d-centrality and local significance.

Definition 16. (x-representativeness): Let \( O_i \in D \), \( d(O_i) > 0 \) and \( x \in \mathbb{R}, x > 1 \). Then \( r_x(O_i) \) is x-representativeness of object \( O_i \) in dataset \( D \) defined by Equation 4.1.

\[
r_x(O_i) = \frac{s(O_i)}{ \log_x(1 + d(O_i)) } \tag{4.1}
\]

An important parameter of the x-representativeness is the base of the logarithm \( x \). Informally speaking, \( x \) affects the degree of representativeness of the object in relation to the number of its neighbors.

Now define the x-representativeness base, which represents the threshold of x-representativeness of the object. Then we define x-representative object of the dataset \( D \).

Definition 17. (x-representativeness base): Let \( O_i \in O \) and \( x \in \mathbb{R}, x > 1 \) where \( r_x(O_i) = 1 \). Then \( x = b(O_i) \) is x-representativeness base of object \( O_i \) in dataset \( D \), defined by Equation 4.2.

\[
b(O_i) = (1 + d(O_i))^{\frac{1}{s(O_i)}} \tag{4.2}
\]

Definition 18. (x-representative): Let \( O_i \in O \) and \( x \in \mathbb{R}, x > 1 \) where \( r_x(O_i) \geq 1 \). Then object \( O_i \) is the x-representative of dataset \( D \).

The smaller the x-representativeness base of the object is, the higher its local importance is. Consequently, this means that with regard to the number of its neighbors the object is to a great extent, their nearest neighbor.

Locally insignificant objects do not have any x-representativeness base, i.e. there is no \( x \), for which locally insignificant objects would be x-representatives of dataset \( D \). Next, we define a x-representative sample of the dataset and its base. Sample is a set of objects which is a subset of \( O \) and for which following conditions hold.

Definition 19. (x-representative sample): Let \( x \in \mathbb{R}, x > 1 \). Then the set of all x-representative objects of dataset \( D \) is x-representative sample \( R_x \) (in short x-RSample) of dataset \( D \). It holds \( R_x \subseteq O \).

Definition 20. (Base of x-representative sample): Let \( R_x \) be the x-representative sample of dataset \( D \). The base \( B(R_x) \in \mathbb{R} \) of this sample is the smallest possible base of x-representative sample for which all objects from sample \( R_x \) are the x-representative objects of dataset \( D \).
As stated in the definition, the largest possible x-representative sample of dataset $D$ is a set $O$ without locally insignificant objects. The base of this sample is the largest base of x-representative samples. For two x-representativeness bases $x < y$ must hold $|R_x| \leq |R_y|$. But if it also holds that $x = B(R_x)$ and $y = B(R_y)$, then $|R_x| < |R_y|$. Total number of x-representative samples of the dataset is thus equal to the number of unique x-representativeness bases of the objects.

**Remark 3.** By selecting the base of x-RSample we choose how many local representatives of a dataset we want to work with.

### 4.3 Deterministic Sampling Based on Local Representativeness

#### 4.3.1 Motivation

In the area of big data, sampling may help facilitate knowledge discovery from large-scale datasets. Using analytical methods it is possible to find patterns and regularities in a sample that is significantly smaller than the original dataset. To be able to validate the observed patterns in the original data, it is necessary to have a representative sample which retains certain statistical properties.

Our approach to obtaining representative objects of a dataset is inspired by the method of finding nearest neighbors. The algorithm is based on the idea that objects which are the nearest neighbors of other objects are the important ones in a dataset. We use a very simple function for defining the representatives of the dataset. This function is based on the neighborhood analysis and depends on the distance between objects.

#### 4.3.2 Related work

Sampling [24, 77] has already been applied in many areas to various types of data. The goal is to reduce the original dataset to a more manageable size. For large n-dimensional dataset sampling methods have been developed in order to optimize data mining tasks [78, 79]. The approaches can be divided into two groups; unbiased and biased. In uniform random sampling, every data point has the same probability of being selected for the sample. This approach has been used extensively in database and data mining tasks [80, 81, 25]. Techniques have been developed to collect a sample in one single sequential pass through the file [82]. In the BIRCH [26] clustering algorithm, a random sample is selected as the initial step. In biased sampling, every data point has a different probability of being selected for the sample. This approach is suitable for clustering tasks where the dataset includes clusters of different sizes. In this case, density-biased sampling [27, 83] gives better results. Kollios et al. [84] also used density-biased sampling to speed up cluster and outlier detection. Kerdprasop et al. [85] proposed a modification of the k-means clustering algorithm where density-biased reservoir sampling is used.
The approach described in this chapter is similar to the biased methods.

4.3.3 Sampling algorithm: x-RSampling.

In this section, we define the basic ideas behind the proposed algorithm. It is based on
the selection of representatives, i.e., data objects with high representativeness. Simply stated, by representativeness we understand the extent to which an object is the nearest neighbor of other objects in the dataset.

Algorithm 6 describes obtaining the representative sample of a dataset. In principle, it should preserve the local structure of the data (e.g., clusters in a dataset).

```
input: dataset D = (O, δ, ε), logarithm base B
output: sample R
1 foreach object o ∈ O do
2     initialize the local significance s(o) to zero
3 end
4 foreach object o ∈ O do
5     foreach neighbor r of o do
6         if r is a nearest neighbor of o then
7             increase the local significance s(r)
8         endif
9     end
10 endforeach
11 foreach object o ∈ O do
12     if s(o) > 0 (o is locally significant) then
13         calculate b(o) (x-representative base of o, according to eq. 4.2)
14         if b(o) ≤ B (o is B-representative) then
15             add o to R
16         endif
17     endif
18 endforeach
```

**Algorithm 4: x-RSampling algorithm**

The complexity of the algorithm may be clearly extracted from the pseudo-code in Algorithm 6. If we suppose that the dataset D contains N objects and the average size of the neighborhood is M, then the complexity of the algorithm is $O(NM)$. Usually, we may assume that $M \ll N$ so the complexity is linear. This is done because of the locality of the algorithm. If we think more deeply about the algorithm, we see that the complexity is highly affected by the complexity of the neighborhood discovery. So we must use a certain data structure which enables fast neighborhood exploration. Many
such structures have been developed in the past, such as R-Tree and KD-Tree and their
variants, or when the data have a small dimension we may use Quadrant tree. These
structures allow us to find neighbors in constant or, in the worst case, logarithmic time,
so the efficiency of the algorithm is still very good.

4.4 LOCAL REPRESENTATIVENESS IN VECTOR DATA

4.4.1 Properties of x-RSampling algorithm

We compare the proposed algorithm with random sampling using dataset with two
clusters in 2D space. The clusters were generated using Normal distribution with the
following parameters: the first cluster contains 99000 objects, the mean is at (0, 0) and
the standard deviations are (20, 20) along the horizontal and vertical axes. The second
cluster contains 1000 objects with mean (50, 50) and standard deviations (2, 2). The
clusters are not proportional and the smaller one contains only 1% of objects of the
larger one. This should demonstrate the ability of the algorithm to preserve both clusters.
Figures 4.1a - 4.1c depict the sampling results of the proposed algorithm. The original
dataset is not visualized due to a lack of space, but may be seen as the light gray cluster.
As can be seen, the algorithm preserves both clusters for any sampling rate. Moreover,
the objects are more concentrated in the cluster centers. The setting of the algorithm
parameters is depicted in Table 4.1.

The result of the unbiased random sampling algorithm is depicted in Figures 4.2a -
4.2c. The comparison with the proposed algorithm shows that the proposed algorithm
(Figures 4.1) is able to sample the data similarly to the random method, but closer inspection
shows that the sample generated by the proposed algorithm is more concentrated
in the cluster centers. Many objects in the random sample are located in the outer area
of clusters and may be considered as noise, which may confuse further processing of the
data.

<table>
<thead>
<tr>
<th>Table 4.1: Two cluster Dataset Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log base</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>-</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

The samples in Figure 4.1 focused on mimicking the random samples with the em-
phasis on the cluster center. However, the different setting of the parameters may lead
to completely different results. Figure 4.3 contains two 4% samples generated by the
4.4 Local representativeness in vector data

Figure 4.1: Two-cluster dataset - x-RSampling algorithm

(a) Sample 20%  (b) Sample 5%  (c) Sample 1%

Figure 4.2: Two-cluster dataset - Random samples

(a) Sample 20%  (b) Sample 5%  (c) Sample 1%

proposed algorithm. The first figure (4.3a) depicts the result similar to random sampling, while the second figure (4.3b) depicts the sample with objects concentrated on the border of the clusters.

4.4.2 Experimental evaluation

The experiments with vector data were focused mainly on the ability of the proposed algorithm to reduce data while preserving important features such as clusters and local density. Before we present the experimental results we must define the pre-processing steps of the algorithm. Then we show results for the artificial and real-world datasets.
4.4 Local representativeness in vector data

Pre-processing of the datasets, algorithm setup and comparison methodology

As was mentioned in Section 6.4, we assume that the data do not contain any outliers. This is a very strict condition that is also difficult to define. In our approach, we detect outliers using Quartiles method [86] according to the distances between the object and its nearest neighbors. We remove an object from the dataset if its distance to its nearest neighbor is not in range \( Q_1 - k(Q_3 - Q_1), Q_3 + k(Q_3 - Q_1) \), where \( k \) is a coefficient that we set to 1.5. \( Q_1 \) and \( Q_3 \) are the First and Third Quartile of the distances between each object and its nearest neighbor. The search for nearest neighbors in high-dimensional data is a well-known problem without a general solution. The standard Euclidean Distance works well for lower-dimensional data.

When the outliers are removed from the datasets we must define the epsilon distance which we take as a maximum significant neighborhood of each object. Moreover, we figured out that the simple distance measurement should be very strict in cases when the difference between the distances of the examined object and its two neighbors are very small. Therefore we selected the normalization factor \( F \) that removes those small differences. Normalization factor \( F \) was always set to the \( Q_1 \) and \( \epsilon \) to the \( Q_3 \). This setting produced the best results in our experiments. Normalization factor \( F \) is used on computed distance \( d \) between two objects as described in Equation 4.3.

\[
d_{\text{normalized}} = F \ast \left\lfloor \frac{d}{F} \right\rfloor
\]

Figure 4.3: Two-cluster dataset - Different parameters of x-RSampling algorithm
Setting normalization factor $F$ to $Q_1$ means that 25 percent of the minimal distances defines the interval for which we do not distinguish the distance to the nearest neighbors. Setting $\varepsilon$ to $Q_3$ then means that points with the distance within the largest 25 percent do not belong to the neighborhood of examined points.

We considered two quantitative measures for the comparison of sample representativeness: d-centrality distribution and cluster assignment changes. The d-centrality distribution is a simple measure very similar to the degree centrality distribution in networks. We computed the number of neighbors of each object in an original dataset and in an x-representative sample and then we compared these distributions using well-known Kolmogorov-Smirnov test [87]. The second measure is based on the cluster centers assignments. We manually set the cluster centers and then we let the K-means algorithm put the center to the proper position. We did this for the original dataset as well as for all x-representative samples. Then we took the centers from each x-representative sample and computed the assignments of objects to clusters in the original dataset. Finally, we computed the amount of object changes in the cluster assignment with respect to the original dataset.

Remark 4. The manual initialization was necessary for the repeatability of the experiments and because the random initialization is not unambiguous. The manual cluster assignments were made according to the natural centers in a dataset.

Experimental datasets

We used two different datasets in our experiments: the first dataset "birch3" is artificial 2-dimensional and contains several more or less dense, random sized, clusters in random locations. The dataset has 100,000 objects. The second one "Czech Map" is real-world 2-dimensional dataset which contains all address points in the Czech Republic provided by the government. This data set contains 2,764,903 address points with coordinates in the S-JTSK coordinate system (S-JTSK is a coord. system which has been used since the beginning of the 20th century in Czechoslovakia and the length unit is approximately one meter). The points are distributed more densely in the area of large cities, e.g. the densest place is in the middle of the image where the capital city Prague is located, but very dense areas are also found in the north and south, although the most populated area of the Czech Republic is in the east, where the Moravian-Silesian region is located. The summary of the datasets with original object count, number of outliers and quartiles of nearest neighbor distances with normalization factor and $\varepsilon$ distances is depicted in Table 4.2 and the visualization of the original datasets is depicted in Figures 4.4 and 4.11.

Table 4.2: Dataset properties

<table>
<thead>
<tr>
<th></th>
<th>Birch3</th>
<th>Czech Map</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objects</strong></td>
<td>100,000</td>
<td>2,740,972</td>
</tr>
<tr>
<td>( Q_1 = F )</td>
<td>186.682</td>
<td>12.831</td>
</tr>
<tr>
<td>( Q_2 = \text{median} )</td>
<td>539.016</td>
<td>19.687</td>
</tr>
<tr>
<td>( Q_3 = \epsilon )</td>
<td>1108.697</td>
<td>28.529</td>
</tr>
<tr>
<td><strong>Outliers</strong></td>
<td>5,511</td>
<td>176,209</td>
</tr>
<tr>
<td><strong>Remaining objects</strong></td>
<td>94,489</td>
<td>2,564,763</td>
</tr>
</tbody>
</table>

**Experiment 1: Birch3 dataset**

The Birch3 dataset contains many clusters with different density. Therefore, it is clear that the proposed algorithm must deal with all these densities to maintain the representatives of each neighborhood. First, we compute all representativeness bases of the dataset and depict them in Figure 4.5.

![Birch3 dataset](image)

As may be seen, we are able to generate a representative sample which will contain up to 70 percent of objects, the remaining 30 percent contain only locally insignificant objects. Moreover, to distinguish between 0% and 60% of representative objects we need only logarithm bases between 1 and 2, but we are not able to generate an x-RSample with any specific number of objects. This is more precisely demonstrated in Table 4.3 that contains some logarithm bases and corresponding sizes of x-RSamples. The sample sizes decreases smoothly except for the jumps from 44.92 to 39.98, 29.54 to 13.30 and 10.98 to 6.42 percent. We are not able to get any sample size between the mentioned size jumps.
Table 4.3: Birch3 - Representativeness bases

<table>
<thead>
<tr>
<th>Log base</th>
<th>Objects</th>
<th>Size [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>94489</td>
<td>100,00%</td>
</tr>
<tr>
<td>651.0000</td>
<td>68195</td>
<td>72,17%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>001.7782</td>
<td>42440</td>
<td>44,92%</td>
</tr>
<tr>
<td>001.7313</td>
<td>37777</td>
<td>39,98%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>001.4420</td>
<td>27910</td>
<td>29,54%</td>
</tr>
<tr>
<td>001.4142</td>
<td>12564</td>
<td>13,30%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>001.3333</td>
<td>10378</td>
<td>10,98%</td>
</tr>
<tr>
<td>001.3159</td>
<td>6067</td>
<td>6,42%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>001.1096</td>
<td>941</td>
<td>1,00%</td>
</tr>
</tbody>
</table>

The results for the Kolmogorov-Smirnov test on Birch3 dataset are depicted in Figure 4.6a. The D-statistic for x-RSamples larger than 10% is very good, around 0.3. Smaller x-RSamples have worse values, but this is caused by the complexity of this dataset, where with less than 10% of objects it is very difficult to maintain neighborhoods. This will be clearly visible in results with clustering.

The last comparison was made using clustering assignments. The number of changes is depicted in Figure 4.6b. The results show that the number of objects that changed its cluster assignment in x-RSample is really low for samples bigger than 6%. The results
for smaller samples are unsatisfactory but it is because the smaller x-RSamples do not maintain several clusters at all.

The cluster assignments for four different sample sizes are depicted in Figures 4.7 to 4.10. Cluster centers are shown as red crosses.

Figure 4.7: Birch
log.base 1.31

Figure 4.8: Birch
log.base 1.41

Figure 4.9: Birch
log.base 1.44

Figure 4.10: Birch
log.base 651

Experiment 2: Czech Map dataset

The Czech Map dataset doesn’t contain any clear clusters at all. The objects are distributed according to the real urban construction and two main different situations exist. First one is the situations when the address points are concentrated into large cities such as Prague, Ostrava, Brno, Pilsen, Olomouc, Liberec, etc. The second one is the situation of small cities and villages where most address points are along the main roads. So the algorithm must preserve both situations, i.e. the overall density of smaller and larger groups of objects. The representativeness bases of this dataset are depicted in Figure 4.12.

Figure 4.11: Czech Map dataset

Figure 4.12: Czech Map Representativeness bases
4.4 Local Representativeness in Vector Data

Table 4.4: Czech Map - Representativeness bases

<table>
<thead>
<tr>
<th>Log base</th>
<th>Objects</th>
<th>Size [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>2564763</td>
<td>100,00%</td>
</tr>
<tr>
<td>90.0000</td>
<td>2164541</td>
<td>84,40%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>01.7772</td>
<td>1467098</td>
<td>57,20%</td>
</tr>
<tr>
<td>01.7294</td>
<td>1261103</td>
<td>49,17%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>01.4419</td>
<td>809874</td>
<td>31,58%</td>
</tr>
<tr>
<td>01.4142</td>
<td>383215</td>
<td>14,94%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>01.3478</td>
<td>296456</td>
<td>11,56%</td>
</tr>
<tr>
<td>01.3159</td>
<td>115080</td>
<td>4,49%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>01.2596</td>
<td>23621</td>
<td>0,92%</td>
</tr>
</tbody>
</table>

As may be seen, we are able to generate an x-RSample which will contain up to 80 percent of objects, the remaining 20 percent contain only locally insignificant objects. Similarly to the previous dataset, we are not able to generate x-RSample of any specific size. The logarithm bases contain many smaller or larger gaps (vertical lines in a plot). More details is depicted in Table 4.4.

![Figure 4.13: Czech Map - x-RSamples comparison](image)

(a) d-centrality statistics  
(b) Cluster assignment changes

Figure 4.13: Czech Map - x-RSamples comparison

The results for the Kolmogorov-Smirnov test on Czech Map dataset are depicted in Figure 4.13a. The D-statistic values are slightly higher than for the previous dataset but

42
this dataset is much more complex and it is not possible to maintain whole neighborhoods at all.

The last comparison was made using clustering assignments. The number of changes is depicted in Figure 4.13b. The results show that the number of objects that changed its cluster assignment in x-RSample is even lower than for the previous dataset. This measure is very low even for x-RSamples with one or two percent of size.

Figure 4.14: Czech Map - log.base 1.30

Figure 4.15: Czech Map - log.base 1.46

Figure 4.16: Czech Map - log.base 1.99

Figure 4.17: Czech Map - log.base 90

The cluster assignments for four different sample sizes are depicted in Figures 4.14 to 4.17. Cluster centers are again shown as red crosses.

4.5 Local representatives in weighted networks

Throughout this chapter we assume the existence of a social network, $G = (N, E)$, where $N$ is a set of nodes and $E$ is a set of undirected weighted edges. We will use a notation $d(N_i)$ for the degree centrality of the node $N_i \in N$. Nodes that have $d(N_i) = 0$ are outliers. Unless otherwise stated, we assume that in the network we work with, are no outliers (or have been removed). The nearest neighbor of a node $N_i \in N$ we call a node $N_j \in N$ with which the node $N_i \in N$ has its strongest edge $E_{ij} \in E$. When more such nodes exists, node $N_i$ has more nearest neighbors. We define the function of local sig-
nification, x-representativeness, x-representativeness base, x-representative and x-representative sample in the same manner as in section 4.2.

From the definitions it follows that the largest possible x-representative sample of network G is a network G without locally insignificant nodes. The base of this sample is the largest base of x-representative samples. For two x-representativeness bases \( x < y \) must hold \( |R_x| \leq |R_y| \). But if it also holds that \( x = B(R_x) \) and \( y = B(R_y) \), then \( |R_x| < |R_y| \). Total number of x-representative samples of the network is thus equal to the number of unique x-representativeness bases of the nodes.

The calculation of x-representativeness is based on two steps. The first is global ranking (we use the degree centrality measure). The second is a transformation of global ranking into local ranking (representativeness) while taking into account information about the node’s importance for its neighbors (local significance).

Using the base of representativeness (the base of x-RSample), we choose how many local representatives of a network we want to work with. As we will show in our experiments, from the global perspective, the value of x-representativeness does not prioritize any specific type of nodes or type of connection between them.

In the definitions we only work with undirected networks. If we use in-Degree instead of the degree, then our approach is also directly applicable to directed networks.

Illustrative example

In Figure 4.18(I) there is a small example network with fourteen nodes where node L is an outlier. The arrows indicate the relationship between the neighbors. E.g. node F is the nearest neighbor of nodes E, G, H, I and the nearest neighbors of node M are nodes I and N. Table 4.5 summarizes degree centrality \( d(N) \), local significance \( s(N) \) and x-representativeness base \( b(N) \) for all nodes from the example network.

<table>
<thead>
<tr>
<th>Node (N)</th>
<th>d(N)</th>
<th>s(N)</th>
<th>b(N)</th>
<th>Node (N)</th>
<th>d(N)</th>
<th>s(N)</th>
<th>b(N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>H</td>
<td>4</td>
<td>2</td>
<td>~2.24</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>I</td>
<td>4</td>
<td>2</td>
<td>~2.24</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>J</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>K</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>L</td>
<td>0</td>
<td>0</td>
<td>outlier</td>
</tr>
<tr>
<td>F</td>
<td>6</td>
<td>4</td>
<td>~1.63</td>
<td>M</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>N</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

From the Table 4.5 it is clear that nodes A and G are locally insignificant because they do not have any x-representativeness base. Their x-representativeness \( r_x(N) \) in the network is equal to zero for any value of x. It is also clear from the table, that nodes have
4.5 Local Representatives in Weighted Networks

![Example network with five x-RSamples](image)

Figure 4.18: Example network with five x-RSamples

A total of five unique x-representativeness bases ranging from $\sim 1.63$ to 5. Therefore we get five x-RSamples with those bases in total. They are all listed in Table 4.6 and shown in Figure 4.18(II-VI).

<table>
<thead>
<tr>
<th>Log base - $x$</th>
<th>x-RSample</th>
<th>Nodes %</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>B C D E F H I J K M N</td>
<td>78</td>
</tr>
<tr>
<td>3</td>
<td>C D E F H I J K M N</td>
<td>71</td>
</tr>
<tr>
<td>$\sim 2.24$</td>
<td>C D E F H I J K</td>
<td>57</td>
</tr>
<tr>
<td>2</td>
<td>C D E F J K</td>
<td>43</td>
</tr>
<tr>
<td>$\sim 1.63$</td>
<td>F</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 4.6: x-RSamples for the example network

4.5.1 Experimental evaluation

The goal of the experiments is the investigation of three different real-world networks. For all of them we follow the same procedure in several separate steps. The aim of the experiment is to show how x-RSamples of different sizes maintain a variety of topological properties of weighted networks.
In the initial step of the experiment we remove all outliers from the examined network. The next step is to find local significance for all nodes in the network. Then we work only with the x-representativeness of each node from Definition 16. The last step is to determine how many different x-representativeness bases exist for the examined network.

The first output of the experiment is the distribution of x-representativeness bases that is cumulatively displayed in the chart. From this chart we can see what percentage of the network’s nodes is locally significant and also how many nodes from the original network are in the x-RSample for the selected base $x$.

As a second output of the experiment, we present several x-RSamples for each network with different values of bases and thus different sizes. The results are summarized in two tables. The first table shows the relationship between the x-representativeness base and the size of x-RSample. The second table then summarizes the comparison of
cumulative distributions of selected five measured properties of the examined networks with all their x-RSamples. In network sampling, the quality of the produced samples is evaluated according to the sampling goals. In the state of the art paper on sub-graph sampling from Leskovec et al. [88] the authors measure several graph properties (nine for a ‘scale-down sampling goal’ and five for a ‘back-in-time sampling goal’). We selected two most commonly used from the ‘scale-down sampling goal’, they are degree distribution (DD) and distribution of local clustering coefficient (CC) [12], we added also their weighted variants (weighted degree centrality and weighted clustering coefficient as defined by Barrat et al. [31]). We also calculated the distribution of edge weights.

To compare the shape of those distributions, we calculated the Kolmogorov-Smirnov D-statistic [87] (used for this purpose also in [88]) which captures the largest gap between two cumulative distribution functions (CDF) and serves as a measure of distance between them.

We tested our algorithm on the three following datasets. All of them representing undirected weighted networks. The results show that the topological properties of x-RSamples are very well maintained when compared with the original network.

Astrocollab dataset

The first investigated dataset is Astrocollab [89], a co-authorship network between scientists posting preprints on the Astrophysics E-Print archive. It has 16,706 nodes and 121,251 edges (with a maximum weight of 16.5).

<table>
<thead>
<tr>
<th>Log base</th>
<th>Nodes</th>
<th>Edges</th>
<th>Nodes %</th>
<th>Edges %</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>16,706</td>
<td>121,251</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>60.0</td>
<td>12,542</td>
<td>105,672</td>
<td>75</td>
<td>87</td>
</tr>
<tr>
<td>1.8</td>
<td>5,476</td>
<td>51,444</td>
<td>33</td>
<td>42</td>
</tr>
<tr>
<td>1.7</td>
<td>4,471</td>
<td>46,575</td>
<td>27</td>
<td>38</td>
</tr>
<tr>
<td>1.6</td>
<td>3,934</td>
<td>41,477</td>
<td>23</td>
<td>34</td>
</tr>
<tr>
<td>1.5</td>
<td>3,093</td>
<td>35,353</td>
<td>18</td>
<td>29</td>
</tr>
<tr>
<td>1.4</td>
<td>2,253</td>
<td>28,136</td>
<td>13</td>
<td>23</td>
</tr>
</tbody>
</table>

After computing the x-representativeness bases, it is clear that around 25% of nodes in this network are locally insignificant, the resulting cumulative distribution of bases can be seen in Figure 4.19a. These distributions give a hint about which logarithm base x to select, when we want to get a representative subnet of a specific size. The minimum value of the base is 1.05 and would produce an x-RSample with one node. The maximum value of the base is 60 and would produce an x-RSample with 75% of
nodes. There are 1,379 different values of x-representativeness base in total, which is also a total number of x-RSamples that may be extracted.

Since we were looking for representative subnets of different sizes for each dataset, we selected six values of the x-representativeness base $x$ and extracted corresponding x-RSamples. For every obtained x-RSample, we give the summary of its size as a percentage against the original dataset, the base of the logarithm in function $r$, and some other characteristics, see Table 4.7.

Table 4.8 gives a summary of calculated KS-distances for the above mentioned distributions.

Table 4.8: D-statistic for Astrocollab

<table>
<thead>
<tr>
<th>Log base</th>
<th>DD</th>
<th>Wght DD</th>
<th>CC</th>
<th>Wght CC</th>
<th>Edge weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>60.0</td>
<td>0.0660</td>
<td>0.0743</td>
<td>0.0368</td>
<td>0.0433</td>
<td>0.0356</td>
</tr>
<tr>
<td>1.8</td>
<td>0.1644</td>
<td>0.0999</td>
<td>0.0358</td>
<td>0.0503</td>
<td>0.1685</td>
</tr>
<tr>
<td>1.7</td>
<td>0.2346</td>
<td>0.1392</td>
<td>0.0453</td>
<td>0.0743</td>
<td>0.1932</td>
</tr>
<tr>
<td>1.6</td>
<td>0.2437</td>
<td>0.1245</td>
<td>0.0551</td>
<td>0.0741</td>
<td>0.2205</td>
</tr>
<tr>
<td>1.5</td>
<td>0.2951</td>
<td>0.1407</td>
<td>0.0922</td>
<td>0.0966</td>
<td>0.2647</td>
</tr>
<tr>
<td>1.4</td>
<td>0.3569</td>
<td>0.1358</td>
<td>0.1353</td>
<td>0.1339</td>
<td>0.3300</td>
</tr>
</tbody>
</table>

Condmatcollab dataset

The second dataset is Condmatcollab, a collaboration network of scientists posting preprints on the condensed matter archive between January 1, 1995 and March 31, 2005 with weights assigned as described in [90]. It has 40,421 nodes and 175,693 edges (with a maximum weight of 46).

Table 4.9: Condmatcollab Dataset x-RSamples

<table>
<thead>
<tr>
<th>Log base</th>
<th>Nodes</th>
<th>Edges</th>
<th>Nodes %</th>
<th>Edges %</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>40,421</td>
<td>175,693</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>88.0</td>
<td>28,596</td>
<td>135,643</td>
<td>71</td>
<td>77</td>
</tr>
<tr>
<td>1.8</td>
<td>9,834</td>
<td>40,407</td>
<td>24</td>
<td>23</td>
</tr>
<tr>
<td>1.7</td>
<td>7,691</td>
<td>33,326</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>1.6</td>
<td>6,339</td>
<td>25,698</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>1.5</td>
<td>4,431</td>
<td>17,947</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>1.4</td>
<td>2,599</td>
<td>10,457</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

After computing the x-representativeness bases we can see that around 29% of nodes in this network are locally insignificant. The resulting cumulative distribution of bases
is in Figure 4.19b. The minimum value of the base is 1.06 and would produce an x-RSample with one node. The maximum value of the base is 88 and would produce an x-RSample with 71% of nodes. There are 1,268 different values of x-representativeness base in total.

The summary of extracted x-RSamples for the six selected values of the x-representativeness base $x$ is depicted in Table 4.9. There we extracted subnets with 6% to 71% of nodes.

Table 4.10 gives the summary of calculated KS-distances for the above mentioned distributions. The distance between the distributions is increasing slightly with further reduction of the original network, but even on the smallest x-RSample (here 6%) we get very good results.

### Table 4.10: D-statistic for Condmatcollab

<table>
<thead>
<tr>
<th>Log base</th>
<th>DD</th>
<th>Wght DD</th>
<th>CC</th>
<th>Wght CC</th>
<th>Edge weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>88.0</td>
<td>0.0638</td>
<td>0.0784</td>
<td>0.0366</td>
<td>0.0380</td>
<td>0.0374</td>
</tr>
<tr>
<td>1.8</td>
<td>0.0419</td>
<td>0.1313</td>
<td>0.1068</td>
<td>0.0969</td>
<td>0.0705</td>
</tr>
<tr>
<td>1.7</td>
<td>0.0948</td>
<td>0.1775</td>
<td>0.1319</td>
<td>0.1250</td>
<td>0.0886</td>
</tr>
<tr>
<td>1.6</td>
<td>0.0479</td>
<td>0.1618</td>
<td>0.1580</td>
<td>0.1438</td>
<td>0.0887</td>
</tr>
<tr>
<td>1.5</td>
<td>0.0677</td>
<td>0.1836</td>
<td>0.1961</td>
<td>0.1788</td>
<td>0.1069</td>
</tr>
<tr>
<td>1.4</td>
<td>0.0704</td>
<td>0.2137</td>
<td>0.2527</td>
<td>0.2233</td>
<td>0.1434</td>
</tr>
</tbody>
</table>

**DBLP dataset**

The last dataset² is a co-authorship network constructed from a DBLP³ dataset. Data was downloaded in April 2013 and preprocessed for the Forcoa.NET system. Edge weights are based on network evolution and a forgetting function that takes into account the frequency and regularity of publishing. For details see [57]. After the preprocessing, a total of 318,971 nodes (active authors) and 786,384 edges (with a maximum weight of 433) remained in the network.

After computing the x-representativeness bases, it is clear that around 27% of nodes in this network are locally insignificant. The resulting cumulative distribution of bases is in Figure 4.19c. The minimum value of the base is 1.04 and would produce an x-RSample with two nodes. The maximum value of the base is 35 and would produce an x-RSample with 73% of nodes. There are 1,557 different values of x-representativeness base in total.

The summary of extracted x-RSamples for the six selected values of the x-representativeness base $x$ is depicted in Table 4.11. There we extracted subnets with 7% to 73% of nodes.

---


3 [http://www.informatik.uni-trier.de/~ley/db/](http://www.informatik.uni-trier.de/~ley/db/)
4.5 Local Representatives in Weighted Networks

Table 4.11: DBLP Dataset x-RSamples

<table>
<thead>
<tr>
<th>Log base</th>
<th>Nodes</th>
<th>Edges</th>
<th>Nodes %</th>
<th>Edges %</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>318,971</td>
<td>786,384</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>35.0</td>
<td>232,238</td>
<td>622,768</td>
<td>73</td>
<td>79</td>
</tr>
<tr>
<td>1.8</td>
<td>108,128</td>
<td>276,483</td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td>1.7</td>
<td>76,720</td>
<td>227,500</td>
<td>24</td>
<td>29</td>
</tr>
<tr>
<td>1.6</td>
<td>69,333</td>
<td>199,621</td>
<td>22</td>
<td>25</td>
</tr>
<tr>
<td>1.5</td>
<td>45,504</td>
<td>148,465</td>
<td>14</td>
<td>19</td>
</tr>
<tr>
<td>1.4</td>
<td>23,553</td>
<td>94,194</td>
<td>7</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 4.12 gives the summary of calculated KS-distances for the above mentioned distributions.

Table 4.12: D-statistic for DBLP

<table>
<thead>
<tr>
<th>Log base</th>
<th>DD</th>
<th>Wght DD</th>
<th>CC</th>
<th>Wght CC</th>
<th>Edge weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>35.0</td>
<td>0.0241</td>
<td>0.0360</td>
<td>0.0046</td>
<td>0.0084</td>
<td>0.0159</td>
</tr>
<tr>
<td>1.8</td>
<td>0.0248</td>
<td>0.0852</td>
<td>0.0732</td>
<td>0.0773</td>
<td>0.0156</td>
</tr>
<tr>
<td>1.7</td>
<td>0.1302</td>
<td>0.1659</td>
<td>0.0465</td>
<td>0.0474</td>
<td>0.0209</td>
</tr>
<tr>
<td>1.6</td>
<td>0.1152</td>
<td>0.1439</td>
<td>0.0580</td>
<td>0.0595</td>
<td>0.0260</td>
</tr>
<tr>
<td>1.5</td>
<td>0.2039</td>
<td>0.2063</td>
<td>0.0506</td>
<td>0.0500</td>
<td>0.0437</td>
</tr>
<tr>
<td>1.4</td>
<td>0.2960</td>
<td>0.2734</td>
<td>0.0876</td>
<td>0.0861</td>
<td>0.0807</td>
</tr>
</tbody>
</table>

Figure 4.20: DBLP x-RSamples metrics
The cumulative degree and edge weight distributions of x-RSamples copy the distribution of the original network. See Fig. 4.20a for the weighted degree distribution and Fig. 4.20b for the edge weights distribution.

To show how the reduction of the network to the sample of representatives progresses, we extracted the sub-network which captures co-authors of Christos Faloutsos, see Figure 4.21. For the resulting x-RSamples from this part of network see Figures 4.22-4.24.

Graph construction is a known method of transferring the problem of classic vector data mining to network analysis. The advantage of networks is that the data are extended by links between certain (similar) pairs of data objects, so relationships in the data can then be visualized in a natural way. In this area, there are many algorithms, often with significantly different results. A common problem for all algorithms is to find
relationships in data so as to preserve the characteristics related to the internal structure
of the data.

Simply put, the methods of data mining analyze data arranged in a table whose rows
are data objects and columns are attributes (features). Typical tasks include, e.g., clus-
tering and classification. These two areas have different objectives, but both are based on
the assumption that there are internal data structures which can be utilized. These struc-
tures are then meant for solving clustering, classification, and other problems. While
examining structures, the distance (similarity) of individual data objects, in particular,
is examined. Assuming that we are working with numeric attributes, using e.g. the
Euclidean distance or Gaussian function, the distance or similarity, respectively, can be
precisely measured. Due to these measurable properties, we can apply many different
algorithms that reveal information about the structures mentioned above.

However, it is not easy to find the way to visualize such data and structures because
usually we work with more than three dimensions. Visualization is one of the essential
tools allowing for an easier understanding of the data. Graphs (networks) have the prop-
erty that, thanks to various layout algorithms, provide a clear visualization depicting the
natural structure of data, particularly for small networks (for large-scale networks with
higher computational costs). Besides visualization, networks also provide space for the
application of analytical methods which can describe other characteristics (node ranking,
local and global clustering coefficient, community structure, etc.).

The basic problem of graph construction is to find the parameters for conversion,
which retain the essential properties of transferred data. The nodes in the constructed
depict individual objects where interconnection between nodes can be done
graph represent various ways. The result of conversion should be a graph in which the clusters,
the nearest neighbors (in the sense of distance or similarity, respectively), outliers, etc.,
are preserved. One of the known approaches based on the nearest neighbor analysis
published Huttenhower et al. [91]. In this approach, in addition to graph construction,
the main objective is to find strongly interconnected clusters in the data. However,
the method assumes that the user must specify the number of nearest neighbors with
which the algorithm works. Methods using the principle based on the use of k
nearest neighbors are referred to as the k-NN networks and assume the k parameter to be a
previously known value.

In our method, we use the nearest neighbors in another way. We assume that data
objects (and consequently future graph nodes) have a different representativeness. The
representativeness is a local property based on the number of objects that are the nearest
neighbors of a selected node. We showed in section 4.5 that local representativeness
could be used for data reduction in which the structure and properties of the original
data are very well preserved.

In our approach we use representativeness in the construction of the graph so that
(1) we create edges between all pairs of nearest neighbors, and (2) we create additional
edges between the individual data objects in the number proportional to the representativeness of these objects. Representativeness of nodes in the constructed graph then corresponds, approximately, to the representativeness of objects in the data. This makes a natural graph representation of the original data, which preserves its local properties.

4.6.1 Related work

Graphs are powerful tools for data analysis, and graph-based algorithms are currently widely used in various fields. Construction of a graph from vector data represents a precursor of many tasks in machine learning, computer vision, and signal processing. Examples include semi-supervised learning (SSL) or spectral clustering. For graph-based SSL algorithms, a graph is a central object on which the subsequent learning takes place. The nodes of the graph represent data objects, and weighted edges express the relationships between the nodes.

For some application domains, the relationships between data are natural (e.g. social networks, collaboration networks, citation networks, the web). In this case, graph-based methods, such as methods of social network analysis (SNA) which use a graph as the basic representation of data, can be applied straightforwardly. However, most data mining tasks consider data to be independent and identically distributed (i.i.d.) variables so we do not have access to any explicit graph structure. Therefore, a graph of independent objects is often generated first, and then graph-based machine learning methods are used. Even though graph-based machine learning methods are currently very popular, little attention is paid to the impact of graph construction methods and their results [92] and there is not a general way to establish high-quality graphs [93]. As an example study of the impact of graph construction see Maier et al. [94], where the authors discuss the influence of graph construction on graph-based clustering measures such as the normalized cut. However, little attention is paid to analyzing the properties of the resulting graph using SNA methods.

From a wider perspective, graph construction techniques can be divided into the following two groups [95]:

1. **Task-dependent graph construction**: algorithms in this group use both labeled and unlabeled data to construct the graph. Labeled data can be used for adapting the graph to the task at hand. This group of algorithms receives less attention. An example is the semi-supervised metric learning algorithm IDML [96] which shows how the available label information can be used for graph construction in graph-based SSL. In particular, authors focus on learning a distance metric using available label information, which can then be used to set the edge weights on the constructed graph.
2. **Task-independent graph construction methods**: algorithms in this group do not use labeled data, so these methods can be considered unsupervised. Their advantage is that they are applicable to any graph-based learning tasks. The disadvantage is that they do not necessarily help subsequent learning tasks.

Graph construction, in general, consists of two consecutive steps. Once a neighborhood graph is constructed, the edge weight assignment step follows. The basic edge weighting methods include Gaussian function (Heat kernel) \[97\], inverse Euclidean distance, local reconstructive relationship \[98\] or e.g. sparse representation (SR) \[99\].

Construction of a graph in which both the selection of neighbors and edge weighting take place in one step, minimizing the weighted sum of the squared distance from each node to the weighted average of its neighbors, was proposed by \[100\] as a concept of hard and \(\alpha\)-soft graphs.

A graph-based algorithm to generate clusters of genes with similar expression profiles was proposed in \[91\]. The k-NN approach is used to construct the graph together with the detection of overlapping cliques of a given size (similarly to \[101\]).

The definitions of representativeness of the object, \(x\)-representative base etc. are in the section 4.2.

### 4.6.2 Proposed LRNet algorithm

The LRNet algorithm for the construction of the weighted graph utilizing local representativeness is composed of four steps:

1. Create a similarity matrix \(S\) of dataset \(D\).
2. Calculate the representativeness of all objects \(O_i\).
3. Create the set \(V\) of nodes of graph \(G\) so that node \(v_i\) of graph \(G\) represents object \(O_i\) of dataset \(D\).
4. Create the set of edges \(E\) of graph \(G\) so that \(E\) contains an edge \(e_{ij}\) between nodes \(v_i\) and \(v_j\) \((i \neq j)\) if \(O_j\) is the nearest neighbor of \(O_i\) or \(O_j\) is the representative neighbor of \(O_i\).

The time complexity of the algorithm is \(O(n^2)\), where \(n = |D|\). This is based on following facts:

- The calculation of the similarity matrix (the first step of the algorithm) has \(O(n^2)\) complexity (if we neglect the complexity of calculating the similarity of two objects).
• The calculation of representativeness, as mentioned above, has in the worst case also $O(n^2)$ complexity (the second step of the algorithm). The worst case is a situation where the similarity matrix is dense; this can be assumed for vector data. For data represented by a sparse similarity matrix, the complexity is $O(n)$; for details see [P7].

• The complexity of selecting representative neighbors of objects from dataset $D$ (the fourth step of the algorithm) depends on the average number of representative neighbors $k$ that we need to select from (in general) $n$ neighbors. In the graph construction task, we can assume that the constructed graph is sparse; then it holds that $k \ll n$ and complexity is $O(n^2)$.

In the proposed LRNet algorithm, each object $O_i$ of dataset $D$ contributes to the graph with the number of edges corresponding to its local representativeness. Regarding local properties and structures in the original data, we understand this principle as crucial to maintaining these properties.

4.6.3 Experimental evaluation

It is not possible to clearly state what properties the constructed graph should have. It depends on the task in which the graph would be utilized. In our approach, we assume that the constructed graph should be designed to best match the converted data. Moreover, because the data are an image of reality, the constructed graph should reflect this reality. Therefore, to assess graph properties, we use methods of social network analysis. The results should then lead to the properties observed in real-world networks.

For comparison, we chose the often used k-NN network method [91]. This method requires a pre-selected number of nearest neighbors $k$. For all of the analyzed datasets, we chose $k = \sqrt{N}$ where $N$ is the number of objects in the dataset (SqrtNN). For evaluation, we selected three datasets meant for classification. The reason for the analysis of labeled data is to assess how the compared methods reflect known information about classes in the data. The chosen datasets are ‘Cortex Nuclear’ [102], ‘Ecoli’ [103] and ‘Seeds’ [104]. For all of the datasets, rescaling the range of features to scale the range in $[0, 1]$ and Gaussian function as a similarity measure were used.

The key properties of real-world networks are so-called small-world (small average shortest path length and diameter of the graph), scale-free structure (power-law degree distribution), high clustering coefficient and community structure. Another property is assortativity, which has positive values for social networks but negative values for e.g. biological or technological networks (see Newman [105]). Table 6.2 shows the properties of graphs constructed from analyzed datasets. The measured properties include the number of nodes $n$ and edges $m$, average degree $<k>$, average shortest path length $<l>$,
4.6 Graph Construction Based on Local Representativeness

diameter $l_{\text{max}}$, average clustering coefficient $CC$, assortativity $r$, number of communities $com_L$ detected by Louvain [106] algorithm, and the corresponding modularity $Q_L$.

<table>
<thead>
<tr>
<th>Graph</th>
<th>$n$</th>
<th>$m$</th>
<th>$&lt;k&gt;$</th>
<th>$&lt;l&gt;$</th>
<th>$l_{\text{max}}$</th>
<th>$CC$</th>
<th>$r$</th>
<th>$com_L$</th>
<th>$Q_L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cortex: LRNet</td>
<td>1080</td>
<td>10648</td>
<td>19.719</td>
<td>3.212</td>
<td>7</td>
<td>0.519</td>
<td>-0.212</td>
<td>8</td>
<td>0.607</td>
</tr>
<tr>
<td>Cortex: SqrtNN</td>
<td>1080</td>
<td>11934</td>
<td>22.162</td>
<td>4.407</td>
<td>10</td>
<td>0.528</td>
<td>0.476</td>
<td>10</td>
<td>0.741</td>
</tr>
<tr>
<td>Ecoli: LRNet</td>
<td>336</td>
<td>2214</td>
<td>13.179</td>
<td>3.519</td>
<td>9</td>
<td>0.599</td>
<td>-0.352</td>
<td>5</td>
<td>0.624</td>
</tr>
<tr>
<td>Ecoli: SqrtNN</td>
<td>336</td>
<td>1986</td>
<td>11.857</td>
<td>4.834</td>
<td>12</td>
<td>0.497</td>
<td>0.521</td>
<td>9</td>
<td>0.703</td>
</tr>
<tr>
<td>Seeds: LRNet</td>
<td>210</td>
<td>997</td>
<td>9.495</td>
<td>3.472</td>
<td>7</td>
<td>0.662</td>
<td>-0.397</td>
<td>5</td>
<td>0.635</td>
</tr>
<tr>
<td>Seeds: SqrtNN</td>
<td>210</td>
<td>1040</td>
<td>9.905</td>
<td>5.207</td>
<td>13</td>
<td>0.509</td>
<td>0.484</td>
<td>7</td>
<td>0.707</td>
</tr>
</tbody>
</table>

It is evident from Table 6.2 that the pairs of graphs constructed by different methods have a similar number of edges, and therefore also an average degree and density. However, graphs constructed by the k-NN algorithm, unlike graphs constructed by LRNet, do not have some properties known for real-world networks. This is, in particular, longer average shortest path, large diameter, and unnaturally high assortativity. Another important property observed in real-world networks, that k-NN graphs do not have, is scale-free structure. The consequence of scale-freeness is so-called hubs, which are nodes with a very high degree, and also the fact that most of the nodes in real-world networks have a low degree. Degree distribution for each of the graphs on a log-log scale is for the LRNet method in Figure 4.25 and for the SqrtNN method in Figure 4.26. In graphs constructed by the SqrtNN method, most nodes have a higher degree and hubs are missing. Also, only a few nodes have a lower degree. This characteristic is also clearly documented by constructed graphs in Figures 4.27 and 4.28. The size of the nodes in those graphs corresponds to their degree; colors represent classes (labels).

![Graphs](image_url)

Figure 4.25: Degree distribution, LRNet
4.6 Graph Construction Based on Local Representativeness

![Graphs](image)

(a) Cortex: SqrtNN  (b) Ecoli: SqrtNN  (c) Seeds: SqrtNN

Figure 4.26: Degree distribution, SqrtNN

### Table 4.14: Classification accuracy of constructed graphs

<table>
<thead>
<tr>
<th>Graph</th>
<th>Weighted Precision</th>
<th>Precision by Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cortex: LRNet</td>
<td>0.607</td>
<td>0.870</td>
</tr>
<tr>
<td>Cortex: SqrtNN</td>
<td>0.643</td>
<td>0.887</td>
</tr>
<tr>
<td>Ecoli: LRNet</td>
<td>0.764</td>
<td>0.842</td>
</tr>
<tr>
<td>Ecoli: SqrtNN</td>
<td>0.777</td>
<td>0.869</td>
</tr>
<tr>
<td>Seeds: LRNet</td>
<td>0.872</td>
<td>0.914</td>
</tr>
<tr>
<td>Seeds: SqrtNN</td>
<td>0.891</td>
<td>0.929</td>
</tr>
</tbody>
</table>

#### 4.6.4 Analysis using classes

All three pairs of analyzed graphs are constructed from labeled data. This allows us to measure how precisely the constructed graphs connect nodes belonging to the same class. In Table 4.14, three different values are calculated. The first is the average weighted precision which is the average value of

\[
w = \frac{w_{pos}}{w_{all}} \tag{4.4}
\]

for all nodes of the graph, where \(w_{pos}\) is the sum of edge weights with neighbors of the same class and \(w_{all}\) is the sum of edge weights to all neighbors. The second value is the average precision, which is the average value of \(p\). This value \(p = 1\) for the selected node if the sum of the of edge weights with neighbors in the same class is higher than the sum of edge weights with neighbors in each of the other classes. Otherwise, \(p = 0\). The third value is modularity \(Q\) (Blondel et al. [106]) by classes defined as follows:
4.6 Graph Construction Based on Local Representativeness

\[ Q = \frac{1}{2m} \sum_{ij} \left[ w_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j), \]  

(4.5)

where \( w_{ij} \) is the edge weight between nodes \( v_i \) and \( v_j \), \( k_i \) and \( k_j \) are the sums of the weights of the edges between nodes \( v_i \) and \( v_j \), \( m \) is the sum of all of the edge weights in the graph, \( c_i \) and \( c_j \) are classes of the nodes \( v_i \) and \( v_j \), and \( \delta \) is Kronecker delta.

Furthermore, Figures 4.29 and 4.30 show ROC (receiver operating characteristic) curves with calculated values of AUC (area under the curve) describing the accuracy of as-
signing objects to each class. ROC curves are calculated from the values of weighted precision $w$ of individual nodes of constructed graphs.

The results show that the accuracy of connecting the nodes to the nodes of the same class for the two methods differ a little. The advantage of the LRNet method is, however, that for the construction of the graph, no parameter has to be selected, which is a required step for the k-NN method.
In this chapter we presented our work in the field of sampling large-scale data. The approach is based on finding representatives in the input dataset. Measurement of representativeness is done by an analysis of the local properties and nearest neighbors. We consider the key features of the method to be its general applicability, natural scalability and flexibility. We compared our algorithm with the unbiased random sampling method and performed tests on three different datasets. The results show that the algorithm is able to efficiently reduce the data using local information only.

Next we presented a method for assessing the representativeness of nodes in weighted networks with respect to their surroundings. This method is based on the assumption that those nodes that are the nearest neighbors of a large number of their adjacent nodes are important. This method is controlled by a single parameter, which determines the...
value of the node’s representativeness. For locally significant nodes, a definite representativeness base exists. This base was used in experiments to find differently sized samples of representatives from the original network. In experiments we worked with three real-world weighted undirected networks. We investigated the properties of samples of representatives using a statistical approach. The application of our method generates reduced sub-networks that preserve the topological properties of the original network very well.

Finally based on local representativeness measure we developed a method for graph construction. LR-Net does not use any parameters for the construction and ensures a very good preservation of the internal data structure. In our experiments we used methods of social network analysis and showed that the result of the application of the LRNet method on real biological data is a graph with properties observed in real-world networks. Comparing with the most commonly used k-NN method, which is based on estimating the number of nearest neighbors needed to construct the graph, we showed on labeled data that the resulting graphs interconnect nodes belonging to the same class with comparable accuracy.
5

CO-AUTHORSHIP NETWORKS - EXPERIMENTS

This chapter comes from my work previously published in conference proceedings \([P9, P10]\).

5.1 USING SELF-ORGANIZING MAPS FOR IDENTIFICATION OF ROLES

5.1.1 Introduction

The idea of describing the roles in social networks through the structural properties was first formulated in \([107]\). Two nodes of the network share the same role if they are linked with other nodes in the same manner. This is related to the local view of the network because the role is dependent on the community structure of the network. In today’s very large networks with many millions of nodes the role detection (role extraction) is one of the tools that help us to understand the characteristics of the network. Both the local and global perspective is interesting. However, networks need not to be seen only through their structure.

The behavior of participants over time is projected also to other properties that are not always just structural. For example online social networks group people with the same interests but different pattern of behavior. \([108]\) describes a life cycle of online community with the detailed characterization of user behavior in several roles (visitor, novice, regular, leader, elder). In this case the roles are also projected into the structure of the network, although this structure changes over time. Both points of view on the roles, just the structural and the behavioral one, may serve as a starting point to the role detection and extraction. In the first case we do not know beforehand what roles (structural patterns) are present in the network. We are in the area of pattern recognition. In the second case we search for predefined and clearly described roles, which is close to the problem of pattern matching.

Our goal is the automatic extraction of roles for which we selected a co-authorship network as input data. However, our approach is universal because it works only with structural properties of networks. To some extent we work with a deeper information about the behavior of nodes. The co-authorship and publication behavior of participants is projected into the weights of network nodes and relations. Additionally, these weights change over time. Therefore the input co-authorship network we use in our experiments reflects the behavior of participants at a particular moment in time. Our results then
show which roles are present in the network at a selected time (the global view) and how they affect the neighborhood of nodes (the local view).

5.1.2 Related work

Analyzing the properties of nodes, such as node degree and other centrality measures, has long been the interest of SNA. The goal is the discovery of various categories of nodes. Usually the idea behind roles is that participants with similar roles show similar link structure in the network. In [109] a method for finding multiple role labels is presented. They used stochastic equivalence for role definition.

Community based roles are proposed in [110]. In this case roles are predefined and in the immediate neighborhood of a node a community metric is calculated. Roles are then assigned based on relative degree and structure of links in and out of the communities around the node. Similar approach with predefined roles is presented in [111]. Here the community-oriented roles are called bridges, gateways and hubs. Roles are assigned to a node based on the relationship between its neighbors. Our understanding of roles is basically the same, although in our approach the roles are not predefined but discovered. In [112] roles are also automatically discovered through the process of unsupervised learning and a RolX approach is presented.

In [113] the Author-Recipient-Topic model is presented. Participant’s role is assigned based on clustering of topic similarity, so each role is associated with a set of topics.

A conceptual definition of roles from a social point of view is presented in [114]. They use the term social role and describe measures and analysis techniques for the identification of such roles.

5.1.3 The self-organizing map

Self-organizing map (SOM) is a type of neural-network described by Tuevo Kohonen [115], that provides a mapping from the high-dimensional data onto (usually) two dimensional grid of neurons. Using unsupervised learning process, SOMs are non-parametric, computationally efficient approach for data exploration. SOMs partition the input vectors into sets that share a common distance (e.g. Euclidean). Fully trained SOM network can be used for pattern classification. The SOMs have been previously used in many papers and applications for exploratory data analysis, pattern recognition or image analysis.

We provide a brief overview of the SOM, for more complete description see [116]. The SOM network consists of a set of neurons organized into a grid (typically rectangular or hexagonal). Each neuron has a set of input weights. The weights are usually represented by a real vector. In the first phase, these vectors are initialized to random values. During the second phase particular vectors from the training set are presented to the network.
For each vector, the nearest neuron (according to the distance between neuron weights and the training vector) is considered as a winner. The weights of the neuron (including the weights of neurons in neighborhood) are shifted by a factor towards the presented vector. Size of the neighborhood decreases over time as well as the shift (learning) factor. This leads to stabilization of the network. The shifting of the neurons can be mathematically expressed as follows:

\[ W_v(t + 1) = W_v(t) + \Theta(u, v, t) \alpha(t)(X - W_v(t)), \]

where \( W_v \) is the weight vector of the SOM network node \( v \) in the iteration \( t \), \( \Theta(u, v, t) \) is the neighborhood function of winner node \( u \), current node \( v \) and current iteration \( t \), \( \alpha(t) \) is the shift factor function and \( X \) is the particular input vector from training set. In each step we can calculate the particular error of vector representation (distance between presented vector and the winner node) as well as the total error of the iteration (sum of particular errors for all vector in the training set).

### Methodology

![Diagram](image)

**Figure 5.1: Role extraction process**

Figure 5.1 shows the sequence of steps in our approach. It contains data preprocessing, training of SOM network and final role extraction.

#### Data preprocessing

In the preprocessing phase we need to convert the input network into the form of individual input vectors based on nodes. In the input vector for one node we need to capture node’s properties as well as the neighborhood of that node. There are several possible ways of composing the vector taking into account degree of a node and it’s neighbors, weights of relations, weights of nodes, etc.

The vector construction is heavily based on the understanding of the term *role* and on the type of data available. When dealing with social networks, we can discuss several basic network types:

- **Unweighted networks** - The role of the node is usually characterized by the number of links to surrounding nodes and the types of these nodes.
5.1 USING SELF-ORGANIZING MAPS FOR IDENTIFICATION OF ROLES

- Weighted networks - Similarly to the unweighted case, the number and type of surrounding nodes is relevant to the role definition. But also the weight of the node itself and weight of the links should be taken into account.

- Networks with time aspects - Evolving networks contain much more information than their static counterparts. Usually we accept some sort of simplification when modeling the role in this type of networks.

Precise description of data preprocessing is presented along with the experiment description. In general the goal of data preprocessing phase is to transform the input data, in our case either weighted or unweighted network of nodes, into numerical vectors.

**Remark** We can assign a weight to the particular components of the vector in order to prioritize different aspects. This issue may be addressed in the preprocessing phase or during the training of the SOM network.

**SOM training**

Once the preprocessing is finished, we can train the SOM network using input vectors. The idea behind this procedure is that particular nodes are described as vectors. These vectors are presented to the SOM network and the SOM network adapts to them while maintaining repetitive patterns.

The parameters of the SOM network are crucial for the whole process. We use hexagonal neighborhood as it provides more natural results contrary to the rectangular grid. Classical Euclidean distance is used. Size of the SOM network should be chosen according to the expected outcomes. In accordance with the principles of granularity, the SOM network may provide coarser or finer view of the roles within the dataset. The granularity may be easily controlled using the size of the SOM network being used. Larger network produces detailed exploitation of the roles as well as lower total error of the representation, but may also provide too detailed results without adequate generalization.

Therefore the configuration of the network depends on the context and we describe it in the further text along with particular experiments.

**Role extraction**

Once the SOM network is trained (the total error stabilizes or predefined number of iterations is reached) we can explore the particular nodes of the SOM network. Each node represents a group of original input vectors (node covers these input vectors) and the weights of the node correspond to a point in the original space. That is enough to describe a role. But if we consider every node of the SOM vector as a separate role, we will always get $m \times n$ roles from the SOM network of dimension $m \times n$. From the nature of the SOM network and the input data, these roles will be similar in part. An interesting task is the selection of representative roles from the SOM network nodes.
5.1 Using Self-Organizing Maps for Identification of Roles

There may be a lot of approaches to select a suitable subset of roles from the candidate set (such as top-n with highest coverage, hierarchical clustering, etc.). In our approach we detect local maxima (w.r.t. number of covered input vectors). These local maxima represent the extracted roles. Given that every such node has weights that can still be viewed as the points in the original space, these roles have natural interpretation.

Proof of concept

In order to proof the viability of our approach, we analyze the well known Dolphin dataset [42], which contains a social network of 62 bottlenose dolphins. The social network is based on the dolphins observations and captures the interaction between particular dolphins (dolphins were seen together). This network is depicted on Figure 5.2.

We would like to identify roles in this network based on the basic structural properties of the nodes. Therefore we construct a two dimensional vector for each dolphin. First component of the vector is the number of links between particular dolphin and it’s surroundings. Second component is the average number of links of surrounding dolphins. Created vectors have been normalized using linear transformation to $<0; 1>$.

We have created a self-organizing map of dimension $5 \times 5$ and trained this network using constructed vectors. The visualization of the network can be seen on Figure 5.3. The visualization may be explained as follows: Hexagons correspond to particular neurons of the neural network (i.e. self-organizing map). Each neuron acts as a representant for a group of input vectors (i.e. dolphins). The darkness of the hexagon’s border illustrates the size of the group covered by the neuron. Using above presented role extraction method we have selected four neurons as the role representants. These neurons are marked by black dots in the visualization. For the visualization purposes we have assigned different colors to extracted roles and filled particular neurons with the color of the nearest role.

A different view on the role clustering and extraction process may be seen on the Figure 5.4. This figure contains all dolphins placed in the scatter plot space using the coordinates of their vector (blue cross). Therefore the coordinates of the dolphins are determined by the number of the dolphin’s links and the average number of links in the dolphin’s surroundings. The coordinates of the neurons (based on their weights) of the SOM (trained by the vectors of dolphins) are depicted by violet triangles and red squares. Using the role extraction process we have selected four neurons and considered them as roles (red squares).

Strictly speaking, the process has identified four roles. The first role is represented by neuron in the network with weights corresponding to coordinates $(0, 0.08; 0.88)$. This neuron directly corresponds to five dolphins (represents a winner cell for their input vector). Interpretation of the role indicates that it covers dolphins with very low number of links in the network, but those links lead to dolphins that are connected to many other dolphins. Second role covers also five dolphins and has coordinates $(0.77; 0.57)$.
which means that it is representing the dolphins with large number of social links but connected only to relatively separated dolphins. Third role located at coordinates \((0,25;0,55)\) covers four dolphins with relatively small number of links also to relatively separated dolphins. Last role with coordinates \((0,52;0,94)\) covers four dolphins with ordinary number of links, but these links lead to very connected dolphins.

Rest of the dolphins can be easily classified by the distance between their coordinates and coordinates of particular roles. Using this approach the dolphin may naturally contain multiple roles up to a certain degree. Illustrations of the surroundings of nodes are postponed to the next section.

5.1.5 Experiment

In our experiment we have used preprocessed DBLP dataset\(^1\). This dataset contains authors from the field of computer science and their publications. Based on the co-authorship of authors we have constructed a social network of authors interactions. This network can be freely browsed using our project Forcoa.NET\(^2\). The DBLP dataset has been preprocessed using forgetting function (a natural analogy of the sliding window) described in [76] in order to obtain a weighted social network of links between authors. Both authors and their links are weighted by their stability. Stability has a mathematical background, simply said it means that more stable authors (authors that publish regularly) have higher weight. Also when two authors publish regularly together, the weight of the link between them has higher weight.

\(^1\) [http://www.informatik.uni-trier.de/~ley/db/]
\(^2\) [http://www.forcoa.net]
5.1 Using Self-Organizing Maps for Identification of Roles

Vector construction

For this experiment we have constructed vectors describing particular authors from the DBLP dataset. The dataset contained 313,408 active authors. In order to provide more complex view of the authors in the network we have constructed larger vector than in the previous section. The vector contained nine components. The first component denoted the weight of the node (i.e. stability of the author). Then we have divided the links of particular author into four intervals according to the stability of the link. Number of links in given interval represented next four components of the vector. The same process was performed using the stability of co-authors, which led to the remaining four components of the vector. Roughly speaking, each vector described one particular author using the number of his/her links, stability of these links and the number and stability of his/her co-authors.

Network training

For the training of the network we have created self-organizing map of dimension 8x8. The training utilized linear decrease of both the node surroundings size and learning factor. The network training required a couple of hours on an ordinary PC and 860 training iterations until the total error of the network has been stabilized. Using this

---

3 Available at [http://www.forcoa.net/resources/cason2013](http://www.forcoa.net/resources/cason2013)
5.1 Using Self-Organizing Maps for Identification of Roles

process we have obtained a self-organizing map of the authors which is depicted on Figure 5.5. The visualization is constructed in the same way as in the previous section.

Role extraction

Using the role extraction process we have identified 10 roles in the network. The coordinates of particular roles (weights of corresponding neurons) are presented in Table 5.1 with the number of authors directly covered by particular roles. Many weights were relatively small, therefore led to 0,00 representation in the table. As well as in the previous section, the roles of all authors (not only authors covered by the roles directly) can be easily stated by calculating the distance between the coordinates of their vector and the coordinates of particular roles.

<table>
<thead>
<tr>
<th>authors</th>
<th>i0</th>
<th>i1</th>
<th>i2</th>
<th>i3</th>
<th>i4</th>
<th>i5</th>
<th>i6</th>
<th>i7</th>
<th>i8</th>
</tr>
</thead>
<tbody>
<tr>
<td>70825</td>
<td>0,02</td>
<td>0,01</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,01</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>47919</td>
<td>0,01</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>30256</td>
<td>0,01</td>
<td>0,01</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,01</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>16232</td>
<td>0,01</td>
<td>0,01</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,01</td>
<td>0,03</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>6480</td>
<td>0,07</td>
<td>0,01</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,01</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>4136</td>
<td>0,01</td>
<td>0,04</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,04</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>4116</td>
<td>0,02</td>
<td>0,03</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,03</td>
<td>0,03</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>4090</td>
<td>0,15</td>
<td>0,01</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,01</td>
<td>0,01</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>823</td>
<td>0,11</td>
<td>0,03</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,02</td>
<td>0,03</td>
<td>0,17</td>
<td>0,00</td>
</tr>
<tr>
<td>360</td>
<td>0,53</td>
<td>0,02</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,02</td>
<td>0,03</td>
<td>0,00</td>
<td>0,00</td>
</tr>
</tbody>
</table>

Benefit of this approach is that the coordinates of roles are located in the original vector space of authors and therefore offers their direct interpretation. This interpretation can be easily confronted with reality when we take random authors covered by particular roles and visualize their surroundings on Figures 5.7–5.16 (weight of the node is denoted by the size of the circle and weight of the edge is denoted by the thickness of corresponding line).

The interpretation of first role’s vector corresponds to authors having larger than minimal stability with a small number of links to other non-trivial (w.r.t. stability) authors. Figure 5.7 confirms this interpretation for two random authors. According to the vector, authors with second role will be probably less stable and have a smaller number of links (see Figure 5.8 for confrontation). Authors in the third role will be probably somewhere in between these two roles (Figure 5.9). Fourth role is interesting since the authors should have links to more stable authors (Figure 5.10). Fifth role is very different since it represents stable authors with a small number of links to less stable authors.
5.1 Using Self-Organizing Maps for Identification of Roles

(Figure 5.11). Sixth role represents less stable authors with a larger number of links to other less stable authors (Figure 5.12). Seventh role covers more stable authors with a larger number of links to both unstable and stable authors (Figure 5.13). Eighth role describes very stable authors with a small number of links to both unstable and stable authors. Ninth role covers stable authors with a number of links to other – especially very stable – authors (Figure 5.15). The last role represents the most stable authors with a number of links to both unstable and stable authors.

Remark 5. A significant portion of roles extracted in this experiment is devoted to authors with low stability. Therefore the behavior of the self-organizing map is natural from this point of view. This feature can be easily verified using the graph of stability distribution among the authors\(^4\) which follows the power law distribution.

<table>
<thead>
<tr>
<th>authors</th>
<th>i0</th>
<th>i1</th>
<th>i2</th>
<th>i3</th>
<th>i4</th>
<th>i5</th>
<th>i6</th>
<th>i7</th>
<th>i8</th>
</tr>
</thead>
<tbody>
<tr>
<td>101415</td>
<td>0,01</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>44139</td>
<td>0,01</td>
<td>0,02</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>13745</td>
<td>0,17</td>
<td>0,02</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,02</td>
<td>0,02</td>
<td>0,00</td>
<td>0,00</td>
</tr>
</tbody>
</table>

\(^4\) Available on http://www.forcoa.net
5.1 Using self-organizing maps for identification of roles

Network size

Figure 5.6 depicts the outcomes of the same experiment performed using smaller 5x5 network. Only three roles (see Table 5.2) have been extracted. It can be easily seen, that lowering the dimension of the network caused only certain roles to be extracted. Comparing these roles and their coordinates to the roles extracted from the larger 8x8 network indicates, that the reduction process is natural and that the smaller map still covers the most important properties of the network. Detailed exploration of neuron weights revealed, that many roles identified in the larger network only are present also in the smaller network. The difference is that they have not been extracted as separate roles using the presented local maximum approach.
5.2 EVOLUTION OF AUTHOR’S TOPIC

5.2.1 Introduction

Publication activities make up a large, complex and steadily growing social network. In such a network, the evolution and dynamics contain much hidden information and implicit knowledge. In co-authorship networks ties among authors indicate common professional interests. Those interests usually change over time following trends as the knowledge grows.

In this section, we study the evolution of topics of an individual author. As a topic we understand a set of keywords used in paper titles. If an author uses a keyword (or a phrase) in paper titles frequently and periodically, we can trace a history of this usage. However, there is a lot of noise; e.g. if an author used a keyword only once in the history, we cannot assume the keyword is a descriptor of the author’s research (there are hundreds keywords and short phrases for some authors).

Therefore, we apply the concept of forgetting [76], which enables us to keep only the essential features of the author’s own research. In this approach we measure stability which describes the dynamics of keyword evolution. The stability measure is based on a general assumption that authors tend to repeat successful activities.

Remark 6. All the procedures described in this chapter are implemented in our online Forcoa.NET\textsuperscript{5} system. This system is oriented on social network analysis, especially on

\textsuperscript{5} http://www.forcoa.net
evolutionary aspects of human behavior. The system core was designed to be open, therefore it can be used with different types of data. This system was practically used for analysis of people’s behavior in administrative department or for analysis of data from Learning Management System.

5.2.2 Related work

Topic extraction

Because DBLP records contain only the publication titles, several approaches were taken to extract the topics. For DBConnect project topics are extracted only from the paper titles using method based on keyword frequency and a list of stopwords [117]. Shubhankar et al. [118] identified the topics by forming closed frequent keyword-sets derived from phrases present in the titles and assigned score to each topic using citation edges of papers containing that topic. FacetedDBLP includes topic facet based on GrowBag graphs [119] using DBLP collection enhanced by abstracts and a set of keywords.

Qin et al. [120] focuses on keyword queries over relational databases, their applications, processing and effectivity. This paper presents results over the DBLP dataset concerning authors, the keywords they use and scientific conferences.

The extraction of topics is also performed by Expert Finder applications working with topic-based model [121, 122]. Pathak et al. [123] presents a model for community extraction based on both link and topic information.

Evolution of topics

In [124, 125] were proposed probabilistic models for evolution of topics in social communities. He and Parker study Topic Dynamics by monitoring the topics in a stream of events and detect bursts in these streams [126].

Sun et al. [127] studies the evolution of communities in information networks and presents a case study for DBLP, its largest communities and their topics. Peng and Li [128] work with authors and their keywords as a three-way data (w.r.t. time) and review the algorithms for their clustering.

Huang et al. in [129] explores the communities of DBLP from the point of view of their keywords and the usage of these keywords through the time. Emerging terms and communities are analyzed as well as the vanishing ones. Jo et al. [130] proposed an algorithm for discovering the evolution of topics in a time-stamped document collection.

6 http://www.informatik.uni-trier.de/ley/db/
7 http://dblp.l3s.de
5.2 Evolution of Author’s Topic

Topics visualization

Grand and Soto in [131] discuss the visualization of structured data related to topics, keywords and scientific data with focus on the complexity and information gain of the visualization. Cao et al. in [132] presents ChronAtlas for visually exploring topics in multifaceted dynamic data and provides a case study on DBLP data. His hybrid visualization combines clustered tag cloud with a radial tag view.

MS Academic Search\(^8\) provides a domain trend page which displays the volume of publications in the subdomains of computer science for a selected time interval. The visualization of topics across the field of computer science is available as a stacked area chart\(^9\).

Contrary to mentioned works, in our approach we focus on the evolution of topics of one particular author over the time and on the methods of visualization of this evolution. In this work we neglect the topics trends over the whole network or community and we specialize on giving the most accurate image of an author and his interests in every moment of his scientific career. For this purpose, we created an animation of author’s keywords from paper titles to present the user the overall view of an author through his/her whole career.

5.2.3 Forgetting function

In this section we explain our concept of forgetting and how it is used w.r.t. author’s topics.

Forgetting function

We use a forgetting function for a time-dependent calculation. In this calculation we were inspired \(^{[76]}\) by forgetting curve \(^{[133]}\). The forgetting curve defines the probability that a person can recall information at time \(t\) since previous recall (see Fig. 5.17). It can describe long-term memory and is usually presented using the following equation.

\[
R = e^{-\frac{t}{S}}
\]

\(R\) (memory retention) - the probability of recalling information at time \(t\) since the last recall.

\(E\) Euler number (aprox. 2.718).

\(T\) time since the last recall.

---

\(^8\) http://academic.research.microsoft.com/

\(^9\) http://academic.research.microsoft.com/DomainTrend?TopDomainId=2
5.2 Evolution of Author’s Topic

$s$ (relative strength of memory - stability) - approximated time since the last recall for which is the information stored in memory.

![Figure 5.17: Forgetting Curve](image)

The computation depends on the type of memory, especially on the estimated time $S$ (this value is not constant in the long term). For simplicity, assume that if we work with the information for the first time, then the time of storing information in memory is $S_{ini} > 0$ and this default value is constant.

An important feature of long-term memory is that after reproduced information recall in the time $t > 0$, the time of storing information in memory - stability - $S$ changes. The change is dependent on the previous time $S$ and on the time of recall $t$. Ideally, the reproduced recall multiplies this time (in comparison with the previous value) by factor $F > 1$.

The other important feature of long-term memory is that immediate reproduced recall (too quick) of information has no bigger effect on the learning. On the other hand, the reproduced recall too lately (in time near $S$) causes substantial forgetting. There is an optimal time between these two extreme situations in which the reproduced information recall causes a high level of remembering (and consequently the maximum increase of time $S$ by factor $F$).

In the ideal case (reproducing the information in optimal time), the remembering of information is gradual and very effective - after each recall, the time of storing information in memory $S$ (remembering) is multiplied by factor $F$.

For updated $S_{new}$, after new information recall should hold:

1. If $t > S$ then $S_{new} = S_{ini}$ (information is considered as new)
2. If $t \rightarrow S$ then $S_{new} \rightarrow S_{ini}$ (late recall is considered as almost new information).
3. If $t \rightarrow 0$ then $S_{new} \rightarrow S$ (early recall has almost no influence)
4. If $t \rightarrow \text{opt}(S)$ then
\[ S_{\text{new}} \rightarrow F \cdot S, \]
where $\text{opt}(S)$ is the function returning optimal time for recalling the information and $F$ is the factor of optimal improvement.

Remark 7. For reproduced information recall is $R = 1$. This follows from the fact, that $t = 0$ at this moment. For the factor of optimal improvement holds, that when the information is recalled at optimal time, the value of $S$ is multiplied by two (depending on the type of memory). Therefore we can assume that $F \in (1;2)$.

![Figure 5.18: Calculation of $S$ in time $t$](image)

For new stability $S_{\text{new}}$ calculation we have to consider three things:

1. The function $\text{opt}(S)$ for the calculation of optimal information recall time.

2. The choice of optimal improvement factor $F$.

3. Function $f(t, S, F)$ for calculation of $S_{\text{new}}$.

Available sources present the optimal time for reproduced information recall in the range of 10–30% of time $S$. The setting of this function $\text{opt}(S)$ is dependent on the type of memory (e.g. $\text{opt}(S) = 0.2 \cdot S$).

The factor $F$ is involved in the computation of time $S$ for which the information is held in memory (is remembered). This factor is again dependent on the type of memory. For the calculation of $S$ with the same type of memory the value of $F$ is constant (e.g. $F = 1.2$).

The value of $S_{\text{new}}$ is dependent on the type of memory, on the time of repetitive information recall and on the previous value of $S$ (this incorporates the history of learning mentioned information). For the calculation of $S_{\text{new}}$ we need to design the function of $\text{ch}(t, S, F, S_{\text{ini}})$ for the calculation of the coefficient of change of the value $S$. Then holds:

\[ S_{\text{new}} = \text{ch}(t, S, F, S_{\text{ini}}) \cdot S \]

Available sources contains various approaches for the computation of value of function $\text{ch}(t, S, F, S_{\text{ini}})$. For example we will use simple relation based on linear functions (see Fig. 5.18):
5.2 EVOLUTION OF AUTHOR’S TOPIC

1. If $0 \leq t \leq \text{opt}(S)$ then
   \[ ch(t, S, F, S_{ini}) = 1 + (F - 1) \cdot \frac{t}{\text{opt}(S)} \]

2. If $\text{opt}(S) \leq t \leq S$ then
   \[ ch(t, S, F, S_{ini}) = F - (F - \frac{S_{ini}}{S}) \cdot \frac{t-\text{opt}(S)}{S-\text{opt}(S)} \]

3. If $t > S$ then
   \[ ch(t, S, F, S_{ini}) = \frac{S_{ini}}{S} \]

To set the parameters of stability calculation we have used the following values: $S_{ini} = 12$, $F = 1.2$. Interpretation: if an author publishes a paper containing the keyword for the first time, then he/she will be active in the corresponding area for the next 12 months. If he/she publishes again in the optimal time, his/her stability is multiplied by 1.2 (if the keyword is not used again, it is forgotten).

Pre-processing algorithm

As previously mentioned, we divided entire recorded publication period into one-month time periods. The calculation of stability based on forgetting function for a keyword is described in algorithm 6. We calculate it for all keywords of every author from the first publication to the last.

<table>
<thead>
<tr>
<th>input</th>
<th>Author’s keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>MOURS - all months from first publication of the keyword to the present time</td>
</tr>
<tr>
<td>output</td>
<td>Stability of keyword for each month</td>
</tr>
<tr>
<td>Data:</td>
<td>Constants: $S_{ini} = 12$, $F = 1.2$</td>
</tr>
<tr>
<td>Data:</td>
<td>$t$ - number of months from last publication of the keyword</td>
</tr>
</tbody>
</table>

for each $m$ in MOURS do
  $S_{new} = ch(t, S, F, S_{ini}) \cdot S$
  if keyword published in $m$ then
    $t = 0$
  else
    $t = t + 1$
  endif
end

Algorithm 5: Data pre-processing

For illustration see example of calculated stability $S$ on Fig. 5.19, where an author published a paper with a keyword in month 1, then two months later, seven months later after that, etc. Every time a keyword is used its stability increases, the more often
5.2 Evolution of Author’s Topic

Figure 5.19: Example of calculation of stability for a keyword over a three year period.

it is used, the higher its stability will be. On the other hand, if it is not used, its stability keeps dropping following the forgetting curve.

Figure 5.20: Example of visualization of three keywords over a three year period.

In the next section we describe Forcoa.NET interface where an animation is used as a method of visualization of this pre-processed data.

5.2.4 Topic evolution visualization

When we observe author’s publication activity we may look at it from two different perspectives. The first one is to focus on the whole history to find out what areas is the author an expert in, which topics were most studied throughout his publication history. This perspective we call author’s profile (see Fig. 5.21 - right part).
5.2 Evolution of Author’s Topic

The second one is more immediate. We are interested in what is the author currently working on, what are the uppermost topics at a particular time (see Fig. 5.21 - left part). Both of these perspectives have their own evolution, which we are trying to depict with an animation. It is obviously not easy to describe an animation in text form, but all readers may check this up online on Forcoa.NET.

Dataset

On April 27, 2012, we downloaded the DBLP dataset in XML and preprocessed it for further usage. For every record we identified the month and year of the publication. In the next step we extracted all authors having at least one published paper (1,060,175 authors). Then we extracted keywords and phrases from paper titles. The approach was based on Faceted DBLP set; 1,179 keywords and phrases were used in total.

We have truncated the selected time period to December 2011 to get the most complete dataset. Then we divided the entire recorded publication period of conferences (the first record from 1963) into one-month time periods. If an author has published a paper in a month we set a keyword records corresponding to the paper title. For each author we obtained a list of months with collected keywords.

Then we applied the forgetting function to compute the weight - stability - of each keyword (or phrase) per month for each author (see Fig. 5.19).
Visualization of the data

The animation consists of two parts. One we call ‘evolution’ serves to show in simple way author’s most recently used keywords. Only some of them are visible, we chose only seven to keep this part illustrative. The second part we call ‘profile’ contains selected keywords which outweighed others from the historic perspective. For example see Fig. 5.20 with three snapshots from different time in history.

User interface

Figure 5.21 presents a screenshot of the Forcoa.NET ‘topic history’ window for one particular author (Philip S. Yu\(^{10}\)). The beginning date of history is the date of author’s first publication activity, then the history goes up to the present. One month is one step in the history as the measures are calculated for each month.

In the left part of the window is an area we call ‘evolution’ which contains seven selected keywords. We select those keywords which have the highest stability at a particular time (month of the year). The size of the text relates to the stability of the keyword. We can see there some keywords either grow, diminish or even fade away once the stability drops under a certain threshold. For example, see Fig. 5.22 - A, where the keyword ‘data streams’ does not stand out, but how it grows in Fig. 5.22 - B. The goal of the left part is to present an actual interest of the author.

![](image)

Figure 5.22: Two Snapshots from Forcoa.NET ‘topic history’ window for Philip S. Yu - A) from month 1/2005, B) from month 12/2011.

The right part of the window represents author’s profile and contains a list of ordered keywords. The author’s profile provides a summary of the author’s activity in history. On the top of the list are keywords with highest maximum of stability until a particular time. So in motion we can see some keywords work their way up to the top of the list while others drop down, see the difference in the order of keywords in Fig. 5.22 - A and

---

\(^{10}\) Philip S. Yu has most records in DBLP database
5.3 DISCUSSION

Fig. 5.22 - B. At the end of the animation we can see the present profile with the topic that has been most studied on the top of the list, even though we might not be able to find it on the left part of the window because it is not actual anymore and was already forgotten.

The ‘actualness’ of the keyword is indicated by the opacity of the text, see Fig. 5.22 - B, where the keyword ‘adaptation’ is grey, because it was not used recently. In fact, most of the keywords from the top of the list had their maximum years ago, see Fig. 5.23.

Figure 5.23: The evolution of the top 7 keywords from the profile of Philip S. Yu.

5.3 DISCUSSION

In this chapter we have shown how self-organizing map can be used to identify various roles of nodes in the social network based on their topological properties. We have demonstrated a process which can be to a great extent parametrized in order to allow different points of view. The experiment with the DBLP co-authorship dataset proved that our approach is viable in large scale networks. The results have been validated by confronting surroundings of randomly picked authors.

In other experiment, we proposed a simple way to visualize the evolution and dynamics of author’s topic in our online system Forcoa.NET. We extracted keywords and short phrases from paper titles and used them as a descriptor of author’s research. The visualization provides a good overview on the evolution of the author. Our approach can be validated in our online tool.
The analysis of networks affects the research of many real phenomena. In our research, we study one characteristic that significantly affects the development of a co-authorship network structure. We detected this characteristic on the basis of our observation and analysis of co-authorship networks; it is a Poisson distribution of the number of co-authors of publications. To assess the relevance of this characteristic, we define a network growth model which uses this characteristic as its only principle. Our experiments show that networks generated by this model have properties known from the environment of real-world networks.

6.1 Introduction

The complex network structure can be viewed from the perspective of the state of the network at the time of the analysis. Networks can, therefore, be described by the properties known from the environment of real-world networks, including, in particular, the small-world, scale-free, high average clustering coefficient, assortativity [105], community structure, or shrinking diameter [134], but also others, such as core-periphery structure [135]. Underlying processes that take place during the evolution of real-world networks have also been examined. Some network models are based on analyzing these processes using the formally described underlying process as a generative mechanism. Such a mechanism can generate networks possessing one or more known properties.

The aim of our research was to determine the statistical distribution of different types of co-authors of publications and how it affects the structure of co-authorship networks. At the start, we formulated two assumptions. The first is that the publication always has one main author. In the second assumption, similarly to Abbasi et al. [136], we recognize different types of co-authors who must create a link to each other (if not already connected) to collaborate on the publication: (1) co-authors who already published with the main author, (2) new authors without publications, and (3) existing authors who do not have any previous publications with the main author. Our analysis of two co-authorship network datasets shows that both the total number of co-authors and the number of all types of co-authors of the main author can be, in agreement with [137], well described by a Poisson distribution. Another part of the research was to determine how strong a role the Poisson distribution plays in the development of the network. Therefore, using only this single characteristic, we define a process which results in network growth. One

1 The supplement is available online at http://homel.vsb.cz/~kud007/asoonam2018supplement/
step in the growth of the network is the interaction between the main author and groups of co-authors of various types in numbers on the basis of the Poisson distribution. In our experiments, we show through a wide range of properties that the model described by such a process generates networks with properties close to real-world co-authorship and social networks in general.

6.2 RELATED WORK

In the last two decades, the analysis of real-world networks has received extraordinary attention. Co-authorship and collaboration networks in general are long-investigated sources in this area. A common feature of this type of network is that underlying processes proceed in cliques, which then become a fundamental building block of the network.

Barabasi et al. [138] presented and analyzed in detail a network model inspired by the evolution of co-authorship networks. Their measurements of real-world networks revealed that the distribution of degrees in collaboration networks has a power law, and they are small-world networks with an increasing average degree. Newman confirmed the above observations in [139] and extended the knowledge about the properties of collaboration networks by assessing that they have a high clustering coefficient and a positive assortativity coefficient. The model presented by Ramasco et al. [11] combines preferential edge attachment with a bipartite structure and depends on the act of collaboration. They also demonstrated that assortativity in collaboration networks depend on the aging of nodes. Stasa [137] introduced a model which accurately reproduces the evolution of empirical team size distribution over time. Core teams form by a Poisson process and produce a Poisson distribution of team sizes in which larger teams are exceedingly rare. The “extend” teams, which started as core teams, subsequently accumulate new members proportional to the past productivity of their members; this kind of teams generate a power-law tail of large teams.

Preferential attachment is a universal principle. Implicit or explicit, it is a basic mechanism of many network models explaining the power law distribution and inhomogeneous network structures. At the moment of the connecting of new nodes to the network during its growth, there is a preference for selecting high-degree nodes. The well-known Barabasi-Albert model [140] generates networks with a constant diameter and a logarithmically growing average degree. Unlike traditional views of preferential attachment, the forest fire model (Leskovec et al. [134]) works with empirically observed properties such as the densification power law (real graphs have degrees that grow over time) and shrinking diameter (diameter decreases with network growth).

In model of network growth, proposed by Toivonen et al. [141], communities arise from a mixture of random attachment and implicit preferential attachment. Implicit
preferential attachment is the result of interconnection between a new node and the neighbors of a randomly selected existing node.

Another important characteristic that has been found in many real-world networks, especially in social networks, is the community structure. Unlike previous methods that focused on the strongly connected cores of communities, the approach of Girvan and Newman [20] uses edge betweenness to detect communities. Understanding the principles upon which communities emerge is a key task. Triadic closure as a principle for connecting the nodes has been described in many models (e.g., Holme and Kim in [142]). As a key principle from the perspective of communities, the triadic closure mechanism is used to describe the growth of a network with a non-trivial community structure in Bianconi et al. [143]. The majority of network growth models involving triadic closure models assume that preferential attachment is only an outcome of a local process and usually consider only the new node node as the source of triadic closure. However, in real-world networks, the majority of links are created by existing nodes. Shekatkar and Ambika [144] reflect on this and include the formation of internal links in their growth model through an additional property for nodes called mediating capacity. A common feature of models generating community structure is that they do not use an explicit preferential attachment which leads to scale-free property. The scale-free structure is, in this case, the result of underlying processes.

Observing real-world networks leads to the realization that communities may overlap. Cliques as the basis for a community were used by Palla et al. [145] in their clique percolation method on the basis of the premise that a node may belong to more cliques simultaneously. Yang and Leskovec introduced the community-affiliation graph model [146], based on the observation that overlaps between communities are denser than communities themselves.

The starting point for designing models is always a set of properties that are known from real-world networks. The aim of the models is to find the principles behind these properties. In our case, this principle is the Poisson distribution, together with the manner in which the nodes are organized in the interactions. We discovered this principle through observation of two co-authorship networks. With the generative model that works with this single principle, we show the impact of this principle on a broad set of known properties of networks and relationships between them.

6.3 Analysis of co-authorship datasets

We studied the DBLP dataset, which contains basic bibliographical information on publications from the computer science field. These data are freely available² and contain highly relevant information about publication activity over a period of nearly fifty years, even though they are not complete. We downloaded this dataset in July 2016 and, on

² http://www.informatik.uni-trier.de/~ley/db/
the basis of the co-authorship of authors, we constructed a social network in which au-
thors are linked if they co-authored a paper. The basic characteristics of this network
are shown in Table 6.1.

### Table 6.1: DBLP and PubMed sample datasets

<table>
<thead>
<tr>
<th></th>
<th>DBLP</th>
<th>PubMed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of publications</td>
<td>2,865,672</td>
<td>209,688</td>
</tr>
<tr>
<td>Total number of authors</td>
<td>1,622,828</td>
<td>483,152</td>
</tr>
<tr>
<td>Total number of edges</td>
<td>6,930,745</td>
<td>1,029,923</td>
</tr>
<tr>
<td>Average papers per author</td>
<td>5.113</td>
<td>1.359</td>
</tr>
<tr>
<td>Average authors per paper</td>
<td>2.895</td>
<td>3.584</td>
</tr>
<tr>
<td>Average degree</td>
<td>8.915</td>
<td>5.153</td>
</tr>
<tr>
<td>Average local clustering coefficient</td>
<td>0.734</td>
<td>0.773</td>
</tr>
<tr>
<td>Assortativity</td>
<td>0.121</td>
<td>0.351</td>
</tr>
<tr>
<td>Modularity (Louvain)</td>
<td>0.718</td>
<td>0.925</td>
</tr>
<tr>
<td>Number of communities (Louvain)</td>
<td>422</td>
<td>357</td>
</tr>
</tbody>
</table>

In an additional two steps of pre-processing of the dataset, we first attributed to each
publication a month which corresponded to the date of a conference or date of publi-
cation in a journal, respectively. If the record in the DBPL did not contain the month
of publication, we chose the month for a given year randomly. In the second step, we
ordered all publications in each month randomly. As a result, we obtained an ordered
list of publications.

Furthermore, we assumed that an existing author of a given publication is the author
who had at least one publication preceding the given publication in the ordered list.
In the next step, we dropped all publications that did not contain any already-existing
authors. For the rest of the publications, we randomly chose one existing main author.
Later in our paper, we describe the exact procedure we used to choose the main author
of the publication.

The first objective of the experiment with the DBLP dataset was to discover the shape
of the distribution of the number of publications, depending on the number of co-
authors of the main author. Fig. 6.1 shows the distribution for the first twenty values
(i.e. up to 20 co-authors) and cumulative distribution of all values (except for one publi-
cation with 286 authors). This distribution is compared to a Poisson distribution with a
$\lambda$ value equal to the average number of co-authors, which is 1.99.

#### 6.3.1 Co-author roles

The second objective of the experiment was to find authors who are most often in the
role of the main author of a publication and to analyze their co-authorships. First, we
6.3 Analysis of co-authorship datasets

need to define who exactly the main author is. Publication activities can be projected into a growth network. As mentioned in the Introduction, one step in the growth of the network is the interaction between the main author and groups of co-authors of various types. Below, authors will be represented by the nodes of the network and publications by the interaction of a group of nodes. Both new and existing nodes are involved in the interaction, and after the interaction, there are edges between all the pairs of nodes that are involved. Some (or all) of these edges between pairs of involved authors could exist before the given publication. The nodes involved in the interaction have four different roles:

**Definition 21.** (Proactive): The proactive node is that already-existing node of the network which has the majority of the previously existing interconnections (edges) with other nodes in the interaction.

**Definition 22.** (Old connection): An old connection node of the interaction is an already-existing node of the network which is the neighbor of the proactive node.

**Definition 23.** (Newbie): A newbie node of the interaction is a new node in the network.

**Definition 24.** (New connection): A new connection node of the interaction is an already-existing node of the network which is not a neighbor of the proactive node.

Furthermore, we assume that the main node of the interaction is the proactive node. If there are more nodes meeting the Definition 21, then we randomly select one of them. It is essential that each interaction always contains exactly one proactive node and it may (but does not have to) include nodes in three other roles. Furthermore, we assume
6.3 Analysis of Co-Authorship Datasets

that the numbers of co-authors in different roles are independent random variables. We detected, by definition, a proactive node of the interaction for each interaction (publication) in an ordered list of publications (resulting from the preprocessing of the DBLP dataset). Furthermore, for individual authors, we investigated the distribution of the number of co-authors among the publications in which these authors were in the role of a proactive node. The empirical distribution, together with the theoretical value of the Poisson distribution for the first ten authors with the highest number of publications in the role of the proactive author, is shown in Fig. 6.2.

![Co-authors histograms for top 10 authors (by number of publications) in DBLP](image)

Figure 6.2: Co-authors histograms for top 10 authors (by number of publications) in DBLP

6.3.2 Poisson distribution

The last part of the experiment with the DBLP dataset was to determine the distribution of the co-authors of proactive nodes overall and in individual roles. Cumulative distributions comparing empirical and theoretical Poisson distribution are shown in Fig. 6.3. For individual Poisson distributions the following values are expected: $\lambda = 1.99$ (total), $\lambda_1 = 1.16$ (old connection), $\lambda_2 = 0.50$ (newbies), and $\lambda_3 = 0.36$ (new connections).

The experiments show that the probabilities of a publication with a certain number of co-authors in different roles follow, at least in part, the Poisson distribution. However, Stasa discovered in [137] that there is also a transition from Poisson to the fat-tailed distribution which cannot be generated by Poisson distribution.

In the last experiment, we worked with a freely available sample of publications from the PubMed database\(^3\). For dataset preprocessing and conversion to a network, we

6.3 ANALYSIS OF CO-AUTHORSHIP DATASETS

Figure 6.3: Distribution of the co-authors of proactive nodes overall and in individual roles - DBLP

used the same procedure as for the DBLP data. However, the data sample is relatively small and fragmented (the publications are from the years 1955, 1965, 1974-5, 1985, 1995, 1999, 2006, and 2012) and most of the authors in this sample have only one publication (390,329 authors in total). The basic characteristics of this network are shown in Table 6.1.

The cumulative distributions comparing the empirical and theoretical Poisson distribution of the number of co-authors of proactive node overall and in individual roles are shown in Fig. 6.4. For individual Poisson distributions the following values are expected: $\lambda = 2.57$ (total), $\lambda_1 = 0.32$ (old connection), $\lambda_2 = 1.86$ (newbies), and $\lambda_3 = 0.39$ (new connections).

Figure 6.4: Distribution of the co-authors of proactive nodes overall and in individual roles - PubMed

Despite the low reliability of the PubMed sample, the results of the analysis show that the distributions for authors with a large number of publications and the total cumulative distributions can be considered the Poisson distribution.

88
The growth of a co-authorship network is influenced by many factors, including, e.g., different types of behavior, changes in the behavior of the author over time [136, 137], or aging [11]. The observation of Poisson distribution in the numbers of nodes in interactions, however, implies a simple model of a growing network for which the Poisson distribution is its only characteristic. Our objective is to show that even in the absence of other aspects of the development of networks such a model generates networks with characteristics known from the environment of real-world networks.

The model is based on the following premises: (1) one step in the growth of the network is the interaction of a proactive node with nodes in three different roles; (2) the variables describing the numbers of co-authors in different roles are independent
random variables that follow Poisson distribution; (3) the proactive node, its neighbors
(old connections), and new connection nodes are selected at random; (4) the expected
values of $\lambda_1$ (old connections), $\lambda_2$ (newbies), and $\lambda_3$ (new connections) of the Poisson
distributions are preselected.

A natural characteristic of the model is implicit preferential attachment; nodes with
a higher degree are more likely to participate in an interaction in which the proactive
node has at least one neighbor. Even though the proactive node of the interaction and
its neighbors are selected at random, a high-degree node has a greater chance of being
selected as a neighbor of the proactive node.

$\lambda_1$, $\lambda_2$, and $\lambda_3$ significantly affect the density of the network. If we assume that a
randomly selected interaction has, on the basis of the corresponding distributions, $b$
nearbys of a proactive node, $n$ new nodes, and $e$ nodes unconnected to the proactive
node, then the number of nodes involved in this interaction (interaction size) is as shown
in Equation 6.1.

$$s = 1 + b + n + e \quad (6.1)$$

and the following applies:

- $n$ new nodes which must connect to a proactive node and each other are created.

- There are $b$ nodes adjacent to the proactive node which must first become con-
  nected with each other, then with $e$ nodes that are not adjacent to the proactive
  node (these edges may already exist prior to the interaction). Finally, they must
  connect with the new $n$ nodes.

- There are $e$ nodes not adjacent to the proactive node, which must become con-
  nected with it. Next, they must become connected with each other (these edges
  may already exist prior to the interaction) and $n$ new nodes.

Each of $\lambda_1$, $\lambda_2$, and $\lambda_3$ affects a different property of the network generated by the
3-lambda model.

- $\lambda_2$ (newbies) defines the growth rate of the network and provides a tree-like net-
  work structure; see Fig. 6.5. To construct a network with $N$ nodes requires approx-
  imately $\frac{N}{\lambda_2}$ interactions.

- $\lambda_1$ (neighbors) constitutes a network community structure through the local con-
  nections of existing neighbors and new nodes; see Fig. 6.6.

- $\lambda_3$ (new connections) ensures the linking of nodes that are not adjacent, thereby
  linking communities. The consequence is the emergence of a core-periphery net-
  work structure; see Fig. 6.7.
For a deeper assessment of the effect of varying lambdas on the properties of the network, we used the settings from Fig. 6.7. For each of the three lambdas, we set the interval and the step of growth for one lambda and fixed the remaining two lambdas at the setting shown in the picture. Using the algorithm described below, we generated networks while moving along the interval with a value of one lambda, and examined changes in the mean clustering coefficient, modularity (Louvain and Infomap), and assortativity (see Fig. 6.12), and also changes in the average degree and the average shortest path and diameter (see Fig. 6.16).

It is evident from Fig. 6.12 that increasing $\lambda_1$ and $\lambda_3$ reduces the average clustering coefficient and a rapid decrease in modularity is associated with this (the network community structure weakens); when $\lambda_2$ increases the values of these properties increase. Assortativity is negatively affected by increasing $\lambda_1$ and positively by increasing $\lambda_3$. Fig. 6.16 shows that increasing $\lambda_1$ and $\lambda_3$ causes a decline in the average shortest path and network diameter. The average degree, and therefore the density of the network,
is most influenced by the growth of $\lambda_3$ and least by the growth of $\lambda_2$. The strange behaviors of characteristics at low values of $\lambda_2$ are caused by proactive nodes, which on average do not have enough neighbors to interact with as a result of the values of the Poisson distribution for $\lambda_1$.

### 6.4.1 Network generator

The network generator uses a simple algorithm which comes directly from the model description. The only extra step is setting up the initial network state. The model is memory-less, which allows us to work with an arbitrary initial state. For our generator we chose a complete graph with a number of nodes equal to the round of $(1 + \lambda_1 + \lambda_2 + \lambda_3)$ as the default state. Algorithm 6 describes the whole process of generating a network. The network that is generated is a connected graph; the algorithm starts with a complete graph, and each interaction contains at least one existing node.

<table>
<thead>
<tr>
<th>input</th>
<th>number of nodes $N$, $\lambda_1$, $\lambda_2$, $\lambda_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>output:</td>
<td>generated network $G$</td>
</tr>
</tbody>
</table>
1. choose $s = \text{ROUND}(1 + \lambda_1 + \lambda_2 + \lambda_3)$
2. create $G = (V, E)$ as a complete graph with $s$ nodes
3. while $V$ has less than $N$ nodes do
   4. choose from $V$ randomly proactive node $a$
   5. $o = \text{number of old connections by Poisson}(\lambda_1)$ ($o$ is the number of neighbors of $a$ at most)
   6. $n = \text{number of newbies by Poisson}(\lambda_2)$
   7. $c = \text{number of new connections by Poisson}(\lambda_3)$
   8. create a list $I$ with a proactive node $a$
   9. add to list $I$ $o$ randomly selected neighbors of node $a$ from $V$
   10. create $n$ new nodes, add them to $V$ and $I$
   11. add to list $I$ $c$ randomly selected not-neighbors of node $a$ from $V$ (if such nodes exist)
   12. foreach pair of nodes $(v_i, v_j) \in I$ do
         13. if no edge $e$ between $v_i$ and $v_j$ exists then
         14. create $e$ and add to $E$
         15. endif
   16. end
17. end

**Algorithm 6:** 3-lambda model network generator

The average number of nodes in an interaction is approximately $s = 1 + \lambda_1 + \lambda_2 + \lambda_3$. However, the average is slightly lower because the number of neighbors selected for
interaction through the simulation of the Poisson distribution is limited by the actual (maximum) number of neighbors of the proactive node. The time complexity of the algorithm is \( O(s^2 \cdot \frac{N}{\lambda^2}) \), which is based on the fact that:

- To generate \( N \) nodes requires approximately \( \frac{N}{\lambda^2} \) interactions.
- The main operation is the test of the existence of an edge among the nodes participating in the interaction.
- The number of edges between the nodes in an interaction is in quadratic relation to the number of nodes involved in this interaction (the interaction takes place in a complete sub-graph with \( n \) nodes and \( m = \frac{n^2(n-1)}{2} \) edges).

The calculation of the complexity of the algorithm does not include the complexity of the simulation of the Poisson distribution for individual lambdas (in our case, Knuth’s algorithm with the complexity \( O(\lambda) \) was used).

When the complexity is being calculated, the value of \( s \) is influenced by the values of all three lambdas, whose impact on the number of operations needed to generate the network is not, at first glance, very clear. Fig. 6.17 is a ternary plot that shows the dependence of the number of operations on the changing values of individual lambdas for a network with millions of nodes for a thousand different settings. The settings combine ten different values for each lambda, so that \( \lambda_1 \in \{0, 1, 2, \ldots, 9\} \), \( \lambda_2 \in \{0.5, 1.5, 2.5, \ldots, 9.5\} \), and \( \lambda_3 \in \{0, 0.5, 1, \ldots, 4.5\} \). Individual points in the ternary plot match one setting.

![Figure 6.17: Time complexity: changing lambdas](image)

The plot clearly shows that the number of tests for the existence of edges between nodes starts growing relatively rapidly with a combination of low levels of \( \lambda_2 \) on one side and high values of \( \lambda_1 \) and \( \lambda_3 \), respectively, on the other side. The maximum value of the number of operations corresponds to the setting \([9, 0.5, 4.5]\). Networks with 1,000
nodes generated with this setting have approximately 110,000 edges and a diameter of 3, and the average length of the shortest path is 1.80.

### 6.5 Experimental Evaluation

For the purposes of our experiments, we defined three different settings, Setting = \([\lambda_1, \lambda_2, \lambda_3]\), in the experiments. In Setting_1 = [1.6, 0.35, 0.05], the average interaction involved three nodes (a triad). This setting presumes that the interaction is dominated by neighbors of the proactive node, with the occasional participation of new nodes and the rather exceptional participation of existing nodes not yet connected to the proactive node. In Setting_2 = [3, 6, 1] new nodes predominate, and the average number of nodes in the interaction is 11. In Setting_3 = [0.45, 0.45, 0.1] the interaction involved two nodes (a dyad) on average, wherein the numbers of neighbors and new nodes are balanced and occurrences of new connections less likely.

![Figure 6.18: Setting_1](image1)

![Figure 6.19: Setting_2](image2)

![Figure 6.20: Setting_3](image3)

![Figure 6.21: Degree distribution (power law)](image4)

### 6.5.1 Properties of generated network

<table>
<thead>
<tr>
<th>Setting</th>
<th>(n)</th>
<th>(m)</th>
<th>(\langle k\rangle)</th>
<th>(\langle l\rangle)</th>
<th>(l_{max})</th>
<th>CC</th>
<th>(r)</th>
<th>(Q_{IM})</th>
<th>(Q_{com})</th>
<th>(Q_{long})</th>
<th>(Q_{front})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting_1 mean</td>
<td>10,001.10</td>
<td>40,108.97</td>
<td>8.0209</td>
<td>4.8101</td>
<td>12.07</td>
<td>0.6555</td>
<td>0.14522</td>
<td>609.40</td>
<td>54.70</td>
<td>0.6424</td>
<td>0.7119</td>
</tr>
<tr>
<td>Setting_1 sd</td>
<td>0.31</td>
<td>98.80</td>
<td>0.0797</td>
<td>0.0424</td>
<td>0.64</td>
<td>0.0020</td>
<td>0.01769</td>
<td>12.37</td>
<td>7.04</td>
<td>0.0062</td>
<td>0.0085</td>
</tr>
<tr>
<td>Setting_2 mean</td>
<td>10,003.87</td>
<td>88,620.40</td>
<td>7.7172</td>
<td>4.2369</td>
<td>8.43</td>
<td>0.8087</td>
<td>0.1296</td>
<td>422.60</td>
<td>42.90</td>
<td>0.6772</td>
<td>0.7334</td>
</tr>
<tr>
<td>Setting_2 sd</td>
<td>1.53</td>
<td>797.95</td>
<td>0.1600</td>
<td>0.0277</td>
<td>0.63</td>
<td>0.0018</td>
<td>0.00913</td>
<td>10.30</td>
<td>2.78</td>
<td>0.0049</td>
<td>0.0067</td>
</tr>
<tr>
<td>Setting_3 mean</td>
<td>10,001.17</td>
<td>21,494.57</td>
<td>4.2984</td>
<td>6.8965</td>
<td>16.23</td>
<td>0.4784</td>
<td>0.19907</td>
<td>953.30</td>
<td>68.70</td>
<td>0.7276</td>
<td>0.8129</td>
</tr>
<tr>
<td>Setting_3 sd</td>
<td>0.46</td>
<td>142.64</td>
<td>0.0285</td>
<td>0.0609</td>
<td>0.63</td>
<td>0.0044</td>
<td>0.01222</td>
<td>15.55</td>
<td>3.19</td>
<td>0.0035</td>
<td>0.0038</td>
</tr>
</tbody>
</table>
In the first experiment, we show the properties of networks generated with different settings. For each setting we generated 100 networks with approximately 10000 nodes. Table 6.2 summarizes the average values and standard deviation of the measured properties for each setting. The measured properties include the number of nodes $n$ and edges $m$, average degree $\langle k \rangle$, average shortest path length $\langle l \rangle$, diameter $L_{\text{max}}$, average clustering coefficient $\text{CC}$, assortativity $r$, number of communities detected by Infomap \cite{infomap} $\text{com}_{\text{IM}}$ and Louvain \cite{louvain} $\text{com}_{\text{L}}$ algorithm, and the corresponding modularities $Q_{\text{IM}}$ and $Q_{\text{L}}$, respectively. Fig. 6.21 shows the degree distribution. The experiment indicates that all three settings generate networks with small-world and scale-free characteristics. The first and second settings generated networks with a high average clustering coefficient. The networks have a tendency to be assortative. The assortativity values for all settings correspond with the values known from social networks \cite{social_networks}. The networks that are generated also have a community structure and high modularity for all settings. Fig. 6.22 shows a network with 1000 nodes and 4,031 edges generated with Setting$_1$, size of the nodes corresponds to their degrees. The network has an overlapping community and core-periphery structure. The 23 colored communities were detected by means of the Louvain method; the modularity is 0.689.

Figure 6.22: Network: 1000 nodes, Setting$_1$
For a more detailed examination of the impact of individual lambdas on selected properties of the networks that were generated, we used the same setting ranges as in Section 6.4. For each setting, we generated a network with 10,000 nodes (a total of 1,000 networks), and for each network, we calculated the average degree, average clustering coefficient, and assortativity. The results are shown in ternary plots in Fig. 6.26. The behavior of the average degree corresponds to the analysis performed for the calculation of complexity. It confirms that the average degree (network density) grows rapidly at low values of $\lambda_2$ together with high values of $\lambda_1$ and $\lambda_3$. The interesting point appears to be the relation of $\lambda_3$ to the value of the average clustering coefficient and assortativity, which was seen in the experiment with fixed lambda values shown in Fig. 6.12. With an increasing value of $\lambda_3$ the clustering coefficient decreases (and the modularity too) and assortativity increases, and vice versa.

Figure 6.23: Degree

Figure 6.24: Clustering coeff.

Figure 6.25: Assortativity

Figure 6.26: Global properties influenced by changing lambdas

### 6.5.2 Growth of generated network

The subject of the second experiment is one network generated with Setting1. The aim is to show the growth of the properties of the network during its growth. The values for each characteristic are measured when the network has 10, 20, 50, 100, ..., and 10,000 nodes. The results are summarized in Table 6.3.

The key properties (average degree, shortest path, clustering coefficient, assortativity, modularity) happen to stabilize their values between 1,000 and 10,000 nodes. Our experiments show that networks generated with other settings also exhibit similar behavior.
6.5 Experimental Evaluation

Table 6.3: Evolution of network properties

<table>
<thead>
<tr>
<th>n</th>
<th>(m)</th>
<th>(&lt;k&gt;)</th>
<th>(&lt;l&gt;)</th>
<th>(l_{\text{max}})</th>
<th>CC</th>
<th>(r)</th>
<th>(c_{\text{IM}})</th>
<th>(c_{\text{L}})</th>
<th>(Q_{\text{IM}})</th>
<th>(Q_{\text{L}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>27.00</td>
<td>5.4000</td>
<td>1.4000</td>
<td>20</td>
<td>0.8195</td>
<td>-0.41738</td>
<td>1</td>
<td>3</td>
<td>0.0000</td>
<td>0.0590</td>
</tr>
<tr>
<td>20</td>
<td>53.00</td>
<td>5.3000</td>
<td>2.0684</td>
<td>5</td>
<td>0.5804</td>
<td>-0.09355</td>
<td>2</td>
<td>4</td>
<td>0.0361</td>
<td>0.2398</td>
</tr>
<tr>
<td>50</td>
<td>184.00</td>
<td>7.3600</td>
<td>2.5004</td>
<td>7</td>
<td>0.6248</td>
<td>-0.00886</td>
<td>5</td>
<td>5</td>
<td>0.2057</td>
<td>0.3742</td>
</tr>
<tr>
<td>100</td>
<td>388.00</td>
<td>7.7600</td>
<td>2.8731</td>
<td>6</td>
<td>0.6793</td>
<td>0.09177</td>
<td>11</td>
<td>9</td>
<td>0.4259</td>
<td>0.4613</td>
</tr>
<tr>
<td>200</td>
<td>812.00</td>
<td>8.1200</td>
<td>3.1859</td>
<td>7</td>
<td>0.6824</td>
<td>0.05107</td>
<td>20</td>
<td>11</td>
<td>0.5002</td>
<td>0.5121</td>
</tr>
<tr>
<td>500</td>
<td>2,112.00</td>
<td>8.4480</td>
<td>3.6095</td>
<td>8</td>
<td>0.6394</td>
<td>0.11911</td>
<td>44</td>
<td>11</td>
<td>0.5540</td>
<td>0.5849</td>
</tr>
<tr>
<td>1,000</td>
<td>4,053.00</td>
<td>8.1060</td>
<td>3.9402</td>
<td>9</td>
<td>0.6592</td>
<td>0.14528</td>
<td>83</td>
<td>17</td>
<td>0.5812</td>
<td>0.6341</td>
</tr>
<tr>
<td>2,000</td>
<td>8,306.00</td>
<td>8.3060</td>
<td>4.2054</td>
<td>10</td>
<td>0.6538</td>
<td>0.14016</td>
<td>151</td>
<td>24</td>
<td>0.5972</td>
<td>0.6714</td>
</tr>
<tr>
<td>5,000</td>
<td>20,316.00</td>
<td>8.1264</td>
<td>4.5344</td>
<td>12</td>
<td>0.6603</td>
<td>0.15823</td>
<td>342</td>
<td>44</td>
<td>0.6203</td>
<td>0.6827</td>
</tr>
<tr>
<td>10,000</td>
<td>40,476.00</td>
<td>8.0952</td>
<td>4.8015</td>
<td>11</td>
<td>0.6557</td>
<td>0.15351</td>
<td>645</td>
<td>64</td>
<td>0.6307</td>
<td>0.7003</td>
</tr>
</tbody>
</table>

6.5.3 Model limitations

As Leskovec et al. [134] demonstrated, real-world networks satisfy the so-called densification power law, which means that during the growth of the network the average node degree grows too; it holds that \(e(t) \propto n(t)^a\). In our case, thanks to the constant values of all lambdas, \(a = 1\), because, in one step of the growth, an unchanging number (on average) of new nodes and edges is created (see also Table 6.3). As mentioned above, it is true for the model that the growth of the \(\lambda_1\) and \(\lambda_3\) values raises the average node degree. If these lambda values grew during the growth of the network, the average node degree would also grow. So the natural question is whether, and how, the values of individual lambdas change during the growth of real-world networks. We took the list of interactions (publications) extracted from the DBLP data and divided it regularly into equally large sections, each with approximately 140,000 interactions. Then we created 20 lists of interactions, all of which started with the first interaction, the first list containing only the first section, the second list two sections, \ldots, and the last list containing all 20 sections. Furthermore, we processed each list using the procedure presented in Section 6.3 and extracted the average number of co-authors of a proactive author in total and in individual roles. How these average numbers change with the growing number of interactions, and thus how the network grows over time, is shown in Fig. 6.27. The interpretation is based on a simple observation.

Observation. When authors collaborate on a publication, the average number of existing co-authors of the proactive author grows with the size of the network. This number grows faster for the neighbors of the proactive author than for authors with whom the proactive author has not yet published. The average number of new co-authors (newbies) of a proactive author is essentially unchanged. This observation evokes the question for further research of how strong a basis a change in the parameters in connecting authors is for the densification power law.

97
We presented our observations of the behavior of authors in two co-authorship networks. We defined the proactive author of the publication and then worked with three co-authorship roles. In the analysis, we confirmed that the number of co-authors of a proactive author of the publication corresponds well to a Poisson distribution. To verify how essential a role the Poisson distribution plays in the growth of the network, we defined the 3-lambda network model based on this property. Thanks to the model, we described the impact of the average number of co-authors in various roles on the structure of the network. The model presented here does not conform to the densification power law because of its simplicity. In the experiment with the DBLP data, we showed that the average numbers of co-authors of the proactive author change during the growth of the network. These changes correspond well with the described impact of the parameters of the model on the networks that were generated. In addition to the observation of Poisson distribution, we see one further observation as a major contribution. The growing values of the average number of co-authors over time cause an increase in the average node degree. Further investigation is needed to provide verification of the results and observations presented here for other types of collaboration networks.
Email communication is a source of important information, much of which is at first sight hidden. This chapter presents an analytical tool that was created to analyze the deeper relationships in the email data. Those include relationships based on an interaction of multiple users in a team. The proposed analytical methods are based on two factors. The first factor is the context, which is a group of multiple users in combination with terms used in the communication. The second factor is the time interval in which the communication was conducted. Based on these factors, we analyze the conversations that take place and get results that are in several different forms presented to the users. We present methods for weighting conversations, users and relationships, as well as methods for finding communities associated with the specified context. Additionally, the concept of the explorative user interface is introduced.

This chapter comes from my work previously published in [P11], I have also participated in the development of the TeamNet system.

7.1 INTRODUCTION

Teams in many organizations nowadays constitute major units for dealing with various tasks. Teams consist of members with high task interdependency and shared common goals [148]. Consequently, much research has focused on models of team performance, teamwork or team effectiveness resulting in a science of teams. Some of the factors that influence team performance are communication structure, work assignments, workload, task type, interdependency, etc. [149].

Our research focuses specifically on communication structure, workload and roles that we extract from emailing activities of several persons in an organization. The focus is on project/team-oriented long-term communication. To gather information necessary for analysis we use mailboxes which are a great source of data since they include information about users (sender, recipient, CC, and BCC), mails (subjects, message bodies, attachments) and sequences (answers, forwarded emails). Every team, formed formally or informally, has throughout the time a different workload. Long-term email communication in the team is reflected in the mailboxes of individual team members. Analysis of email data can reveal interesting and, at first sight, unseen information. Much of semantics is hidden in the data. This semantics can be visualized in different ways to obtain a result that can then be used for decision support.
The first step is to obtain the data. It is necessary to choose the team members, from whose email addresses the data for analysis will be extracted. After that, a simple procedure starting with extraction and ending with exploratory analysis of the communication, is applied.

During the analysis of team’s performance, the demandingness of individual conversations, user’s share in them and important terms (keywords) are measured and assessed. The result is a detailed knowledge of the community, especially the major users and strong ties between them, and also about important terms in communication. This knowledge is used to produce an understandable visualization of the community and the details of its communication.

We present a novel approach to email mining with a focus on teams and exploration using the TeamNet application. We evaluate the method in a case study of real-life email communication of Inflex LLC company.

7.2 RELATED WORK

The major tasks that involve investigation of email involve spam detection, email categorization, contact analysis, email network property analysis and email visualization [150]. Our work covers the areas of email network property analysis and visualization. We will focus on previous work in those areas.

7.2.1 Email network analysis

Techniques and models of social network analysis (SNA) have been applied to discover the relationships between people, groups, and organizations from email networks. Many studies [151, 152, 153, 154, 155] used Enron email corpus as a dataset due to the lack of large public email corpus.

Diesner et al. [156] constructed directed graph from relations (sender to recipient) where edges were weighted by the cumulative frequency of emails exchanged in a time range, then applied SNA methods (network measures, centralities, etc.). Rowe et al. [155] constructed an undirected graph with edges representing communication between two accounts and introduced a social score measure for a user (based on number of emails, response score, clique scores, centrality measures etc.) which is then used to automatically reconstruct the social hierarchy. Chapanond et al. [152] used network metrics and spectral analysis to study both directed and undirected email graphs constructed by changing the value of the threshold (e.g. minimum number of emails). The difference between such studies and our approach is in the construction of the communication network (relations through conversations), also in our approach this network is only one of the views derived from email activities. Furthermore, our tool is exploratory, it allows
the user to refine the context, i.e. view of the data, by adding relevant persons or topics to the context, or changing the time span.

Some models were designed to work with the language content or topics besides relations. McCallum et al. [153] presented the Author-Recipient-Topic (ART) model that discovers discussion topics (using probabilistic language modeling) in a corpus of messages, extended also to the Role-Author-Recipient-Topic (RART) model to capture multiple roles (by clustering sets of correlated topics). In our approach, only email subjects are processed to extract keywords that are then associated with a conversation, but roles are defined independently of those keywords.

Stuit and Wortmann [157] use the business process modeling language to visualize the process. They construct message threads (conversations) in the same manner used in our approach (utilizing message header fields) and also use the conversation as a basic unit of observed communication. However, we do not analyze each conversation as a unit but use a set of them together to assess the social aspect of communication.

7.2.2 Email visualization

Visualization tools can be used to analyze an individual email account or email archives. NodeXL [158] tool includes an email import tool where analysts can generate email networks based on the sender/recipient fields of an email corpus. It generates direct reply networks [159] and SNA metrics (in-degree, out-degree, centrality, etc.) of those networks can be used to identify important people or social roles (by creating ego-networks of each contributor). NetLens-Email [160] is a system designed to support exploration (using queries) of the content-actor network in large email collection providing a visual representation of the data over several attributes (e.g., distribution of the number of emails by a time period), or distribution of people by status (e.g. sender vs. receiver). Themail [161, 162] is another tool for visualization of relationships (interaction histories) in email archives. It is designed for an individual’s email archive, extracting keywords from the content of exchanged messages and visualizing how they change over different time periods. Our approach provides ego-centered visualization of email network around selected user, but the main focus is on teams, by selecting a group of users, the visualization provides the view of email network for this group.

7.3 BACKGROUND

In this section, we define the key concepts that we use in our email analytics tool.

7.3.1 Conversations

Definition 25. (Conversation): Conversation is a set of emails which:
begins with a single email
• has at least two emails where senders are different
• other emails are either replies or forwarding of some email from that conversation

Conversations can contain multiple branches of different length. A conversation can be more or less taxing for the participants. From our observations, we estimated several factors that influence the difficulty of conversations. Conversation is demanding when:
• has many emails in total
• has many emails in a longest thread (branch)
• goes through many days
• has many emails in days when users are conversing
• has many emails where senders are different

**Definition 26. (Conversation demandingness):** Let C be the conversation, M be the total number of emails in a conversation, \( x \) is the logarithm base empirically set to 1.4, \( M_{\text{max}} \) be the number of emails in the longest branch, \( N \) be the number of days in which participants conversed, \( N_{\text{max}} \) be the maximum number of emails in one conversation day, \( S \) be the number of senders in the conversation. Then \( D(C) = D(M, M_{\text{max}}, N, N_{\text{max}}, S) \), \( D(C) \) is demandingness of conversation \( C \), defined by Equation 7.1.

\[
D(C) = \sqrt{\frac{\log(M - 1)^2 + (N - 1)^2 + N_{\text{max}}^2 + F^2}{M}}
\]  

(7.1)

**Remark 8.** For further calculations, we use the normalized conversation demandingness \( Q(C) \in [0, 1] \), \( Q(C) = D(C) / R \), \( R \) is the maximum value of \( D(C) \).

### 7.3.2 Roles

User’s roles are detected from his/her participation in a conversation; it can be either active or passive. On that account we divide the roles into two groups, we call the initiator, solver and co-solver active roles and the rest passive roles.

**Initiator:** the sender of the first mail in the conversation

**Solver:** the recipient of the first mail (not a copy) who sends at least one mail during the conversation

**Co-solver:** the sender of some mail (not a copy) in the conversation

**Invited:** the recipient of the first mail (not a copy) in the conversation
7.3 Background

- **Co-invited**: the recipient of some mail (not a copy) in the conversation
- **Notified**: the rest

**Definition 27. (User weight in conversation):** Let $C$ be the conversation, $U$ be the user in the conversation $C$, $All$ be the number of emails in a conversation, $From$ be the number of emails in which the user $U$ is the sender, $To$ be the number of emails in which the user $U$ is the receiver, $Copy$ be the number of emails in which the user $U$ is in CC or BCC (and not in $To$ or $From$). Parameters $f, t, c$ give importance to each component; their sum equals 1. Then $w(U, C)$ is user weight in the conversation, defined by Equation 7.2.

$$
w(U, C) = f \cdot F + t \cdot T + c \cdot C
$$

$$
F = \frac{\text{From}}{\text{All}}, f = 0.7
$$

$$
T = \frac{\text{To}}{\text{All}}, t = 0.2
$$

$$
C = \frac{\text{Copy}}{\text{All}}, c = 0.1
$$

(7.2)

Because weight $w(U, C)$ does not reflect the intensity of the conversation from a global perspective (different user weight in easy and challenging conversation), we use normalized conversation demandingness $Q(C)$ to convert the user weight according to the demandingness of conversation, so $w_n(U, C) = Q(C) \cdot w(U, C)$.

**Definition 28. (Term weight):** Let $C$ be the conversation, $T$ be the term, then $w(T, C)$ is the weight of term $T$ in conversation $C$, $w(T, C) = Q(C)$ if conversation $C$ contains term $T$, $w(T, C) = 0$ if conversation $C$ does not contain a term $T$.

**Remark 9.** Based on statistical analysis, we found out that the vast majority of conversations does not change terms in the email subject. Therefore, the weight of each term in the conversation equals normalized demandingness of the conversation.

**Definition 29. (Relationship weight):** Let $C$ be the conversation, $U_1$ and $U_2$ be users in the conversation $C$, then $w(U_1, U_2, C)$ is the weight of relationship of these users in the conversation, defined by Equation 7.3.

$$
w(U_1, U_2, C) = \text{MIN}(w(U_1, C), w(U_2, C))
$$

(7.3)

7.3.3 Context and community

A context is a group of specified users and terms (from email subjects). The specified context is a subject for further analysis where the joint communication of specified users concerning the occurrence of specified terms is examined. Key results of this analysis are:
Finding the community is based on context parameters (a group of users, a group of terms, or a combination of users and terms). A community based on the specified group of users is constructed as follows:

1. Enter the users and mark them as *context users*.
2. Find all conversations in which all context users together in an *active* role. Mark these conversations as *community conversations*.
3. To the context users add users who are in an active role in at least one community conversation. Mark these users as a *community*.
4. Find all terms that appear at least in one community conversation. Mark these terms as *community terms*.

A community based on the specified group of terms is constructed as follows:

1. Enter the terms and mark them as *context terms*.
2. Find all conversations in which all context terms appear together. Mark these conversations as community conversations.
3. Find all users who are in active role in at least one community conversation. Mark these users as community.
4. To the context terms add terms that appear in at least one community conversation. Mark these terms as community terms.

Community based on the specified combination of users and terms is constructed as follows:
1. Enter the users and mark them as context users.

2. Enter the terms and mark them as context terms.

3. Find all conversations in which are all context users together in an active role and where all context terms appear together. Mark these conversations as community conversations.

4. Find all users who are in an active role in at least one community conversation. Mark these users as a community.

5. To the context terms add terms that appear in at least one community conversation. Mark these terms as community terms.

7.4 System Overview

In this section, we describe basic steps of our approach starting with an individual’s mail account and ending with calculated statistics, roles, etc. The process includes data collection, pre-processing of data, aggregation of data from multiple user accounts and data analysis, see Fig. 7.1.

![Figure 7.1: System overview.](image)

7.4.1 Pre-processing

To extract data from email account the system is using a separate application that contains the connectors to email clients (MS Outlook, Thunderbird) or connects via IMAP or Gmail. The output of the application is an XML file (dataset) ready for further use. The pre-processing tool is using the header fields [163] (Message-ID, In-Reply-To, References) to reconstruct conversations (threads) and message subjects to extract keywords (terms). Pre-processing doesn’t utilize the message body nor attachments; the reasons are legal restrictions on access to personal information. After the pre-processing of
emails from an individual email account produced XML file can be then imported into TeamNet application. In case of a single file from just one person, the resulting views provide insight only from this ego-centric perspective. The intended usage for TeamNet is with a set of multiple XML files collected (in the organization) from different persons.

### 7.4.2 Data import

Collected XML files are processed together and aggregated while removing duplicities, into one set and stored in SQL database. SQL database enables effective and quick work with five types of entities: users and the relationships between them, with the keywords (terms) from communication and relationships between them, and with conversations. Specially designed algorithms work with these entities.

### 7.4.3 Analysis method

Conversations are weighted according to their demandingness in terms of time cost or complexity. User roles are derived from the type of user’s participation in the conversation (see Section 7.3.2), we also calculate the user role’s ratio (weight) in the conversation. The calculation of weights is based on definitions and equations in Section 6.4.

### 7.4.4 User interaction

After the pre-processing, the work with TeamNet application starts with a query that defines the context. It can be a single user (term), a group of users or a combination of user(s) and term(s). Part of this query is also the selection of desired time span. Based on the selected context, the conversations are filtered, and user (term) weights and user role ratios for this context are aggregated. System algorithms automatically detect the characteristic of the team (defined by context) and provide different ways of visualization. The flow of user interaction is depicted in Fig. 7.2.

![Figure 7.2: TeamNet usage after importing email accounts.](image-url)
The views include the charts (basic statistics), activity heat map, graph of main domains, user network, term cloud/network, communication roles, etc. Every view offers a different visualization of the same context. The views themselves are interactive and allow for further refinement of the context. By changing the time span, it is possible to observe how roles in the team change over time, or how the surrounding network evolves. The information from the views about the team’s workload may help to manage resources more efficiently.

Remark 10. Context (users, terms and the time span) may be chosen randomly. The purpose of the analysis of the context is a selection of a maximum number of key users and terms that are most frequent in the communication within the specified context. In our case, this value is set to 50. Therefore, the output of the analysis is at most 50 weighted users and terms and, weighted relationships between them. As described in Section 7.3.3, the basis for identifying users and terms is common conversations. Over the time, however, common conversations are changing, depending on the occurrence of users and terms in conversations. At the end of the calculation, algorithms only select a limited number of users and terms and then filter conversations. Selected users and terms must be understood in a context of the selected set of conversations and the specified time span.

7.5 CASE STUDY - INFLEX LLC

Inflex LLC is a small computer software company founded in 2004, developing desktop, web, mobile and Smart TV applications tailored to customers’ needs. The company provides help desk support to their customers through phone and particularly email. Number of employees is around ten and number of projects is around fifty. All projects depend heavily on email communication. Due to the small number of employees the hierarchy structure is flat, most of them are developers, one businessman/manager plus one more executive.

7.5.1 Inflex LLC dataset

Email data were collected from six individual accounts (most included more than one email address per account), this team consisted of four developers, one businessmen and one executive. Altogether, 156,876 messages from 16,740 users (email addresses) in the period from 1.1.2010 to 21.1.2015 were analyzed. For details see Tables 7.1 and 7.2.

7.5.2 Project example

The outputs from TeamNet application are always context-specific. Due to the absence of one comprehensive report with statistics about the whole collection (that could be
Table 7.1: Inflex LLC - Dataset Overview

<table>
<thead>
<tr>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emails</td>
</tr>
<tr>
<td>Conversations</td>
</tr>
<tr>
<td>Conversation branches</td>
</tr>
<tr>
<td>Users (email addresses)</td>
</tr>
</tbody>
</table>

Table 7.2: Inflex LLC - Dataset Details

<table>
<thead>
<tr>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users who are in at least one conversation</td>
</tr>
<tr>
<td>Emails sent from someone from Inflex LLC</td>
</tr>
<tr>
<td>Conversations in which at least one user is from Inflex LLC</td>
</tr>
<tr>
<td>Conversations initiated by someone from Inflex LLC</td>
</tr>
</tbody>
</table>

evaluated), we evaluate outputs using one selected company project, IntraDoc, as an example. IntraDoc is a company’s long-term project with several slightly different instances deployed for several customers. So the keyword ‘intradoc’ is selected as the context term and the time span is the whole interval (2010-2015).

First view is charts, see Fig. 7.3, where the most important persons are listed on top. The system correctly identified the manager (Hovorka), head developer (Spacil) and executive (Kudelka) as being most involved. Key topics associated with this project revolve around words like ‘version, problem, new application, modification’, etc. Key persons for all roles are also listed.

Next view is a network of all users associated with the IntraDoc project and their relations, see Fig. 7.4. Around two prominent nodes, manager (Hovorka) and developer (Spacil), are nodes (customers) with direct link only to those two. It is clear that the rest of the company employees rarely communicate directly with customers, which corresponds to the reality. This network as an output can be further analyzed using SNA methods, i.e. simple community detection algorithm [P3] was implemented to detect interconnected subgroups, see Fig. 7.5, where all company employees are detected as one of those subgroups.

Another view in Fig. 7.6 displays the distribution of roles among the team members. We can see the manager and head developer on top again, now with additional information about their roles, and ratio of each role w.r.t. others. We see another developer (Kvapil) on 4th place (in involvement) who acts as ‘solver’, which reflects his real role on this project. He handled the data/business layer of the product.
### 7.5 Case Study - Inflex LLC

**Figure 7.3:** TeamNet chart - major team members for IntraDoc

**Figure 7.4:** TeamNet network - users relations in IntraDoc

#### 7.5.3 Project example with limited time interval

In next example, we use the same project but focus on the evolution of relationships and workload by changing the time span. We restrict it to the interval of only one
7.5 CASE STUDY - INFLEX LLC

Figure 7.5: TeamNet network - detected subgroup - Inflex employees

Figure 7.6: TeamNet roles - team members distribution of roles in IntraDoc

year (1/2014 - 1/2015). By that time, the system was deployed and running for years, so the team’s workload is gradually declining. The communication structure between individual team members changed which can be seen in the communication network, see Fig. 7.7, where the central position of manager (Hovorka) in Fig. 7.7a is taken by the head developer (Spacil) in Fig. 7.7b, who by being at the support desk developed strong relations with customers. The communication structure also changed between organizations, see Fig. 7.8, where the partners are different in each interval.
The distribution of roles evolved in the same manner as the communication structure, see Fig. 7.9, where the most involved person became the head developer (Spacil) by being at the support desk, and the two passive addresses below him are support for customers, see Fig. 7.9b. Except for the manager (Hovorka) the rest of Inflex employees is not involved significantly anymore.

Remark 11. Take notice of the selection of time interval in the two examples above. We do not compare two different intervals of the same length; the second interval is part of the first (five years vs. one year). In the first case, we look at the project globally (using whole interval) and the importance of team members is assessed correctly (overall, the manager played the most important role). However, it would be wrong to draw con-
clclusions from those charts about how the situation looks like at the present moment, as evidenced by the different results in the second example.

### 7.5.4 Accuracy

As described in Section 6.4, the presented approach is based on measuring the demand-ingness of conversations. The calculated values are then used for weighting users, terms and relationships between them. To assess the accuracy of calculations, we performed a simple experiment. We automatically detected the list of the forty most important users of the entire dataset over the time. Furthermore, we selected all the important projects that the company has worked on. Each of the six employees (their accounts were used for the analysis) completed a questionnaire and identified most important persons in the individual projects. Afterward, a list of all persons who were identified in at least one project (75 persons in total) was created. These persons, together with the automatically detected persons, were aggregated to the small experimental dataset (85 persons in total). We classified all persons from the questionnaires as positive and the others (10 persons) as negative. Then we took into account the first ten (to forty) automatically detected persons by weight and calculated precision and recall values. The results are shown in Table 7.3.

### 7.5.5 Evaluation

By a combination of two factors, the context, and the time interval, we get a set of conversations. With deeper analysis of this set, we can detect community (participants of the conversations) and topic (terms of conversations) that characterize the behavior of the team at that time. There are thousands of such sets of conversations defined by

---

**Table 7.3**

<table>
<thead>
<tr>
<th>INVOLENCE</th>
<th>ROLES</th>
<th>SNT</th>
<th>RECIEVED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(a) 1/2010-1/2015</td>
<td>(b) 1/2014-1/2015</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Figure 7.9: TeamNet roles evolution](image)
the specified context. The communities matching these contexts vary in size from two participants to hundreds of participants. One person or a single term may appear in hundreds of conversations, and dozens or even hundreds found communities of different sizes. This effect implies the importance of explorative approach, allowing refine the selected context interactively. For reasons of efficiency of calculations, the data are preprocessed in the system, so the exploratory interface responds very quickly.

The system was piloted in two other organizations with teams of three to eight members (the first organization specialized in project management, and the second was the university). Despite skeptical expectations, the result for team members was surprising. Mainly for two reasons:

1. Outputs from directly specified context matched expectations.

2. The exploratory interface facilitated discovering situations that in detail showed surprising outcomes (e.g. low or excessive workload and interconnectedness of some people).

7.6 Discussion

In this chapter, we proposed methods that are focused on the analysis of email communication of small teams. These methods were designed so that the basic unit of communication was the conversation. A weighing method was designed for those conversations, being also the basis for weighting users, terms and relationships in the communication. We also presented algorithms for finding communities specified by the context and selected time interval. All the procedures presented in this chapter were used in the implementation of the analytical tool that was deployed in several organizations (working in project teams to a large extent) as a pilot project. This analytical tool is built on an exploratory user interface that provides a combination of different views of the analyzed data. Despite the very good feedback, pilot users missed automatic interpretation for displayed outputs. These are largely sociological tasks that are not simple. The second problem is the ability continuously to process incoming messages and to provide more detailed information about the development of communication.

Table 7.3: Precision and Recall (the first n-selected persons)

<table>
<thead>
<tr>
<th>Selected</th>
<th>TP</th>
<th>FP</th>
<th>TN</th>
<th>FN</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>0</td>
<td>10</td>
<td>65</td>
<td>1.00</td>
<td>0.13</td>
</tr>
<tr>
<td>20</td>
<td>15</td>
<td>5</td>
<td>5</td>
<td>60</td>
<td>0.75</td>
<td>0.20</td>
</tr>
<tr>
<td>30</td>
<td>21</td>
<td>9</td>
<td>1</td>
<td>54</td>
<td>0.70</td>
<td>0.28</td>
</tr>
<tr>
<td>40</td>
<td>30</td>
<td>10</td>
<td>0</td>
<td>45</td>
<td>0.75</td>
<td>0.40</td>
</tr>
</tbody>
</table>
CONCLUSIONS

This thesis was focused on investigating local characteristics of social networks and application of local methods on large-scale networks. The advantage of this approach is that the knowledge of the entire graph structure is not required. Although the methods are mostly applied to collaboration networks, they are applicable to any other types of networks, e.g. various biological networks, the world wide web, citation and friendship network, etc. Only depending on the context, the local properties (centrality, communities) have a different meaning. Some methods work also on any vector data, as was done in works on sampling. Vector data in general can be transferred into network representation and visualized which may help investigation of their structure.

Contributions of this thesis can be summarized in the following way:

This thesis proposes two new centrality measures for node importance:

- The first proposed measure - local dependency is designed to reflect the unequal (non-symmetrical) relationship of nodes. Dependency centrality is computationally similar to classical degree centrality, however, it is able to capture more precisely the importance of a node in a network. It may be used for the ranking of nodes or for the transformation of an originally unweighted network to a weighted network.

- The second proposed measure - local representativeness aims at finding the representatives - object which due to being nearest neighbors of most of the nodes in their surrounding may serve as a substitute to them. Sampling based on selecting only those representative objects is able to efficiently reduce the data using local information only. Local representativeness is also the basis for our method of graph construction LR-Net, which can be used for transferring vector data to networks, so that the relationships in the data can then be visualized in a natural way.

Next, we presented our experiments with real large-scale co-authorship networks:

- Using self-organizing map to identify various roles of nodes in the social network based on their topological properties.

- We proposed a simple way to visualize the evolution and dynamics of author’s topic (keywords and short phrases extracted from paper titles).
8.1 future work

- We determined the statistical distribution of different types of co-authors of publications and tried to determine how strong a role the Poisson distribution plays in the development of the network. Therefore, using only this single characteristic, we defined a process which results in network growth. In our experiments, we show through a wide range of properties that the model described by such a process generates networks with properties close to real-world co-authorship and social networks in general.

Finally, we proposed methods for the analysis of email communication of small teams:

- The proposed methods take the conversation as a basic unit of communication. A weighing method was designed for those conversations, being also the basis for weighting users, terms and relationships in the communication.

- The implementation of the analytical tool TeamNet was developed and deployed in several organizations (working in project teams to a large extent) as a pilot project.

8.1 future work

For future work these approaches will be applied to new areas such as data from health sciences to help presenting and interpreting data from medical practice. Also other applications for either reduction or construction of networks will be considered when investigating the properties of representative samples.
8.2 Author’s Bibliography

Articles in Journals with Impact Factor


Articles in Conference Proceedings (Web of Knowledge, Scopus)


8.2 Author’s bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Andrew McCallum, Andres Corrada-Emmanuel, and Xuerui Wang. “Topic and role discovery in social networks”. In: Computer Science Department Faculty Publication Series (2005), p. 3.


126
Bibliography


[124] C Lin, Qiaozhu Mei, Yunliang Jiang, Jiawei Han, and Shanxiang Qi. “Inferring the diffusion and evolution of topics in social communities”. In: Social Network Mining and Analysis 3.d4 (2011), p. d5.

[125] Cindy Xide Lin, Bo Zhao, Qiaozhu Mei, and Jiawei Han. “PET: a statistical model for popular events tracking in social communities”. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM. 2010, pp. 929–938.


[127] Yizhou Sun, Jie Tang, Jiawei Han, Manish Gupta, and Bo Zhao. “Community evolution detection in dynamic heterogeneous information networks”. In: Proceedings of the Eighth Workshop on Mining and Learning with Graphs. ACM. 2010, pp. 137–146.

Bibliography


Bibliography


Bibliography


