Detection and Analysis of Overlapping Community Structures for Modelling and Prediction in Complex Networks

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

MSc
Mohsen Shahriari
aus Birjand, Iran

Berichter: Universitätsprofessor Dr. rer. pol. Matthias Jarke
Privatdozent Dr. rer. nat. Ralf Klamma
Universitätsprofessor Dr. techn. Wolfgang Nejdl

Tag der mündlichen Prüfung: 3. Juli 2018

Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online verfügbar.
Abstract

Many real-world systems are known as complex networks that can be modeled by networks of interacting agents. In complex networks, vertices can be a member of densely connected components named community structures, which sparsely connect to the rest of the net. Community structures can be overlapping, such that vertices are the member of more than one community. We can see examples of overlapping communities in real life, for instance, a scientist is not only active in scientific communities but also may belong to communities of relatives, family members, colleagues, a sports club, etc. We can also observe overlapping communities in other types of networks, e.g., biological and online social networks. Overlapping members often help communities to scale up and to share their knowledge, e.g., diffusion of information in learning environments.

In this dissertation, we propose overlapping community detection algorithms that use properties such as degree mixing and information diffusion. We also build prediction models for the evolution of overlapping communities. Besides, we show the importance of overlapping community structures in the prediction of mixing patterns in networks. For this purpose, dynamics of overlapping community structures are used to propose ranking and recommender models. Our proposed algorithms in many cases outperform state-of-the-art techniques. Moreover, the algorithms have been made widely available in our WebOCD software framework.

Different applications can use the algorithms proposed in this thesis. Recommender systems can use overlapping community structure dynamics to recommend items to users in overloaded information spaces. Additionally, overlapping community structures can contribute to recommend overlapping members to experts that this helps to increase the information flow and to scale up communities. Algorithms, WebOCD framework, and applications have been validated in several settings, including large-scale informal Learning networks in the European Learning Layers project.
Acknowledgement

I would like to express my deepest appreciations to all the persons who contributed to accomplishing this work.

First, I would like to thank Professor Matthias Jarke for accepting me as a researcher in his chair and supporting me with his thoughtful comments.

Second, my most profound appreciation goes to Dr. Ralf Klamma for his supervision and continued support. He not only gave me the most thoughtful and most in-depth feedback but also his motivating comments could extend my view and passion towards my scientific career. I would extend my gratitude to my colleagues at the i5 Institute of RWTH Aachen University including Dr. Dominik Renzel, Petru Nicolaescu, Istvan Koren, Peter de Lange, Kateryna Neulinger, Dr. Anna Hannemann and Dr. Zinayida Petrushyna, Dr. Michael Derntl, Georgios Toubekis, Reinhard Linde, Tatiana Liberzon, Daniele Glöckner, Claudia Puhl and Bernhard Göschlberger.

I am very blessed to have such a great role model, my late father Parviz Shahriari, whose thoughts have always been with me during my whole life. His thoughts inspired the passion and strength for completing this mission. My dearest, most affectionate, merciful and the heart of my life, my mother Aghdas Bolouri whom I owe her my every single piece of improvement. Moreover, I am grateful to my dear brother Saeed Reza, who has always been there for me, a great brother, my closest friend and an excellent motivator. Furthermore, my appreciation goes to my elder brother Hamid Reza who has always encouraged me to pursue my scientific career. For sure, I would not have made it in science without his support. I would not forget my brothers Vahid Reza and Mahmoud Reza for all their help, support and kindness in life.

Patience, resilience and hard working are an integral part of scientific research, which I tried my best to fulfill them. I hope these could have made me not only a better researcher but also a better human being.
Kurzfassung


# Contents

1 Introduction 1

1.1 Overlapping Communities: Underpinning Tenet Governing Laws of Networks 1
1.1.1 Overlapping Community Detection 2
1.1.2 Overlapping Community Analysis and Prediction 3
1.1.3 Supporting Online Applications 3

1.2 Research Questions and Contributions 4
1.2.1 Research Question 1 (RQ1) 4
1.2.2 Research Question 2 (RQ2) 5
1.2.3 Research Question 3 (RQ3) 6
1.2.4 Research Question 4 (RQ4) 7

1.3 Thesis Overview 7

2 Concepts and Definitions 9

2.1 Introduction 9

2.2 Basic Concepts and Definitions 9
2.2.1 Complex Networks 9
2.2.2 Overlapping Community Detection 10
2.2.3 Community Evolution Prediction 11
2.2.4 Community Mapping 11
2.2.5 Ranking Algorithms and Centrality Measures 12
2.2.6 Signed Networks 15
2.2.7 Machine Learning: Classifiers and Optimization Techniques 17
2.2.8 User/Item Recommendation 18
2.2.9 Cooperation and Defection 19

2.3 Metrics and Measures 20
CONTENTS

2.3.1 Network and Community Synthetic Generators .......................... 20
2.3.2 Classical Prediction Metrics ........................................... 21
2.3.3 Information Retrieval Metrics ......................................... 21
2.3.4 Community Evaluation Metrics ....................................... 23
2.3.5 Correlation and Distance Measures .................................. 26

3 Related Work ................................................................. 31
3.1 Introduction .................................................................... 31
3.1.1 Overlapping Community Detection (OCD) ......................... 31
3.1.2 Community Mapping ..................................................... 38
3.1.3 Community Evolution Analysis and Prediction .................. 40
3.1.4 Signed Social Networks ................................................. 41
3.1.5 User/Item Recommender Algorithms ............................... 44
3.1.6 Expert Identification ..................................................... 46
3.1.7 Cooperation & Defection ............................................... 47
3.1.8 Community and Network Analytic Tools .......................... 48
3.2 Conclusion .................................................................... 49

4 Overlapping Community Detection ........................................ 51
4.1 Introduction .................................................................... 51
4.2 Proposed Overlapping Community Detection in Unsigned Networks ........................................ 53
4.2.1 Disassortative Degree Mixing and Information Diffusion ........ 53
4.2.2 Evaluation Results ....................................................... 56
4.3 Proposed Overlapping Community Detection in Signed Networks ........................................ 60
4.3.1 Signed Disassortative Degree Mixing and Information Diffusion ........................................ 60
4.3.2 Evaluation Results ....................................................... 61
4.4 Conclusion .................................................................... 72

5 Overlapping Community Analysis and Prediction Models ............. 73
5.1 Introduction .................................................................... 73
5.2 Community Evolution Prediction ....................................... 74
5.2.1 Community Evolution Prediction Features ......................... 74
5.2.2 Evaluation Results ....................................................... 76
5.3 Community-Based Sign Prediction ..................................... 81
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.1 Proposed Intra, Overlapping and Extra Features</td>
<td>81</td>
</tr>
<tr>
<td>5.3.2 Proposed Overlapping Community-Aware Ranking Algorithms</td>
<td>82</td>
</tr>
<tr>
<td>5.3.3 Evaluation Results</td>
<td>84</td>
</tr>
<tr>
<td>5.4 Conclusion</td>
<td>89</td>
</tr>
<tr>
<td>6 Applications of Overlapping Communities</td>
<td>91</td>
</tr>
<tr>
<td>6.1 Introduction</td>
<td>91</td>
</tr>
<tr>
<td>6.2 Proposed Community-Aware Baseline Predictor for Item Recommender Algorithm</td>
<td>94</td>
</tr>
<tr>
<td>6.2.1 Obtaining User-User and Item-Item Graphs</td>
<td>94</td>
</tr>
<tr>
<td>6.2.2 Build Graphs from Ratings</td>
<td>95</td>
</tr>
<tr>
<td>6.2.3 Build Graphs from Tags</td>
<td>95</td>
</tr>
<tr>
<td>6.2.4 Identifying Community Detection Algorithm</td>
<td>96</td>
</tr>
<tr>
<td>6.2.5 Estimate Ratings</td>
<td>96</td>
</tr>
<tr>
<td>6.2.6 Neighborhood-Integrated SVD (NSVD) Baseline Estimation</td>
<td>97</td>
</tr>
<tr>
<td>6.2.7 Time-Aware NSVD (TNSVD) Model</td>
<td>98</td>
</tr>
<tr>
<td>6.2.8 Community-Aware NSVD (CNSVD) Model</td>
<td>100</td>
</tr>
<tr>
<td>6.2.9 Time and Community-Aware NSVD (TCNSVD) Model</td>
<td>103</td>
</tr>
<tr>
<td>6.2.10 Community-Aware NSVD for Faster Learning (CNSVD-Fast) Model</td>
<td>108</td>
</tr>
<tr>
<td>6.2.11 Time and Community-Aware NSVD for Faster Learning (TCNSVD-Fast) Model</td>
<td>109</td>
</tr>
<tr>
<td>6.2.12 Evaluation Results</td>
<td>112</td>
</tr>
<tr>
<td>6.3 Expert Identification</td>
<td>127</td>
</tr>
<tr>
<td>6.3.1 Proposed Community-Aware Ranking Algorithms for Expert Finding</td>
<td>127</td>
</tr>
<tr>
<td>6.3.2 Evaluation Results</td>
<td>128</td>
</tr>
<tr>
<td>6.4 Cooperation and Defection: Individual and Collective Properties</td>
<td>131</td>
</tr>
<tr>
<td>6.4.1 Cooperation and Defection</td>
<td>133</td>
</tr>
<tr>
<td>6.4.2 Ranking and Cooperation</td>
<td>137</td>
</tr>
<tr>
<td>6.4.3 Cooperativity of Community Structures</td>
<td>141</td>
</tr>
<tr>
<td>6.4.4 Evaluation Protocol</td>
<td>142</td>
</tr>
<tr>
<td>6.5 Conclusion</td>
<td>161</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Online social networks have received increasing attention recently. People join these platforms due to their interests. While participating in online social networks, they generate a tremendous amount of data. Often, researchers apply data mining and machine learning methods to analyze the data and to find useful patterns. In social networks, we denote people as vertices, and their connections represent edges of the network. Online social networks are examples of complex networks showing fundamental properties such as power-law degree distribution and small-world-ness [LKFa07, DuSt98]. In social networks, some vertices are structurally more similar, and thus we may consider them as communities [Newm04]. Research has identified several measures to define communities. One of them is density, that shows vertices have a higher tendency to communicate with the nodes inside a community than the nodes in other communities [Newm04]. The other measure specifies that communities are connected components in the network [BaPo16]. Besides, network science researchers have also proposed other criteria and definitions for communities, as clusters of people who almost all know each other [BaPo16]. Next, researchers proposed the concept of strong and weak communities, which all members of strong communities have more connections to inside than outside, on the contrary, weak communities allow members to be among communities, i.e., nodes can have approximately an equal number of neighbors who reside inside and outside of the community [BaPo16]. Often, communities are overlapping that nodes can be the member of more than one community [PDFV05].

By only considering disjoint community detection algorithms, we may lose meaningful information. Hence, overlapping communities are the more realistic representation of community structures. We can observe overlapping communities in real life, for instance, a person can belong to multiple communities, e.g., communities of colleagues at the university, members of a family, relatives, people in a sports club, etc [BaPo16]. We also see traces of overlapping communities in online social networks, for instance, a person may activate in online forums with different contexts, i.e., topics about health and topics about computer science, and thus this user can be counted as overlapping among these two settings. Having overlapping roles gives the nodes the possibility to obtain information from multiple communities, which they can have a significant effect on networks. In contrast to previous research on community detection which considered overlapping communities as separate clusters or merged them into one group, recent studies showed that overlapping community structures are dense structures in networks [JaLe12]. Also, overlapping members can play important
Introduction

roles, for instance, they can connect people from different disciplines in science, or they may help the diffusion of information in learning communities. Recently, researchers have proposed various algorithms for the detection of overlapping communities; however, few studies use overlapping communities in real applications [DoJe15, CNZh15, ShPK15].

1.1 Overlapping Communities: an Underpinning Tenet

Governing Laws of Networks

The study of overlapping communities is connected with three main areas. First, we need algorithms to detect overlapping community structures. Usually, the explicit community structures in networks are unknown, and we use overlapping community detection algorithms to identify them implicitly. Second, communities evolve, and they can support networks in dynamic situations. On the one hand, communities are temporal, and we need algorithms to predict their evolutions. On the other hand, we can use overlapping community structures to predict mixing patterns in networks. Finally, we can apply (overlapping) communities to support online applications. Examples include recommender systems, expert identification, and cooperation & defection. In the following subsections, we investigate these directions.

1.1.1 Overlapping Community Detection

Several challenging problems are concerned with communities that attract network science researchers to approach them. Although we require algorithms to detect communities, we need first to define and describe them. Network science researchers may not find a unique definition for communities; however, some research works enumerated connected-ness and density among properties of communities [Newm04, BaPo16]. These criteria may not hold in all circumstances and may narrow our minds to define and detect communities. In contrast to the density of connections, other criteria including users’ similar interests, properties and innovations may cause more dense interactions and help to form communities [BHKL06]. In this regard, the challenge of detecting community structures in a meaningful and useful way still exists. Many research works have approached the topic from different aspects and improved both the complexity and precision of the overlapping community detection algorithms since 2004 [CWZh11, PrTh13, JaLe12, PRES11, WLW*12, XiKS13].

Local or global, static or dynamic, structural or contextual are among properties of available overlapping community detection (OCD) algorithms. However, these algorithms are sometimes complicated to implement and do not reach a suitable performance; moreover, no one proved their goodness in specific applications. To put it another way, simple algorithms with reliable social dynamics may be more effective than others. Furthermore, most of the recent studies do not have an integrative perspective regarding community analysis. In other words, they propose an algorithm; however, they barely applied the proposed algorithm on community application domains and community prediction models.
1.1.2 Overlapping Community Analysis and Prediction

Not only detection of community structures is important, but also their analysis should be comprehensive and manifold. To put it differently, we should investigate communities from different aspects and discover the dynamics behind them. Corresponding findings regarding communities can be used to devise community detection algorithms; otherwise, algorithms may not be practical in real cases. For instance, a critical property of communities is the overlapping structures; however, our knowledge regarding the importance of overlapping members and overlapping community structures is imperceptible, and few applications employ the importance of overlapping members [JaLe12, JaLe14]. Traces of overlapping members can be found in more than one community, which we assume they might contain more information in comparison to other nodes. For instance, in informal learning environments, they can scale up community borders and help the diffusion of innovations. As such, we are interested in figuring out how important overlapping members are in social networks. Additionally, communities encounter various evolution stages such as birth, death, growth, shrink, split and merge. In other words, two communities combine to a bigger community or another community may be divided into several smaller communities [BSKa13, TRZa14]. However, our knowledge regarding the essential indicators for such particular evolution pattern is imperceptible. This dissertation approaches to identify critical signs to detect future of communities.

1.1.3 Supporting Online Applications

Communities are related to other areas of research. We investigate three main applications - item recommender, expert identification and cooperation & defection - by using dynamics of community structures. First, let us consider the example of a item recommender. Although current recommender algorithms consider personalization and user preferences, they do not use the collective behaviors and patterns of users. They use daily, monthly user/item preference drifts but they barely consider community drifts. In other words, temporal dynamics of community structures have not been used to recommend items to users.

Communities can also be used to support identifying of experts in social networks such as question & answer forums. Experts respond to questions raised by other users. Often, they are eligible to judge the success of an approach, strategy or activity [ChNa08]. In other words, they may help novices to accomplish a task in a question & answer forum via answering askers’ questions [LCKo05]. Expert identification algorithms sort the users of a network based on importance to assign a rank value to them. Expertise of a person is a domain-specific concept, which a person might be an expert in one domain; however, he might not have the high level of knowledge and importance in other domains. To identify experts, we rank members of a social network by putting more weights to opinions of users sharing the same community. We presume that people in the same community might have more information regarding the community context.

Concerning cooperation and defection, it is possible to investigate cooperativity of overlapping communities. Network science researchers have already studied network cooperativity by using the game theoretic approaches, and they have identified the network properties which favor cooperation [RBH*14, LPTo08]. However, cooperation and the defection of community structures in
Introduction
different contexts is yet an open area for research. Knowing about the cooperativity of community structures and its relation to community properties may help us to estimate the willingness of users to cooperate in specific settings, e.g., informal learning environments. Furthermore, we can apply cooperation and defection to determine whether an expert is willing to cooperate with other users. Or as another example, we can compare the amount of collaboration in two different environments by measuring their communities’ cooperativity.

1.2 Research Questions and Contributions

In all the iterations of this work, we had three principles in mind. First, we exposed ourselves to analytic areas such as overlapping community detection to propose proper models and algorithms to describe and predict network evolution. We evaluated the suitability of the proposed algorithms by simulations and analytic metrics. Second, we applied the proposed algorithms and models to different applications as well as various domains. As such, community-aware item recommender models, expert identification, link prediction, and cooperation & defection are among application areas that algorithms and models are employed. Third, we combined technological innovations with the analytic part to improve social interaction with users. We implemented algorithms and models as RESTful Web services that can be useful for different stakeholders.

1.2.1 Research Question 1 (RQ1)

To what extent can we improve detection of overlapping community structures by using simple but rather effective dynamics such as degree mixing, information diffusion, and opinion formation?

To address this research question, we have proposed novel overlapping community detection algorithms using different properties as follows.

- The first algorithm is DMID (Disassortative Degree Mixing and Information Diffusion). It is a two-phase OCD algorithm based on two social dynamics. The first used phenomenon is disassortative degree mixing, which is a standard measure showing dissimilarity among neighbors of nodes. In the first phase, disassortative degree mixing alongside the degree of nodes are combined to compute influential nodes. DMID uses information diffusion in its second step. To simulate information diffusion, we may consider several strategies that one of them is the network coordination game. Considering binary states for opinions like A and B and equal resistance threshold for the nodes contribute to agree to a simple diffusion strategy. We evaluated DMID with different metrics including statistical and knowledge-driven. We also applied DMID on various application domains such as community evolution prediction, expert identification, and recommender systems. DMID, in many cases, outperformed the-state-of-the-art techniques concerning the evaluation metrics. In our experiments, we concretely identified to what extent DMID improves accuracies.
1.2. RESEARCH QUESTIONS AND CONTRIBUTIONS

- We also extended DMID to the case of signed social networks, which positive and negative connections exist among members of the social network. The dominance of negative connections is considered both in computing leaders and the cascading process. While this algorithm is suitable for signed social networks, it is named Signed Disassortative degree Mixing and Information Diffusion (SDMID). We extensively evaluated SDMID on synthetic and real-world networks. SDMID outperformed other OCD algorithms regarding running time and statistical metrics, e.g., modularity. Moreover, we applied SDMID in a sign prediction task that it produced competitive prediction accuracies.

1.2.2 Research Question 2 (RQ2)

*To what extent can we predict the evolution of network structures, i.e., mixing patterns and community structures, using collective dynamics of overlapping communities?*

We constructed prediction models that use overlapping community structures of the network. We mention community evolution prediction and sign prediction models.

- We used static and temporal community properties to predict the evolution of overlapping communities. We considered several evolution events such as dissolve, split, merge and survive and used a machine learning classifier to predict the events using community properties. Our results identified to what extent each feature contributes to the prediction accuracies. Moreover, we could compare the effect of different community detection algorithms on prediction accuracies.

- We connected the sign prediction problem in network science with predictive features of signed graphs such as *intra*, *overlapping* and *extra* node feature types. Also, we applied community-aware ranking algorithms to the prediction of positive and negative connections in online social networks. We employed the sign prediction problem to make models based on simple in-degree and out-degree features of trustor and trustee. In other words, each of incoming and outgoing nodes’ neighbor of trustor and trustee can fit in one of intra, extra and overlapping categories. Additionally, we extended original HITS and PageRank to Overlapping Community-Based HITS (OC-HITS) and Overlapping Community-Based PageRank (OC-PageRank). OC-HITS and OC-PageRank differentiate among intra, overlapping and extra nodes in their updating rules through $\alpha$, $\beta$ and $\gamma$ coefficients. Ranking algorithms connected the OCD with the link prediction problem of signed networks. Our experimental results indicated that overlapping community structures play an important role, which their prediction accuracies are competitive compared to other cases.
1.2.3 Research Question 3 (RQ3)

How much effective is dynamics of overlapping community structures to improve application areas such as recommender systems and expert identification?

We applied dynamics of overlapping communities to three application domains such as recommender systems, expert identification, and cooperation & defection.

- We proposed temporal and community-aware recommender algorithms that use neighborhood and factor models of recommendation. As such, we considered the concept of community drifts in the proposal of a recommender system. We proposed models such as Community-Aware Neighborhood-Integrated SVD (CNSVD) and Time and Community-Aware Neighborhood-Integrated SVD (TCNSVD) that are extensions to Neighborhood-Integrated SVD and Time Neighborhood-Integrated SVD models. Our experimental results showed that TCNSVD model outperforms the state-of-the-art approaches as well as the baseline methods with a large margin concerning item ranking and accuracy metrics. Furthermore, we could make a compromise between precision and runtime by proposing TCNSVD-Fast and CNSVD-Fast models that produce competitive accuracies.

- We extended classical ranking algorithms to obtain community-aware ranking algorithms to use them in expert identification application (see chapter 6). We assumed that when nodes reside in the same community, their opinions towards each other are more reliable. In HITS (see chapter 3), we made changes to hub and authority vectors, on the contrary, the random walker was adapted in the PageRank algorithm (see chapter 3) to activate the effects of community structures. Our experiments showed that using community structures can improve expert identification - we mention the concrete results in chapter 6.

- We connected cooperation and defection of agents to ranking values of nodes and properties of implicit community structures. Researchers simulate cooperation and defection problem through prisoner’s dilemma game. First, we calculated the correlation of rank values of nodes with their cooperativity through node level and neighborhood level strategies. We could indicate a high amount of correlation among specific ranking algorithms and willingness of nodes to cooperate (see chapter 6). Besides, we applied prisoner’s dilemma on implicit community structures detected from two network types, i.e., open source software (OSS) development, and learning forums. As such, we computed the correlation between network and community properties that we observed a negative correlation between main community properties, i.e., density, size, etc., and their respective average cooperativity values. Last but not least, in our simulations, we figured out that OSS forums are more cooperative than learning forums.
1.2.4 Research Question 4 (RQ4)

Can we offer a framework which facilitates the analysis of overlapping communities for a broader range of users?

We developed and evaluated an analytic framework named WebOCD.

- WebOCD handles preprocessing, postprocessing, calculation of OCD metrics, generation of graphs and community benchmarks, and detection of overlapping communities. We developed OCD as a RESTful Web service to give it the flexibility to be easily integrated with other applications and services. We supported WebOCD with a Web client that users, i.e., researchers or students can log in and perform analytic tasks. Development of WebOCD also gave us the platform to conduct experiments on expert identification and recommender system tasks. Moreover, we used WebOCD in the Learning Layers project as well as a seminar on the topic of overlapping community detection. Our online evaluation results with users show that WebOCD facilitates interaction of users with an analytic software on the subject of overlapping community detection (see chapter 7).

1.3 Thesis Overview

The organization of the dissertation is as follows:

- In chapter 2, we introduce complex networks and define the basic concepts and notations required to track the rest of the document. We also describe the necessary metrics for evaluation of the algorithms and prediction models.

- In chapter 3, we investigate the related works and essential literature regarding overlapping community detection, community evolution prediction and community applications. We also explain related works concerning signed social networks, recommender systems, and expert finding algorithms.

- Chapter 4 discusses the structural overlapping community detection algorithms that answer RQ1. We introduce DMID and SDMID algorithms; afterward, we compare them with other methods.

- Chapter 5 is about prediction models, which we answer to RQ2. In the beginning, we define the community evolution prediction model which is implemented to predict future of communities. In the second part of this chapter, we describe the link prediction model that indicates the significant effect of overlapping nodes in social networks.

- In chapter 6, we answer RQ3 and investigate the applications of overlapping community structures. In the first section of this chapter, we investigate the user/item recommender algorithms, which is an extension to neighborhood and factor models. In the second part of this chapter, we formally define expert identification. Here in this chapter, we introduce community-aware ranking algorithms to identify experts in question & answer forums.
the last part of this chapter, we investigate community cooperativity through game theory approaches.

- In chapter 7, we address RQ4 and describe the architecture of the developed overlapping community detection framework as well as required technologies. We as well mention case studies concerning the WebOCD.

- Finally, we summarize the findings, the achievements and the future works of this dissertation in chapter 8.
Chapter 2

Concepts and Definitions

2.1 Introduction

We have dedicated this chapter to introduce the necessary concepts and terms to follow the thesis. At the beginning of this chapter, we define networks and primary metrics such as diameter and shortest path. Afterwards, we explain overlapping community detection, evolution prediction, and mapping. Moreover, we demonstrate ranking approaches including centrality metrics and classical ranking algorithms and expert identification. We describe what a social network with positive and negative links is. Regarding signed networks, we demonstrate overlapping community detection and sign prediction problems. Subsequently, we discuss frequently used machine learning classifiers including Logistic Regression, Bagging, J48 and Decision Table. Additionally, we introduce user/item recommender algorithms and cooperation & defection. Finally, we debate metrics and measures to evaluate the algorithms and approaches. These metrics include standard supervised and unsupervised machine learning measures, information retrieval measures and metrics to assess overlapping community detection algorithms.

2.2 Basic Concepts and Definitions

2.2.1 Complex Networks

A system is a collection of different elements, which is supposed to achieve specific goals. These elements might have internal interactions with each other. When the number of items in a system increase then it is not simple anymore and becomes complicated and even complex. Complex systems have some properties. Static or dynamic, casual or non-casual, discrete or concrete and reversible or irreversible features can be mentioned as for system properties [Karu95]. One example of a complex system is a complex network which elements of the system are nodes and connections among them. Moreover, we might maintain properties for either the nodes or edges. As there are a tremendous number of nodes and edges, analysis and knowledge extraction from these types of
systems, i.e., networks, are non-trivial. Large scale temporal behavior, chaotic and periodic phenomena, robustness and complicated influence dynamics, etc., make the analysis of these networks complex.

Network science researchers analyze online social networks, biological, technical and protein-protein interaction networks by mapping them to graphs. In other words, when there are vertices with their respective connections, we can assign them to graphs with nodes as vertices and edges as connections among them. To better understand a complex network, we can indicate it with \( G(V,E) \) which \( V \) is the set of nodes and \( E \) is the collection of edges among them. In other words, \( V \) is a set that comprises \( V = \{V_1,V_2,\ldots,V_N\} \) and \( E \) contains \( E = \{E_1, E_2, \ldots, E_m\} \), respectively. Each connection \( E_i \) is constituted between two arbitrary nodes \( V_i \) and \( V_j \). Neighbors of node \( i \) are denoted by \( \text{Nei}(i) \) and degree of node \( i \) is denoted by \( \text{deg}(i) \). Moreover, list of nodes linking to node \( i \) is denoted by \( \text{Indeg}(i) \) and neighbors which node \( i \) linking to them are denoted by \( \text{Outdeg}(i) \). We denote size of these sets by \( |\text{Indeg}(i)| \) and \( |\text{Outdeg}(i)| \) and size of sets \( V \) and \( E \) is denoted by \( N \) and \( M \). In unsigned social networks, all the connections are positive, and the adjacency matrix can either take 0 or 1, i.e., \( A_{ij} = 0, 1 \). Additionally, networks are time-evolving, and we require to consider temporality in our notation sets. Hence, we can denote a time-evolving graph at time \( t \) with \( G^t(V^t,E^t) \) which \( V^t = \{V_1^t, V_2^t, \ldots, V_N^t\} \) and \( E^t = \{E_1^t, E_2^t, \ldots, E_m^t\} \). If we consider two connected nodes \( i \) and \( j \), then the shortest path length of all the paths between these two nodes is known as the shortest path, which is also referred as Geodesic or Characteristic path. Another important metric is the diameter, which can be defined as the longest shortest path between all the nodes in the whole network. Based on these two distance measures, one can estimate the distance between two nodes. Moreover, one characteristic of real-world networks is small-world-ness. That means networks are large-scale; however, nodes have relatively short paths among each other [HuGu08].

2.2.2 Overlapping Community Detection

Networks consist of densely connected components named community structures. These components are most of the time overlapping, which nodes are the member of more than one community. Often, the explicit information concerning node membership to communities is not available. Thus, networks require algorithms for the detection of the implicit community structures. In other words, these algorithms detect (overlapping) community structures based on different criteria, i.e., density, etc. As such, we identify components that are densely connected internally and sparsely connected to each other. Different algorithms generate various community resolution levels. For instance, if you consider a network among users in a learning forum, neither the information concerning the communications nor the community structures is unknown. In this case, users do not mention explicitly which community of topics, they belong, or they have expertise. So, we require applying OCD algorithms on the graph of a social network, and then we obtain several overlapping communities. Communities can be denoted by \( C = \{C_1, C_2, \ldots, C_L\} \) that \( L \) is the number of found communities and \( C_i \cap C_j \neq \emptyset \). While networks evolve, we can integrate time symbols with the notation sets. For instance, clusters can be denoted as \( C^t = \{C_1^t, C_2^t, \ldots, C_N^t\} \).
2.2. BASIC CONCEPTS AND DEFINITIONS

2.2.3 Community Evolution Prediction

Community structures are temporal, which people may join or leave the communities at different times. As community structures are implicitly detected, identifying such changes in community structures are hard to recognize or track. In a node level, one can track small changes, e.g., changes in the user behavior; however, for larger scales, i.e., community scales, one requires collective algorithms to track the implicit community changes. As an example, suppose a group of people applying for a university position and need eligibility checks via the admission committee. This group of people gathers online to share their thoughts, innovations, and experiences with each other. The users of this forum create threads of questions. Since the opening of such a position, this community has emerged. Applicants post their questions about, e.g., the required documents, the probability of acceptance and the quality of the university. Later on, this community might grow or even shrink to smaller communities based on the event and time of the year. Moreover, external and internal reasons might split this community and create some new communities. Altogether, one may recognize that communities experience different life cycles. In such an environment, we are interested in predicting such evolution over time. In the following, we formulate such developments.

Considering two snapshots $t$ and $t + 1$, we would like to match the communities, in other words, given $C^t_i$, it is intended to figure out its fate. Overlapping communities evolve, and they face different life cycles. Research has mainly identified evolutions such as birth ($C^{t+1}_i$ is only observed at time $t + 1$), death ($C^t_i$ is not observed in snapshot $t + 1$), grow and shrink ($C^t_i$ is observed at snapshots $t$ and $t + 1$ but with different sizes), split and merge (sometimes a community at time $t$ is split into several communities at time $t + 1$; similarly several communities may be merged and form a bigger community at time $t + 1$) [BSKa13]. Figure 2.1 shows different evolutions that may happen to a community. In the community evolution prediction, we are interested in applying either supervised or unsupervised learning algorithms to reliably predict what happens to a certain community $C^t_i$. The objective for community evolution prediction is to reach to a common understanding of communities and how to predict them over time.

2.2.4 Community Mapping

As communities evolve, it is required to map them over time. In other words, it is necessary to know how a specific community has changed in two consecutive time steps. In this regard, we can apply similarity metrics to map the communities. Among the approaches, we select Group Evolution Discovery (GED) to track the evolution of communities (see chapter 3).

Summary of complex networks and overlapping communities

In this subchapter, we introduced the basic concepts and problems to follow this document. First, we formally defined complex networks and community structures. We introduced community detection problem because several algorithms are proposed in chapter 4. We learned that communities in real cases are overlapping and thus we addressed overlapping community detection algorithms. Afterwards, we described community evolution prediction problem. We introduced community evolution prediction because we build community prediction models in chapter 5.
Concepts and Definitions

(a) Grow

(b) Merge.

(c) Split.

(d) Shrink.

(e) Dissolve.

(f) Birth.

Figure 2.1: This figure indicates major events that may happen to a community. They include grow, merge, continue (split, shrink), dissolve and birth. Community at time $t$ and $t+1$ are indicated on the left and right of each sub-figure, respectively.

2.2.5 Ranking Algorithms and Centrality Measures

To calculate significance and centrality of nodes in a network, we require applying ranking algorithms. Researchers have employed ranking methods in different application domains such as search engines, recommendation systems, and expert identification. Experts are nodes with a higher level of domain-specific knowledge in the network. Often, they can judge quality and correctness of an approach or strategy. Most of expert finding tasks were involved in enterprises and organizations; however, one can currently extract traces of expertise from online social network data. Experts can help novices in a question & answer forum, i.e., an informal learning or an OSS forum, to find their intended content or to find their solution [BBC*13]. In this problem, whenever a user asks a question, the system returns lists of relevant experts regarding the query. As such, we define experts from a structural point of view in the network. In other words, we mention ranking algorithms as a method of identifying experts [ShPK15]. Furthermore, in expert finding problem, it is ideal to figure out people with higher level of knowledge. Although expertise usually depends on different structural and contextual factors, we may categorize the vertices based on their ranking. Hence, in expert finding problem, we are interested to reliably rank the nodes that possess the higher level of expertise in comparison to other users in a specific domain. As such, the query context can identify such domain knowledge [YQG*13]. Furthermore, many of the ranking algorithms and expert identification approaches return experts with the higher level of expertise; however, they do not consider task management and experts’ willingness for help. Ranking algorithms are related to the tendency of experts to cooperate, and thus we investigate expert identification and cooperation of experts in chapter 6.

Based on the connection context, we can interpret rank values as reputation, expertise or social rank; we denote this value for node $i$ as $R_i$. We investigate a couple of ranking algorithms in the following:
2.2. BASIC CONCEPTS AND DEFINITIONS

Simple Degree

Simple degree (SD) measure is a standard metric, which its understanding is comfortable and its computational complexity is low (O(N)). Naturally, we count the number of neighbors of each node ($|\text{deg}(i)|$). Simple degree, which is also called prestige, had good performance in several studies, e.g., [EaKl10].

PageRank

In the 1990s, Larry Page proposed PageRank which is the most widely used ranking algorithm [PBMW98]. If we denote PageRank value of each node with $R_i$ then it is updated as follows:

$$ R_i = \beta \times \frac{\sum_{j \in E_{ji}} R_j}{\text{outdeg}(j)} + (1 - \beta) \times \frac{1}{N}. \quad (2.1) $$

Here, $\beta$ is the teleporting parameter showing the probability of starting a new walk by the surfer. The surfer is an agent that starts a walk on the network and chooses path randomly based on the probability distribution of outgoing links. This walk can start from any vertex in the network and the probability of spending the time at node $i$ is independent of its initial position.

Hyperlink-Induced Topic Search

Kleinberg proposed Hyperlink-Induced Topic Search (HITS) algorithm around 1999, and it was employed to extract useful information from World Wide Web. HITS works based on two vectors named hubs and authorities. Hubs are nodes which refer to other nodes and authorities receive links from other nodes in the network [Klei99]. We initialize the hub and authority vectors with $\frac{1}{N}$ value and apply the equation 2.2 and 2.3, as follows:

$$ a_i = \sum_{j \in \text{indeg}(i)} h_j, \quad (2.2) $$

and

$$ h_i = \sum_{j \in \text{outdeg}(i)} a_j, \quad (2.3) $$

where $a_i$ and $h_i$ show authority-ness and hub-ness of node $i$. We let the algorithm run until the convergence condition is met. Convergent $a^*$ vector is considered as the rank vector ($R$). In vector terms, we consider HITS algorithm as a power-method eigenvector computation, and thus we can rewrite the above summations as vectors as follows:

$$ a^t = A^T Aa^{t-1}, \quad (2.4) $$

$$ h^t = AA^T h^{t-1}, \quad (2.5) $$

which authority vector $a$ and hub vector $h$ are the eigenvector of $A^T A$ and $AA^T$, respectively.
Concepts and Definitions

Closeness Centrality

Closeness Centrality (CL) can be defined based on the average shortest distance between a node to all other nodes [EaKI10]. We can write as follows:

\[ CL_i = \frac{1}{N} \sum_{j=1}^{g_{jk}}. \]  

(2.6)

We can normalize this value with \( N \), and consider the \( CL \) vector as the \( R \) vector.

Betweenness Centrality

Betweenness Centrality (BC) is a measure which shows how many of nodes should go through you to reach other nodes [EaKI10]. It can be defined as follows:

\[ BC_i = \sum_{j < k} \frac{g_{jk}}{g_{jk}}. \]  

(2.7)

where \( g_{jk} \) is the number of shortest paths between node \( j \) and \( k \), and \( g_{jk}^i \) is the number of these paths going through \( i \). Similarly, we consider \( BW \) vector as the \( R \) vector.

Clustering Coefficient

Clustering Coefficient (CC) shows local connectivity and local information exchange in the network. We adopt the definition of CC from Watts and Strogatz [HuGu08] and compute it based on the following formula:

\[ cc_i = \frac{2(|E_{deg(i)}|)}{(|deg(i)|)(|deg(i)| - 1)}. \]  

(2.8)

where \( |E_{deg(i)}| \) is the number of edges between neighbors of node \( i \). Here, we consider \( CC \) vector equal to \( R \) vector.

Assortative degree mixing

Assortative degree mixing is the tendency of nodes to initiate connections with similar nodes, and thus this metric shows similarity and homophile among vertices [Newm03]. Litvak et al. [Liva13b] presented an approach to compute assortative degree mixing in the network as follows:

\[ \rho = 1 - \left( \frac{6 \sum d^2}{V(V^2 - 1)} \right), \]  

(2.9)

where \( d \) shows the distance between the rank values of nodes.
2.2. BASIC CONCEPTS AND DEFINITIONS

Summary of centrality metrics

We introduced ranking algorithms and centrality metrics in this subchapter. Network science applications widely used these algorithms. For instance, ranking algorithms are used to search in networks or to identify experts. Moreover, they have been used in recommender systems as well as to determine the importance of nodes in a network. Also, they have been used to identify community structures, to recommend items, etc. In this thesis, we considered classical algorithms - majorly used in chapter 6 - for expert finding. In chapters 5 and 6, we investigate how using of community structures can improve returned results by these classical ranking algorithms. Moreover, we correlate users’ cooperativity with rank values of nodes. Although ranking algorithms have the wide range of applications, here we use them for the sign prediction and expert identification problem. We also use some of these centrality metrics to the community evolution prediction problem and to compute willingness of users to cooperate.

2.2.6 Signed Networks

Signed networks are examples of online social networks where connections can be either positive or negative. Positive relationships indicate trust or friendship, and negative connections show distrust or enmity. We can observe real examples of networks with both positive and negative links in online social networks. For instance, users in Wikipedia apply positive and negative votes to select administrators for this online glossary. As another example, users of Epinions Website as a customer review site can show their (dis)trust to each other. Administrators of this Website use this information together with the ratings to decide which reviews to show to the users. We can also extract positive and negative sentiments out of texts in a forum, and thus construct networks with both positive and negative connections. Trust information can even have significant effects in judging the reliability of learning materials. Often, learners rely more on learning resources produced by users who they trust. Research has also used networks of positive and negative connections to model and analyze international relationships [EaKI10].

In this section, we formally define networks with both positive and negative connections and introduce two essential problems including sign prediction and overlapping community detection. Often researchers like to predict how the signed networks evolve in future, i.e., who will be friends and who will be enemies. Sign prediction can have practical applications in online social network Websites to recommend possible friends for users. Respectively, we can also find overlapping community structures in networks where negative connections even exist. In this regard, detecting community structures can help recommender systems, prediction models and other applications using networks of positive and negative relationships. Signed social networks are graphs \( G(V, E) \), which \( V \) is the set of nodes and \( E \) is the set of edges. Adjacency matrix regarding signed networks can take two possible values of -1 and +1. We indicate \( \text{indeg}^+(i) \) and \( \text{indeg}^-(i) \) as nodes who positively and negatively vote toward node \( i \). Respectively, we define \( \text{outdeg}^+(i) \) and \( \text{outdeg}^-(i) \) as the sets of nodes which node \( i \) positively and negatively points to them. The set of neighbors of node \( i \) is represented with \( \text{deg}(i) \). Alternatively, we represent \( \text{Nei}(i) = \text{deg}(i) = \text{indeg}^-(i) \cup \text{indeg}^+(i) \cup \text{outdeg}^-(i) \cup \text{outdeg}^+(i) \) as the set of nodes adjacent to \( i \).
Concepts and Definitions

Figure 2.2: In signed graphs, triangles can form 4 states which 2 of them are balanced and the other 2 are unbalanced and have a tendency to become stable at some point in time.

Sign Prediction and Overlapping Community Detection

The sign prediction is an extension of the link prediction problem which we build a probabilistic model to reliably predict sign for a hidden edge from a trustor to a trustee. We denote trustor by $u$ and trustee by $v$. Additionally, we extend the in-degree notation for node $i$ to $\text{indeg}^{\text{sign}}_{\text{type}}(i)$, which sign can have values of $+$ and $-$. $\text{type}$ can get three values of $\text{intra}$, $\text{extra}$ and $\text{Ovl}$ (overlapping). For example, for a sample node $j$, $\text{indeg}^{\text{sign}}_{\text{Ovl}}(j)$ indicates the set of in-degree nodes not only positively link to node $j$ but also are overlapping among identified communities. Similarly, $\text{indeg}^{\text{sign}}_{\text{intra}}(j)$ is the set of nodes that negatively points to node $j$, and also are in the same community which node $j$ belongs. Respectively, we can extend it to the extra and outgoing case by mentioning that $\text{outdeg}^{\text{sign}}_{\text{extra}}(j)$ is the set of nodes which node $j$ negatively points to them and are the member of different communities than $j$ belongs to (they are not on the same cover). To denote the size of sets, we use $||$ or $\#$ throughout this manuscript. In other words, $|\text{outdeg}^{\text{sign}}_{\text{Ovl}}(j)|$ or $\#\text{outdeg}^{\text{sign}}_{\text{Ovl}}(j)$ is the number of nodes which node $j$ positively vote for them, and these nodes are also overlapping among communities.

Like other online social networks, signed networks can have overlapping community structures. In other words, nodes can be the member of more than one community. OCD algorithms in signed networks do not only consider the density of connections but also they should consider balancing theory. Community detection algorithms look for more connected components named communities. These algorithms usually perform based on finding more dense parts of the graph. We denote the covers found by a community detection algorithm by $C = \{C_1, C_2, ..., C_L\}$, which $L$ is the number of communities and these communities overlap. In other words, for communities $C_i$ and $C_j$, $C_i \cap C_j \neq \emptyset$. To explain the balancing theory, we require investigating triangles in signed networks. Triangles can take four states of balanced and unbalanced in networks with positive and negative links. As Figure 2.2 indicates, two of the states are balanced, and the other two are unbalanced [CaHa56]. One can prove that a signed network is balanced when nodes inside communities have positive relationships and nodes among clusters have negative relationships with each other. One can estimate the quality of a community detection algorithm by the number of edges that do not satisfy the balancing property.

Summary of signed Networks

We introduced networks with both positive and negative connections, and we defined problems such as link prediction and community detection for such network types. One can model trust relations among people by using signed social networks, and one can map relations by connections of friendship or enmity types. In chapter 4, we present an algorithm to detect overlapping communities in signed graphs. In chapter 5, we perform sign prediction using community-related features.
2.2.7 Machine Learning: Classifiers and Optimization Techniques

In this section, we introduce classifiers which we employ in the sign prediction and community evolution prediction models.

**Logistic Regression**

Logistic regression uses a set of features for training and makes a probabilistic model out of the training data. The logistic regression utilizes a cost function to find the parameters of the model. The cost function is as follows:

\[
J(h_\theta(x), y) = \frac{1}{2} \times \left( \frac{1}{1 + \exp(-\theta^T x)} - y \right)^2.
\]

(2.10)

Where \( J \) is the error in the model, \( x \) is the vector of features, \( \theta \) is the vector of parameters which should be computed by an optimization approach. Finally, \( y \) contains the real class values. In our case, it is a vector of +1 and -1 values [Bish06].

**Bagging**

The bagging classifier uses some weak classifiers, which their performance is better than a random classifier. Each of them is trained with some portion of the training data. Finally, the predicted class is decided based on the aggregated votes of the whole classifiers. This aggregation decision process is known as bagging or bootstrap aggregation [Bish06].

**J48**

J48 is another classifier which is categorized in the family of decision trees and uses features to constitute decision tree over the training data [Mitc97]. In sign prediction, leafs of the tree would be -1 and +1 nodes.

**Decision Table**

Decision tables infer some rules for predicting the classes. Flowcharts and if-then-else statements are created to deduce the closest classes.

**Bayesian Network and Naive Bayesian**

Bayesian Network (BayesNet) and Naive Bayesian (BayesNaive) are based on the Bayesian probability theory. Bayesian Network is a form of probabilistic graphical models which uses directed graphs to show probability distributions. Each node in BayesNet indicates a random variable and edges denote the transition probability to transfer from one state to the other. It uses training data to
build the model and compute the probability dependencies. Based on statistical inference, it clas-
sifies test cases to their predicted classes. Additionally, BayesNaive applies the original Bayesian
theorem to make the statistical inference of the classes which in our case are -1 and +1 values
[Bish06].

Stochastic Gradient Descent

Stochastic gradient descent (SGD) is an extended version of gradient descent for optimization of a
cost function or objective function. SGD iterates over the training records and updates the true value
of gradient step by step. In other words, SGD shuffles the training records, which several iterations
on the training data is possible. Each training record can be used to adjust the true gradient value.
We run the algorithm until meeting the convergence condition [Bott12]. In simple form, we can
write the updating equation as follows:

\[ x(t + 1) = x(t) - \alpha f'(X). \] (2.11)

Summary of Machine Learning

In this subchapter, we introduced a couple of machine learning classifiers including logistic re-
gression, bagging, J48, decision table, Bayesian Network and Naive Bayesian. One can apply
these classifiers to binary classification problems as well as other wide range of applications in
machine learning. In this dissertation, we use them for the sign prediction, and we compare their
performance with different evaluation metrics. In chapter, 5, we reveal the performance of varying
community properties on community prediction. Also, stochastic gradient descent has a wide range
of applications for optimization. As such, we use it for optimization of parameters in our proposed
recommender models in chapter 6.

2.2.8 User/Item Recommendation

Recommender systems have been the center of much attention more than a decade. They have been
employed to recommend movies, kinds of music, hotels and other types of items to users. Even
users can be recommended to users which one can consider friend or expert recommenders as this
type. In the recommendation process, some of the ratings from user \( u \) to item \( i \) are already known,
which we denote them with \( r_{ui} \), and we require to predict \( \bar{r}_{ui} \) based on the known rating matrix
\( R \). Each element of this matrix is a cell from user \( u \) to item \( i \). Sometimes, we prefer to refer to
the adjacency matrix of ratings that we denote it with \( RA \). Each element in \( RA \) indicates whether
the corresponding user has rated the item or not. Recommender algorithms have been under inves-
tigation more than a decade, and significant approaches are collaborative filtering, content-based
filtering, community-based recommenders and the combined approaches [RRSh11]. Collaborative
Filtering (CF) is a classical recommender algorithm, which variations have been devised based on
it. The idea behind CF is that users who rated the same sets of items might have similar preferences
and interests regarding other items in future. In chapter 6, we will explain a recommender model
using dynamics of community structures.
2.2.9 Cooperation and Defection

Researchers have studied cooperation and defection through Prisoner’s Dilemma (PD) game which two players can play an evolutionary game through time. In this game theoretic perspective, each player pays a cost in return for a benefit. Nowak and May considered the payoffs as with fitness measure [NoMa92]. There have been some variations to the original PD game such as Tit for Tat, Joss, Friedman and random strategies. In Tit for Tat strategy, an agent cooperates in the first round of the game, afterward, replicates the behavior of the other player. Joss strategy is based on Tit for Tat; however, it is combined with some random defection in between. Friedman strategy is a bit different, and the players usually do not defect at the start of the game but would start not to cooperate as soon as the opponent start defection. Then the player continues defection until the end of the game. In random strategy, each player has a random probability of cooperation and defection which usually is considered equal for the players [KiNa07, Axel84].

We use evolutionary game theory to study cooperation and defection of individuals and groups while it provides a mathematical framework to model interactions among players (agents). This model can be called a game, which it consists of a set of players, a set of strategies and a payoff function. In a repeated game, players remain unchanged in different rounds of iterations. We may simulate the game in a way that participants select their strategies without their knowledge about strategies of others. In this case, the game can be called simultaneous. A player is a participating individual in a game, which can also be called an agent. In game theory terms, we consider the set of participating players or agents as a population, which they can interact with their neighbors. Moreover, we name a population well-mixed, when the graph \( G \) is complete, otherwise, it is structured.

A action determines a player’s behavior while interacting with others, which his strategy defines possible actions applied by a player. The action used by player \( i \) is defined by \( \alpha(t) \) in round \( t \) according to his strategy. A pure strategy identifies actions for all situations. Suppose strategy space \( S^i = \{s_1, s_2, \ldots\} \). It defines the pure strategies available for player \( i \). A mixed strategy assigns probabilities to available pure strategies. We denote a strategy by a column vector \( \vec{s}_i \), which the \( k \)-th component identifies the probability of choosing strategy \( s_k \). For a player, interaction with neighbors generates an outcome called payoff. In a normal form game with two players, we show the payoff with a matrix \( P \). In this matrix, rows and columns show the action of each player while every cell represents the received payoff. Usually, games are considered symmetric, which the same payoff matrix is used for all agents. One can show the received payoff of player \( i \) resulting from an interaction with player \( j \) as \( \pi_{ij} = \vec{s}_i^T \star P \star \vec{s}_j \).

Summary of Applications

Besides expert identification, we introduced two network science applications named recommender systems and cooperation & defection. As such, we propose a novel algorithm named TCNSVD using temporal dynamics of community structures in chapter 6. Additionally, cooperativity of users, as well as communities, are investigated through the cooperation & defection in the same chapter.
2.3 Metrics and Measures

In this section, we introduce the metrics and measures used to evaluate the algorithms and approaches employed in this thesis. We mention standard machine learning metrics such as prediction accuracy, RMSE, MAE and information retrieval metrics such as Prec@k, mean reciprocal rank, mean average precision, etc. Moreover, we discuss measures to evaluate overlapping community detection algorithms in online social networks. These metrics include statistical and knowledge-driven metrics such as modularity and Normalized Mutual Information (NMI).

2.3.1 Network and Community Synthetic Generators

In general, the ground-truth community structure of real-world networks are not available and network science researchers face challenges to evaluate OCD algorithms. One way of evaluating the algorithms is to produce synthetic network data with known community structures that resemble real-world networks. In this regard, Girvan and Newman proposed a simple model which generates a network with 128 nodes and four communities; each community has 32 nodes. There are two parameters $\lambda_1$ and $\lambda_2$ for each node in the network. These parameters specify the number of internal community connections and number of connections to the outside [GiNe02]. However, Girvan and Newman network models do not create overlapping community structures. Moreover, they may be suitable for flat networks with assortative degree mixing structures. Thus, we employ a more complex synthetic network that is suitable for OCD experiments in both signed and unsigned networks.

LFR Synthetic Networks

Synthetic networks, or computer generated complex graphs, consider several properties from the real world. The parameters of the LFR model include the number of nodes $n$, the average and maximum degree of each node $k$ and $maxk$, the minus exponents for the degree and community size distributions following power laws $t_1$ and $t_2$. It also contains minimum and maximum community size $minc$ and $maxc$, the number of nodes in overlapping communities on, the number of communities nodes belong to, om, and the fraction of edges that each node shares with the other nodes outside of its community, $\mu$. Overlapping community structures in LFR model, and having power-law distribution for the number of nodes and size of communities, make them more similar to real-world networks.

The OCD in signed networks has a similar challenge regarding the ground-truth of real-world data. Therefore, we employ LFR models to the case of signed networks. If we adopt the idea of [LiLJ14], synthetic networks will be developed based on the directed unweighted LFR model$^1$. In this regard, LFR model uses two more parameters; the fractions of negative connections within communities $P_-$ and positive connections between communities $P_+$. In other words, we negate weights of all edges between communities; then we randomly select $P_-$ fractions of edges inside communities. Finally, we choose randomly $P_+$ fractions of edges between communities and negate them.

$^1$https://sites.google.com/site/santofortunato/inthepress2
2.3. METRICS AND MEASURES

2.3.2 Classical Prediction Metrics

In this section, we define and explain measures for the evaluation of prediction models such as community evolution and sign prediction models.

**Mean Absolute Error**

Mean Absolute Error (MAE) is a metric which is used to evaluate the goodness of an algorithm for prediction of ratings. To compute the MAE error, we average over the instance values [Bish06, TSKu05]. If we consider the real observed value with $y_i$ and the predicted class with $f_i$ for the instance $i$, then the MAE can be computed as follows:

$$MAE = \frac{1}{n} \sum_{i=0}^{n} |f_i - y_i|,$$

(2.12)

where $n$ is the number of instances.

**Root Mean Square Error (RMSE)**

Another metric to evaluate the error, which penalizes large values of errors is RMSE [Bish06, TSKu05]. RMSE averages over the square values of errors and can be computed as follows:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{n} |f_i - y_i|^2}{n}}.$$  

(2.13)

**Precision in Classification Context**

While having a classification problem, known classes are binary or multi-target cases. To evaluate such supervised learning problems, one needs to calculate true positive and false positive rates. Positive or negative are signs of the classifier's expectations and better to be named predictions [Bish06]. In contrast, true and false prove to be correspondent with ground-truth labels. If one denotes the true positive rates with (TP) and false positive rates with (FP), then precision can be calculated as follows:

$$Precision = \frac{TP}{TP + FP}.$$  

(2.14)

While having two distinct labels or classes in a binary classification problem, the Prediction Accuracy (PA) may be a better measure indicating the goodness of the prediction as follows:

$$PA = \frac{TP + TN}{TP + TN + FP + FN}.$$  

(2.15)

2.3.3 Information Retrieval Metrics

Information retrieval metrics aim to evaluate the searching and ranking algorithms. In fact, one can use these metrics for the evaluation of expert identification task and community-aware ranking algorithms.
Mean Average Precision

We considered a set of queries to compute the mean average precision in information retrieval systems. Mean average precision is the average precision score of each query that can be computed as follows:

\[
MAP = \frac{\sum_{q \in Q} AP(q)}{|Q|},
\]

(2.16)

where \( AP \) is the average precision for the results returned by each query, which can be calculated as follows:

\[
AP = \frac{\sum_{k \in K} P@k(q)}{|K|}.
\]

(2.17)

\( K \) determines the cut-off to ignore documents ranked lower than the threshold value. \( P@k \) computes the fraction of relevant results in the top \( k \) [BBC*13].

Mean Reciprocal Rank

Another metric which is applied to evaluate the results of expert ranking lists is Mean Reciprocal Rank (MRR) which can be stochastically defined as follows:

\[
MRR = \frac{\sum_{q \in Q} \frac{1}{\text{rank}_q}}{|Q|},
\]

(2.18)

where \( \text{rank}_q \) is the rank position of the first relevant document for the query \( q \) [BBC*13].

Precision and Recall

One can evaluate recommender systems based on different approaches and metrics. Herlocker et al. and Shani et al. described a couple of these metrics in surveys [HKTR04] and [ShGu11], respectively. In this dissertation, we intend to evaluate our proposed recommender models regarding precision and computational costs. For rating prediction accuracy, we can employ RMSE and MAE from formulas 2.13 and 2.12. Moreover, for the top \( k \) item ranking or the accuracy of recommendation, precision and recall can be employed from equations 2.19 and 2.20; however, we require to consider a list of \( k \) recommended items. We denote them with retrieval metrics with Prec@\( k \) and Rec@\( k \). Respectively, they can be defined as follows:

\[
\text{Prec@}k = \frac{\#tp}{\#tp + \#fp},
\]

(2.19)

\[
\text{Rec@}k = \frac{\#tp}{\#tp + \#fn}.
\]

(2.20)
Normalized Discounted Cumulative Gain

Normalized Discounted Commutative Gain (NDCG) is a metric that evaluates the utility of a returned recommended items’ list. In other words, the utility of items is aggregated to compute the total utility of a recommendation list. NDCG is a metric that measures the utility of a recommendation list as the sum of the utilities of each recommended item, where the utility of each recommendation is discounted corresponding to its position on the list with items on the bottom of the list being discounted more heavily. The idea behind NDCG is to penalize the items that appear lower regarding the position in the list while they are more relevant compared to their position.

2.3.4 Community Evaluation Metrics

For the evaluation of community detection algorithms, we use knowledge-driven and statistical measures such as NMI and modularity which we define them in the following.

Modularity

Newman and Girvan proposed modularity as a measure to evaluate the quality of covers. It considers the degree distribution of nodes in addition to the network clustering. It supposes that the number of edges in the communities is higher than the number of edges between communities in comparison to a network with randomly shuffled edges. The original modularity formula proposed by Newman is not suitable for overlapping communities, hence a version ideal for both directed graphs and overlapping communities is applied [NMCM09]. The implemented modularity considers a simple random graph known as a null model. The null model contributes to figure out how the cover is modular. This random structure has the same number of nodes, edges, in-degree and out-degree distribution as the original graph. We define a belonging factor as follows to compute to what extent an edge belongs to a community:

\[
\gamma_{(i,j),l} = f(M_{i,l}, M_{j,l}),
\]

(2.21)

where \( \gamma \) is the belonging factor, \( f \) is a function computing the factor, and \( M \) is the membership matrix. \( f(M_{i,l}, M_{j,l}) = M_{i,l}M_{j,l} \) is suggested in the original paper. The probability that a node belongs to a community is assumed to be stochastically independent of the probabilities of other nodes. Similar to the belonging edge factor, one may proceed by considering the belonging probability of nodes to a certain community. Hence, we define \( \gamma_{(i,j),l}^{+} \) as the belonging factor of an edge \((i, j)\) which starts from node \( i \) and belongs to community \( C_l \):

\[
\gamma_{(i,j),l}^{+} = \frac{\sum_{j \in V} M_{i,l}M_{j,l}}{|V|},
\]

(2.22)

Similarly \( \gamma_{(i,j),l}^{-} \) indicates the belonging factor of an edge \((i, j)\) referring to node \( j \) which is defined as follows:

\[
\gamma_{(i,j),l}^{-} = \frac{\sum_{i \in V} M_{i,l}M_{j,l}}{|V|}.
\]

(2.23)
Finally, to compute the modularity, we proceed by subtracting the probabilities related to original graph and the null model as follows:

\[ Q = \frac{1}{m} \sum_{l=1}^{\left| C_l \right|} \sum_{i,j \in V} \left( \gamma_{(i,j),l} A_{ij} - \gamma'_{(i,j),l} \right), \tag{2.24} \]

where \( \gamma'_{(i,j),l} \) is the belonging probability regarding the null model.

**Signed Modularity**

It is required to extend the definition of modularity to evaluate overlapping clusters in signed networks. Gomez et al. [GJA09] extended modularity to the case of signed networks; however, it is suitable for disjoint community detection algorithms. Deviation of positive and negative connections from the positive and negative edges of the null model is effective in the calculation of signed modularity. We can formalize signed modularity as follows:

\[ Q_s = \frac{1}{2w^+ + 2|w^-|} \sum_i \sum_j \left[ w_{ij} - \left( \frac{w^+_i w^+_j}{2w^+} - \frac{w^-_i w^-_j}{2|w^-|} \right) \right] \delta(C_i, C_j), \tag{2.25} \]

where \( w^+ \) and \( (w^-) \) represent the total weight sum of all positive and negative connections. Moreover, \( w^+_i \) and \( w^-_i \) are the total weight sum of all positive and negative edges incident to node \( i \). Finally, \( \delta(C_i, C_j) \) is equal to 1 if node \( i \) and \( j \) are in the same community. Otherwise \( \delta(C_i, C_j) \) is mapped to 0.

We extend the definition of signed modularity to consider overlapping clusters and the expected number of positive and negative connections. In the new definition of a community, an edge can reside inside and between more than one community. Thus, we expect that an edge takes positive value if it is within at least one community and obtains negative value if it resides within no overlapping community. Correspondingly, \( \delta(C_i, C_j) \) is an integer less and equal to \( k \), which \( k \) is the total number of communities a node can belong to. In addition, edges with the repetition frequency of more than one will also be considered multiple times. To rephrase the modularity formula, we consider effective weight sum of positive and negative connections as \( \frac{w^+_i}{(2w^+)_e} \) (\( w^-_i \)). Thus, we can write the extended signed modularity employed for the evaluation in this thesis as follows:

\[ Q_{so} = \frac{1}{(2w^+)_e + 2|(w^-)_e|} \sum_i \sum_j \left[ w_{ij} - \left( \frac{w^+_i w^+_j}{(2w^+)_e} - \frac{w^-_i w^-_j}{2|(w^-)_e|} \right) \right] \delta(C_i, C_j). \tag{2.26} \]

**Frustration for Signed Networks**

Another statistical metric to evaluate communities in signed networks is frustration. It can be calculated based on the number of positive and negative edges that break the balancing theory [DoMr96]. We define the frustration as for the evaluation of overlapping communities in signed networks, and
thus we employ the effective values in our formulations. In other words, each edge may be considered multiple times. Frustration can be defined as follows:

$$Frustration = \frac{\alpha \times |(w_{\text{intra}})^e| + (1 - \alpha) \times (w_{\text{inter}}^+)^e}{(w^+)^e + |(w^-)^e|},$$  

(2.27)

where $\alpha$ is the weighting parameter affecting the importance of positive and negative violations. Moreover, $(w_{\text{intra}})^e$ and $(w_{\text{inter}}^+)^e$ denote the effective weight sum of all negative and positive edges located within and between communities, respectively. $(w^+)^e((w^-)^e)$ is the effective weight sum of all positive (negative) edges. When frustration is smaller, the detected community structure is closer to balancing theory [CaHa56].

Normalized Mutual Information

Normalized Mutual Information (NMI) is intended to calculate the quality of overlapping building blocks [DDDA05]. If we consider that each node $i$ is a member of a community $C_l$ then the entry $ij$ in the membership matrix $M$ can be considered as a random variable with the probability distribution as follows:

$$P(X_{il} = 1) = \frac{|C_l|}{|V|}, \quad P(X_{il} = 0) = 1 - P(X_{il} = 1).$$  

(2.28)

Where $\frac{|C_l|}{|V|}$ is defined as the number of nodes in community $C_l$. The same probabilistic relationship is assumed to be held for the ground-truth cover. The uncertainty of node $i$ belonging to communities can be considered by conditional entropy as follows:

$$H(X_k|Y_l) = H(X_k, Y_l) - H(Y_l),$$  

(2.29)

where it only depends on distributions of $P(Y)$ and $P(X_k, Y - l)$. Observing the vector $Y_l$ then the the entropy of $X_k$ given the entire $Y$ is defined as follows:

$$H(X_k|Y) = \min_{l \in \{1,...,|C_L|\}} H(X_k|Y_l),$$  

(2.30)

if we normalize and average over the communities, it can be written as follows:

$$H_{\text{norm}}(X|Y) = \frac{1}{|C_L|} \sum_k H(X_k|Y),$$  

(2.31)

similarly we can compute $H_{\text{norm}}(Y|X)$ and proceed to compute the total NMI value as follows:

$$NMI(X|Y) = 1 - \frac{1}{2}[H_{\text{norm}}(X|Y) + H_{\text{norm}}(Y|X)].$$  

(2.32)

The actual NMI values range between 0 and 1. 1 indicates the highest and the best match. To apply the measure, we can use synthetic generators like LFR networks.
Omega Index

Similar to NMI, omega index [CoDe88] is suitable for comparison of overlapping communities with ground-truth information. For two pairs of node $i$ and $j$, it checks whether they are in agreement, in other words, if they share the same amount of communities. The omega index is as follows:

$$w(\Gamma, \Gamma') = \frac{w_u(\Gamma, \Gamma') - w_e(\Gamma, \Gamma')}{1 - w_e(\Gamma, \Gamma')},$$

which $w_e$ is the expected omega index and can be computed by normalizing the expected number of nodes in agreement as follows:

$$w_e(\Gamma, \Gamma') = \frac{1}{n_{\text{pairs}}} \sum_{j=0}^{\min(|\Gamma|, |\Gamma'|)} |A_j(\Gamma)||A_j(\Gamma')|,$$

where $A_j(\Gamma)$ is all pairs of nodes that share exactly $j$ communities. $n_{\text{pairs}}$ is the total number of node pairs in the graph.

Summary of Information Retrieval and Community Detection Metrics

In this subsection, we introduced a couple of metrics for the evaluation of problems investigated in this thesis. First, we studied network generators such as LFR synthetic networks, which they are used in almost assessment of all OCD algorithms. We need LFR networks while ground-truth information lack in real-world datasets, so networks with ground-truth community structures are statistically generated with this algorithm. We also use LFR synthetic networks for the evaluation of OCD algorithms proposed in chapter 4 as well as computing cooperativity in these synthetic networks in chapter 6. Moreover, we introduced MAE, RMSE, MAP, and MRR for the evaluation of recommender systems as well as expert identification, which we investigate in chapter 6. They are either used to evaluate rating prediction accuracy or item ranking in information retrieval problems. These metrics are specific to information retrieval, as such, we have used them for the evaluation of our proposed recommender algorithms. Precision metrics, in the classification context, are used to evaluate the community evolution prediction as well as sign prediction problem, which we investigate in chapter 5. In this regard, precision metrics are used for either binary or multiple target classification problems. Next, we mentioned evaluation metrics for community detection problem. Respectively, we specified metrics for the evaluation of overlapping communities in unsigned, forums and signed networks. Moreover, metrics such as NMI are also introduced to evaluate overlapping communities on networks with ground-truth information. We widely use evaluation metrics such as modularity and NMI in chapter 4.

2.3.5 Correlation and Distance Measures

In the following, we introduce some of the distance measures that we applied in various parts of this manuscript, i.e., recommender systems and ranking algorithms.
Adar Coefficient

To compute the relatedness of personal home pages, Adamic and Adar proposed a metric which weighs more to the unusual features extracted from the pages. This formula is as follows:

\[
\sum z \frac{1}{\log(\text{frequency}(z))},
\]  

(2.35)

where \( z \) is the shared feature vector between feature vectors \( x \) and \( y \) from personal home pages [LiKl03].

Hamming Distance

Hamming distance between two vectors \( V \) and \( W \) is the element-wise difference of these two vectors that can be computed as follows:

\[
H(W, V) = \sum_{k=1}^{K} |V_k - W_k|,
\]  

(2.36)

which \( K \) is the number of elements of the two vectors.

KL Divergence

KL divergence is usually applied to compute dissimilarities of two random variables. KL divergence can be computed as follows:

\[
KL(p, q) = \sum_{k=1}^{K} p_k \log\left(\frac{p_k}{q_k}\right),
\]  

(2.37)

which \( p \) and \( q \) are two vectors that we need to compute their relative information. KL divergence can be rewritten by entropy terms as follows:

\[
KL(p, q) = \sum_{k=1}^{K} p_k \log(p_k) - \sum_{k=1}^{K} p_k \log(q_k) = -H(p) + H(p, q),
\]  

(2.38)

where \( H(p, q) \) is named the cross entropy. We can show that when exactly \( p = q \) then \( KL(p, q) \) equals zero and always \( KL(p, q) \geq 0 \) [Murp12].

Pearson Correlation

One suitable approach to reveal linear relations between two variables is to use Pearson correlation. Pearson correlation can be applied when variables are discrete or continuous. The Pearson correlation between two vectors \( u \) and \( v \) can be computed as follows:

\[
\text{corr}(u, v) = \frac{\text{covariance}(u, v)}{\text{StandardDeviation}(u) \times \text{StandardDeviation}(v)} = \frac{S_{uv}}{S_u \times S_v},
\]  

(2.39)
which the covariance can be defined as follows:

\[
\text{covariance}(u, v) = S_{uv} = \frac{1}{n-1} \sum_{k=1}^{n} (u_k - \bar{u})(v_k - \bar{v}).
\] (2.40)

Moreover, the standard deviation of variable \( u \) can be defined as follows:

\[
\text{StandardDeviation}(u) = S_u = \sqrt{\frac{1}{n-1} \sum_{k=1}^{n} (u_k - \bar{u})^2},
\] (2.41)

and finally the mean value for variable \( u \) can be defined as:

\[
\bar{u} = \frac{1}{n} \sum_{k=1}^{n} u_k.
\] (2.42)

Several studies have investigated the Pearson correlation and showed its satisfactory performance [HKBR99, CMBo07, SuKh09]. As we employ this similarity metric for recommender algorithms, we adapt it to the recommendation problem where it is required to build the user-user or item-item graphs. In other words, the correlation among two users \( u \) and \( v \) is computed as follows:

\[
\text{pearson}(u, v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)^2 \sum_{i \in I_u \cap I_v} (r_{v,i} - \bar{r}_v)^2}},
\] (2.43)

Here, items which have been rated by users \( u \) and \( v \) are denoted by \( I_u \) and \( I_v \), respectively. Moreover, items rated by users \( u \) and \( v \) are denoted by \( I_u \cap I_v \) and \( \bar{r}_u \) and \( \bar{r}_v \) represent the mean ratings generated by users \( u \) and \( v \) [DeKa11]. Similarly, we can calculate similarity of two items \( i \) and \( j \) as follows:

\[
\text{pearson}(i, j) = \frac{\sum_{u \in U_i \cap U_j} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U_i \cap U_j} (r_{u,i} - \bar{r}_i)^2 \sum_{u \in U_i \cap U_j} (r_{u,j} - \bar{r}_j)^2}},
\] (2.44)

which \( \bar{r}_i \) and \( \bar{r}_j \) represent the mean rating shared for items \( i \) and \( j \), respectively by users who have co-rated these items, i.e. \( U_i \cap U_j \). Pearson correlation can take values in the range -1 and +1 which +1 (-1) indicates a perfect linear relationship between two random variables.

**Spearman Correlation**

Spearman’s rank correlation does not assume whether the relationship among two variables \( x \) and \( y \) is linear. Moreover, it does not suppose any predefined frequency distributions for two variables. The Spearman measure can be computed as follows:

\[
\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)},
\] (2.45)

which \( d_i \) is the difference between the corresponding value of two elements \( x \) and \( y \), and \( n \) is the number of pairs of values [KuVa10].
2.3. METRICS AND MEASURES

Cosine

Cosine similarity is a widely used metric to find similarities in documents; however, it has been employed in collaborative filtering [CMBo07, SuKh09, BOHG13]. It is mainly related to vector spaces, and it measures the angle between two vectors; however, we may require calculating the size of vectors. In this regard, we can compute cosine similarity between two vectors \( u \) and \( v \) as follows:

\[
\cos(u, v) = \frac{u \cdot v}{|u||v|},
\]

(2.46)

where \( u \cdot v \) is the vector dot product which can be computed as \( u \cdot v = \sum_{k=1}^{n} u_k v_k \) and as well \( |u| \) is the length of the vector \( u \) which is computed via \( \sqrt{\sum_{k=1}^{n} u_k^2} \). Similar to Pearson correlation, we employ cosine similarity to construct the user-user and item-item graphs [DeKa11]. In other words, cosine similarity among two vectors \( u \) and \( v \) can be calculated as follows:

\[
\cos(u, v) = \frac{\sum_{i \in I_u \cap I_v} r_{u,i} r_{v,i}}{\sqrt{\sum_{i \in I_u} r_{u,i}^2 \sum_{i \in I_v} r_{v,i}^2}}.
\]

(2.47)

Similarly, the cosine similarity between two items can be computed as follows:

\[
\cos(i, j) = \frac{\sum_{u \in U_i \cap U_j} r_{u,i} r_{u,j}}{\sqrt{\sum_{i \in U_i} r_{u,i}^2 \sum_{i \in U_j} r_{u,j}^2}}.
\]

(2.48)

K-NN graph construction [PPLJ14] which is explained in chapter 6 employs cosine similarity as its default similarity metric and thus it is selected for graph construction in this dissertation.

Jaccard Mean Squared Distance

Bobadilla et al. [BSBe10] proposed Jaccard Mean Squared Distance as a similarity measure which combines mean squared distance (msd) and the Jaccard coefficient. Several applications including collaborative filtering have employed Mean squared distance; however, one major problem with msd is that it does not consider the size of sets. In case of collaborative filtering, two users might have high similarity although they have co-rated a small set of items which might be unrealistic. Hence, msd is combined with the Jaccard index and those users with the higher number of co-rated items will be weighted higher. In the following, we respectively define the msd and the Jaccard index for two sample users \( u \) and \( v \):

\[
\text{msd}(u, v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - r_{v,i})^2}{|I_u \cap I_v|},
\]

(2.49)

\[
\text{jac}(u, v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|}.
\]

(2.50)

And the combined Jaccard mean squared distance is computed as

\[
\text{jmsd}(u, v) = \text{jac}(u, v) \cdot (1 - \text{msd}(u, v)),
\]

(2.51)
Concepts and Definitions

where $I_u \cap I_v$ is common items co-rated by both users $u$ and $v$. Similarly, we can define jmsd for two sample items $i$ and $j$ as follows:

$$\text{msd}(i,j) = \frac{\sum_{u \in U_i \cap U_j} (r_{u,i} - r_{u,j})^2}{|U_i \cap U_j|},$$  \hspace{1cm} (2.52)

$$\text{jac}(i,j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|},$$  \hspace{1cm} (2.53)

$$\text{jmsd}(i,j) = \text{jac}(i,j) \cdot (1 - \text{msd}(i,j)).$$  \hspace{1cm} (2.54)

Babadilla et al. [BOHG13] proved the satisfactory performance of jmsd regarding precision and accuracy.

Kendall Distance

The Kendall distance measures the number of pairwise disagreement between two ranking lists. It can be defined as:

$$k(L_1, L_2) = \Sigma_{(i,j) \in \text{elements}} K_{i,j}(L_1, L_2),$$  \hspace{1cm} (2.55)

where, elements show the set of unordered items in list $L_1$ and $L_2$. The normalized Kendall distance can take a value between 0 and one [KuVa10].

Summary of Distance Metrics

We introduced correlation and distance metrics, which are used in graph construction, e.g., in recommender systems, or to find correlations among cooperativity and rank vectors of nodes. Distance metrics have a wide range of applications; however, we have primarily used them in recommender systems or for analysis of the algorithms on the used social networks. We used correlation metrics including cosine similarity, Pearson, and JMSD for graph construction in the proposal of a recommender system using temporal dynamics of community structures in chapter 6, which we compare their performance on rating prediction and item ranking. Several other works have also investigated these metrics in the proposal of recommender systems in specific scenarios, and compared their performance. Besides, hamming distance and KL Divergence can be used to find the distance between any two binary or real-valued vectors; however, we use them to find correlation among rank values of nodes and their cooperativity in chapter 6. Moreover, we use Covariance, Pearson, Spearman and Kendal distance for computing correlations between community properties and cooperativity in chapter 6. Other applications can also use them, for instance, finding correlation among ranking algorithms as well as using them in prediction models; similarly, we have applied them in chapter 5.
Chapter 3

Related Work

3.1 Introduction

In this chapter, we study the state-of-the-art related to the posed research questions. First, we describe algorithms for detection of overlapping communities. Then, we analyze the literature regarding the community mapping, evolution, and prediction models. Afterwards, we explain the literature on signed social networks, sign prediction and overlapping community detection in networks with both positive and negative edges. Finally, the literature on applications such as recommender systems, expert identification, and cooperation & defection are studied. We can see an overview of the literature in Figure 3.1. We can observe that study of communities is in the center of the diagram. As it is not possible to show all the fields related to the study and investigation of communities, we only plot a couple of them related to this thesis. The big community node in the center is connected to OCD. We can classify major categories in the studying of OCD as Leader-based, Agent-based, Clique Percolation and Line-graphs that we plotted them in the figure and cited corresponding papers. Moreover, the Community node in the center is connected to the area of community mapping that describes major techniques to map the communities over time. Approaches based on matching as well as identifying and tracking influential nodes are noticeable in this regard (see page 38). Furthermore, another domain that we investigated in this thesis is community analysis and prediction. Several approaches named as parameter-free and parameter-based are denoted in this regard. Also, certain tools are available for the analysis of communities that the reader can get information regarding a list of them from the node named Community Tools. Last but not least, the Community node is connected with network applications including expert identification, recommender systems and link prediction that the reader can find literature with a swift look at these nodes.

3.1.1 Overlapping Community Detection (OCD)

Numerous disjoint community detection algorithms have been proposed [CINM04, LJZQ11]; however, they are not proper to identify communities in real-world networks. If one studies the literature
Figure 3.1: An overview of the related work regarding overlapping communities. Communities are related to different disciplines in social network analysis such as detection, mapping, evolution, prediction, tools and supporting applications.
on community detection algorithms, she notices several different categorizations. In a global perspective, we designate the approaches to three classes of static or dynamic, local or global and structural or content-based algorithms. In addition to the above categories, we can nominate algorithms in this area as categories such as clique percolation, line graphs or link communities, local optimization or leader-based methods, random walks or agent-based algorithms. Global methods apply universal metrics to identify communities [CINM04]. These metrics can be, for example, modularity, and we can optimize them through a global optimization approach [JiMc12, GhLe10].

Local methods of community detection consider local information of the network and thus approaches using random walk processes, cliques, influential nodes or leaders can be classified as local methods [SCCH09, LSHA08, KKKS08, ToSa15, SSKo11, WLLN17, LND16]. Furthermore, considering other sources of information may provide more realistic data to identify communities. Content-based and attribute-based approaches take attributes of nodes and edges besides to the connection information [YMLe13, RuFP12]. In addition to locality and relationship information, some algorithms are suitable for temporal environments. Complex networks are dynamic; however, static community detection algorithms can also be applied on each snapshot of the graph separately. For temporal network properties, static solutions may not be suitable and stable, and thus adaptive methods of community detection emerged. Adaptive methods approach the issues encountered in static methods, and they behave more stable [AHSu14, CVRa13]. For a comparison of community detection methods, we refer to [YA16, FoHr16]. In the following, we briefly explain each of these categories.

**Leader-based Techniques**

Considering neighbors of nodes to identify communities is informative while nodes’ proximity manifest fine-grained properties that indicate local interactions of nodes. Often, leader-based approaches employ local dynamics to identify several groups of nodes as influential members of the communities. Afterwards, membership of other nodes to the communities is computed based on a specific metric [SSKo11]. Influential nodes can be identified based on different approaches such as random walk, degree and even considering the influence ranges. Computing the membership of nodes to communities can also be calculated via the random walk, cascading behaviors and shortest paths [LZL*, ABL10, HHSG11]. In these approaches, we can detect communities around leaders.

- **Algorithm by Stanoev, Smikov and Kocarev (SSK)**

  Stanoev et al. proposed a two-phase algorithm based on influence dynamics and membership computation [SSKo11]. In the first phase, a random walk is employed to calculate the local and global influence matrices. We nominate this algorithm as SSK, which it assumes that relationships among nodes and their influences are counted more important than considering the direct connection. In other words, proxies among nodes are better established while there exist triangles among nodes. They constituted the influence matrix via adjacency matrix and triangle occurrences (3-cliques) among nodes. They further applied the matrix to achieve the local and global influential nodes, i.e., the hierarchy of the network. In other words, they calculated the transitive link matrix as follows:

  \[
  t_{lj} = t_{lj} + \sum_k t_{lj}^k,
  \]

  (3.1)
where \( tlv_{ji}^k = \min A_{ki} \), \( A_{jk} \) is the transitive link weight for the edge \((i, j)\) which goes through \( k \). The corresponding transition matrix for doing the random walk can be obtained by row normalizing the \( tl \) matrix. After doing the random walk, the most influencing neighbors of node \( i \) is identified based on \( (N_{influential} = j | T_{ji} = \max T_{ki}) \) that \( T \) shows the computed link weight transition matrix. By comparing the influencing neighbors with their neighbors’ influences, we can detect leaders. Afterwards, they identified membership of nodes to set of leaders by considering weighted average membership of neighbors. The updating rule for membership computation is as follows:

\[
M_i(t + 1) = \sum_{j=1}^{n} A_{ij} M_j(t),
\]

where \( A_{ij} \) is the row-normalized adjacency matrix. A row normalized adjacency matrix is computed based on normalizing each row of \( A \) by the sum of the row. The hierarchical and decentralized working behaviors are among properties that SSK possesses.

- **Algorithm By Li, Zhang, Liu, Chen and Zhang (CLiZZ)**
  This algorithm comprises two main steps. One step includes identifying leader nodes, and the other contains computing the membership of nodes to communities [LZL*12]. We nominate this algorithm as CLiZZ. To identify leader nodes, they computed the influence range of members based on shortest distance. It determines the mutual effects of nodes towards each other based on the following formula:

\[
LV_i = \sum_{j=1 ; d_{ij} \leq= \left\lfloor \frac{3}{\sqrt{2}} \right\rfloor}^{n} e^{-\frac{d_{ij}}{\delta}},
\]

where \( d_{ij} \) is the shortest path from node \( i \) to node \( j \). Authors approximate leadership value of node \( i - LS_i \) - within a range of \( \left\lfloor \frac{3}{\sqrt{2}} \right\rfloor \) [LZL*12]. An influential node has strong linkage with other nodes of the network. Afterwards, membership values of nodes to influential nodes need to be computed. To do this, they used a random walk process with the help of initial stationary membership values. \( M \) indicates the membership vector, and they updated each entry of this vector based on the following:

\[
M_i(t + 1) = \frac{1}{1 + \sum_{j=1}^{n} A_{ij}} \left[ M_i(t) + \sum_{j=1}^{n} A_{ij} M_j(t) \right],
\]

The algorithm needs to determine the \( \delta \) based on the topological entropy of nodes. CLiZZ is suitable for directed and weighted networks.

- **Merging of Overlapping Communities (MONC)**
  This algorithm, we name it MONC, starts by assigning each node in one community. The communities of nodes are then merged and expanded based on a predefined fitness function. The fitness function \( f \) is as follows:

\[
f(C, \alpha) = \frac{k_{inside}^C + 1}{(k_{inside}^C + k_{outside}^C)^\alpha},
\]
where $C$ is a community and $\beta$ is a parameter to identify community resolution, $k_{in}$ is the internal degree and $k_{out}$ is the external degree of the intermediate community $C$ [HHSG11]. Each of communities with minimum size is visited and checked to see whether the fitness value increases. Hence the change in the fitness function can be calculated as follows:

$$
\Delta f(C, i) = \frac{\log(k_{in}^{C \cup \{i\}} + 1) - \log(k_{total}^{C \cup \{i\}}) - \log(k_{total}^{G})}{\log(k_{total}^{C \cup \{i\}} + 1)}.
$$

(3.6)

After investigating the nodes, those with highest $\Delta f$ will be added to the community. The process is continued until no communities further expanded. Meanwhile, communities are checked to see whether there are duplicate ones with the same number of nodes.

**Agent-based Approaches: Random Walks and Label Updates**

A subclass of OCD algorithms is the agent-based methods. In this category, algorithms based on the random walk are noticeable. In the random-walk-based methods, the surfer starts a walk which is assumed to be an infinite journey. It selects the probability of its outgoing path based on a transition matrix. Many of the algorithms employ the steady state of the random walker and its spectrum of its walk to identify overlapping communities. Sometimes, for instance, they address every node of the network by unfolding a community around it through a random walk process then community structures around nodes can be combined [JYB*13, CWZh11]. Other methods may consider walks with limited length or constraint for the walks [ToSa15]. For instance, Jin et al. apply a limited-length random walk and find the walks of length $n^2$ and then identify paths of length $k$. In this method, edges overlapping in the paths gain weights which determine the communities [LZH14]. Other techniques impose constraints on the walk such as visiting active nodes (nodes with a higher degree than average degree of the network) and nodes with a predefined property named associate degree [FuWW13]. Another approach combines modularity with the random walk and considers the difference between random walk value in the network and its corresponding value in the null model [AsHu14]. Aston and Hu discovered communities based on hierarchical clustering; therefore it needs to compute distances. To define the distances, they computed node-node and community-node distances based on random walks [PoLa04]. Finally, other ideas such as applying coding techniques are incorporated with optimization techniques or including the history of previous states besides current states are among them [RoBe08]. Agent-based methods are also based on updating labels of the nodes [Greg10b]. For instance, Speaker-listener Label Propagation Algorithm (SLPA) is an agent-based technique that applies memories for each node [XSL11]. Labels can take several different states and can be updated based on some predetermined rules. Other agent-based techniques consider each node of the network as a separate agent deciding on its community. As such, they used game theoretic approaches, which each agent interacts with the neighboring agents with specific strategies and thus receives a payoff. They computed the payoff obtained by agents and detected communities as a result of gain and loss updates [AHSu14, AHHa11, CLSW10b]. In this context, agents are different than leader nodes. In other words, we can select agents from the nodes of the network, or they can be external entities that perform a specific behavior on the network. Leaders, on the contrary, are vertices that have high degree or rank which distinguish them from other nodes in the network - a ranking algorithm may be used to identify leaders.
Related Work

Clique Percolation Approaches

Clique percolation methods are based on finding the K-clique groups. K-cliques are sub-graphs that are fully connected. The algorithm identifies these set of K-cliques and maps each of these cliques to a new node. These sub-graphs can overlap, and the algorithm maps a set of cliques into a community while it identifies a series of $k$ neighboring cliques. In this regard, sharing of the $k-1$ member can be considered to have two cliques as neighbors. Time complexity and suitability for dense graphs are two problems ascribed to clique percolation techniques [SCCH09, LSHa08, KKK08]. There have been several variations of clique-based methods. For instance, (agglomerative hierarchical clustering based on maximal clique) EAGLE combines agglomerative techniques with maximal cliques and considers maximal cliques instead of nodes to form hierarchical clustering. It neglects all maximal cliques with threshold smaller than a certain $k$ value [SCCH09]. Moreover, Bi-clique communities consider the clique percolation problem in bipartite networks. It defines the concept of $K_{a,b}$ cliques and considers $k_a$ cliques in one set and $K_b$ in the other set [LSHa08]. Wang and Fleury employed clique optimization to identify granular overlaps. Clique optimization as a fine-grained approach helps to detect the nodes, which have connections to distinct communities [WaFl12]. Additionally, Estrada and Hatano define a communicability graph from adjacency matrix and attribute finding of cliques to communicability and binary matrices. They calculated a communicability graph by performing walks of length $k$ between every pair of nodes. Afterwards, by converting the communicability graph to a binary matrix and constituting the similarity matrix of cliques, cliques are merged to form communities [EsHa09]. As the efficiency of cliques has been an issue in several applications such as prediction of protein domains, several research works proposed other extensions. For instance, Fan et al. [FWR*12] eliminate all the edges with weights smaller than a predefined threshold then the algorithm starts with cliques of size $k$ and uses projection.

Line Graphs and Link Communities

The idea behind this category of approaches is to find the overlapping communities on links of a graph rather than nodes. In other words, instead of using partition methods for OCD, these methods convert the original graphs to line graphs and then partition the nodes of the new graph. In this regard, there are two ways, which map the original graph to a line graph: in the first method a vertex of degree $k$ is mapped into $k(k-1)/2$ edges of the line graph, so the random walk will be a link-link random walk. The other method converts the graph to an affiliation graph which edges of the original graph are on one side, and the vertices are on the other side [EvLa10]. In another method, Ahn et al. [ABLe10] applied the edge similarity to form the hierarchical communities. Furthermore, some research works combined link communities with other methods such as cliques and random walks [Evan10, EvLa10].

Content-based OCD Approaches

Content-based methods apply node, edge attributes and the actual content of social network to decide about communities [YMLe13, GEG*08]. Zhou et al. apply a clustering method, a version of k-medoid clustering, to determine the distances between nodes of an executed random-walk model on an attributed augmented graph. They generated this graph by adding dummy nodes for each of
the attribute values and dummy links between nodes and associated attributes of the nodes. k-means and k-medoid are variations of clustering techniques that start with the $k$ initial centroid. k-medoid identifies $k$ initial centroids based on the density point of view [ZCYu09]. Martin et al. devised the connected k-center problem, which uses intrinsic node characteristics to compute features. First, they determined the center nodes, and then they calculated the pairwise node distance. Afterwards, they assigned all non-center nodes to these centers; as a result, nodes have to be within a certain radius of the center, and the connectedness within clusters should be respected [GEG*08]. Cruz et al. [CBPo14] proposed a new approach to use semantic information from social networks to find related groups of nodes as well as a layout algorithm for visualization. Liu et al. introduced the concept of content propagation to determine communities in networks, using content and structure. In this regard, they computed community structure according to the interaction of the nodes. These interactions are modeled using two principles of content propagation. One based on a linear influence propagation model and the second principle employs a random walk to model the interactions directly. Both methods merely calculate content propagation probability from one node to another [LXWC15]. Yu et al. propose two feature integration strategies, to regulate the effect of linkage structure and edge content in OCD. In both methods, TF-IDF is used to transform the content. TF-IDF is an information retrieval technique to statistically calculate the importance of words in a text based on the frequency of their occurrence then one approach combines the content vector with the vector representing the set of neighboring edges and corresponding weights. The other approach first applies Mahalanobis distance (a metric to compute the distance between a point and distribution) to calculate the distance between two nodes based on content [YWZW14].

**Time Evolving Community Detection Approaches**

Research on community detection has emerged to time evolving networks which nodes and edges change through time. One of the early approaches is FacetNET proposed by Lin et al. [LCZ*08] which analyses current status of communities and their evolution through a joint procedure. They assume that communities will not change a lot between two consecutive time steps, and thus they regularize the community at the new snapshot. As such, a cost function consisting of two parts including snapshot cost and the temporal cost is defined. Duan et al. [DLLL12] also proposed a framework to identify communities with the contribution of k-cliques which cliques have access to each other through adjacent k-cliques. Furthermore, Nguyen et al. proposed an adaptive two-phase OCD algorithm named AFOCS to detect overlapping communities in dynamic networks [NSDT11]. It runs a static algorithm on the initial network topology by identifying local high-density clusters while it inspects only the updated nodes/edges as the input to re-evaluate the communities. It considers network changes as small events to improve efficiency and speed at the same time. The only inputs for this algorithm are a control parameter and the overlapping threshold. Additionally, Nguyen et al. proposed a fast and adaptive algorithm for identifying community structures in dynamic social networks named Quick Community Adaptation (QCA). This algorithm only considers previously discovered network structure and only processes network changes. It considers a modularity objective function and handles each small change of the network. In other words, adding/removing nodes/edges are handled separately by considering the modularity function. Results show that this algorithm is efficient and achieves competitive modularity; however, QCA does not detect overlapping nodes [NDYT11]. Cazabet et al. proposed intrinsic Longitudinal Community
Related Work

Detection (iLCD) which is capable of detecting community structures by employing a list of edges carrying snapshot information. To update the network structure and assign nodes to communities, the average number of the second neighbors needs to be higher than a certain threshold. iLCD handles the evolution of communities over time; however, it cannot distinguish between events such as shrinking, merging and splitting [CaAH10]. Only some of the approaches detect overlapping communities and thus may be inefficient for real-world cases.

Summary

In this section, we discussed significant categories of overlapping community detection algorithms. We characterized Leader-based, agent-based, clique percolation and line graphs and link communities as four main types of methods to identify overlapping communities. Figure 3.1 gives an overview of the research on community detection approaches mentioned above. We as well reviewed the literature on content-based techniques and methods suitable for temporal settings. Table 3.1 gives a list of community detection algorithms and some properties of these algorithms. These properties include the respective software, the time complexity, ability to detect overlapping communities and being deterministic. Most of the community detection algorithms are complex and simple but rather effective dynamics can be used for detection of community structures.

In the next section, we will analyze related work concerning community mapping and community prediction.

3.1.2 Community Mapping

Communities evolve, which they may appear, disappear and face other changes. As such, in community mapping, we apply algorithms, e.g., similarity comparisons, to find out how communities have evolved [BSKa13, BhAb14]. In this part, main approaches regarding the community mapping are discussed and addressed. Tracking the evolution of (overlapping) communities is significant to analyze and extract dynamics over time. The main category of approaches investigates either the inclusion of one community in another or they track the influential members of communities. The first category is matching information from one community in one snapshot with all the communities in the next snapshot. Ma and Huang [MaHu13] proposed a community update and tracking (CUT) algorithm which maps the graph information to a bipartite graph. The authors take the idea of connected components named cliques (here 3-cliques) on one side of the bipartite graph and the connections among these cliques on the other side. The algorithm tracks node and edge adding/removal by tracking these clique updates. Hopcroft et al. [HKKS04] defined a tracking method by repeatedly creating of agglomerative clustering and considering a match value ranging between 0 and 1. Greene et al. innovated a simple but rather efficient method which employs aggregate information from other communities. They define a front set and compare the identified community \( C_{ta} \) with front \( F_i \) based on Jaccard similarity \( \text{sim}(C_{ta}, F_i) = \frac{|C_{ta} \cap F_i|}{|C_{ta} \cup F_i|} \) [GrDC10]. Savie et al. devised and implemented a Java code to detect community evolution life cycles including birth, death, contraction, growth, split and stability in Apache Ant class collaboration network. They consider a similarity measure and specify several rules for communities that place transitions in
Table 3.1: This table indicates some of the community detection algorithms offered by open source community analytic frameworks. The corresponding software package, time complexity, community structure, and determinism are given in this table.

<table>
<thead>
<tr>
<th>Algorithm Based on Publication</th>
<th>Time Complexity</th>
<th>Overlapping Communities</th>
<th>Weighted Graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernighan-Lin [KeLi70]</td>
<td>$O(n^2)$</td>
<td>no</td>
<td>unknown</td>
</tr>
<tr>
<td>Edge Betweenness Clusterer [GiNe02]</td>
<td>$O(n^2)$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>VoltageClusterer [WuHu04]</td>
<td>$O(n + m)$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>CIZZ [LZL*12]</td>
<td>$O(m - O(n^2))$</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DMID [ShKK15]</td>
<td>$O(m \cdot n)$</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>SLPA [XXLi11]</td>
<td>$O(n)$</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Link Communities (LC) [ABLe10]</td>
<td>unknown</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MONC [HHSG11]</td>
<td>$O(n^2)$</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>SSK [SSKo11]</td>
<td>$O(n) - O(n^2)$</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DOCA [NDTN11]</td>
<td>$O(n) - O(n^2)$</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Spectral [Newm06b]</td>
<td>$O(n^2 \log n)$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Spinglass [Rebs06]</td>
<td>unknown</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Leading Eigenvector (LE) [Newm06]</td>
<td>$O(n^2)$</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Walktrap [PoLa05]</td>
<td>$O(m \cdot n)$</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Edge Betweenness (EB) [GiNe02]</td>
<td>$O(n^2)$</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>FGMod [WaTs07]</td>
<td>$O(m)$</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>MLMMod [BGLL08]</td>
<td>$O(n) - O(m)$</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Label Propagation (LP) [RAKu07]</td>
<td>$O(m)$</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Infomap [RABe09]</td>
<td>$O(m)$</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
predefined life cycles [SRLv12]. Cuzzocrea and Folino track the flow of members through a confusion matrix and they define internal and external transitions for communities. External transitions leverage the relationships of the community to other communities, and internal transitions handle transformations happening inside the community. They further apply similarity and overlap measures to find sharing of clusters [CuFo13]. Correspondingly, Palla et al. consider clusters of two consecutive time slots $t_i$ and $t_{i+1}$ as one cluster and apply the clique percolation method. This method can identify, merge, grow and unchanged events by considering an overlapping coefficient [PBVi07]. CommTracker follows influential nodes to discover how their properties change and how they are updated. This algorithm is suitable for large-scale networks and does not require any input parameter [WWPe09]. Similarly, Chen et al. considered community representatives to track the community dynamics. They defined communities as cliques and community representative as nodes with the minimum number of appearances in other communities. They constitute a decision three to describe the events happening to community representatives [CWJ*10]. HOCTracker is a node-based processing approach that maintains intermediate transition through a Log list. The Log-based approach contributes to avoiding the requirements to define overlapping coefficient and similarity measures and enhances the performance of community comparisons [BhAb14]. Finally, Group Evolution Discovery (GED) [BSKa13, BKKo12] is a widely used approach that we also use it in this thesis, and thus we explain it in more details in the following.

**Group Evolution Discovery**

GED technique considers the consecutive snapshot of communities and takes their representative graphs. In other words, inclusion of a community in another community is a significant decision rule to extract events like survive, merge, split and dissolve. The inclusion can be based on quantitative and qualitative metrics. Quantitative metrics are only about the percentage of overlapping nodes in two consecutive communities. Qualitative inclusion metrics, on the contrary, may take into account other information such as rank and position of nodes. If we indicate the overlaps of communities $C_i^t$ and $C_j^{t+1}$ with $OL_{C_{ij}}$, then their inclusion can be computed as follows:

$$ OL_{C_{ij}} = \frac{|C_i^t \cap C_j^{t+1}|}{|C_i^t|} \times \frac{\sum_{v_i \in C_i^t \cap C_j^{t+1} \& v_i \in V} (R(C_i^t(v_i)))}{\sum_{v_i \in C_i^t} R(C_i^t(v_i))}, \tag{3.7} $$

where $R$ is a representation of a node that can be its centrality, degree or position of the node. In simulations, we consider $R$ as the centrality of a node; $0 \leq OL \leq 1$. We also applied some threshold values to assign the communities to certain events. Two main threshold values that can be selected are $\alpha$ and $\beta$ values. By considering different threshold values, we made an event decision tree For instance, if $OL \geq \alpha$ and $OL \geq \beta$, then it will be a survive event [BSKa13].

**3.1.3 Community Evolution Analysis and Prediction**

Researchers have investigated community evolution analysis and prediction to a certain extent. Community prediction methods apply either parameter-free or parameter-based approaches for pre-
3.1. INTRODUCTION

diction of communities. Parameter-based approaches learn the parameters of the model by employing supervised learning algorithms and apply train and test sets. Takaffoli et al. applied a disjoint community detection method and mapped the communities over time. They predicted events such as survive, merge, split by employing several static and dynamic features [TRZa14]. However, their method did not consider overlapping communities and balancing the distribution of the classes. Similarly, Brodka et al. applied GED technique for community evolution tracking. Moreover, they employ features such as group size and event types in previous time steps [BKKo12]. Sekara et al. proposed a framework for the changes happening in the community. As such, they shed lights on the high stability of core gatherings and prediction of gatherings through cores. In other words, gatherings change which users join and leave but they consist of stable core individuals. They further discussed the concept of soft boundaries (soft membership participation), the context of meeting or community and recreational gatherings versus work gatherings [SSLe16]. Palla et al. mapped communities over time and indicated that stability of large groups depends highly on their adaptability to change the group composition. In other words, gatherings change which users join and leave but they consist of stable core individuals. They further discussed the concept of soft boundaries (soft membership participation), the context of meeting or community and recreational gatherings versus work gatherings [SSLe16]. Palla et al. mapped communities over time and indicated that stability of large groups depends highly on their adaptability to change the group composition. In other words, gatherings change which users join and leave but they consist of stable core individuals. They further discussed the concept of soft boundaries (soft membership participation), the context of meeting or community and recreational gatherings versus work gatherings [SSLe16]. Palla et al. mapped communities over time and indicated that stability of large groups depends highly on their adaptability to change the group composition. In other words, gatherings change which users join and leave but they consist of stable core individuals. They further discussed the concept of soft boundaries (soft membership participation), the context of meeting or community and recreational gatherings versus work gatherings [SSLe16]. Palla et al. mapped communities over time and indicated that stability of large groups depends highly on their adaptability to change the group composition. In other words, gatherings change which users join and leave but they consist of stable core individuals. They further discussed the concept of soft boundaries (soft membership participation), the context of meeting or community and recreational gatherings versus work gatherings [SSLe16]. Palla et al. mapped communities over time and indicated that stability of large groups depends highly on their adaptability to change the group composition. In other words, gatherings change which users join and leave but they consist of stable core individuals. They further discussed the concept of soft boundaries (soft membership participation), the context of meeting or community and recreational gatherings versus work gatherings [SSLe16].

3.1.4 Signed Social Networks

We address the literature regarding networks with both positive and negative connections including sign prediction and community detection in this section.

Community Detection in Signed Networks

According to the network types, we can categorize the research on community detection into unsigned and signed networks. There exist a lot of research on community detection in unsigned networks. However, these algorithms may not be eligible to be applied to signed networks due to balancing theory which enforces positive edges inside communities and negative connections among them [Dori04]. Balancing theory has been investigated and used in different studies which one of them is community detection. To this aim, Dorian et al. suggested a frustration metric calculating positive connections (friendship) inside communities and negative edges (hostilities) among

1https://www.ning.com/
them. They considered the frustration of an error value which increases when balancing theory violates. In this regard, a local search algorithm is employed to find the optimal number of communities [DoMr09]. Yang et al. assumed that it is more probable for a walker to traverse inside communities than across community borders and thus could detect communities by examining localized aggregated transition probabilities [YaCL07]. Gomez et al. identified communities by adapting modularity to the case of correlated data which is suitable for signed networks [GJAr09]. Shen applied generative models to identify regular structures of networks which they find similar patterns by using statistical inference and expected-maximization. Shen et al. estimated parameters of a statistical model such as maximum likelihood by expected-maximization that leans upon unobserved (latent) variables [Shen13b]. Wu et al. innovate the idea of mapping nodes from node to sphere coordinate and applying k-means clustering to detect communities [WWLL14].

Regarding spectral approaches, a two-phase spectral method is proposed by Anchuri and Magdon-Ismail to find the covers that employ a two-phase approach to maximize modularity and minimize frustration [AnMa12]. In this algorithm, leading eigenvector of generalized modularity matrix initially assigns all nodes to two communities. Afterwards, it reshuffles the nodes with the lowest values in the eigenvector if their reassignment causes higher values of modularity or lower values of frustration. The reshuffling will continue until the overall value cannot be improved any more. Moreover, Amealio and Pizzuti proposed a multi-objective approach which simultaneously optimize two objectives [AmPi13]. The first objective increases the density of positive intra-connections and decreases the density of negative inter-connections. In contrast, the second objective is to minimize both negative intra-connections and positive inter-connections. Finally, they achieve the best solution by opting either minimum frustration or maximum modularity [AmPi13].

The literature as mentioned above regarding community detection in signed networks are limited to disjoint communities, i.e., a node belongs to one single community. However, research on overlapping communities is rather scattered in this regard. In the following, we briefly describe them.

- **Signed Probabilistic Mixture Model (SPM)**

SPM algorithm [CWYT14] applies expected-maximization (EM) approach which maximizes the probability of latent variables. Latent variables are unobserved variables in statistics that can be inferred indirectly from observed variables. This algorithm requires an input parameter that needs to be set beforehand, i.e., the number of communities. \( \omega_{rs} \) is the probability of an edge \( e_{ij} \) choosing a community pair \( \{ r, s \} \) (1 \( \leq \) \( r, s \) \( \leq \) \( k \)) with the constraint \( \sum_{rs} \omega_{rs} = 1 \). \( e_{ij} \) is located in one community if \( r = s \) and is between two communities if not. The probability of community \( r \) (\( s \)) choosing node \( i \) (\( j \)) is denoted as \( \theta_{ri} \) (\( \theta_{sj} \)). For any community \( r \), given \( n \) nodes in the network, \( \sum_{i} \theta_{ri} = 1 \). As a result, the edge probability by SPM is:

\[
P(e_{ij}|\omega, \theta) = \left( \sum_{rt} \omega_{rt} \theta_{ri} \theta_{rj} \right)^A_{ij} \left( \sum_{rs(r \neq s)} \omega_{rs} \theta_{ri} \theta_{sj} \right)^{A_{ij}}.
\]  

s.t. \( \sum_{e_{ij} \in E} P(e_{ij}|\omega, \theta) = 1 \). They generated Soft partition of a network that is according to the probability of a node belonging to each community. SPM has some deficiencies such as its inability to tackle directed networks and the prime requirement regarding the total number of communities [CWYT14].
• Multiobjective Evolutionary Algorithm for Community Detection from Signed Networks (MEA)

Structural similarity is the core part for MEA as an evolutionary algorithm. The similarity between any two nodes can be calculated as follows:

\[ s(u, v) = \frac{\sum_{x \in B(u) \cap B(v)} \psi(x)}{\sqrt{\sum_{x \in B(u)} w_{ux}^2} \cdot \sqrt{\sum_{x \in B(v)} w_{vx}^2}} \]

(3.9)

where \( B(u) \) (\( B(v) \)) is the set of node \( u \)'s and \( v \)'s neighbours and \( w_{ux}(w_{vx}) \) is the weight of the edge connecting \( u(v) \) and \( x \). In this algorithm, two objective functions are employed to maximize positive similarities inside communities and negative similarities outside communities [LiLJ14].

Sign Prediction

To compute the likeliness of connections between two users, we may use classical machine learning approaches. Unsupervised learning considers topology of the network and extracts relevant information of two users. Regarding unsupervised techniques, similarity measures are popular. The similarity metrics include local measures such as the number of common neighbors, Jaccard index, Adar Coefficient and global metrics such as Katz index [DKWW11, LiKl03, TAbE09, WXWZ15]. In addition to structural-based distances, the similarity of users can be computed based on user's attributes. For instance, family names, location, marital status, and age can be considered to compute the similarity of users [Thel09]. Computation of user's similarities and using a particular threshold can identify whether two users may have a connection in future. Another class of link prediction methods uses supervised learning algorithms. This category employs the labels (targets or classes) and train data to learn some patterns automatically. When we show the test data to the classifier, it decides whether there may be a link between two users. More complex unsupervised and supervised techniques have also been devised such as probabilistic graphical models and low-rank matrix factorizations [LLCh10, WPS*11, MeEl11].

Sign prediction approaches can be categorized as supervised and unsupervised learning algorithms to predict the sign of links. Similar to link prediction, sign prediction are based on unsupervised methods. In this case, similarity-based as well as propagation-based techniques are noticeable [SyNi13, GKRt04]. Symeonidis and Nikolaos proposed similarity measures for nodes residing in the same or different communities [SyNi13]. They also introduced metrics that consider status theory for computation of node similarities. Guha et al. considered different trust propagation strategies including direct and transpose trust [GKRt04]. Regarding supervised sign prediction, we consider -1 and +1 as the binary classes, and the goal is to apply suitable features to predict the sign of a given link reliably. Leskovec et al. showed that using some simple features for the nodes, e.g., the number of positive/negative links outgoing from or incoming to a node, we can predict the sign of the links to a certain extent [LHKl10]. They also showed that the link sign could be partially described based on social balance and status theories [LHKl10]. Mostly, graphs are sparse, and triangle features may not be suitable [TCAI15]. Chiang et al. improved the performance of the sign prediction by employing more complex features such as longer cycles [CNDT11].
computing nodes similarities [STMo10] and collaborative filtering on signed community structure [JaJa14b]. Not only structural features are applied to train the models, but also homophily properties such as gender, interests, and location are employed [DGSr11]. In addition to the node level features, properties of interactions among users can give valuable information about their future activities [PABh12]. Yang. Et al. combined latent factor models with users and items [YSL*12]. They introduced a framework called behavior relation interplay to consider relations between social interactions and decision making. Regarding sign prediction based on community detection approaches, Symeonidis et al. apply a spectral clustering approach to the case of link prediction [SIMM13]. They use eigenvalues and eigenvectors of a Laplacian matrix for partitioning the data and predict missing links. Their approach is suitable for unsigned networks. Javari and Jalili use clustering to the case of sign prediction [JaJa14]. They apply cluster-based collaborative filtering for the sign prediction [JaJa14]. Community-based features are also devised to the case of sign prediction [SSNo14]. Incoming and outgoing links residing inside and outside community connections are differentiated and employed for the sign prediction [SSNo14]. For a detailed survey regarding signed graphs and sign prediction methods, refer to [TCAL15].

3.1.5 User/Item Recommender Algorithms

In this section, we introduce the related works on recommender algorithms. We investigate temporal and collective recommender systems separately.

Collaborative Filtering: Neighborhood and Latent Factor Approaches

The idea behind most of the recommender algorithms is collaborative filtering, which users with similar interests in the past would have similar preferences in the future. In other words, we may count on the existing rating patterns and predict rating of a particular user on a specific item. The rating matrix can usually be analyzed based on two different approaches. One is the neighborhood model, and the other is the factor model. A neighborhood model can be itself considered as user-based neighborhood and item-based neighborhood estimations. In other words, when we want to estimate the rating of user \( u \) on item \( i \) then the rating based on the user-based neighborhood models consider the neighborhood of the users similar to the targeted user, and thus the rating can be computed as follows:

\[
\hat{r}_{ui} = \frac{\sum_{v \in U} r_{vi} w_{uv}}{\sum_{v \in U} w_{uv}},
\]

(3.10)

Where \( w_{uv} \) is the similarity weight between user \( v \) and the target user \( u \). Similarly, we define the item neighborhood models by considering the items similarly rated by the targetted item and thus the rating of user \( u \) on target item \( i \) can be defined as follows:

\[
\hat{r}_{ui} = \frac{\sum_{j \in I} r_{uj} w_{ij}}{\sum_{i \in I} w_{ij}},
\]

(3.11)

where \( w_{ij} \) is similarity weight among neighborhood item \( j \) and target item \( i \). To compute the similarity weights, we employ one of the similarity factors introduced in chapter 2. Furthermore, to compute the rating of user \( u \) on item \( i \), we map each user to user factors vector \( p_u \in \mathbb{R}^f \) and each
item $i$ to a item factors vector $q_i \in \mathbb{R}^f$. The estimated rating for user $u$ on item $i$ is then computed using the dot product

$$\hat{r}_{ui} = q_i^T p_u. \quad (3.12)$$

Vectors $q_i$ and $p_u$ can be computed by Singular Value Decomposition (SVD).

**Time-Aware Recommendation**

Research has investigated algorithms dealing with temporal data in recommender systems. Cam- pos et al. [CDCa14] performed an extensive survey of recommender systems suitable for temporal settings and mentioned time as one important data dimension as for the recommendation. In the following, we describe some of the leading approaches in this regard. Koren proposed one of the approaches [Kore09], which we extend it to propose community-aware and temporal community-aware recommender algorithms (see chapter 6). He applies collaborative filtering based on neighborhood and factor models by employing individual drifts, i.e., short-lived and gradual user and item biases, user preferences and item characteristics drifts. Daneshmand et al. propose the idea of hidden item network which can be induced based on the history of user’s item selection. In other words, two related items in the hidden item network may cause associate rules such as if a user selects one of the two related items then it probably selects the other one as well. They consider temporal information of ratings and employ time-stamped user ratings to model information cascades to construct the network of relations among items. Baltrunas and Amatriain [BaAm09] increased the time-aware recommender algorithms by recommending music based on time, which can be the day, week or year. They employ the repetitive user preferences together with collaborative filtering as for the recommendation. They divided user profiles into micro profiles that each spans a period. On the contrary, Koren did not consider such daily, weekly and yearly repetitive patterns [Kore09].

**Community-Aware Recommendation**

Community structures have also been employed to recommend either users and items. In the following, we describe a couple of these approaches. Dolgikh and Jelinek [DoJe15] employ the tags and community information to recommend kinds of music to users. A tag-network is constructed based on user preferences regarding artists and the assigned tags for artists. Community detection is applied on the tag-network to identify the sub-interests related to a user, and thus artists are recommended based on the detected subfields. Cao et al. [CNZh15] employ the neighborhood of users and items to construct graphs and apply community detection to filter the neighborhood set. It is intended to increase recommendation precision and reduce the time complexities; however, it is not clear if they could improve the performance by reducing the scope of each user’s neighborhood. Here ratings are employed to construct the user-user network to recommend movies as items. Choo et al. [CYCS14] consider a similar approach, which they recommend items to users leaning upon neighborhood-based collaborative filtering. The difference is the construction of the user network based on a review-reply pattern, in other words, if a user generates a review, other users can reply to them, and thus a network can be built. We can use this network for the recommendation and identify its communities. Furthermore, some approaches reduce the cold start problem in collaborative filtering by applying community structure. For instance, metadata information such as age,
Related Work

gender and occupation are applied to identify clusters of users and generate the top-n recommendations for each community. Thus it avoids individual recommendations, which can alleviate the cold start problem. Furthermore, relation information among users is also applied to construct similarity graphs, and the user’s community membership is employed to compute similarity metrics in collaborative filtering [SaCo11]. These two latter approaches are examples of using metadata information as for reducing the cold start problem. None of the research works in the literature applied both temporal and collective dynamics to support recommender systems.

3.1.6 Expert Identification

Researchers have applied community structures to various domains such as sign and link prediction, misbehavior detection, routing, biology and analyzing criminal networks [MOTs14, AHHa11, ShSN14, ShKl15, TSTB16, CBPi17]. Another application domain is to identify communities of people for the expert finding. Peer production systems such as question & answer forums require experts that possess higher levels of knowledge in particular domains. There have been several different approaches accosting the challenge. Bozzon et al. found the corresponding profiles of workers in Facebook, Twitter, and LinkedIn and looked for experts in them. To achieve that, they identified some people and asked them questions about their expertise with values ranging from 1 to 7. These people were from different disciplines like computer science, music and so on. They employed information retrieval techniques and came to conclusions like information by others on the profiles of people are informative, and Twitter appeared to be the best social network for expert matching [BBC*13]. In a similar work, Bozzon et al. targeted the IBM company and extracted corresponding persons and their information in LinkedIn and Facebook.

Similar to social networks, researchers investigated the expert finding in DBLP networks. Deng et al. applied statistical graphical models like generative probabilistic models. In addition to the statistical model, research has considered weighted language model together with citation of papers for finding experts. In the same paper, they used a topic-based model, which they converted documents to topics, and they found the relationship between the topics and the queries [DKLy08]. Reichling and Wulf combined self-reported directory information with keyword mining from users’ files and folders. NIA used their recommender system in their real-world organization [ReWu09]. Macdonald and Ounis considered the expert search task as a voting problem where documents vote for the candidates with relevant expertise [MaOu06]. Data fusion techniques are also applied and evaluated by utilizing different document weighting models [BRWe08]. They conducted the actual evaluation within the context of expert search task of the TREC enterprise track.

Researchers have studied question & answer forums with two main categories of algorithms. In some of the approaches models and methods are not directly applied to find experts but rather to find similar and related items. One of such directions is feed distillation, that related blogs to a query are retrieved. First approach identifies resources associated with specific queries and calculates contribution of people in those resources. Structural-based ranking algorithms and expert identification based on HITS and PageRank reside in this category. In other words, graph induced from the query is employed to apply classic or tuned HITS and PageRank algorithms on them. Zhou et al. applied Latent Dirichlet Allocation (LDA) to extract topics and contextual information from generated askers’ and answers’ activities. They devised a tuned PageRank algorithm to
3.1. INTRODUCTION

take advantage of topical similarity in addition to structural information [ZLLZ12]. Yang et al. challenged previous works on topic and expertise modeling. They considered tags of users while posting questions as for their topic interests and similarities. Moreover, they employed generated votes of users as traces of their expertise. Two information sources, tags and votes, are leveraged to be consumed for expert finding [YQG*13]. Zhu et al. innovated a new approach for computation of relevant topics and categories. They further created a ranking method employing two information resources of the target and related categories [ZCX*11, ZCX*14]. Bouguessa et al. did not appreciate methods generating a ranked list of nodes while they had problems identifying the threshold in different categories in Yahoo! Answers. Thus, they leveraged authoritative and non-authoritative authority values with the gamma distribution. They estimated the number of components and the parameters of the model via Bayesian Information Criterion and EM algorithm [BoDW08]. Our proposed community-aware ranking algorithms belong to these categories, and they assimilate the effect of community structures with ranking algorithms.

3.1.7 Cooperation & Defection

Study of cooperation and selfish behavior has received much attention recently. Literature identified several mechanisms that favor cooperation. Hamilton [Hami64] proposed the inclusive fitness theory, which is also known as kin selection. Kin selection [Smit64] is based on two fundamental concepts. First, it states that a higher relatedness favors cooperative interactions. The second mentions that the higher payoff obtained by a player’s relatives also increases its fitness. Trivers [Triv71] proposed the concept of reciprocal altruism, which states that cooperation occurs through mutual help. By studying the literature, we get to know about other types of reciprocity, i.e., direct and indirect reciprocity, which we can observe in social systems. Nowak proposed a common framework to investigate reciprocity through evolutionary game theory [Nowa06].

Neumann and Morgenstern [NeMo07] invented game theory as a robust framework to study economic and strategic human decisions. Similarly, Smith and Price [SmPr73] used game theory as a tool to study biological structures. They offered a new perspective on human behavior, which they studied the cooperation between humans through game theory [Axel84, Nowa06]. Axelrod [Axel84], for the first time, studied the evolution of cooperation, which he investigated the mechanism of direct reciprocity through game theory approaches. Axelrod considered a sequence of strategies for the iterated prisoner’s dilemma. He could show that Tit-for-Tat is the most successful strategy, which favors cooperation. Indirect reciprocity was studied by Alexander [Alex87], which an individual does not need to get an immediate benefit for interactions. In other words, in indirect reciprocity, a player may cooperate with another player to get a benefit from a later interaction with a third player. For instance, in an open source software development environment, developers participate to gain the reputation as well as skills while cooperating with others. Several variants of games used for direct reciprocity have been applied to study indirect reciprocity [NoSi98, BoRi89].

Network cooperativity studies, on the other hand, investigated the cooperation and defection on a network of nodes connected to each other. As such, Nowak and May studied evolutionary prisoner’s dilemma on regular networks [NoMa92]. They showed that evolution of cooperation depends on the structure of the network and regular networks can promote cooperation. There exists other models and studies on network cooperativity [SzFa07, RCSa09, DoHa05, HCSy11]. Scale-free networks
Related Work

were investigated by Santos et al. [SaPa05], which they showed that preferential attachments in complex networks favor cooperation.

Because studying of community structures is the main focus in this thesis, we also reviewed cooperation and defection in networks with communities. As such, Luthi et al. [LPTo08] detected community structures on several networks including scale-free, synthetic and real-world social networks. They showed that communities tend to adopt the same structure and thus cooperators get separated from defectors. Another study by Chen et al. [CFWa07] on community structures showed that reducing of inner-community and inter-community links can favor cooperation. They used a model based on Santos et al. [SaPa05] on subgraphs of community structures with different average degrees. Yong-Kui et al. [YZXL09] used evolutionary prisoner’s dilemma to show that higher community size results in smaller cooperativity. Salehi et al. [SRJa10] studied cooperation in statistical repeated structures in complex networks i.e. motifs [SMMA02]. They used a variant of prisoner’s dilemma based on replicator dynamic to show that cooperativity correlates with the lower significance of motif structures. We see, therefore, that we still know very few regarding cooperativity of implicit community structures detected by (overlapping) community detection algorithms.

3.1.8 Community and Network Analytic Tools

During last decade, researchers have considered and developed different tools addressing various community analytic problems. GANXiS² is programmed in Java and is an extended version for SLPA. JUNG provides Java implementation of some graph theory, social network analysis, and data mining techniques and it also includes undirected and unweighed implementation of Link Communities³. Moreover, pylouvain-igraph renders a python implementation of multiple versions of Louvain method⁴. Community detection modularity suit⁵ is based on C++ and R and contains three community detection algorithms based on modularity measure. It further contains boot-strapping facilities to test cluster robustness. Additionally, CFinder is a Java-based implementation of clique percolation method. The software is suitable for dense graphs and provides suitable visualization and computation of graphs and covers through proper diagrams⁶. Furthermore, InfoMap has a corresponding Java implementation named Map Equation, which provides dynamic visualization of algorithm execution and the community structure⁷. Finally, Jmod⁸ is an open source tool implemented in Java that can be integrated with third-party software applications. It comprises several community detection algorithms with the possibility of parsing networks into different formats. Similar to our framework, it contains benchmark graphs and the LFR method for generation of synthetic networks. Other tools such as Apache Commons Graph⁹, JGraphX¹⁰, graph stream¹¹

²https://sites.google.com/site/communitydetectionslpa/
³http://jung.sourceforge.net/
⁴https://launchpad.net/pylouvain-igraph
⁵https://sourceforge.net/projects/cdmsuite/
⁶https://sourceforge.net/projects/cdmsuite/
⁷http://www.mapequation.org/index.html
⁸http://tschaffter.ch/projects/jmod/
⁹http://commons.apache.org/sandbox/commons-graph/project-reports.html
¹⁰https://github.com/jgraph/jgraphx
¹¹http://graphstream-project.org/doc/Tutorials/Getting-Started/
3.2. CONCLUSION

and JGraphT\textsuperscript{12} provide functionalities regarding handling and processing of graphs. Additionally, igraph\textsuperscript{13} is available for C, Python and R programming languages and offers a wide range of algorithms for community detection. There are also other software packages that implicitly or explicitly support community detection, as such, we can refer to SNAP\textsuperscript{14}, MOSES\textsuperscript{15}, DCM\textsuperscript{16}, GAMER\textsuperscript{17}, and bipartiteSBM\textsuperscript{18}. Among them, DCM and GAMER are based on description-oriented methods and attributed graphs; however, bipartiteSBM is suitable for bipartite network structures. It is obvious that there is quite a wide range of tools; however, they are rather scattered, and users (researchers) still encounter challenges while using them.

3.2 Conclusion

In this chapter, we reviewed the state-of-the-art approaches related to the raised research questions. We described the literature regarding community detection, evolution prediction, and community analysis. Moreover, we analyzed related work concerning signed networks, recommender systems, expert identification, cooperation \& defection, and so on. Some of the overlapping community detection algorithms are somewhat complex, and there may not exist proper social dynamics behind their formulations. Moreover, the analysis of communities’ future states is not realistic while many research works did not consider the overlapping communities.

Regarding signed networks, research works have poorly investigated the role of overlapping nodes in networks with both positive and negative connections. Additionally, the sign prediction approaches do not use community structures to study the link prediction problem. Furthermore, researchers have not yet applied temporal dynamics of community structures in the proposal of a recommender system. As for expert finding, overlapping community structures can be used to identify experts, i.e., through classical ranking metrics.

Besides, we studied significant works on cooperation and defection, which we focused on cooperativity of communities and sub-graphs. It is not yet determined, what is the relation among community properties and their cooperativity values in different network structures. Finally, regarding community analytic tools, many of them lack a suitable graphical interface, and the tools are scattered based on different programming languages. Studying the literature could help us to identify the research questions in this thesis work, which we mentioned in chapter 1.

\textsuperscript{12}http://jgrapht.org/
\textsuperscript{13}http://igraph.org
\textsuperscript{14}https://snap.stanford.edu/snap/description.html
\textsuperscript{15}https://sites.google.com/site/aaronmcdaid/moses
\textsuperscript{16}http://www.patternsthatter.org/software.php
\textsuperscript{17}http://dme.rwth-aachen.de/de/gamer
\textsuperscript{18}http://danlarremore.com/bipartiteSBM/
Related Work
Chapter 4

Overlapping Community Detection

4.1 Introduction

With the growth of computational technologies and development of online social networks, studying human interactions, and investigation of communities in specific have received attention. Moreover, human interactions cannot be researched on individual interactions while people perform most of the practical communications in societies in groups [PDFV05, Evan10]. Not only we find social phenomena while studying communities, but also we can develop and support social software that people use in daily life. An example of this social software is recommender systems that can be supported by community structures. As we investigated communities in chapter 3, they are connected with different domains and sub-disciplines such as detection, tracking, evolution analysis and prediction, and supporting of online applications. In fact, we require detecting community structures in networks before doing further analysis. Research has considered different viewpoints and definitions for online communities. First, they mention properties such as dense interactions inside components and sparse ones among them [Newm04]. Second, they view communities as groups of people that perform activities and interact due to their interests [LXWC15, BKNK09]. Moreover, one can employ other criteria such as the creation of content, innovations, and ideas, and social theories like status and balancing to the detection of overlapping communities [DoMr96]. We emphasize on overlapping structures while social network analysis indicates that there exist activity traces from people in more than one community.

In this regard, overlapping community detection is a cornerstone in this thesis. We base most of our analysis and conclusions on the applied overlapping community detection algorithms. The first section of this chapter introduces an algorithm based on two social dynamics named disassortative degree mixing and information diffusion (DMID). We adapt DMID to the case of signed social networks by defining the signed cascading process and considering effective degrees. We introduce Signed Disassortative degree Mixing and Information Diffusion (SDMID) in section 2 of this chapter.

Summary. We employ social properties and combine them with mathematical interpretations to devise algorithms for the detection of overlapping communities. We realize these properties as follows:
• **Disassortative degree mixing:** This property shows two states regarding similarity with the neighbors. One state is assortative degree mixing, where nodes tend to communicate with similar nodes concerning rank, degree and other aspects. For instance, assortative low-degree nodes tend to communicate with other low-degree nodes. Disassortative degree mixing, on the other hand, is a sign of dissimilarity, which high-rank or high-degree nodes tend to communicate with low-rank or low-degree nodes and vice versa. As a real example of this phenomenon, we may think of scientists that cite important papers in their fields or students who like to co-author with well-known and high-rank professors.

• **Opinion Formation and Information Cascade:** We may think of this property as opinion formation in social networks. For instance, one may bring the example of purchasing a mobile phone. It may be common that one pays attention to some of the close friends having a new brand device, and this may affect him/her to some extent based on the influence of his/her neighbors and how much s/he counts on them. Sometimes, this property is simulated through network coordination game when players can take binary opinions and receive payoff by changing their behaviors.

Considering the above-mentioned social properties, we devise two overlapping community detection algorithms leaning upon social properties such as disassortative degree mixing and information diffusion. This chapter approaches the first research question raised in chapter 1. We use social properties and combine them with mathematical interpretations and artificial intelligence techniques such as random walk and game theory methods like network coordination to design algorithms. The devised methods can be employed to detect traces of collective behaviors crawled from tracked online social network interactions. We denote the list of symbols we shall use for the description of OCD algorithms in this chapter in Table 4.1.
4.2. PROPOSED OVERLAPPING COMMUNITY DETECTION IN UNSIGNED NETWORKS

Table 4.1: An overview of the symbols used for the description of OCD algorithms.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$deg(i)$</td>
<td>Degree of node $i$</td>
</tr>
<tr>
<td>$AS$</td>
<td>Assortative matrix</td>
</tr>
<tr>
<td>$TAS$</td>
<td>Row-normalized disassortative transition matrix</td>
</tr>
<tr>
<td>$DV$</td>
<td>Disassortative vector</td>
</tr>
<tr>
<td>$LV$</td>
<td>Vector containing relative leadership values</td>
</tr>
<tr>
<td>$AFD$</td>
<td>Average follower degree</td>
</tr>
<tr>
<td>$RT$</td>
<td>Resistance threshold</td>
</tr>
<tr>
<td>$RG$</td>
<td>Received gain</td>
</tr>
<tr>
<td>$RG_A$</td>
<td>Resistance threshold for behavior $A$</td>
</tr>
<tr>
<td>$Follower(i)$</td>
<td>List of followers of node $i$</td>
</tr>
<tr>
<td>$GL$</td>
<td>Number of leaders</td>
</tr>
<tr>
<td>$M$</td>
<td>Membership matrix</td>
</tr>
<tr>
<td>$M_{il}$</td>
<td>membership dependence of node $i$ on community $l$</td>
</tr>
<tr>
<td>$Successor(i)$</td>
<td>The set of successors of node $i$</td>
</tr>
<tr>
<td>$t_i$</td>
<td>The time point node $i$ changes its behavior</td>
</tr>
<tr>
<td>$ED(i)$</td>
<td>Effective degree of node $i$</td>
</tr>
<tr>
<td>$DASS(i)$</td>
<td>Normalized disassortative-ness of node $i$ in signed networks</td>
</tr>
<tr>
<td>$LLD(i)$</td>
<td>Local leadership value of node $i$</td>
</tr>
<tr>
<td>$PayOff(i)$</td>
<td>Payoff that node $i$ receives</td>
</tr>
<tr>
<td>$k$</td>
<td>Average degree in LFR networks</td>
</tr>
<tr>
<td>$\mu$</td>
<td>The faction of edges sharing with other communities (mixing parameter)</td>
</tr>
<tr>
<td>$minc$</td>
<td>Minimum community size</td>
</tr>
<tr>
<td>$maxc$</td>
<td>Maximum community size</td>
</tr>
<tr>
<td>$on$</td>
<td>Number of nodes in overlapping communities</td>
</tr>
<tr>
<td>$om$</td>
<td>Number of communities which nodes in overlapping communities belong to</td>
</tr>
<tr>
<td>$P_+$</td>
<td>Fraction of positive connections between communities</td>
</tr>
<tr>
<td>$P_-$</td>
<td>Fraction of negative connections between communities</td>
</tr>
<tr>
<td>$maxk$</td>
<td>Maximum node degree</td>
</tr>
</tbody>
</table>

4.2 Proposed Overlapping Community Detection in Unsigned Networks

In the first part of chapter 4, we address overlapping community detection in unsigned social networks.

4.2.1 Disassortative Degree Mixing and Information Diffusion

We propose and introduce DMID algorithm in this section. DMID (Disassortative Degree Mixing and Information Diffusion) works based on two simple social properties named disassortative degree mixing and information diffusion. Often, the algorithm assumes that communities form around high degree vertices. We consider these nodes as leaders or influential members. We may consider
simple degree, closeness, and betweenness centralities or rank values as leadership degree; however, these metrics do not take into account disassortative degree mixing, which is a prevailing property of social networks. In other words, influential nodes not only possess high structural degree level but also tend to have disassortative degree mixing with their neighbors. Disassortative degree mixing is a homophilic measure which indicates any dissimilarity; for simplicity, we consider structural differences. For instance, high degree nodes tend to connect with low degree nodes and vice versa. In the first phase, DMID finds hubs with high disassortative degrees. We begin the first phase by defining a $N \times N$ assortative matrix named $AS$ that can be computed as follows:

$$AS_{ij} = \begin{cases} 
|\text{deg}(i) - \text{deg}(j)|, & \text{if } j \in \text{deg}(i) \\
0, & \text{otherwise}
\end{cases}, \quad (4.1)$$

where $|\text{deg}(i)|$ is the degree of node $i$ and $|\text{deg}(i) - \text{deg}(j)|$ is the absolute value. Now, we have a matrix that contains the degree difference corresponding to nodes. If we apply a random walker on the corresponding row-normalized transition matrix, we can identify disassortative paths. In other words, the walker tends to flow in directions with higher degree differences. The row-normalized disassortative matrix can be computed as follows:

$$TAS_{ij} = \frac{AS_{ij}}{\sum_{k=1}^{N} AS_{ik}}. \quad (4.2)$$

Afterwards, a Disassortative Vector (DV) is considered to hold the disassortative value of each node. We initialized $DV$ with $\frac{1}{|N|}$ and we updated it with the help of disassortative transition matrix $TAS$; therefore the update is based on:

$$DV^t = DV^{t-1} \times TAS. \quad (4.3)$$

After enough iterations, the process converges, and we obtain the disassortative value of each node. To be considered as a leader, the simple degree of nodes needs to be combined in addition to homophile. We can calculate the leadership value of node $i$ ($LV(i)$) as follows:

$$LV(i) = DV(i) \times |\text{deg}(i)|. \quad (4.4)$$

So far, each node $i$ can be represented by its relative leadership value $LV(i)$. To further proceed toward final leaders, vertices need to locally decide regarding local leadership. In other words, for each node $i$, we need to select a local leader as follows:

$$LV(i) > LV(j) \quad \forall j \in \text{deg}(i). \quad (4.5)$$

As the formula represents, node $j$ is the follower of node $i$. Therefore a forest is formed. Leafs of the forest are not good candidates to be considered as global leaders. However, they are good enough to be the local leaders, which can be achieved by comparing the number of followers of local leaders to the average number of followers in the forest. The average number of followers ($AFD$) is defined as follows:

$$AFD = \frac{\sum_{i \in LL} |\text{Follower}(i)|}{|LL|}. \quad (4.6)$$
Here $LL$ is the local leader set and $\text{Follower}(i)$ is the set of followers of node $i$ (nodes that have lower leadership degree in comparison to node $i$ and are connected to it). At last, we extract global leaders from local leaders based on the following formula:

$$|\text{Follower}(i)| > AFD.$$ (4.7)

Now $|GL|$ (number of leaders) has been identified which corresponds to the number of communities; however, the dependency of nodes to communities is missing. To compute membership degree of vertices to covers, we merely apply an information diffusion strategy with profitability gain for each member. We can consider different strategies and behaviors in the network and some nodes share common behaviors. In this regard, nodes’ opinions and beliefs are influenced by the opinion of their neighboring vertices. To simulate opinion formation and information diffusion, we may consider some resistance threshold (RT) and received gains (RG). For node $i$, the $RG$ value for behavior $A$ is aggregated based on the following:

$$RG_A(i) = \frac{|\{j \in \text{Nei}(i) : j \text{ has behavior } A\}|}{\text{deg}(i)}.$$ (4.8)

For each node $i$, if the $RG$ value is higher than the $RT$ value then it will accept the new behavior; otherwise, it will resist with its opinion. To further clarify our membership calculation strategy, consider a node $i$ with three neighbors and belief $A$. If two of the neighbors believe in behavior $A$ and one of them believe in behavior $B$ then the $RG(i) = 0.33$. If the RT value for node $i$ is 0.5, then it will resist on its current belief.

Similarly, in our simulations, we consider a unique behavior ($B$) for all of the nodes. All of a sudden, one of those detected global leaders changes its behavior to a new behavior $A$ which influences the neighboring nodes and cascades the new behavior $A$ through the network. To simplify the second phase, we consider equal $RT$ values for all of the nodes. For each of global leaders, it is needed to initiate such a changeover in behaviors. We start by $RT = 1$ and reduce it gradually; repeating the cascading process until all the nodes are a member of at least one community. In this information diffusion phase, the sooner a node adopts the new behavior, the stronger it depends on the corresponding leader and the community. In our simulations, we consider the following formula to calculate soft degree membership value of node $i$ to the leader (actually community) $l$:

$$M_{il} = \frac{1}{l_i^2},$$ (4.9)

where $M_{il}$ is the membership dependence of node $i$ on community $l$.

**Time Complexity**

The time complexity of the first phase of the algorithm is $O(M)$. To be more concrete, we can calculate the disassortative matrix with $O(M)$ and simulate the random walk process in $O(M)$. Moreover, we can compute the leadership vector in $O(N)$ and identify the local leaders in $O(M)$. Because sequential steps in the first phase, we can estimate the total time complexity of this phase as $O(M)$. To analyze time complexity of the second phase of the algorithm, we consider communities with the same size $O(\sqrt{N})$. In the worst case, only one
node at each time may adopt the behavior among the neighbors and if average degree of
nodes is $d$ then the time complexity can be $O\left(\frac{N^2}{d}\right)$. In other words, the second phase can be
estimated as $O(M \frac{N}{|L|})$. Finally the time complexity of the whole algorithm can be deduced
as $\max(O(M),O(M \frac{N}{|L|}))$, which is $O(M \frac{N}{|L|})$ in the worst case.

- **DMID weighted and Directed**

We further developed the directed and weighted version of DMID. In the first phase, the
disassortative and the transition matrices are calculated based on nodes’ weighted in-degrees
instead of node degrees. Moreover, the payoff definition slightly changes by:

$$RG_A(i) = \frac{\left|\{j \in Successor(i) : j \text{ has behavior } A\}\right|}{|Successor(i)|}, \quad (4.10)$$

that $Successor(i)$ determines the set of successors of node $i$.

### 4.2.2 Evaluation Results

In this section, we evaluate DMID by comparing it to other structural-based algorithms in unsigned
social networks. We implemented DMID and compared it with a couple of baselines. To eval-
uate OCD algorithms, we can use both real world and synthetic networks. Real-world networks
show fundamental properties of real-world social networks; however, the ground-truth information
regarding the communities are unknown. In this regard, synthetic networks would help to obtain
more extensive and broader view of the evaluation results. First, we introduce the evaluation of
results on LFR synthetic networks [LaFo09] which we introduced in chapter 2. Afterwards, we
mention the results on a couple of real-world networks by modularity and running time as metrics.
The prototypical implementation of this part of the research was supported in a bachelor thesis
guided by the author of this work [Krot14].

### Results on Synthetic Networks

Synthetic networks can be generated based on different parameters. In general, average node de-
grees, mixing parameter and number of overlapping nodes may challenge the algorithms in the
evaluation. In this case, DMID algorithm is compared with a couple of algorithms including SLPA,
SSK, CLiZZ and Link Community (Link) (see chapter 2). Figure 4.1 indicates a clear picture re-
garding the comparison of synthetic networks based on different parameters. $k$ indicates the average
node degree and $\mu$ is mixing parameters that connect communities. As we can observe $k = 12$ and
$\mu = 0.1$ indicate a network with a rather low node average degree and low mixing parameter. When
the percentage of overlapping nodes are small from 0 to 10 percent, SLPA has superior NMI value
in comparison to others. It is mainly followed by SSK, MONC, DMID, Link and CLiZZ. However,
as the number of overlapping nodes increase, the performance of SLPA slumps over DMID, Link
and CLiZZ. CLiZZ has the worst NMI value when the number of overlapping nodes increases in
networks with assigned parameters. DMID also has almost a stable NMI value (around 0.2) as
the number of overlapping nodes increase. Best cases are MONC followed by Link when overlap
enhances which results in NMI value of around 0.3.
4.2. PROPOSED OVERLAPPING COMMUNITY DETECTION IN UNSIGNED NETWORKS

Figure 4.1: This figure shows the NMI values of different OCD algorithms on LFR synthetic networks. The upper plots show networks with $k = 12$ and the lower plots indicate networks with $k = 24$. The left and right plots show networks with $\mu = 0.1$ and $\mu = 0.3$, respectively.

We can observe a bit different results when average node degree increases and is set to $k = 24$. Now MONC is quite improved to 0.5 when overlapping percentage even increases to 40%. Similarly, at the beginning in case of zero overlapping percentage, SLPA and SSK have the best NMI performance in comparison to others. SLPA performance decreases but this time is a bit better than SSK, CLiZZ, and DMID. Again DMID has a stable performance of around 0.2 for all of the overlapping percentages for these parameter settings. For instance, if we look at 40% of overlapping, MONC (0.49), SLPA (0.24), Link (0.22), DMID (0.21), SSK (0.15) and CLiZZ (0.07) obtain highest NMI values, respectively. Furthermore, for cases where mixing parameter is increased to $\mu = 0.3$, the same pattern is kept for both $k = 12$ and $k = 24$.

Figure 4.2 shows the running time in seconds for different algorithms and various parameter settings. The first interesting point is the high running time of Link which could be plotted in none of the figures. Moreover, SSK and MONC approximately had worse running time performance in comparison to other algorithms. As for all of the parameter settings and the number of overlapping percentages, SLPA, CLiZZ, and DMID finish in less than 10 seconds. SLPA wins with a bit better performance, and it is followed by DMID and CLiZZ, respectively.

Results on Real World Networks

To compare the appropriateness of different algorithms, we applied them to a wide range of available datasets. Table 4.2 shows some fundamental properties of these networks such as the number of nodes, edges and the type of these networks. The results of the modularity and the CPU times are inserted in Tables 4.3 and 4.4. In Table 4.3, we can observe that DMID has satisfactory modularity
performance in comparison to other algorithms. As we can observe, SLPA wins the first rank in 7 cases including DBLP (0.7115), Dolphins (0.7457), Email (0.6730), Hamsterster (0.5703), Internet (0.5410) and Zachary (0.6993). Moreover, it obtains the second rank in two cases including Sawmill (0.6812) and Facebook (0.9318). Additionally, DMID receives the first rank in two cases including Sawmill (0.6854) and Sawmill Strike (0.7523) and obtains the second rank for two cases in Zachary (0.6914) and Internet (0.5229). MONC is neither among the first nor the second rank algorithms, and CLiZZ only reaches the second rank for Power Grid (0.6810). Furthermore, SSK obtains the first rank in two cases of Power Grid (0.7014) and Facebook (0.9407), and it obtains the second rank for JAZZ (0.4135) and Sawmill Strike (0.7029). Finally, Link achieves the second rank in two cases of Email (0.4410) and Hamsterster (0.5110).

Regarding the running times of these algorithms, on the other hand, SLPA has the best running times in almost all of the datasets except in very much small datasets that the difference may be negligible. Link and SSK have the worst performance among the other algorithms. Moreover, DMID surpasses the others in many cases including Dolphins, Email. However, CLiZZ gets the superior performance in DBLP, Facebook, Hamsterster, Internet, and JAZZ. Finally, MONC wins only in Power Grid network.

Figure 4.2: This figure indicates the running times of the different algorithms on LRF synthetic networks. The upper plots show networks with $k = 12$ and the lower plots indicate networks with $k = 24$. The left and right plots show networks with $\mu = 0.1$ and $\mu = 0.3$, respectively.
4.2. PROPOSED OVERLAPPING COMMUNITY DETECTION IN UNSIGNED NETWORKS

Table 4.2: This table shows real-world networks that are employed to compare OCD algorithms. This table includes the number of nodes, edges, and type of each network.

<table>
<thead>
<tr>
<th>Graph</th>
<th>DBLP</th>
<th>Dolphins</th>
<th>Email</th>
<th>Facebook</th>
<th>Hamsterster</th>
<th>Internet</th>
<th>Jazz</th>
<th>Power Grid</th>
<th>Sawmill</th>
<th>Sawmill Strike</th>
<th>Zachary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>1959</td>
<td>62</td>
<td>1133</td>
<td>4039</td>
<td>2000</td>
<td>6474</td>
<td>198</td>
<td>13188</td>
<td>24</td>
<td>156</td>
<td></td>
</tr>
<tr>
<td>Edges</td>
<td>16354</td>
<td>318</td>
<td>10902</td>
<td>17646</td>
<td>32196</td>
<td>25144</td>
<td>5484</td>
<td>124</td>
<td>76</td>
<td>24</td>
<td>34</td>
</tr>
<tr>
<td>Type</td>
<td>Co-Authorship</td>
<td>Social</td>
<td>Social</td>
<td>Social</td>
<td>Social</td>
<td>Technological</td>
<td>Social</td>
<td>Technological</td>
<td>Social</td>
<td>Social</td>
<td>Social</td>
</tr>
</tbody>
</table>

Table 4.3: Modularity values for different OCD algorithms on real-world datasets.

<table>
<thead>
<tr>
<th>Graph</th>
<th>DBLP</th>
<th>Dolphins</th>
<th>Email</th>
<th>Facebook</th>
<th>Hamsterster</th>
<th>Internet</th>
<th>Jazz</th>
<th>Power Grid</th>
<th>Sawmill</th>
<th>Sawmill Strike</th>
<th>Zachary</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMID</td>
<td>0.4203</td>
<td>0.5140</td>
<td>0.3457</td>
<td>0.8257</td>
<td>0.2801</td>
<td>0.5229</td>
<td>0.3461</td>
<td>0.6433</td>
<td>0.6854</td>
<td>0.7523</td>
<td>0.6914</td>
</tr>
<tr>
<td>CLiZZ</td>
<td>0.4281</td>
<td>0.5218</td>
<td>0.2483</td>
<td>0.8820</td>
<td>0.3683</td>
<td>0.2253</td>
<td>0.0000</td>
<td>0.6810</td>
<td>0.4550</td>
<td>0.0000</td>
<td>0.5929</td>
</tr>
<tr>
<td>MONC</td>
<td>0.2102</td>
<td>0.3218</td>
<td>0.0886</td>
<td>0.0000</td>
<td>0.1727</td>
<td>NaN</td>
<td>0.0000</td>
<td>0.3638</td>
<td>0.4957</td>
<td>0.3321</td>
<td></td>
</tr>
<tr>
<td>SLPA</td>
<td>0.7115</td>
<td>0.7457</td>
<td>0.6730</td>
<td>0.9318</td>
<td>0.5703</td>
<td>0.5410</td>
<td>0.7532</td>
<td>0.5345</td>
<td>0.6812</td>
<td>0.6925</td>
<td>0.6993</td>
</tr>
<tr>
<td>SSK</td>
<td>0.4855</td>
<td>0.5506</td>
<td>0.4283</td>
<td>0.9407</td>
<td>0.3665</td>
<td>0.4065</td>
<td>0.4135</td>
<td>0.7014</td>
<td>0.5882</td>
<td>0.7029</td>
<td>0.5929</td>
</tr>
<tr>
<td>Link</td>
<td>0.3497</td>
<td>0.3599</td>
<td>0.4410</td>
<td>NaN</td>
<td>0.5110</td>
<td>0.0808</td>
<td>0.3386</td>
<td>0.0902</td>
<td>0.3407</td>
<td>0.3680</td>
<td>0.2370</td>
</tr>
</tbody>
</table>

Table 4.4: Running time of OCD algorithms on real world networks.

<table>
<thead>
<tr>
<th>Graph</th>
<th>DBLP</th>
<th>Dolphins</th>
<th>Email</th>
<th>Facebook</th>
<th>Hamsterster</th>
<th>Internet</th>
<th>Jazz</th>
<th>Power Grid</th>
<th>Sawmill</th>
<th>Sawmill Strike</th>
<th>Zachary</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMID</td>
<td>110.77</td>
<td>0.380</td>
<td>5.51</td>
<td>3102</td>
<td>734.19</td>
<td>2523</td>
<td>0.203</td>
<td>3040</td>
<td>0.040</td>
<td>0.028</td>
<td>0.018</td>
</tr>
<tr>
<td>CLiZZ</td>
<td>84.91</td>
<td>0.418</td>
<td>7.59</td>
<td>528.90</td>
<td>29.55</td>
<td>2484</td>
<td>0.140</td>
<td>400037</td>
<td>0.108</td>
<td>0.011</td>
<td>0.011</td>
</tr>
<tr>
<td>MONC</td>
<td>159.08</td>
<td>0.441</td>
<td>109.01</td>
<td>3808</td>
<td>87.38</td>
<td>NaN</td>
<td>0.296</td>
<td>62.41</td>
<td>0.020</td>
<td>0.009</td>
<td>0.074</td>
</tr>
<tr>
<td>SLPA</td>
<td>2.89</td>
<td>0.657</td>
<td>2.82</td>
<td>82.09</td>
<td>5.04</td>
<td>5.09</td>
<td>2.29</td>
<td>42.54</td>
<td>0.129</td>
<td>0.018</td>
<td>0.073</td>
</tr>
<tr>
<td>SSK</td>
<td>3911</td>
<td>0.360</td>
<td>129.98</td>
<td>3528</td>
<td>7406</td>
<td>6546</td>
<td>0.656</td>
<td>204797</td>
<td>0.236</td>
<td>0.029</td>
<td>0.023</td>
</tr>
<tr>
<td>Link</td>
<td>12501</td>
<td>0.545</td>
<td>3074</td>
<td>NaN</td>
<td>133532</td>
<td>48736</td>
<td>304.33</td>
<td>5083</td>
<td>0.065</td>
<td>0.014</td>
<td>0.045</td>
</tr>
</tbody>
</table>
4.3 Proposed Overlapping Community Detection in Signed Networks

In this section, we address the overlapping community detection in signed social networks and we propose SDMID for networks with both positive and negative connections. Signed social networks and OCD in signed networks are further introduced in chapter 2 and 3.

4.3.1 Signed Disassortative Degree Mixing and Information Diffusion

In the second part of this thesis, we demonstrate the proposed structural OCD approach suitable for signed social networks named Signed Disassortative Degree Mixing and Information Diffusion (SDMID). Similar to DMID, the first phase involves identifying leaders; however, effective degree and normalized disassortative value of each node are employed. While negative edges also exist, we adopt the simple degree to effective degree. Moreover, to avoid the random walk process for negative connections, we employ simple positive and negative links passing through a node. Afterwards, we adopt network coordination game for networks with both positive and negative connections. In the following, we explain SDMID algorithm in details.

Identifying Leaders

In the first phase, the algorithm identifies most influential nodes named leaders. Leaders not only possess high influence but also they have high dissimilarity values with their neighbors. Due to the existence of negative links, computing influence is different from unsigned networks. We define the influence or effective degree of node $i$ as follows:

$$ED(i) = \frac{\#\text{indeg}^+(i) - \#\text{indeg}^-(i)}{\#\text{indeg}^+(i) + \#\text{indeg}^-(i)}, \quad (4.11)$$

where $ED(i)$ is the effective degree of node $i$. We consider nodes with positive $ED$ values to identify leaders. Moreover, a leader not only has high degree influence, but also it has degree difference with its neighbors. Disassortative degree mixing denotes heterogeneity and disassortative-ness of a node indicate how much the node differs from its neighbors’ degree. We can define the normalized disassortative-ness of node $i$ as follows:

$$DASS(i) = \frac{\sum_{j \in \text{Nei}(i)} (\#\text{deg}(j) - \#\text{deg}(i))}{\sum_{j \in \text{Nei}(i)} (\#\text{deg}(j) + \#\text{deg}(i))}, \quad (4.12)$$

where $DASS(i)$ is the disassortative-ness of node $i$. To compute this value, we consider negative edges as positive and only take into account the number of connections. Now, with the contribution of $DASS$ and $ED$, we can further define the local leadership of each node $i$ as follows:

$$LLD(i) = \alpha \times DASS(i) + (1 - \alpha) \times ED(i), \quad (4.13)$$
where $LLD$ is the local leadership degree of node $i$, $\alpha$ is a parameter which weight to either disassortative-ness or influence of node $i$. In the experiments, we set $\alpha$ to 0.5 because both effective degree and disassortative degree mixing are two significant factors to identify leaders and it is better not to underestimate one of them. Similar to DMID, we proceed to determine local leaders. To this aim, we compare local leadership value of each node to its neighbors. In other words, a node $i$ is a local leader if for $\forall j \in Nei(i)$

$$LLD(i) > LLD(j),$$

(4.14)

We denote the local leader set with $LL$. The algorithm also selects global leaders from local leaders with a sufficient number of followers based on formula 4.6 and 4.7.

### Signed Cascading Process

Similar to DMID, we compute the degree membership of each node to the leaders by using a cascading process named network coordination game. As mentioned before, in the network coordination game, all nodes have the same behavior, but at a certain point in time, a set of nodes changes the behavior. We are interested to know which and how many nodes are affected by this change. In other words, if all the nodes in the network have behavior $A$ and at some point in time a set of nodes changes the behavior to $B$ then we are interested to know which nodes will be affected by behavior $B$. In signed social networks, this process is a little different. For instance, let us consider an example which node $i$ has behavior $B$ and node $j$ and $k$ have behavior $A$. Although the link between $i$ and $j$ is negative and the link between $i$ and $k$ is positive, $j$ will not receive any payoff from node $i$ and even resists with behavior $B$. On the contrary, $k$ gets some payoff and prefers to change its behavior to the behavior of $B$. We should compute the received pay-off by a node to check whether the node changes its behavior. Therefore, by adapting formula 4.8, node $i$’s payoff for a new behavior $A$, can be computed as follows:

$$\text{PayOff}(i) = \frac{\#(j_A \in Nei^+(i)) - \#(j_A \in Nei^-(i))}{\#(j_A \in Nei^+(i)) + \#(j_A \in Nei^-(i))},$$

(4.15)

In the above formula, $j_A$ has behavior $A$ which is different than the current behavior of node $i$. If the received pay-off by node $i$ is more significant than the node $i$’s threshold, then the node changes its behavior. In the experiments, we equal the number of leaders to the number of communities. To compute degree membership of a node to a particular community $j$, we change the behavior of the community $j$’s leader. In other words, other nodes in the network are affected based on their dependency on the corresponding leader of community $j$. Membership of each node to the community is computed based on the time iteration a node changes their behavior. We apply equation 4.9 for this purpose.

### 4.3.2 Evaluation Results

We implemented SDMID and compared it with two methods from the-state-of-the-art. Similar to the evaluation of OCD algorithms in unsigned social networks, we can employ both synthetic and
real-world networks. While there are few big real-world networks with both positive and negative connections, we only consider Epinions, Slashdot, and Wikipedia; however, the running times of the baseline methods such as SPM and MEA are too high that cannot finish within a reasonable time on big networks of Slashdot and Epinions. Hence, we consider WikiElec as for the real-world network. Because of the limited number of real-world networks, we consider a more comprehensive analysis over the signed LFR networks (see chapter 2). First, we mention the evaluation on synthetic networks. The prototypical implementation of this part of the research was supported in a bachelor thesis guided by the author of this work [Li2016].

In the experiments, we only changed one parameter and kept fixed all other parameters. We used a network of size $n=100$ to avoid data volume complexities. The default parameter values for the experimented synthetic networks are $k=3$, $maxk=6$, $\mu=0.1$, $t_1=-2.0$, $t_2=-1.0$, $minc=5$, $maxc=30$, $on=5$, $om=2$, $P_-=0.1$, $P_+=0.1$. We realize experiments for three times and average over the results. Regarding the algorithm parameters, we may require specifying the number of communities in the SPM algorithm. As such, we considered two communities as for SPM algorithm and three as for the number of trials of the EM method.

![Figure 4.3: Performance overview of MEA, SDMID and SPM in signed LFR networks with different network size.](image)

**Network Size $n$**

The first parameter that we change is network size. In this experiment, we change the network size $n$ from 50 to 400 and observe running times, frustration, modularity, and NMI of three algorithms including SDMID, MEA and SPM in Figure 4.3. Regarding execution time, SDMID achieves
superior performance in comparison to MEA and SPM. The running time for SPM is volatile, and we observe two peaks at network sizes 150 and 350. MEA's running time grows exponentially with the network size, which has the worst performance in comparison to SPM and SDMID.

As for modularity, we can observe that SDMID defeats MEA with a high difference and it achieves competitive and in many cases better modularity performance in comparison to SPM. For instance, if we consider network size of 200, SDMID achieves 0.42 value which is higher than SPM (0.28) and MEA (0.40). As network size increases, SDMID proves to hold the best position regarding modularity although SPM is quite competitive and reaches SDMID performance for bigger networks. Regarding frustration, SPM achieves lower frustration values that show its better performance in comparison to SDMID and MEA. SDMID obtains frustration values around 0.05 which is close to SPM. Last but not least, regarding the ground-truth-ness of communities, MEA and SPM have the highest performance. In small networks, wherein principle number of communities are small, SPM proves better performance due to the default number of communities as 2; however, when the network size increases, SDMID and MEA accelerate. Overall regarding network size parameter, SDMID achieves the best performance as for modularity and running time in this configuration. SPM obtains the best frustration values, and MEA produces the highest NMI values.

![Figure 4.4: Performance Overview of MEA, SDMID and SPM in signed LFR networks with different average node degree.](image)

**Average Node Degree \( k \)**

We also experimented the performance of these three algorithms using average degree which is indicated in the Figure 4.4. We altered the parameter between 3 and 12 with the maximum degree of 15. First of all, SDMID is the fastest algorithm for networks with higher average degrees which
is followed by MEA with stable but slower running time values. Next, SPM achieves unstable running times with three peaks ranging between 40000 and 45000 milliseconds which is quite high in comparison to others. If we investigate NMI diagram, we figure out that SDMID achieves more or less the best performance among the other two algorithms. MEA only surpasses SDMID when the average degree is 5. So for $k = 8$, SDMID obtains the best NMI values. NMI comparison of MEA and SPM is somewhat volatile; however, we can mention that MEA achieves better performance for networks with average degree between 3 and 7 and lower NMI values when average degree increases up to 12.

Regarding modularity, SDMID is quite competitive and in many cases better in comparison to other algorithms. The modularity performance is quite better while average degrees are low. It might be because disassortative degree mixing property of SDMID distinguishes better the community structure in these types of networks and the lower average degree might show better this property. With average degrees 10 and 11, SDMID, SPM and MEA achieve similar values of around 0.3 and 0.29. Finally, concerning frustration, except average degrees 3 and 4, SDMID achieves the worst frustration values; however, MEA and SPM obtain quite competitive and lower frustration values. Overall, SDMID could achieve satisfactory performance about modularity, running time and NMI when average degree parameter changes.

![Performance overview of MEA, SDMID, and SPM in signed LFR networks with different maximum node degree.](image)

**Maximum Node Degree $maxk$**

We also experimented the algorithms based on the maximum size of communities which range between 5 and 16. As we can observe in Figure 4.5, SDMID achieves a clear advantage over
the other two algorithms regarding the time complexity which is followed by SPM that is slightly slower than SDMID. SPM falls behind MEA; however, in two points with maximum sizes 6 and 16, its running time rise. Regarding modularity, SDMID obtains the best performance followed closely by SPM; however, MEA produces fewer modularity values. As for NMI, the results are blended; however, MEA achieves slightly better NMI values approximately for all of the networks and SDMID is ranked second regarding the highest NMI values in many cases. With maximum community sizes 12 and 14, SDMID and MEA both obtain 0.26 as for the NMI. Finally, SPM defeats the other two algorithms regarding frustration and it is followed by SDMID that slightly obtains higher frustration values. MEA acquire high frustration values in all of the networks which are not as satisfactory as SDMID and SPM.

Figure 4.6: Performance overview of MEA, SDMID and SPM in signed LFR networks with a different fraction of edges sharing with other communities.

**Fraction of Edges Sharing with Other Communities $\mu$**

A more difficult experiment for the algorithms is to change the fraction of edges shared with other communities for each node which is depicted in Figure 4.6. Here, we change $\mu$ from 0.05 to 0.185, and higher values of $\mu$ indicate more top interactions among communities which makes it difficult for the algorithms to detect the structures. In this experiment, results for running times are similar for all the algorithms; however, SDMID reaches the best time complexities for all the parameter settings. Regarding frustration, SPM achieves the best frustration values which is catch up by DMID for $\mu=0.08$ and 0.155 - SDMID and SPM achieve 0.02 and 0.04 as for these parameters. In this plot, MEA has quite higher frustration values. Regarding modularity, SDMID surpasses SPM for all $\mu$ values except 0.05 and SPM has much better modularity performance in comparison to
Overlapping Community Detection

Regarding NMI, the situation is better for MEA for almost all of the $\mu$ values; however, for 0.095, 0.14, 0.155 and 0.17, there exist some exceptions of competitive performance. As for 0.14, SPM achieves 0.25 followed by MEA and SDMID with 0.19 as for the NMI value. Overall, SDMID achieves better performance regarding execution times and modularity runs. SPM and MEA are the best algorithms for Frustration and NMI, respectively.

Figure 4.7: Performance overview of MEA, SDMID and SPM in signed LFR networks with different maximum community size.

**Maximum Community Size maxc**

In this experiment, we change the maximum size of communities from 30 to 40 which is depicted in the Figure 4.7. In this regard, SDMID achieves the best time complexity followed by SPM and MEA, similar to previous experiments. As for frustration values, SDMID gets worse performance in comparison to SPM but rather much better frustration results compared to MEA. For instance, as for $maxc=35$, SDMID, SPM and MEA obtain 0.04, 0.02 and 0.14, respectively as for the frustration values. Regarding modularity, the situation for SDMID is better than the other two except $maxc=32$ which SPM and SDMID achieve equal modularity of 0.38. Finally regarding NMI, SDMID is better in many cases although MEA was better in previous experiments. For instance as for $maxc=34$, SDMID, SPM and MEA achieve 0.29, 0.19 and 0.21, respectively. There are also cases where MEA is better such as $maxc=32$ which its NMI value is 0.35 followed by 0.25 and 0.19 as for SDMID and SPM, respectively.
4.3. PROPOSED OVERLAPPING COMMUNITY DETECTION IN SIGNED NETWORKS

Figure 4.8: Performance overview of MEA, SDMID and SPM in signed LFR networks with different number of nodes in overlapping communities

**Number of Nodes in Overlapping Communities on**

All the algorithms were tested similarly to previous experiments; however on value, which is the number of nodes in overlapping communities, is parametrized from 1 to 10. As it can be seen in Figure 4.8, SDMID gets the lowest running times followed by unstable SPM running times - SPM obtains better time complexities as on increases. Regarding frustration, SPM achieves better frustration values thoroughly which is followed by SDMID; however, SDMID frustration values improved as on increases and it is catch up at on=9. On the contrary, MEA obtains the worst frustration values. Regarding modularity, SDMID enjoys the best modularity values, and it generates higher modularity values as on increases. If we consider on value 3, SDMID, MEA and SPM achieve 0.37, 0.27, 0.36 as for modularity values, respectively. Moreover, regarding on value of 9, SDMID, MEA and SPM obtain 0.43, 0.26 and 0.37 as for the modularity values. If we look at NMI values, we observe that MEA is the best in the first three networks on=1, 2, 3; however, as for on=4, 5, 6, 7 the algorithms reach quite similar performance. Finally, for on=8, 9, 10, SDMID beats the other two algorithms. For instance, for on=10, SDMID obtains 0.26 which is higher than MEA (0.22) and SPM (0.19).

**Number of Communities Which Nodes in Overlapping Communities Belong to om**

om is the number of communities of overlapping nodes which is ranged between 2 and 10 and is depicted in the Figure 4.9. On the contrary to the previous set up, we generated networks of size 200 to have a higher number of overlapping communities. Similar to previous experiments, SDMID has the best running time performance with following unstable SPM running times and stable MEA time
Overlapping Community Detection

Figure 4.9: Performance overview of MEA, SDMID and SPM in signed LFR networks with a different number of communities which nodes in overlapping communities belong to.

complexities for all \( om \) values. Regarding frustration, SPM obtains better performance; however, in many cases with equal and competitive values; \( om=6 \) and 8. As for \( om=2, 3 \) and 4, SDMID is slightly worse than SPM; however, the values are close. On the contrary, MEA is far behind these two algorithms. As for modularity, SDMID and SPM are again quite competitive with SDMID obtaining slightly better values; however, both of them achieve satisfactory modularity values. For instance, \( om=10 \), SDMID, SPM and MEA obtain 0.4, 0.4 and 0.3, respectively. You may observe that MEA does not generate modularity values as satisfactory as SDMID and SPM. The situation for MEA improves while we talk about NMI. In the beginning, \( om=2, 3 \) and 4, MEA achieves the best NMI performance with the maximum of 0.32; however, it is surpassed by SDMID for \( om \) values larger than 5. Although SDMID and MEA are still competitive, SDMID (0.24) defeats MEA (0.22) and SPM (0.10).

Fractions of Positive Connections between Communities \( P_+ \)

We also change the fractions of positive connections between communities \( (P_+) \). As we can observe in Figure 4.10, SDMID achieves the best running time values, and it is followed by SPM and MEA, respectively. As for modularity, SDMID has the best performance in almost all of the cases except \( P_+ =0.07 \). SPM also generates satisfactory modularity values similar to SDMID; however, MEA falls behind with around 0.1 difference. Overall, modularities for all the algorithms decrease as the community borders get lost. Moreover, similar to previous experiments, SPM has the best performance as for frustration which SDMID is its competitor with slightly higher values. Regarding \( P_+ =0.08 \), SDMID, MEA and SPM achieve 0.08, 0.18 and 0.05, respectively. As for NMI values, MEA obtains the highest NMI value in almost all of the cases except \( P_+ =0.03 \) and 0.05. For
4.3. PROPOSED OVERLAPPING COMMUNITY DETECTION IN SIGNED NETWORKS

Figure 4.10: Performance Overview with Different Fractions of Positive Connections between Communities.

\( P_+ = 0.03 \), SPM has the highest NMI value of 0.22 which is followed by MEA (0.19) and SDMID (0.16). In addition, SDMID (0.26) is superior to MEA (0.18) and SPM (0.20).

Evaluation of WikiElec Real-World Signed Network

Now we analyze a real-world network named *Wikipedia Election (Wiki-Elec)*. Users from all around the world have collaboratively written Wikipedia as a free glossary. There exist different users with different levels of authorization. Administrators have higher access level for editing and other actions. Also, there exist elections for selecting these administrators. Wikipedia users can positively and negatively vote for the candidates. Hence, it is a signed social network. To analyze this dataset, SPM requires a realistic number of communities; then five is selected based on the experiments of Javari and Jalili [JaJa14] that make sign predictions in large-scale networks. The number of trials of EM algorithm is determined to be 3 to achieve realistic results. As for SDMID, we set the network coordination game to 3 iterations, leaving out loosely connected nodes for each community. As we can observe in Figure 4.11, SDMID achieves superior performance in comparison to its peers regarding modularity and execution time. SPM obtains the smallest frustration error, which is only slightly better than that of SDMID (0.0953 vs. 0.0962).

Figure 4.11 indicates the community distribution of the found covers by these algorithms. We can figure out that MEA detects huge communities while assigning a large number of nodes to be stand-alone. SPM generates the lowest standalone nodes while grouping the remainders in five clusters ranging from 2148 to 3043. SDMID identifies three clusters with approximately uniform sizes. MEA only specifies 5 nodes in overlapping parts, on the contrary, SDMID and SPM detect more than 6,000 overlapping nodes, which may reflect the real world.
Overlapping Community Detection

Finally, experiments on synthetic networks are performed based on changing the value of one parameter and keeping the rest unchanged. We average over the results over all the runs and render the results through radar diagrams. A radar chart or spider diagram shows the performance of a specific algorithm regarding a single evaluation metric through one of its poles. We normalized the values of the metrics and scaled them 100 times at each pole. Here, the higher values indicate better performance. As it can be observed in the Figure 4.12, SDMID achieves the best execution time for both Wiki-Elec and in case of synthetic networks. MEA and SPM behave somewhat similar and much lower performance in comparison to SDMID. Furthermore, regarding modularity, SDMID obtains the best result in Wiki-Elec and synthetic networks, which is followed by SPM and then by MEA. Moreover considering frustration metric, we may figure out that SPM achieves the lowest frustration error in Wiki-Elec which is followed by SDMID and MEA.
Figure 4.12: Evaluation of algorithms with different metrics through radar diagrams. Red, green and blue show SDMID, MEA, and SPM, respectively. The top spider diagram is related to synthetic networks, and the bottom shows WikiElec. The values are normalized such that the more prominent area indicates better performance regarding a metric.
4.4 Conclusion

In this chapter, we introduced two algorithms for overlapping community detection. The first algorithm is DMID, which works based on disassortative degree mixing and information diffusion in social networks. DMID achieved satisfactory results in our experiments. It also has some other features, for instance, it may be suitable for disassortative degree mixing networks, weighted and directed graphs. Moreover, it can identify the hierarchical structure of the networks. However, DMID is not the best performing algorithm regarding running time. In several cases of the experiments, other algorithms showed better performance. In this regard, we may improve the running time by using different dynamics, i.e., using backward propagation with multiple leaders. We will investigate DMID in the next chapters, for community evolution prediction, recommendation, and expert identification. We also extended DMID to signed social networks. Our experiments showed that SDMID achieves a superior performance in comparison to the MEA and SPM algorithms regarding different evaluation metrics. Still, for larger scale real-world networks, SDMID may show time complexity problems. Parallel implementation of the algorithm or using other dynamics, e.g., backward propagation might be helpful.
Chapter 5

Overlapping Community Analysis and Prediction Models

5.1 Introduction

Complex networks evolve which vertices are added or removed and connections are formed or deformed. Similar to networks, communities are also temporal. We are interested in predicting future of communities [LKFa07]. In general, temporality has major applications in different domains. For instance, in an informal learning forum, it is essential to predict the evolution of learning communities. Other prediction models in social networks also exist. One of such models is the link prediction models, which it predicts if two persons will have connections together in future [YSL*12]. Furthermore, we can mention prediction of ratings in recommender systems from users to specific items, and we see how this issue affects the users to find their intended items [Kore09].

Prediction models origin from data mining which supervised, unsupervised and semi-supervised learning models are employed to predict the future. Supervised learning models are those which labels of data are known. On the other hand, unsupervised learning techniques try to find similar patterns without knowing the labels. Finally, semi-supervised learning methods take the approaches from the latter two methods [Murp12]. As such, prediction models have many applications in online social networks. For instance, let us consider the example of presidential polling which content of social networks in different formats, e.g., Tweets from Twitter are used to predict the election outcome. Even specific prediction models are applied to foresee what events will cascade in close future times. Prediction models are extensively being used in marketing and business which we are interested in fitting proper functions to market data. Finally, other applications have employed prediction model metrics like accuracy and precision to prove their performance. As such, ranking algorithms are connected with prediction models. In other words, some research works could not determine the convergence of ranking methods, and thus they applied evaluation prediction metrics to observe how the ranking methods could contribute to prediction [ShJa14].

Summary. This chapter is related to prediction models. It comprises two sections including community evolution prediction and sign prediction models. In the first section, we predict the evolution of overlapping communities. In other words, we map the communities over time with the help of a
method named Group Evolution Discovery (see chapters 2 and 3). Afterwards, we extracted some events from the time-evolving data sets which we labeled the data to prepare it for the prediction model. In this section, we describe the corresponding features applied in the prediction task and their importance. We dedicate the second part of this chapter to sign prediction models. In fact, with the help of the sign prediction models, we investigate the role of nodes in different parts of communities. To put this another way, we introduced three new in-degree and out-degree features to differentiate among intra, overlapping and extra node classes. Then, we explain how we apply these features to the case of sign prediction problem. In this part, we also demonstrate two ranking algorithms for signed networks, which they consider opinions of overlapping nodes. Additionally, we employ the rank values of these proposed community-aware ranking methods to the case of sign prediction model. Finally, we discuss the findings and conclude this chapter.

Here, we consider the second research question from chapter 1 and approach to tie up network science problems such as community and sign prediction, and ranking algorithms to find laws and principles describing the networks. We identify the features which are significant in determining the future traces of overlapping communities. Moreover, we figure out how significant are overlapping members in social networks. In other words, we answer the research questions regarding the significance of overlapping community structures and strong features to predict overlapping communities. To identify the importance of overlapping community structures, we selected networks with positive and negative connections because communities can favor trust enhancement. To put this another way, members of the same community have higher levels of trust than the people that are in different communities. As such, we investigate the effect of overlapping community structures in the sign prediction problem. We could indicate that overlapping members could predict signs of both positive and negative links as reliable as intra and extra members of communities. Moreover, we could figure out a combination of features necessary for different community evolution events - the size of a community was the most important one.

5.2 Community Evolution Prediction

First, we start by the community evolution prediction. In this regard, we introduce the features used for the classifiers. Afterwards, we explain the evaluation results. We applied GED technique (see chapter 2) to label the temporal communities and provide the ground-truth information for the prediction problem. Next, we employed logistic regression as for the classifier to build a model for prediction of overlapping communities using the node and community level features.

5.2.1 Community Evolution Prediction Features

In this section, we will introduce the features to predict the future of overlapping communities. Here, we discuss community and node level features separately. To predict the evolution of communities, we need to map the communities over time and build a prediction model. Any supervised prediction task requires features and labels to train the data. As for the features, we extracted static and temporal community level features. These features are given in the Table 5.1.
We can classify the features in Table 5.1 as three categories. Node level features include leader ratio, leader degree, closeness and eigenvector centralities, which are related to the leaders of a community. Several community level features are also considered such as size ratio, density, cohesion, clustering coefficient, average assortative degree, closeness, degree and eigenvector centralities of communities. These features indicate how good the general structure of a community is. Other nodes and community-based features are also applied. In other words, a change in the features mentioned above in consecutive time slots \( t + 1 \) and \( t \) are taken into account as temporal features as for the classifier. Events that happen to communities are merge, split, survive and dissolve. We consider a binary classification problem that constitute polar labels of \{survive, not – survive\}, \{merge, not – merge\}, \{dissolve, not – dissolve\} and \{split, not – split\}. For instance, not – survive consists of merge, split and dissolve. As mentioned before, we used the logistic regression as for the classifier.

Table 5.1: List of Features Employed in the Community Evolution Prediction Task. These features include node and community level ones that each of them can be regarding the current snapshot or the previous time slot. \( \delta \) also shows the change in two consecutive values.

<table>
<thead>
<tr>
<th>Features</th>
<th>Definition</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>leader ratio</td>
<td>#leaders (\frac{\text{leaders}}{\text{all nodes}})</td>
<td>leader leaders ratio</td>
<td>(0,1)</td>
</tr>
<tr>
<td>leader average degree</td>
<td>(\frac{\sum_{i \in \text{leaders}} \text{deg}(i)}{\text{leaders}})</td>
<td>leader average degree</td>
<td>(0,1)</td>
</tr>
<tr>
<td>leader average closeness</td>
<td>(\frac{\sum_{i \in \text{leaders}} C(i)}{\text{leaders}})</td>
<td>leader average closeness</td>
<td>(0,1)</td>
</tr>
<tr>
<td>leader eigen centrality</td>
<td>(\frac{\sum_{i \in \text{leaders}} E(i)}{\text{leaders}})</td>
<td>leader average eigenvector</td>
<td>(0,1)</td>
</tr>
<tr>
<td>size ratio</td>
<td>(\frac{\text{Size}^t}{\text{Size}^{t+1}})</td>
<td>average community size</td>
<td>(0,1)</td>
</tr>
<tr>
<td>density</td>
<td>(\text{Den}^t = \left[\frac{\sum_{p \in \text{all pairs}} \text{vp}(p)}{\text{leaders} \times</td>
<td>\text{leaders} - 1</td>
<td>}\right])</td>
</tr>
<tr>
<td>cohesion</td>
<td>(\text{CO}^t = \frac{n}{\sum_{v \in \text{leaders}} (\text{vp}(v) - 1)})</td>
<td>how cohesive is the community</td>
<td>(0,∞)</td>
</tr>
<tr>
<td>clustering coefficient</td>
<td>(\text{CC}(V) = \frac{1}{\sum_{p \in \text{all pairs}} \text{vp}(p)})</td>
<td>counting number of triads</td>
<td>(0,1)</td>
</tr>
<tr>
<td>assortative degree mixing</td>
<td>(\rho = 1 - \left(\frac{\sum_{p \in \text{all pairs}} \text{vp}(p)}{\text{leaders} \times</td>
<td>\text{leaders} - 1</td>
<td>}\right))</td>
</tr>
<tr>
<td>degree centrality</td>
<td>(D(u) = \frac{\sum_{v \neq u \in \text{leaders}} \text{deg}(v)}{\text{leaders}})</td>
<td>simple degree measure</td>
<td>(0,1)</td>
</tr>
<tr>
<td>closeness centrality</td>
<td>(C(u) = \frac{\sum_{v \neq u \in \text{leaders}} \text{vp}(v)}{\text{leaders}})</td>
<td>centrality based on closeness</td>
<td>(0,1)</td>
</tr>
<tr>
<td>eigenvector centrality</td>
<td>(E(u) = \frac{1}{\text{leaders}} \sum_{v \neq u \in \text{leaders}} \text{vp}(v))</td>
<td>centrality based on eigenvectors</td>
<td>(0,1)</td>
</tr>
<tr>
<td>(\delta) leader ratio</td>
<td>#leaders (\frac{\text{leaders}^{t+1}}{\text{leaders}^{t}})</td>
<td>change in leader ratio</td>
<td>(0,1)</td>
</tr>
<tr>
<td>(\delta) size ratio</td>
<td>(\text{Size}^{t+1} - \text{Size}^{t})</td>
<td>change in size</td>
<td>(0,1)</td>
</tr>
<tr>
<td>(\delta) density</td>
<td>(\text{Den}^{t+1} - \text{Den}^{t})</td>
<td>change in density</td>
<td>(0,1)</td>
</tr>
<tr>
<td>(\delta) cohesion</td>
<td>(\text{CO}^{t+1} - \text{CO}^{t})</td>
<td>change in cohesion</td>
<td>(0,1)</td>
</tr>
<tr>
<td>(\delta) clustering coefficient</td>
<td>(\text{CC}^{t+1} - \text{CC}^{t})</td>
<td>change in clustering coefficient</td>
<td>(0,1)</td>
</tr>
<tr>
<td>(\delta) degree centrality</td>
<td>(D^{t+1} - D^{t})</td>
<td>change in degree</td>
<td>(0,1)</td>
</tr>
<tr>
<td>(\delta) closeness centrality</td>
<td>(C^{t+1} - C^{t})</td>
<td>change in closeness centrality</td>
<td>(0,1)</td>
</tr>
<tr>
<td>(\delta) eigenvector centrality</td>
<td>(E^{t+1} - E^{t})</td>
<td>change in eigenvector centrality</td>
<td>(0,1)</td>
</tr>
<tr>
<td>(\delta) previous survive</td>
<td>(\text{Survive}^{t}_{c})</td>
<td>survive as previous event?</td>
<td>true, false</td>
</tr>
<tr>
<td>(\delta) previous merge</td>
<td>(\text{merge}^{t}_{c})</td>
<td>merge as previous event?</td>
<td>true, false</td>
</tr>
<tr>
<td>(\delta) previous split</td>
<td>(\text{split}^{t}_{c})</td>
<td>split as previous event?</td>
<td>true, false</td>
</tr>
<tr>
<td>(\delta) previous dissolve</td>
<td>(\text{dissolve}^{t}_{c})</td>
<td>dissolve as previous event?</td>
<td>true, false</td>
</tr>
</tbody>
</table>
5.2.2 Evaluation Results

First, structural properties of the applied OCD algorithm including DMID, SLPA, and AFOCS (see chapter 2 and 3) need to be investigated. In this regard, we apply a couple of datasets, which we introduce some of them in chapter 4 and Table 5.3 shows some of them in this chapter. The prototypical implementation of this part of the research was supported in bachelor and master theses [Guna15, Doma15] guided by the author of this work.

The number of overlapping nodes, average community size and the number of communities is among the explored properties; Table 5.2 shows the information. We can understand that number of overlapping nodes for the DMID algorithm is approximately high in comparison to SLPA and AFOCS. SLPA detects less overlapping nodes; however, it is more than AFOCS. For instance, Power, NetScience, and PolBlogs, DMID has the highest overlapping percentage considering 0.96, 0.55 and 0.839, respectively. AFOCS detects 0.076, 0.1 and 0, respectively and SLPA identifies 0.404, 0.16 and 0.002, respectively. Regarding the number of detected communities, different algorithms show different resolution levels. The number of communities identified by AFOCS is way more than DMID and SLPA. SLPA generates yet more communities in comparison to DMID. Similarly, regarding Power, NetScience and PolBlogs, 4256, 658, 823 are detected by AFOCS. Moreover, SLPA detects 737, 407 and 5 whereas DMID detects 569, 61 and 15. When there is a low number of communities with the high overlapping percentage, we can deduce that size of communities should be big. In this regard, DMID has the biggest community sizes of more than 500 for bigger datasets which is followed by SLPA and AFOCS. In this regard, AFOCS communities are limited to the size of around 3-5 with an exception regarding PolBlogs with the average number of 148 members.

<table>
<thead>
<tr>
<th>Network</th>
<th>SLPA</th>
<th>SLPA</th>
<th>AFOCS</th>
<th>AFOCS</th>
<th>AFOCS</th>
<th>DMID</th>
<th>DMID</th>
<th>DMID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>AvgSize</td>
<td>Ovl</td>
<td>C</td>
<td>AvgSize</td>
<td>Ovl</td>
<td>C</td>
<td>AvgSize</td>
</tr>
<tr>
<td>Karate Club</td>
<td>4</td>
<td>10.5</td>
<td>0.23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>Dolphins</td>
<td>3</td>
<td>216</td>
<td>0.048</td>
<td>34</td>
<td>3.3</td>
<td>0.008</td>
<td>6</td>
<td>25.2</td>
</tr>
<tr>
<td>PolBlogs</td>
<td>5</td>
<td>245.4</td>
<td>0.002</td>
<td>823</td>
<td>148</td>
<td>0</td>
<td>15</td>
<td>1026.2</td>
</tr>
<tr>
<td>NetScience</td>
<td>407</td>
<td>4.1</td>
<td>0.1656</td>
<td>658</td>
<td>4.2</td>
<td>0.1</td>
<td>61</td>
<td>177.31</td>
</tr>
<tr>
<td>Power</td>
<td>737</td>
<td>9.9</td>
<td>0.40376</td>
<td>4256</td>
<td>2.1</td>
<td>0.076</td>
<td>569</td>
<td>462.8</td>
</tr>
<tr>
<td>CA-GrQc</td>
<td>954</td>
<td>6.9</td>
<td>0.24475</td>
<td>2683</td>
<td>3.9</td>
<td>0.038</td>
<td>263</td>
<td>654.7</td>
</tr>
<tr>
<td>p2p-Gnutella08</td>
<td>230</td>
<td>18.24</td>
<td>0.50436</td>
<td>5274</td>
<td>2.39</td>
<td>0.04</td>
<td>614</td>
<td>573.4</td>
</tr>
</tbody>
</table>

Table 5.2: The number of communities, number of overlapping nodes and the average size of communities for three algorithms including AFOCS, DMID, and SLPA are computed on a couple of real-world networks. We can observe the structural property differences of these three algorithms.

Significance of Features

To compare the significance of applied features for prediction of each OCD algorithm, we plotted their HeatMap in Figure 5.1. A HeatMAP figure best describes the significance of various features. The features are plotted in bold blue when the feature is important for the prediction task. The first finding is the significance of size ratio feature in all of the datasets and with the contribution of all algorithms. Although different algorithms lead into approximately different communities and
5.2. COMMUNITY EVOLUTION PREDICTION

different predictions, size of a community has been shown to be a significant and prevailing factor to predict various events happening to a community. Regarding DMID algorithm, in survive degree centrality, in merge temporal features such as previous merge or previous split, in split degree centrality and leader degree centrality and in dissolve density and eigenvector centralities are more important. In Enron dataset, we can observe that leader closeness centrality is more important in survive, eigenvector centrality and previous split are important for merge, previous merge for split and finally previous split, cohesion and change in cohesion can be important for dissolve. Finally, leader eigenvector centrality, change in size ratio and change in density are important for the survive event. Regarding merge leader ratio, regarding split leader closeness centrality and delta leader ratio are important. Moreover, as for dissolve, previous merge is important. If we take a look at other algorithms and datasets, s/he can observe a different pattern for significant features. This may be because of the inherent properties of the algorithms which cause different levels of importance of features and events. For instance, with SLPA and survive event in Facebook, we can observe the change in clustering coefficient as an important feature, but for AFOCS and Facebook, the previous status of the community is important. One challenging point that comes to mind is whether for each dataset, each separate event and each algorithm, we need to employ the methodology mentioned above and look at the result? The answer to this question can be manifold. Yes, we need to do that because correlating the dynamics of the algorithm with the property of the event is not easy and thus this may depend on the context of the social network and the properties of the algorithm. From another aspect, communities need to be defined explicitly, and we may need to consider context and content of communities. Adding contextual and content-based features may help to figure out more stable and consistent community properties; which we put it for our future work. The third aspect regarding the question mentioned above is the application of other statistical prediction methods. Perhaps using logistic regression and a huge number of features may not lead into variant results.

Table 5.3: List of real-world networks that are used to extract some properties from overlapping community detection algorithms. It represents the number of nodes, edges, and type of the networks.

<table>
<thead>
<tr>
<th>Graph</th>
<th>PolBlogs</th>
<th>P2P-Gnutella</th>
<th>General Relativity</th>
<th>NetScience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>1224</td>
<td>6301</td>
<td>5242</td>
<td>1461</td>
</tr>
<tr>
<td>Edges</td>
<td>19022</td>
<td>20777</td>
<td>28968</td>
<td>2742</td>
</tr>
<tr>
<td>Type</td>
<td>Internet topology</td>
<td>Technical</td>
<td>Collaboration</td>
<td>Collaboration</td>
</tr>
</tbody>
</table>

Community Evolution Prediction Accuracies

To observe how different feature sets behave in community evolution prediction problem, we applied different categories of features including structural, temporal and selective features for each dataset and each algorithm. Prediction accuracy results of different events are given in Table 5.4. By structural features, it is meant to apply node and community level features. By temporal, we mean to employ temporal node and community level features. Selective features are the best from temporal and structural feature categories that are selected and applied to give the prediction accuracy results. The first interesting observation is that selective features give better prediction accuracies in
almost all of the cases. There is yet an exception for DMID-Enron and the survive event. There may exist other exceptions, but in the majority of the cases, selective features yield the best performance. If we consider the selective features for the algorithms, datasets and the events, the comparison may become easier. In Facebook dataset and the survive event, AFOCS generates the best prediction accuracy of 82.35 which is higher than 75 and 78.57 of SLPA and DMID, respectively. Regarding 
\textit{dissolve} and Facebook, DMID gives the best result with the prediction accuracy of 78.57. This result can be due to larger communities identified by DMID. Finally, regarding \textit{split}, the highest prediction accuracy (77.94) belongs to SLPA. If we take a look at Enron dataset, DMID gives the best prediction accuracy result (88.71) that is better than SLPA (76.92) and AFOCS (82.14) for the \textit{survive} event. Regarding \textit{dissolve}, SLPA generates the best prediction accuracy of 93.75. Regarding \textit{merge} and \textit{split}, we can figure out that DMID wins with 95.59 and 88.71 on Enron dataset, respectively. As for the DBLP dataset and the \textit{survive} event again DMID wins with the value of 82.14. Results indicate that, regardless of the dataset, selective features based on DMID algorithm achieves the highest prediction accuracy for the \textit{survive} event. Regarding \textit{dissolve} and DBLP, we can observe the DMID superiority in the prediction task. Although Enron dataset indicated SLPA as the best candidate for \textit{dissolve}, DMID gives better results for Facebook and DBLP. Regarding \textit{merge} event, DMID obtains the best predictive value of 66.76 which is higher than 64.78 (SLPA) and 63.62 (AFOCS). Except for the Enron dataset, DMID also leads to the best prediction accuracy results for the \textit{merge} event. Finally, as for \textit{split}, DMID again takes the lead and wins by obtaining the value of 74.29 over the SLPA (65.56) and AFOCS (67.78). Except for the Facebook dataset, DMID also gets the highest prediction accuracy for the \textit{split} event.
5.2. COMMUNITY EVOLUTION PREDICTION

Figure 5.1: Comparison of important features in Facebook, Enron and DBLP datasets through SLPA, DMID and AFOCS algorithms.
Table 5.4: Prediction accuracy results of SLPA, AFOCS and DMID on three datasets including Facebook, Enron and DBLP. These prediction accuracies are shown for four events comprising *survive*, *dissolve*, *merge* and *split*.

<table>
<thead>
<tr>
<th></th>
<th>Survive</th>
<th>Dissolve</th>
<th>Merge</th>
<th>Split</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SLPA-Facebook-Structural</strong></td>
<td>58.33</td>
<td>72.99</td>
<td>66.67</td>
<td>66.18</td>
</tr>
<tr>
<td><strong>SLPA-Facebook-Temporal</strong></td>
<td>50</td>
<td>70.3</td>
<td>69.38</td>
<td>61.77</td>
</tr>
<tr>
<td><strong>SLPA-Facebook-Selective</strong></td>
<td>75</td>
<td>72.82</td>
<td>67.83</td>
<td>77.94</td>
</tr>
<tr>
<td><strong>DMID-Facebook-Structural</strong></td>
<td>58.93</td>
<td>67.7</td>
<td>67.35</td>
<td>70.62</td>
</tr>
<tr>
<td><strong>DMID-Facebook-Temporal</strong></td>
<td>51.79</td>
<td>66.4</td>
<td>65.82</td>
<td>53.75</td>
</tr>
<tr>
<td><strong>DMID-Facebook-Selective</strong></td>
<td>78.57</td>
<td>78.57</td>
<td>67.22</td>
<td>63.75</td>
</tr>
<tr>
<td><strong>AFOCS-Facebook-Structural</strong></td>
<td>75.49</td>
<td>61.52</td>
<td>64.5</td>
<td>60.09</td>
</tr>
<tr>
<td><strong>AFOCS-Facebook-Temporal</strong></td>
<td>80.39</td>
<td>62.27</td>
<td>63.54</td>
<td>63.78</td>
</tr>
<tr>
<td><strong>AFOCS-Facebook-Selective</strong></td>
<td>82.35</td>
<td>63.28</td>
<td>65.02</td>
<td>66.03</td>
</tr>
<tr>
<td><strong>SLPA-Enron-Structural</strong></td>
<td>71.15</td>
<td>43.75</td>
<td>65.15</td>
<td>50</td>
</tr>
<tr>
<td><strong>SLPA-Enron-Temporal</strong></td>
<td>65.39</td>
<td>68.75</td>
<td>56.06</td>
<td>50</td>
</tr>
<tr>
<td><strong>SLPA-Enron-Selective</strong></td>
<td>76.92</td>
<td>93.75</td>
<td>78.79</td>
<td>83.33</td>
</tr>
<tr>
<td><strong>DMID-Enron-Structural</strong></td>
<td>91.94</td>
<td>79.55</td>
<td>89.71</td>
<td>91.93</td>
</tr>
<tr>
<td><strong>DMID-Enron-Temporal</strong></td>
<td>88.71</td>
<td>72.73</td>
<td>85.29</td>
<td>88.71</td>
</tr>
<tr>
<td><strong>DMID-Enron-Selective</strong></td>
<td>88.71</td>
<td>88.64</td>
<td>95.59</td>
<td>88.71</td>
</tr>
<tr>
<td><strong>AFOCS-Enron-Structural</strong></td>
<td>77.38</td>
<td>62.05</td>
<td>56.14</td>
<td>62.5</td>
</tr>
<tr>
<td><strong>AFOCS-Enron-Temporal</strong></td>
<td>77.38</td>
<td>56.25</td>
<td>70.17</td>
<td>64.58</td>
</tr>
<tr>
<td><strong>AFOCS-Enron-Selective</strong></td>
<td>82.14</td>
<td>79.17</td>
<td>74.56</td>
<td>81.25</td>
</tr>
<tr>
<td><strong>SLPA-DBLP-Structural</strong></td>
<td>53.92</td>
<td>53.92</td>
<td>58.98</td>
<td>62.78</td>
</tr>
<tr>
<td><strong>SLPA-DBLP-Temporal</strong></td>
<td>60.78</td>
<td>61.77</td>
<td>62.85</td>
<td>63.61</td>
</tr>
<tr>
<td><strong>SLPA-DBLP-Selective</strong></td>
<td>67.16</td>
<td>67.2</td>
<td>64.78</td>
<td>65.56</td>
</tr>
<tr>
<td><strong>DMID-DBLP-Structural</strong></td>
<td>62.5</td>
<td>62.23</td>
<td>63.35</td>
<td>65.71</td>
</tr>
<tr>
<td><strong>DMID-DBLP-Temporal</strong></td>
<td>62.5</td>
<td>65.69</td>
<td>64.49</td>
<td>67.86</td>
</tr>
<tr>
<td><strong>DMID-DBLP-Selective</strong></td>
<td>82.14</td>
<td>66.42</td>
<td>66.76</td>
<td>74.29</td>
</tr>
<tr>
<td><strong>AFOCS-DBLP-Structural</strong></td>
<td>54.62</td>
<td>55.07</td>
<td>58.38</td>
<td>62.03</td>
</tr>
<tr>
<td><strong>AFOCS-DBLP-Temporal</strong></td>
<td>55.65</td>
<td>56.09</td>
<td>60.88</td>
<td>64.04</td>
</tr>
<tr>
<td><strong>AFOCS-DBLP-Selective</strong></td>
<td>55.99</td>
<td>55.99</td>
<td>63.62</td>
<td>67.78</td>
</tr>
</tbody>
</table>
5.3 Community-Based Sign Prediction

In this section, we introduce the sign prediction models that we proposed with the contribution of overlapping community dimension and community-aware ranking approaches. List of symbols we shall use to describe the models in signed networks is denoted in Table 5.5. In this regard, we propose intra, overlapping and extra feature types to the case of sign prediction. We propose two novel ranking algorithms for signed social networks combined with overlapping community structures on classical ranking algorithms. Next, we introduce the evaluation protocol.

5.3.1 Proposed Intra, Overlapping and Extra Features

We introduce the proposed features for the prediction task to differentiate intra, overlapping and extra nodes. Afterwards, we reveal the importance of overlapping members in social networks.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_i^+$</td>
<td>Positive authority value of node $i$</td>
</tr>
<tr>
<td>$a_i^-$</td>
<td>Negative authority value of node $i$</td>
</tr>
<tr>
<td>$h_i^+$</td>
<td>Negative hub value of node $i$</td>
</tr>
<tr>
<td>$PR_i^+$</td>
<td>Positive PageRank value of node $i$</td>
</tr>
<tr>
<td>$PR_i^-$</td>
<td>Negative PageRank value of node $i$</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>Damping factor</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Coefficient which contributes to extra hub and authority nodes</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Coefficient which contributes to overlapping hub and authority nodes</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Coefficient which contributes to intra hub and authority nodes</td>
</tr>
<tr>
<td>$\text{indeg}_{\text{intra}}(u)$</td>
<td>Nodes that negatively point to node $u$ and are in the same community as node $u$ is</td>
</tr>
<tr>
<td>$\text{indeg}_{\text{intra}}^+(u)$</td>
<td>Nodes that positively point to node $u$ and are in the same community as node $u$ is</td>
</tr>
<tr>
<td>$\text{outdeg}_{\text{intra}}(u)$</td>
<td>Nodes that node $u$ negatively points to them and are in the same community as node $u$ is</td>
</tr>
<tr>
<td>$\text{outdeg}_{\text{intra}}^+(u)$</td>
<td>Nodes that node $u$ positively points to them and are in the same community as node $u$ is</td>
</tr>
<tr>
<td>$\text{indeg}_{\text{ovl}}(u)$</td>
<td>Nodes that negatively point to node $u$ and are overlapping</td>
</tr>
<tr>
<td>$\text{indeg}_{\text{ovl}}^+(u)$</td>
<td>Nodes that positively point to node $u$ and are overlapping</td>
</tr>
<tr>
<td>$\text{outdeg}_{\text{ovl}}(u)$</td>
<td>Nodes that node $u$ negatively point to them and are overlapping</td>
</tr>
<tr>
<td>$\text{outdeg}_{\text{ovl}}^+(u)$</td>
<td>Nodes that node $u$ positively point to them and are overlapping</td>
</tr>
<tr>
<td>$\text{indeg}_{\text{extra}}(u)$</td>
<td>Nodes that negatively point to node $u$ and are in different community that node $u$ is</td>
</tr>
<tr>
<td>$\text{indeg}_{\text{extra}}^+(u)$</td>
<td>Nodes that positively point to node $u$ and are in different community that node $u$ is</td>
</tr>
<tr>
<td>$\text{outdeg}_{\text{extra}}(u)$</td>
<td>Nodes that node $u$ negatively points to them and are in different community that node $u$ is</td>
</tr>
<tr>
<td>$\text{outdeg}_{\text{extra}}^+(u)$</td>
<td>Nodes that node $u$ positively points to them and are in different community that node $u$ is</td>
</tr>
</tbody>
</table>

Prediction approaches have applied degree features to prediction tasks such as link or sign prediction. Leskovec et al. employed simple degree features such as $\text{indeg}^+$, $\text{indeg}^-$, $\text{outdeg}^+$, $\text{outdeg}^-$ and the number of common neighbors between trustor and trustee as for the features [LHK110]. Moreover, status and balancing theory in signed networks have been applied to the case of sign prediction. Other proximity measures such as Jaccard index and embeddedness are also applied to predict proximity of users in social networks. Similarly, for the features, we employ simple in-degree and out-degree features which correspond to prestige and opinions of nodes. However, we delicately change them to examine the effect of overlapping, intra, and extra nodes. Table 5.6 shows three sets of features applied for training and test purposes.
Overlapping Community Analysis and Prediction Models

Table 5.6: Three sets of features including intra, overlapping and extra features which have been used for the classifier. Here \( u \) and \( v \) respectively refers to trustor and trustee. All set is not listed and it comprises all the 12 features.

<table>
<thead>
<tr>
<th>Intra</th>
<th>Ovl</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{indeg}_{\text{intra}}(u) )</td>
<td>( \text{indeg}_{\text{Ovl}}(u) )</td>
<td>( \text{indeg}_{\text{extra}}(u) )</td>
</tr>
<tr>
<td>( \text{outdeg}_{\text{intra}}(u) )</td>
<td>( \text{outdeg}_{\text{Ovl}}(u) )</td>
<td>( \text{outdeg}_{\text{extra}}(u) )</td>
</tr>
<tr>
<td>( \text{indeg}_{\text{intra}}(v) )</td>
<td>( \text{indeg}_{\text{Ovl}}(v) )</td>
<td>( \text{indeg}_{\text{extra}}(v) )</td>
</tr>
<tr>
<td>( \text{outdeg}_{\text{intra}}(v) )</td>
<td>( \text{outdeg}_{\text{Ovl}}(v) )</td>
<td>( \text{outdeg}_{\text{extra}}(v) )</td>
</tr>
</tbody>
</table>

To compute in-degree features’ type for a node \( i \) we use the following formula:

\[
\text{indeg}_{\text{type}}(i) = \frac{|\text{indeg}_{\text{type}}^+(i)| - |\text{indeg}_{\text{type}}^-(i)|}{|\text{indeg}_{\text{type}}^+(i)| + |\text{indeg}_{\text{type}}^-(i)|},
\]

(5.1)

where \( \text{type} \in \{\text{Intra}, \text{Extra}, \text{Ovl}\} \) and \( i \) can be either \( u \) or \( v \). Respectively \( \text{outdeg}_{\text{type}}(i) \) features’ type can be computed as follows:

\[
\text{outdeg}_{\text{type}}(i) = \frac{|\text{outdeg}_{\text{type}}^+(i)| - |\text{outdeg}_{\text{type}}^-(i)|}{|\text{outdeg}_{\text{type}}^+(i)| + |\text{outdeg}_{\text{type}}^-(i)|},
\]

(5.2)

Therefore, for each edge corresponding to a trustor and trustee, we can consider these three sets of features. In the \( \text{Intra} \) type, nodes which are in the same community as trustor and trustee, are considered, and in-degree and out-degree features of this type are computed based on these nodes. Similarly, overlapping type comprises nodes which are overlapping among communities and are neighbors to trustor and trustee. Finally, \( \text{Extra} \) features’ type considers nodes which are in different communities than trustor and trustee but are the neighbors to them. An overview of these features is given in Table 5.6.

5.3.2 Proposed Overlapping Community-Aware Ranking Algorithms

To employ other community-aware features, we adapt classical ranking algorithms to the case of signed social networks. In this regard, we base our proposed ranking algorithms on HITS and PageRank and adopt them to overlapping community-aware versions. In the following, we introduce these approaches.

Overlapping Community-Aware HITS

HITS algorithm works based on two primary vectors named hubs and authorities (see chapter 2). First, we initialize these two vectors with some random values; then they are updated until convergence. However, they only take into account incoming and outgoing connections and do not consider overlapping community structures. To consider overlapping community structures, we propose an extended version of the HITS algorithm. We not only differentiate between intra and
5.3. COMMUNITY-BASED SIGN PREDICTION

Figure 5.2: Prediction accuracy, RMSE, and MAE results on the Wiki-Elec dataset. Intra, Overlapping (Ovl) and all of them together (all) are used for the sign prediction task. We can observe that in both balanced and imbalanced cases, Ovl feature set has better performance in comparison to Intra and Extra. This holds for all the metrics including prediction accuracy, RMSE, and MAE.

extra community links but also we take into account overlapping community structures. As both positive and negative connections exist, we consider four types of vectors as follows:

\[
\begin{align*}
    a^+_{i} &= \alpha \times \sum_{j \in \text{indeg}_{\text{intra}}^+(i)} h^+_j + \beta \times \sum_{j \in \text{indeg}_{\text{ovl}}^+(i)} h^+_j + \gamma \times \sum_{j \in \text{indeg}_{\text{extra}}^+(i)} h^+_j \\
    a^-_{i} &= \alpha \times \sum_{j \in \text{indeg}_{\text{intra}}^-(i)} h^-_j + \beta \times \sum_{j \in \text{indeg}_{\text{ovl}}^-(i)} h^-_j + \gamma \times \sum_{j \in \text{indeg}_{\text{extra}}^-(i)} h^-_j \\
    h^+_{i} &= \alpha \times \sum_{j \in \text{outdeg}_{\text{intra}}^+(i)} a^+_j + \beta \times \sum_{j \in \text{outdeg}_{\text{ovl}}^+(i)} a^+_j + \gamma \times \sum_{j \in \text{outdeg}_{\text{extra}}^+(i)} a^+_j \\
    h^-_{i} &= \alpha \times \sum_{j \in \text{outdeg}_{\text{intra}}^-(i)} a^-_j + \beta \times \sum_{j \in \text{outdeg}_{\text{ovl}}^-(i)} a^-_j + \gamma \times \sum_{j \in \text{outdeg}_{\text{extra}}^-(i)} a^-_j
\end{align*}
\]

(5.3)

which \(a^+_i\) and \(a^-_i\) are respectively positive and negative authorities of node \(i\), \(h^+_i\) and \(h^-_i\) are positive and negative hubs of node \(i\). Original HITS algorithm considers random values for hubs and authorities initialization. It is proved that the algorithm converges after a sufficient number
of iterations. We initialize hubs and authorities with \( \frac{1}{|N|} \) and run the updating rules for a sufficient number of iterations for OC-HITS. Although we were not able to provide convergence proof for our algorithm, hubs and authorities are updated sufficiently. Finally, \( \alpha \), \( \beta \) and \( \gamma \) respectively contribute to intra, overlapping and extra hub and authority nodes.

**Overlapping Community-Aware PageRank**

In this section, we extend classical PageRank algorithm (see chapter 2) to also consider overlapping members in social networks. For each node, we consider both positive and negative PageRank values. We can describe positive and negative overlapping community-based updating rules as follows:

\[
PR^+_i = \zeta \times (\alpha \times \sum_{j \in \text{indeg}^\text{intra}_i} \frac{PR^+_j}{|\text{Outdeg}^\text{intra}_j|} + \beta \times \sum_{j \in \text{indeg}^\text{ovl}_i} \frac{PR^+_j}{|\text{Outdeg}^\text{ovl}_j|} + \\
\gamma \times \sum_{j \in \text{indeg}^\text{extra}_i} \frac{PR^+_j}{|\text{Outdeg}^\text{extra}_j|}) + (1 - \zeta) \times \frac{1}{N},
\]

\[
PR^-_i = \zeta \times (\alpha \times \sum_{j \in \text{indeg}^\text{intra}_i} \frac{PR^-_j}{|\text{Outdeg}^\text{intra}_j|} + \beta \times \sum_{j \in \text{indeg}^\text{ovl}_i} \frac{PR^-_j}{|\text{Outdeg}^\text{ovl}_j|} + \\
\gamma \times \sum_{j \in \text{indeg}^\text{extra}_i} \frac{PR^-_j}{|\text{Outdeg}^\text{extra}_j|}) + (1 - \zeta) \times \frac{1}{N},
\]

where \( PR^+_i \) and \( PR^-_i \) are positive and negative PageRank values of node \( i \), \( \zeta \) is a damping factor in the original PageRank which is set to 0.85 and \( \alpha \), \( \beta \) and \( \gamma \) coefficients weigh to intra, overlapping and extra nodes linking to node \( i \).

### 5.3.3 Evaluation Results

As for evaluation, we set up two sign prediction tasks. The first one is based on the simple degree features introduced in Table 5.6 and the second prediction model is built based on the community-aware ranking methods.

**Simple Degree Sign Prediction Results**

After identifying communities and computing intra, extra and overlapping (ovl) features’ types, they are applied to the case of sign prediction problem. To this end, we use the WEKA software and recruit bagging, BayesNaive, BayesNet, J48, logistic regression and decision table as classifiers (see chapter 2). Ten fold cross validation is used for all the algorithms. Moreover, to evaluate the results of the models, we use prediction accuracy, MAE, RMSE [TSKu05], which are introduced in the chapter 2. Because the number of positive edges are more than the number of negative
5.3. COMMUNITY-BASED SIGN PREDICTION

![Charts showing prediction accuracy, RMSE, and MAE results on Wiki-RfA dataset]

Figure 5.3: Prediction accuracy, RMSE, and MAE result on Wiki-RfA dataset. Intra, Overlapping (Ovl) and all of them together (all) are used for the sign prediction task. We can observe that the performance of Ovl feature set is higher than Intra and Extra feature sets. Only for the balanced case, Intra is a little bit better than Ovl. However, both Intra and Ovl feature sets performed better than Extra feature set. This also holds for all the metrics including prediction accuracy, RMSE, and MAE.

ones, we consider both balanced and imbalanced datasets. On the one hand, imbalanced refers to the original dataset, which the number of positive and negative edges are unequal. On the other hand, the balanced dataset is a subset of the imbalanced dataset, which the number of positive and negative links are randomly equalized. To prevent errors, the results for the balanced dataset are performed 10 times with random omitting of positive edges. Then, the results are averaged over these 10 realizations. Before train and test phase, we preprocessed the data and omitted those edges which their feature sets are completely empty. As for the dataset, we employ two available datasets including WikiElec (introduced in the chapter 4) and Wiki-RfA. Wiki-RfA is an updated version of the Wikipedia adminship election data. This dataset is enriched with textual information and the result of the voting for the adminship election. Similarly, users vote positively and negatively towards the candidates. We can observe more information about these two datasets in Table 5.7.

Figures 5.2 and 5.3 indicate results of the logistic regression classier on Wiki-Elec and Wiki-RfA, respectively. As for Wiki-Elec in imbalanced case, prediction accuracy values are 82.61, 83.53, 82.61 and 83.9 respectively for Intra, Ovl, Extra and All sets of features. In this case, Ovl feature set has roughly one % better performance in comparison to Intra and Extra features. We could also observe the same pattern for MAE and RMSE. RMSE error for Ovl feature type set is
Table 5.7: Number of nodes, edges, positive and negative links related to Wiki-RfA and Wiki-Elec.

<table>
<thead>
<tr>
<th></th>
<th>#Nodes</th>
<th>#Edges</th>
<th>#PositiveEdges</th>
<th>#NegativeEdges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki-RfA</td>
<td>10835</td>
<td>159388</td>
<td>144451</td>
<td>41176</td>
</tr>
<tr>
<td>Wiki-Elec</td>
<td>8298</td>
<td>103186</td>
<td>83962</td>
<td>23118</td>
</tr>
</tbody>
</table>

0.35, which is lower than Intra (0.37) and Extra (0.37) feature sets. Furthermore, MAE error for Ovl feature set is 0.24 which is lower than Intra (0.27) and Extra (0.27) feature sets. As for Wiki-Elec in the balanced case, the difference is more tangible regarding these measures. As we can observe, prediction accuracy for Ovl features is 69.23 which is approximately 10% higher than Intra (58.93) and Extra (58.29) features. This also holds for MAE and RMSE. MAE value for Ovl (0.39) is lower than Intra (0.48) and Extra (0.48). Finally, Ovl (0.44) surpasses intra (0.49) and Extra (0.48) in terms of RMSE.

Similarly, we can observe approximately the same pattern for the Wiki-RfA. As Figure 5.3 indicates for the imbalanced case, prediction accuracy of Ovl (83.85) supersedes by 0.5% over the Intra (83.38) and Extra (83.38) feature sets. Interestingly, in balanced dataset the pattern is a little bit different from what we observed in Wiki-Elec and the imbalanced case of Wiki-RfA. Here, the Intra feature set obtains prediction accuracy of 71.89 which is higher than the ovl (68.66) and Extra (58.71) feature sets. However, the ovl feature is about ten% higher than the extra feature set, which again approves the significance of overlapping set features. We can observe the same pattern for MAE and RMSE measures. In the imbalanced case, MAE value for the Ovl feature set (0.24) is equal to Intra (0.24) and lower than Extra (0.48). However, this is not true for the balanced case. Intra (0.36) is lower than both Ovl (0.40) and Extra (0.48) in terms of MAE. The same relation also holds for RMSE. In other words, Ovl (0.35) is equal to Intra (0.35) and lower than Extra (0.37) in the imbalanced case. However, intra (0.43) beats both Ovl (0.45) and Extra (0.49) in the balanced case.

Finally, we compared the performance of different classifiers on the feature sets of Intra, Extra, Ovl and All for imbalanced data sets. Results of the prediction accuracy comparison on Wiki-Elec and Wiki-RfA are respectively shown in Tables 5.8 and 5.9. In Wiki-Elec, we can observe for all the classifiers that the Ovl feature set is higher than the Intra and Extra feature sets except for BayesNaive classifier. Both Intra (82.61) and Extra (82.61) feature sets’ prediction accuracies are higher than the Ovl set (80.37). Surprisingly, the All set has a lower performance. Regarding the performance of the classifier, All feature set shows that Bagging (84.80), J48 (84.53), DecisionTable (84.16) and logistic regression (83.90) have better performance in comparison to the other classifiers. Although, we applied logistic regression in experiments of Figure 5.3 and 5.2, performance of Bagging, J48 and DecisionTable are better.

Likewise, Table 5.9 indicates a similar pattern. For all the classifiers, the prediction accuracy of the Ovl feature is higher than the Intra and Extra feature sets except for Bagging and BayesNaive. Furthermore, we can also conclude that performance of Bagging (85.69), J48 (85.65), Logistic (85.43) and Decision Table (85.25) are higher than BayesNet (84.93) and BayesNaive (82.62) for the All feature set.
5.3. COMMUNITY-BASED SIGN PREDICTION

Table 5.8: Prediction accuracy of different classifiers on Wiki-Elec dataset. Each row of the table indicates the set of features which are used for training and test phase.

<table>
<thead>
<tr>
<th></th>
<th>Bagging</th>
<th>Logistic</th>
<th>BayesNet</th>
<th>BayesNaive</th>
<th>J48</th>
<th>DecisionTable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intra</td>
<td>82.61</td>
<td>82.61</td>
<td>82.61</td>
<td>82.61</td>
<td>82.61</td>
<td>82.61</td>
</tr>
<tr>
<td>Ovl</td>
<td>83.82</td>
<td>83.53</td>
<td>83.32</td>
<td>80.37</td>
<td>84.05</td>
<td>84.02</td>
</tr>
<tr>
<td>Extra</td>
<td>82.74</td>
<td>82.61</td>
<td>82.64</td>
<td>82.61</td>
<td>82.61</td>
<td>82.64</td>
</tr>
<tr>
<td>All</td>
<td>84.80</td>
<td>83.90</td>
<td>83.20</td>
<td>80.94</td>
<td>84.53</td>
<td>84.16</td>
</tr>
</tbody>
</table>

Table 5.9: Prediction accuracy of different classifiers on Wiki-RfA dataset. Each row of the table indicates the set of features which are used for training and test phase.

<table>
<thead>
<tr>
<th></th>
<th>Bagging</th>
<th>Logistic</th>
<th>BayesNet</th>
<th>BayesNaive</th>
<th>J48</th>
<th>DecisionTable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intra</td>
<td>83.68</td>
<td>83.39</td>
<td>83.44</td>
<td>73.12</td>
<td>83.58</td>
<td>83.35</td>
</tr>
<tr>
<td>Ovl</td>
<td>83.67</td>
<td>83.85</td>
<td>83.99</td>
<td>82.92</td>
<td>83.88</td>
<td>83.93</td>
</tr>
<tr>
<td>Extra</td>
<td>83.51</td>
<td>83.39</td>
<td>83.4</td>
<td>83.39</td>
<td>83.39</td>
<td>83.42</td>
</tr>
<tr>
<td>All</td>
<td>85.69</td>
<td>85.43</td>
<td>84.93</td>
<td>82.62</td>
<td>85.65</td>
<td>85.25</td>
</tr>
</tbody>
</table>

Overlapping Community-Aware-HITS Sign Prediction

In this section, we apply overlapping community-aware-HITS (OC-HITS) to the case of sign prediction and evaluate the results. In this regard, we employ Bagging classifier to evaluate importance of intra, overlapping and extra nodes. For this purpose, 16 different combination of $\alpha$ (intra), $\beta$ (overlapping) and $\gamma$ (extra) values in Table 5.10 and 5.11 are applied to the case of sign prediction. Table 5.10 indicates prediction accuracies, MAE and RMSE of prediction for different coefficient values in Wiki-RfA dataset. The lowest MAE (0.158), RMSE (0.287) and highest prediction accuracy (88.280) belong to $\alpha = 0.3$, $\beta = 0.4$ and $\gamma = 0.3$, which hub and authority values of overlapping nodes are taken into account more than hubs and authorities of intra and extra nodes. In other cases like combination of 0.2, 0.8 and 0, we can observe that ($\alpha = 0.8$, $\beta = 0.2$, $\gamma = 0$) leads into lowest MAE (0.16), RMSE (0.288) and highest prediction accuracy (88.199). Regarding combination of 0, 0.333 and 0.666, lowest MAE (0.159), RMSE (0.288) and the highest prediction accuracy (88.154) relate to ($\alpha = 0.333$, $\beta = 0.666$, $\gamma = 0$). Regarding 0.3, 0.4 and 0.3, the combination with higher coefficient for overlapping members wins ($\alpha = 0.3$, $\beta = 0.4$, $\gamma = 0.3$).

Overlapping Community-Aware-PageRank Sign Prediction

Here, we applied overlapping community-aware-PageRank (OC-PageRank) as for the sign prediction problem. Table 5.11 reveals MAE, RMSE and prediction accuracy of Wiki-RfA dataset using OC-PageRank. Similarly, we can observe that lowest MAEs, RMSEs and highest accuracies belong to $\alpha = 0.3$, $\beta = 0.3$, $\gamma = 0.4$ and $\alpha = 0.4$, $\beta = 0.3$, $\gamma = 0.3$ with respectively MAE
Table 5.10: MAE, RMSE and prediction accuracy values for different combination values of $\alpha$ (intra), $\beta$ (Ovl) and $\gamma$ (extra) coefficients in community-based HITS updating rules for Wiki-RfA dataset.

<table>
<thead>
<tr>
<th>$\alpha$ (intra)</th>
<th>$\beta$ (Ovl)</th>
<th>$\gamma$ (extra)</th>
<th>MAE</th>
<th>RMSE</th>
<th>Prediction accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.200</td>
<td>0.800</td>
<td>0.203</td>
<td>0.323</td>
<td>85.901</td>
</tr>
<tr>
<td>0.000</td>
<td>0.800</td>
<td>0.200</td>
<td>0.192</td>
<td>0.315</td>
<td>86.437</td>
</tr>
<tr>
<td>0.000</td>
<td>0.333</td>
<td>0.666</td>
<td>0.199</td>
<td>0.321</td>
<td>86.117</td>
</tr>
<tr>
<td>0.000</td>
<td>0.666</td>
<td>0.333</td>
<td>0.194</td>
<td>0.316</td>
<td>86.316</td>
</tr>
<tr>
<td>0.300</td>
<td>0.300</td>
<td>0.400</td>
<td>0.159</td>
<td>0.288</td>
<td>88.198</td>
</tr>
<tr>
<td>0.300</td>
<td>0.400</td>
<td>0.300</td>
<td>0.158</td>
<td>0.287</td>
<td>88.280</td>
</tr>
<tr>
<td>0.400</td>
<td>0.300</td>
<td>0.300</td>
<td>0.160</td>
<td>0.288</td>
<td>88.217</td>
</tr>
<tr>
<td>0.800</td>
<td>0.200</td>
<td>0.000</td>
<td>0.160</td>
<td>0.288</td>
<td>88.199</td>
</tr>
<tr>
<td>0.200</td>
<td>0.800</td>
<td>0.000</td>
<td>0.208</td>
<td>0.327</td>
<td>84.721</td>
</tr>
<tr>
<td>0.333</td>
<td>0.666</td>
<td>0.000</td>
<td>0.159</td>
<td>0.288</td>
<td>88.154</td>
</tr>
<tr>
<td>0.333</td>
<td>0.333</td>
<td>0.333</td>
<td>0.160</td>
<td>0.288</td>
<td>88.225</td>
</tr>
<tr>
<td>0.666</td>
<td>0.333</td>
<td>0.000</td>
<td>0.209</td>
<td>0.327</td>
<td>84.777</td>
</tr>
<tr>
<td>0.333</td>
<td>0.000</td>
<td>0.666</td>
<td>0.210</td>
<td>0.328</td>
<td>84.668</td>
</tr>
<tr>
<td>0.666</td>
<td>0.000</td>
<td>0.333</td>
<td>0.209</td>
<td>0.327</td>
<td>84.777</td>
</tr>
<tr>
<td>0.200</td>
<td>0.000</td>
<td>0.800</td>
<td>0.210</td>
<td>0.327</td>
<td>84.756</td>
</tr>
<tr>
<td>0.800</td>
<td>0.000</td>
<td>0.200</td>
<td>0.208</td>
<td>0.327</td>
<td>84.721</td>
</tr>
</tbody>
</table>

Values 0.225 and 0.226, RMSE values 0.339 and 0.339 and prediction accuracies of 83.300 and 83.298. For other combinations, high coefficient overlapping PageRank values are also competitive in Wiki-RfA. Other cases with high overlapping weight also possess competitive error values and prediction accuracies. Although prediction accuracy for OC-HITS is higher than OC-PageRank, the significance of overlapping community structures remain approximately the same. Moreover, comparing prediction accuracy results of simple in and out-degree, OC-HITS hub and authority and PageRank features, we observe that OC-HITS outperforms the others in Wiki-RfA.
5.4. CONCLUSION

Table 5.11: MAE, RMSE and prediction accuracy values for different combination values of $\alpha$ (intra), $\beta$ (Ovl) and $\gamma$ (extra) coefficients in community-based PageRank updating rules for Wiki-RfA dataset.

<table>
<thead>
<tr>
<th>$\alpha$ (intra)</th>
<th>$\beta$ (Ovl)</th>
<th>$\gamma$ (Extra)</th>
<th>MAE</th>
<th>RMSE</th>
<th>Prediction accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.200</td>
<td>0.800</td>
<td>0.309</td>
<td>0.393</td>
<td>80.327</td>
</tr>
<tr>
<td>0.000</td>
<td>0.800</td>
<td>0.200</td>
<td>0.309</td>
<td>0.393</td>
<td>80.331</td>
</tr>
<tr>
<td>0.200</td>
<td>0.800</td>
<td>0.000</td>
<td>0.230</td>
<td>0.342</td>
<td>82.931</td>
</tr>
<tr>
<td>0.800</td>
<td>0.200</td>
<td>0.000</td>
<td>0.230</td>
<td>0.342</td>
<td>82.921</td>
</tr>
<tr>
<td>0.200</td>
<td>0.000</td>
<td>0.800</td>
<td>0.231</td>
<td>0.344</td>
<td>82.933</td>
</tr>
<tr>
<td>0.800</td>
<td>0.000</td>
<td>0.200</td>
<td>0.232</td>
<td>0.344</td>
<td>82.966</td>
</tr>
<tr>
<td>0.333</td>
<td>0.666</td>
<td>0.000</td>
<td>0.230</td>
<td>0.342</td>
<td>82.938</td>
</tr>
<tr>
<td>0.333</td>
<td>0.333</td>
<td>0.333</td>
<td>0.226</td>
<td>0.340</td>
<td>83.237</td>
</tr>
<tr>
<td>0.333</td>
<td>0.000</td>
<td>0.666</td>
<td>0.231</td>
<td>0.344</td>
<td>82.950</td>
</tr>
<tr>
<td>0.666</td>
<td>0.000</td>
<td>0.333</td>
<td>0.232</td>
<td>0.344</td>
<td>82.906</td>
</tr>
<tr>
<td>0.000</td>
<td>0.333</td>
<td>0.666</td>
<td>0.309</td>
<td>0.393</td>
<td>80.326</td>
</tr>
<tr>
<td>0.000</td>
<td>0.666</td>
<td>0.333</td>
<td>0.310</td>
<td>0.393</td>
<td>80.283</td>
</tr>
<tr>
<td>0.300</td>
<td>0.300</td>
<td>0.400</td>
<td>0.225</td>
<td>0.339</td>
<td>83.300</td>
</tr>
<tr>
<td>0.300</td>
<td>0.400</td>
<td>0.300</td>
<td>0.227</td>
<td>0.340</td>
<td>83.211</td>
</tr>
<tr>
<td>0.400</td>
<td>0.300</td>
<td>0.300</td>
<td>0.226</td>
<td>0.339</td>
<td>83.298</td>
</tr>
<tr>
<td>0.666</td>
<td>0.333</td>
<td>0.000</td>
<td>0.315</td>
<td>0.397</td>
<td>80.324</td>
</tr>
</tbody>
</table>

5.4 Conclusion

In this chapter, we applied supervised learning algorithms to predict both the evolution of overlapping communities and mixing patterns. Regarding community evolution prediction, we identified the size of the communities as the most essential feature. Moreover, among different community detection algorithms, DMID achieved the highest accuracy values. As for various events, we could not identify stable properties.

Besides, we showed that overlapping nodes are reliable for building sign prediction models and they achieve competitive performance compared to other node types. This observation could be explained while overlapping nodes possess broader perspective compared to intra and extra nodes of a community. We only used structural properties to predict the sign of missing links. As such, using context-based personal features such as gender, age, preferences and activity-based properties such as sentiments may reveal further meaningful and informative results.
Overlapping Community Analysis and Prediction Models
Chapter 6

Applications of Overlapping Communities

6.1 Introduction

Advancement of computational technologies, the emergence of social networks and social software, have created an enormous amount of data that cause problems for users from different aspects. If you remember your experience of purchasing in an online Website such as Amazon, you figure out the massive number of options regarding a particular product. Suppose, as another example; you would like to watch a movie; however, you do not want to dedicate your time to a movie that you may not like. In other domains, such as learning environments, people search for experts related to their inquiries while experts help people to find their intended content. Considering these examples, we figure out that overload of information items and the complexity of the tasks have expanded the requirement for scaffolding software [CYCS14]. Literature has approached recommender systems as a way of supporting users. Related works are available proposing a variety of approaches to solve this problem considering user (expert) and item recommender systems (see chapter 3). However, we cannot detach social networks from community structures. Because dynamics of community structures provide temporal and collective information related to groups of people [LKFa07, BHKL06].

We address the third research question raised in chapter 1 in three sections.

The first section is related to recommender systems. Recommender systems are capable of facilitating choosing suitable items or finding the desired information. Researchers have proposed a different range of algorithms based on collaborative filtering leaning upon social properties and human characteristics. These approaches use time or community as contextual information; however, temporal community structures are not examined in the proposal of a recommender model. To this aim, we base our work on neighborhood and factor models and extend them to community-aware and temporal community-based recommender algorithms. Two models are also proposed to reduce the time complexity of these models by still keeping the compromise of precise recommendations.

In the second part of this chapter, we investigate expert identification task. Experts possess a higher level of information and more context-specific knowledge regarding a specific domain. Expertise
is a context-dependent concept, which a user might be expert and thus useful within a particular context but not in other areas. Expert identification tasks sort the users based on their knowledge and return the top ones, as such, we require considering the context of the domain when fabricating the network. In this section, community-aware ranking algorithms have investigated expert identification in question-answer forums. We extend classical HITS and PageRank algorithms (see chapter 2) and apply them to the case of identifying experts.

In the third section of this chapter, we study the ranking and the cooperation problems to find out patterns and correlations among the rank value of a person and its cooperativity. We investigate users’ cooperativity via connecting ranking algorithms with cooperation/defection challenge in complex networks. As such, prisoner’s dilemma as a game theoretic approach is employed to compute the amount of a node’s cooperation. Second, we know few about cooperativity of community structures detected by community detection algorithms. Thus, we apply prisoner’s dilemma on several network structures. As such, we consider different releases of OSS forums as well as various months of learning forums and apply game theory to compute cooperation in these networks.

In the following, we address each of these sections.
Table 6.1: An overview of the symbols used in the recommender models’ formulation.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>User</td>
</tr>
<tr>
<td>i,j</td>
<td>Item</td>
</tr>
<tr>
<td>t</td>
<td>Timestamp</td>
</tr>
<tr>
<td>t_{ui}</td>
<td>Time of rating given by user u to item i</td>
</tr>
<tr>
<td>C</td>
<td>Community</td>
</tr>
<tr>
<td>C_u</td>
<td>Set of communities that user u belongs to</td>
</tr>
<tr>
<td>n_{ui}C</td>
<td>Membership level of user u in community C</td>
</tr>
<tr>
<td>n_{i}C</td>
<td>Membership level of item i in community C</td>
</tr>
<tr>
<td>I_{C}</td>
<td>Set of items rated by user u</td>
</tr>
<tr>
<td>I_{C_u}</td>
<td>Set of items rated by community co-members of user u</td>
</tr>
<tr>
<td>r_{ui}</td>
<td>Rating given by user u to item i</td>
</tr>
<tr>
<td>r_{C,i}</td>
<td>Mean rating of users belonging to communities in C_u for item i</td>
</tr>
<tr>
<td>r_{t}</td>
<td>Estimated rating of user u to item i</td>
</tr>
<tr>
<td>µ</td>
<td>Mean rating over all users and items</td>
</tr>
<tr>
<td>b_{u}</td>
<td>Bias of user u</td>
</tr>
<tr>
<td>b_{i}</td>
<td>Bias of item i</td>
</tr>
<tr>
<td>b_{u,i}</td>
<td>Combined user and item bias of u and i</td>
</tr>
<tr>
<td>b_{u,t}</td>
<td>Bias of user u at time t</td>
</tr>
<tr>
<td>b_{u,Period(t)}</td>
<td>Recurring time-dependent user bias of u and time t</td>
</tr>
<tr>
<td>b_{u}(t)</td>
<td>Combined time-(in)dependent bias of user u</td>
</tr>
<tr>
<td>b_{i,Bin(t)}</td>
<td>Bias of item i at time bin of t</td>
</tr>
<tr>
<td>b_{i,Period(t)}</td>
<td>Recurring time-dependent bias of item i and time t</td>
</tr>
<tr>
<td>b_{i}(t)</td>
<td>Combined time-dependent and time-independent bias of item i</td>
</tr>
<tr>
<td>b_{C}</td>
<td>Bias of community C</td>
</tr>
<tr>
<td>b_{C_u}</td>
<td>Combined bias of communities in C_u</td>
</tr>
<tr>
<td>b_{C}</td>
<td>Combined bias of communities in C</td>
</tr>
<tr>
<td>b_{C,t}</td>
<td>Bias of (user) community C at time t</td>
</tr>
<tr>
<td>b_{C,Bin(t)}</td>
<td>Bias of (item) community C at time t</td>
</tr>
<tr>
<td>b_{C,u,i}</td>
<td>Combined user and item community bias for u and i</td>
</tr>
<tr>
<td>w_{ij}</td>
<td>Explicit rating feedback of item j to targeted item i</td>
</tr>
<tr>
<td>c_{ij}</td>
<td>Implicit rating feedback of item j to targeted item i</td>
</tr>
<tr>
<td>d_{ij}</td>
<td>Implicit community rating feedback of item j to targeted item i</td>
</tr>
<tr>
<td>y_{i}</td>
<td>Implicit user factor contribution of item j</td>
</tr>
<tr>
<td>z_{i}</td>
<td>Implicit user community factor contribution of item j</td>
</tr>
<tr>
<td>q_{i}</td>
<td>Item factor of i</td>
</tr>
<tr>
<td>p_{u}</td>
<td>User factor of u</td>
</tr>
<tr>
<td>p_{u}(t)</td>
<td>Time-dependent user factor of u</td>
</tr>
<tr>
<td>o_{C}</td>
<td>Community factor of C</td>
</tr>
<tr>
<td>o_{C,k}</td>
<td>k-th component of the community factor of C</td>
</tr>
<tr>
<td>o_{C,k,t}</td>
<td>k-th component of the community factor of C and time t</td>
</tr>
<tr>
<td>o_{C}(t)</td>
<td>Time-dependent community factor of C</td>
</tr>
<tr>
<td>o_{C,t}</td>
<td>Combined time-(in)dependent factors of communities in C_u for time t</td>
</tr>
<tr>
<td>o_{u}</td>
<td>Decay factor of user u</td>
</tr>
<tr>
<td>ϕ_{u}</td>
<td>Community-related decay factor of user u</td>
</tr>
<tr>
<td>α_{u}</td>
<td>Coefficient for linear drift in user bias of user u</td>
</tr>
<tr>
<td>α_{C}</td>
<td>Coefficient for linear drift in community bias of community C</td>
</tr>
<tr>
<td>α_{C,k}</td>
<td>Coefficient for linear drift in k-th component of the community factor C</td>
</tr>
<tr>
<td>dev_{u}(t)</td>
<td>Time distance between t and u’s mean rating time</td>
</tr>
<tr>
<td>dev_{C}(t)</td>
<td>Time distance between t and C’s mean rating time</td>
</tr>
<tr>
<td>Bin(t)</td>
<td>Time bin of time t</td>
</tr>
<tr>
<td>Period(t)</td>
<td>Recurring period of time t (e.g. day of week, season, etc.)</td>
</tr>
</tbody>
</table>
Applications of Overlapping Communities

6.2 Proposed Community-Aware Baseline Predictor for Item Recommender Algorithm

In the first section of this chapter, we propose an algorithm to improve precision and running time of previous recommender methods using temporal community structure dynamics. Table 6.1 shows the list of symbols we shall use for describing the recommender models. The primary job of a recommender algorithm is to estimate/predict a rating from a user to an item based on the observation of other ratings information. Often, recommender systems employ these algorithms to provide a top $k$ choice to be selected by users. The basic idea behind our approach is the works of [Kore08, Kore09] which are extended to consider temporal user and item community structures. Regarding the proposed model, we would like to proceed and answer a couple of research questions and concerns. First, we are interested to know what the effect of community structures on the rating estimation proposed by Koren is. In other words, how useful it is to combine the community structures with contextual information such as time and how they affect the accuracy of the recommendation. Second, we would like to compare the role of overlapping and non-overlapping community detection algorithms on item ranking and accuracy metrics. To this end, we employ Walktrap and DMID which are respectively disjoint and overlapping community detection algorithms (see chapter 3). As community structure adds extra dimensions to the model and the learning phase, we are also interested in observing how the time complexities can be improved through overlooking individual user and item biases. Thus, we also propose two fast models for community-aware and temporal community-aware versions.

Finally, as the proposed algorithm requires different parameters, i.e., graph construction approach, similarity metrics and community detection algorithms, it is essential to observe the effect of these parameters on the final evaluation results, and also to find the best parameter settings as for future runs. While the proposed algorithm requires different input parameters, we may need to explain them separately. These parameters include building the user-user and item-item graphs, the applied community detection algorithms and the extended baseline predictor that we will describe in the following subsections.

6.2.1 Obtaining User-User and Item-Item Graphs

For the user and item communities, we require building user-user and item-item graphs which we can apply a community detection algorithm. These networks can be constructed directly from the dataset if it contains explicit relation information for users or items. For instance, if the dataset contains user friendship information, then this can be used to build the user-user graph directly. And if the data do not contain such information then the graphs must be generated from the rating information. We assume that these graphs need to be undirected and may be weighted. In principle, the graphs also could be directed, e.g., considering trust levels between users to generate the user-user graph. In this case, user $u$ may trust user $v$ but $v$ may not trust $u$. For simplicity, we only consider undirected graphs. On the other hand, we do consider weighted graphs since we can generate graphs from implicit information.
6.2. PROPOSED COMMUNITY-AWARE BASELINE PREDICTOR FOR ITEM RECOMMENDER ALGORITHM

6.2.2 Build Graphs from Ratings

To apply community detection algorithms, we require building graphs from the implicit or explicit information available in the datasets. For instance, explicit information may refer to friendship networks already formed among users; however, implicit information can be inferred either from ratings or tags. Regarding implicit construction, if two users have rated the same item or the same user has rated two items then we may connect these users and items, respectively. As for simplicity, the constructed graph is supposed to be undirected but weighted. Weighted graphs can convey further information and thus reflect some similarities among users and items. To compute the similarities among users and items from the rating information, we employ Pearson Correlation, Cosine Similarity, and Jaccard Mean Squared Distance, which we defined in chapter 2. In this regard, Park et al. [PPLJ14] proposed a graph construction approach mainly suitable for information retrieval and recommender algorithms.

Improving the running time is one of the objectives of the proposed algorithm, and thus we require to employ an efficient graph construction method. As time complexity of community detection algorithms depends on the density of graphs and number of edges, k-NN graph construction algorithm reduces running times by only considering $k$ adjacent neighbors for each user or item. Using only a subset of the whole neighbors reduces time complexity of a search from $O(n^2)$ to $O(n)$. This algorithm reaches its best performance when it filters the nodes represented by sparse user/item ratings. In other words, user pairs and item pairs will be filtered when there is no common item or user with high weights among the pairs. We process these vectors by applying a TF-IDF approach. The idea behind it is to produce weights that are commensurate with reverse frequency. Especially, in search engines, the terms that happen more frequent will be ranked lower using the TF-IDF scheme. As for the recommender task, the TF-IDF scheme is adapted to consider rating data [PPLJ14].

The extended TF-IDF scheme regarding item $i$ in the sparse vector containing user $u$ can be computed as follows:

$$\text{tf-idf}_{\text{user}}(i, u) = \left( 0.5 + \frac{0.5 \cdot r_{ui}}{\max_{j \in I_u} \{r_{uj}\}} \right) \cdot \left( \log \frac{|U|}{|U_i|} \right), \quad (6.1)$$

and the TF-IDF regarding user $u$ in the sparse vector containing item $i$ can be computed as:

$$\text{tf-idf}_{\text{item}}(u, i) = \left( 0.5 + \frac{0.5 \cdot r_{ui}}{\max_{u \in U_i} \{r_{ui}\}} \right) \cdot \left( \log \frac{|I|}{|I_u|} \right). \quad (6.2)$$

After the filtering phase, we calculate the similarities among the remaining users and items.

6.2.3 Build Graphs from Tags

Another way of graph construction among users and items as for the recommender task is to employ the tag information. Users usually assign tag information to items, for instance in MovieLens dataset, these tags can be genre or the director of movies. We may connect users who have used the
same tag for any item or items that have received the same tag from any user in the system. This graph is supposed to be unweighted; however, we may consider weighted schemes by employing frequency of common tags between two users/items or applying frequency schemes.

### 6.2.4 Identifying Community Detection Algorithm

We need to apply community detection algorithms on user-user and item-item graphs. We can denote the user $U$’s and item $I$’s community via $C_U$ and $C_I$, which for an OCD algorithm $|C_U|>1$ and $|C_I|>1$. Referring to chapter 3, there are many OCD algorithms available; however, we may need to consider a couple of criteria for the recommender task. Scalability of the algorithm, and generating deterministic results should be the properties of the candidate algorithms. As in the $K$-NN graph construction phase weighted networks appear, we also need to select an algorithm that is suitable for these types of networks. We did some investigation on the current OCD algorithms regarding time complexity and number of found communities, and we could select two out of them to apply them as parameters for the recommender task.

Some of the algorithms cannot accompany the large data volume such as MONC, Link Communities, SSK. On the other hand, some others detect very large (CLiZZ) or very low (SLPA, LPA) number of communities. Algorithms such as SLPA and Infomap scale well; however, they may not generate deterministic results. Finally, we preferred to select DMID as it generates an average number of detected communities in comparison to the size of the network and also scales quite linearly. Moreover, it can handle (un)weighted, directed networks. To compare the results with a disjoint community detection algorithm, Walktrap is also selected which is quite fast and generate an average number of communities in the scale of the network. By considering Walktrap and DMID, we consider both disjoint and overlapping community detection algorithms. For constructed user and item graphs based on the tags, we employ SLPA in some parts of the runs where possible.

### 6.2.5 Estimate Ratings

Many recommender algorithms have been proposed considering the precision of recommendation either in rating level or the returned top $k$ item list. In other words, two fundamental problems in recommender systems are estimating rating of a user on a particular item or suggesting the top $k$ suitable items to a specific user. In this recommender framework, we propose an algorithm which exploits the community structures to extend the basic model proposed by Koren. He primarily employed neighborhood and factor models to build a precise recommender algorithm and he extended the proposed model to consider the temporal effects in the baseline predictor. By using stochastic gradient descent, they learned the parameters of the model and used it to estimate the ratings for users on the item sets. In our approach, we employ a similar perspective; however, we require to build the graphs among users and items. To give a better view regarding the recommendation process, we refer to the Figure 6.1.
6.2. PROPOSED COMMUNITY-AWARE BASELINE PREDICTOR FOR ITEM RECOMMENDER ALGORITHM

Figure 6.1: Steps to compute the ratings.

Table 6.2: This Table indicates the models described in this chapter. CNSVD, TCNSVD, CNSVD-Fast, and TCNSVD-Fast are extensions of NSVD and TNSVD models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Time-Aware</th>
<th>Community-Aware</th>
<th>Main Objective</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSVD</td>
<td>no</td>
<td>no</td>
<td>Accuracy</td>
<td>6.2.6</td>
</tr>
<tr>
<td>TNSVD</td>
<td>yes</td>
<td>no</td>
<td>Accuracy</td>
<td>6.2.7</td>
</tr>
<tr>
<td>CNSVD</td>
<td>no</td>
<td>yes</td>
<td>Accuracy</td>
<td>6.2.8</td>
</tr>
<tr>
<td>TCNSVD</td>
<td>yes</td>
<td>yes</td>
<td>Accuracy</td>
<td>6.2.9</td>
</tr>
<tr>
<td>CNSVD-Fast</td>
<td>no</td>
<td>yes</td>
<td>Speed</td>
<td>6.2.10</td>
</tr>
<tr>
<td>TCNSVD-Fast</td>
<td>yes</td>
<td>yes</td>
<td>Speed</td>
<td>6.2.11</td>
</tr>
</tbody>
</table>

6.2.6 Neighborhood-Integrated SVD (NSVD) Baseline Estimation

Collaborative filtering methods show a high amount of user and item biases in ratings. In other words, some users vote generously, and some items may receive overly positive votes from users. In this regard, Koren reflected these biases in a baseline estimator as follows:

\[ b_{ui} = \mu + b_u + b_i. \] (6.3)

In which \( b_{ui} \) is the baseline prediction of user \( u \) on item \( i \). Moreover, three parameters are involved in the least-square learning optimization process including \( \mu, b_u \) and \( b_i \). Here, \( \mu, b_u \) and \( b_i \) are the average rating, the user bias, and the item bias, respectively. In other words, \( \mu \) is the average rating of all the users over all the items, \( b_u \) is the user \( u \)'s deviation from the average rating and \( b_i \) is the item \( i \)'s deviation from the average rating. Suppose the average rating \( \mu \) over all the movies is 4.0 and a sample movie receives 0.7 above the average rating and user \( u \) rates 0.6 lower than the
average. Hence, the baseline estimate for this particular user on the sample item is 4.0 + 0.7 - 0.6. The following formula can also express the neighborhood estimation model:

\[ \hat{r}_{ui} = |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} ((r_{uj} - b_{uj})w_{ij} + c_{ij}). \] (6.4)

In the above formula, user \( u \) rates a set of items which is denoted by \( I_u \). Moreover, \( w_{ij} \) and \( c_{ij} \) indicate the deviations of the ratings from the baselines or the offsets. In a neighborhood model, weights are interpolation coefficients that map unknown ratings to known ones; however, in this model weights are offsets from the baseline estimates.

Hence, \( r_{uj} - b_{uj} \) are coefficients multiplied by the offsets or the weights. In other words, \( w_{ij} \) and \( c_{ij} \), which are related to explicit and implicit feedbacks, respectively, enhance estimated rating from the user to each item. Some users might rate too many items which their contribution is overestimated. Thus, we reduce their effect by applying \( |I_u|^{-\frac{1}{2}} \) coefficient. We describe the factor model as follows:

\[ \hat{r}_{ui} = q_i^T(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j), \] (6.5)

which \( q_i \) and \( p_u \) are latent vectors describing item \( i \) and user \( u \). We embed sum of vectors \( y_j \) in the preference vector \( u \) exploiting implicit feedback from each item \( j \in I_u \). Overall the above formulations show the basic neighborhood and factor models together. Several parameters need to be learned - we applied stochastic gradient descent for this purpose.

### 6.2.7 Time-Aware NSVD (TNSVD) Model

Often real-world data and networks face temporal drifts which help of data mining techniques are required to model these temporal effects. Similarly, recommender systems experience these dynamic changes both regarding items and users. Correspondingly, concept drift is a term which refers to the modeling of temporal effects. In fact, these temporal changes can be the emergence of new products or services that alter the focus of customers. For instance, a change in family structure, income, and the job may change shopping patterns. Overall, user preferences might vary at different times of the year, and this may depend on the seasonal changes and even feeling of the users. Although item properties are stable, their reception by the users might be changing while new brands and products arrive at the market. As an example, the reception of some mobile devices reduce when new versions are produced and delivered to the market, or as another example, some movies may lose their attention when new ones arrive.

In this regard, Koren proposed an extension to the neighborhood and factor models which tackles the temporal effects related to users and items. In this extension, baseline, neighborhood and factor estimators are affected by temporal dynamics. First, the baseline estimation (\( b_{ui} \)) from a user \( u \) to an item \( i \) is defined as follows:

\[ b_{ui}(t) = \mu + b_u(t) + b_i(t). \] (6.6)
6.2. PROPOSED COMMUNITY-AWARE BASELINE PREDICTOR FOR ITEM RECOMMENDER ALGORITHM

In the above formula, dynamic user and item biases are embedded via $b_u(t)$ and $b_i(t)$, respectively. $b_u(t)$ shows the difference between average rating of user $u$ and $\mu$ at time point $t$. Similarly, $b_i(t)$ indicates the deviation of average rating toward item $i$ and $\mu$ at time $t$. They parametrize $b_u(t)$ and $b_i(t)$ based on reasonable effects, for instance, the following linear function captures the gradual drift of user bias:

$$\text{dev}_u(t) = \text{sign}(t - t_u) \cdot |t - t_u|^\beta,$$

which $t_u$ is the mean time of rating and $t - t_u$ describes the time difference between $t$ and $t_u$. Moreover, we identify value for $\beta$ through cross-validation and here is set to $\beta = 0.4$. Moreover, we may require considering time-independent user bias and also the short-lived effects through $b_u$ and $b_{u,t}$ parameters. Spikes and sudden changes happen in many applications, for instance, the movie ratings generated by a single user on a single day may depend on her mood, and thus one may require reflecting these short-lived effects through $b_{u,t}$. Hence, we write the temporal bias for user $u$ as follows:

$$b_u(t) = b_u + \alpha_u \cdot \text{dev}_u(t) + b_{u,t},$$

where $\alpha_u$ parameter captures the gradual drifts related to each user.

The items have more stable properties while time-dependent item bias is less complex. Hence, $b_i(t)$ can be defined as follows:

$$b_i(t) = b_i + b_{i,\text{Bin}(t)},$$

which $b_i$ is the time-independent item bias. Moreover, biases imposed by temporal effects can be considered through $b_{i,\text{Bin}(t)}$. In fact, each bin is correspondent to a range, and the bias in that range can be computed.

Item bias does not contain the short-lived properties related to user biases. We revise the neighborhood model rating as follows:

$$\hat{r}_{ui}(t) = |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} e^{-\phi u_{i\j} |t - t_u|} ((r_{uj} - b_{uj}) w_{ij} + c_{ij}),$$

where the offsets and their coefficients are multiplied by a decay function $e^{-\beta u_{i\j} |t - t_j|}$. This function reduces the effect of ratings far from the time point $t$. Moreover, we revise the factor model as follows:

$$\hat{r}_{ui}(t) = q^T_i (p_u(t) + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j).$$

Here, the user preference vector $p_u$ is replaced with $p_u(t)$ that indicates the user preference changes through time. Likewise, user preferences are complex, and one needs to consider three components of gradual drift, short-lived effects, and time-independent factors. Therefore, time-dependent user preference drift can be computed as follows:

$$p_{uk}(t) = p_{uk} + \alpha_{uk} \cdot \text{dev}_u(t) + p_{uk,t} \quad k = 1, \ldots, n,$$

which $p_{uk}$ shows user stable preferences, $p_{uk,t}$ indicates sudden effects and $\alpha_{uk} \cdot \text{dev}_u(t)$ captures the gradual drifts. In other words, each component of $p_u(t)$ can be represented by a function $p_{uk}(t)$ which $p_u(t)^T = (p_{u1}(t), \ldots, p_{un}(t))$. 

99
6.2.8 Community-Aware NSVD (CNSVD) Model

In this section, we propose community-aware neighborhood-integrated SVD model, which is an extension to NSVD model, to improve recommendation accuracies. Similarly, we require applying the community structure on three components of baseline estimation, neighborhood and factor models. First, we start with baseline estimation.

**Baseline Estimation**

Collective behaviors of users in an online social network affect the user preferences as well as on item receptions. For instance, when someone wants to buy a mobile phone, she may listen to word of mouth and also to the opinions of the communities which she belongs to. In this regard, users belonging to the same community may possess similar interests and similarities. To apply the community structure on the baseline estimation, we may consider the community bias. In other words, average communities rating has a deviation from the average ratings of all users and also average rating related to a particular user has a deviation from the average rating of its community. To parametrize these effects, we consider community biases and embed them in the baseline estimation as follows:

\[ b_{ui} = \mu + b_u + b_{C_u} + b_i + b_{C_i}, \]  

which \( b_{C_u} \) shows user \( u \)’s community bias, \( C_u \) represents the set of user communities that user \( u \) belongs to. Similarly, \( b_{C_i} \) indicates the community bias for item \( i \) which \( C_i \) is the set of item communities that \( i \) is a member of.

To compute the community biases, we may require to consider bias corresponding to a community \( b_{C} \) as well as the membership of a user or an item to the corresponding community. In other words, the community bias can be defined as follows:

\[ b_{C_u} = \sum_{C \in C_u} b_C \cdot m_{uC}. \]  

(6.14)

Here, all the biases regarding user \( u \) to different communities are added, which can be calculated based on the multiplication of the community bias it belongs to, with the membership degree of the user \( u \) to community \( C \). Similarly, we can define the item community bias as follows:

\[ b_{C_i} = \sum_{C \in C_i} b_C \cdot m_{iC}. \]  

(6.15)

which \( b_{C} \) shows the community bias of community \( C \) and \( m_{iC} \) represents the membership level of item \( i \) belonging to community \( C \). Combining with the original baseline estimation, we reach to the extended community-aware baseline estimation as follows:

\[ b_{ui} = \mu + b_u + \sum_{C \in C_u} b_C \cdot m_{uC} + b_i + \sum_{C \in C_i} b_C \cdot m_{iC}. \]  

(6.16)
6.2. PROPOSED COMMUNITY-AWARE BASELINE PREDICTOR FOR ITEM RECOMMENDER ALGORITHM

Neighborhood Model

The next step is to apply the community structure on the neighborhood model. To this end, equation 6.4 is employed to take the effect of implicit feedback from a member of the users’ communities. This issue is reflected as follows:

\[
\hat{r}_{ui} = |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} ((r_{uj} - b_{uj})w_{ij} + c_{ij}) + |I_{Cu}|^{-\frac{1}{2}} \sum_{j \in I_{Cu}} d_{ij},
\]

(6.17)

which \(I_{Cu}\) represents the set of items that have been rated by any user belonging to one of user \(u\)’s communities. Moreover, item \(j\) contribution for item \(i\)’s rating is represented through offset \(d_{ij}\).

Latent Factor Model

We also require applying the community dimension in the latent factor model. As such, users member of a particular community and those who have rated specific items will be used to capture the implicit feedback, and they will be employed to compute the user’s preference vector \(p_u\). Using equation 6.5 and applying the item implicit feedback from user communities, we will obtain:

\[
\hat{r}_{ui} = q_i^T (p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j + |I_{Cu}|^{-\frac{1}{2}} \sum_{j \in I_{Cu}} z_j),
\]

(6.18)

which vector \(z_j\) reflects the implicit feedback from item \(j\). To consider the user preference and item characteristics, \(o_C\) represents the preferences or characteristics of community \(C\). Since a user can belong to multiple communities, we define \(o_{Cu}\) and \(o_{Ci}\). Respectively, they represent the combined community preferences of all communities that user \(u\) belongs to as well as the combined community characteristics of all the communities of the item \(i\). One can calculate \(o_{Cu}\) by aggregating the community factor \(o_C\) multiplied by the user’s membership value \(m_{uC}\) to each community \(C\) which can be computed as follows:

\[
o_{Cu} = \sum_{C \in C_u} o_C \cdot m_{uC}.
\]

(6.19)

Similarly, one can compute \(o_{Ci}\) by aggregating the community factor \(o_C\) multiplied by the item’s membership value \(m_{iC}\) to each community \(C\) that the following formula computes it:

\[
o_{Ci} = \sum_{C \in C_i} o_C \cdot m_{iC}.
\]

(6.20)

By aggregating the implicit item feedback from the set of user communities, user community preferences, and item community characteristics, the estimated rating can be computed as follows:

\[
\hat{r}_{ui} = (q_i + \sum_{C \in C_i} o_C \cdot m_{iC})^T(p_u + \sum_{C \in C_u} o_C \cdot m_{uC} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j + |I_{Cu}|^{-\frac{1}{2}} \sum_{j \in I_{Cu}} z_j).
\]

(6.21)
Combined Model and Parameter Learning

The final rating estimation can be computed by aggregating the baseline estimation (Equation 6.16), the neighborhood model (Equation 6.17) and the factor model (Equation 6.21) as follows:

$$
\hat{r}_{ui} = \mu + b_u + \sum_{C \in C_u} b_C \cdot m_{uC} + b_i + \sum_{C \in C_i} b_C \cdot m_{iC} \\
+ |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} ((r_{uj} - b_{uj}) w_{ij} + c_{ij}) + |I_{C_u}|^{-\frac{1}{2}} \sum_{j \in I_{C_u}} d_{ij} \\
+ (q_i + \sum_{C \in C_i} o_C \cdot m_{iC})^T (p_u + \sum_{C \in C_u} o_C \cdot m_{uC} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j + |I_{C_u}|^{-\frac{1}{2}} \sum_{j \in I_{C_u}} z_j).
$$

(6.22)

We learned the parameters of the model by minimizing the squared-error function as follows:

$$
\min_{b_u, w_u, c, d, q, p, o, y, z} \left( r_{ui} - \mu - b_u - \sum_{C \in C_u} b_C \cdot m_{uC} - b_i - \sum_{C \in C_i} b_C \cdot m_{iC} \\
- |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} ((r_{uj} - b_{uj}) w_{ij} + c_{ij}) - |I_{C_u}|^{-\frac{1}{2}} \sum_{j \in I_{C_u}} d_{ij} \\
- (q_i + \sum_{C \in C_i} o_C \cdot m_{iC})^T (p_u + \sum_{C \in C_u} o_C \cdot m_{uC} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j + |I_{C_u}|^{-\frac{1}{2}} \sum_{j \in I_{C_u}} z_j)^2 \\
+ \lambda (b_u^2 + \sum_{C \in C_u} b_C^2 + b_i^2 + \sum_{C \in C_u} w_{ij}^2 + \sum_{j \in I_u} c_{ij}^2 + \sum_{j \in I_u} d_{ij}^2 + \sum_{C \in C_i} \|o_C\|^2 + \sum_{j \in I_u} \|y_j\|^2 + \sum_{j \in I_{C_u}} \|z_j\|^2).
$$

(6.23)

In the above formula, $\lambda(...)$ is the regularization term to avoid large parameter values. It is a very typical approach in data mining to prevent biases in an optimization problem. The regularization parameter needs to be computed based on the used dataset. We employ stochastic gradient descent to learn the parameters of the model. Parameters of the CNSVD model are updated as follows:

- $\forall C \in C_u$:
  
  $$
  b_C \leftarrow b_C + \gamma \cdot (e_{ui} \cdot m_{uC} - \lambda \cdot b_C)
  $$

- $b_u \leftarrow b_u + \gamma \cdot (e_{ui} - \lambda \cdot b_u)$

- $\forall C \in C_i$:
  
  $$
  b_C \leftarrow b_C + \gamma \cdot (e_{ui} \cdot m_{iC} - \lambda \cdot b_C)
  $$
• $b_i \leftarrow b_i + \gamma \cdot (e_{ui} - \lambda \cdot b_i)$

• $\forall j \in I_u:$
  $w_{ij} \leftarrow w_{ij} + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} \cdot (r_{uj} - b_{uj}) - \lambda \cdot w_{ij})$

• $\forall j \in I_u:$
  $c_{ij} \leftarrow c_{ij} + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} - \lambda \cdot c_{ij})$

• $\forall j \in I_{C_u}:$
  $d_{ij} \leftarrow d_{ij} + \gamma \cdot (e_{ui} \cdot |I_{C_u}|^{-\frac{1}{2}} - \lambda \cdot d_{ij})$

• $q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot (p_u + \sum_{C \in C_u} o_C \cdot m_{uC}) + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j + |I_{C_u}|^{-\frac{1}{2}} \sum_{j \in I_{C_u}} z_j) - \lambda \cdot q_i)$

• $\forall C \in C_i :$
  $o_C \leftarrow o_C + \gamma \cdot (e_{ui} \cdot m_{iC} \cdot (p_u + \sum_{C \in C_i} o_C \cdot m_{uC}) + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j + |I_{C_u}|^{-\frac{1}{2}} \sum_{j \in I_{C_u}} z_j) - \lambda \cdot o_C)$

• $p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot (q_i + \sum_{C \in C_i} o_C \cdot m_{iC}) - \lambda \cdot p_u)$

• $\forall C \in C_u :$
  $o_C \leftarrow o_C + \gamma \cdot (e_{ui} \cdot m_{uC} \cdot (q_i + \sum_{C \in C_i} o_C \cdot m_{iC}) - \lambda \cdot o_C)$

• $\forall j \in I_u :$
  $y_j \leftarrow y_j + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} \cdot (q_i + \sum_{C \in C_i} o_C \cdot m_{iC}) - \lambda \cdot y_j)$

• $\forall j \in I_{C_u} :$
  $z_j \leftarrow z_j + \gamma \cdot (e_{ui} \cdot |I_{C_u}|^{-\frac{1}{2}} \cdot (q_i + \sum_{C \in C_i} o_C \cdot m_{iC}) - \lambda \cdot z_j)$

### 6.2.9 Time and Community-Aware NSVD (TCNSVD) Model

We propose TCSVD model as an extension to TNSVD model which three components of baseline estimation, neighborhood and factor models capture temporal user and item communities. Time-evolving graphs reflect these temporal changes. In other words, we detect community structures over time and identify user and item communities. If we apply a community detection algorithm on the user graph then user $u$ at time $t$ belongs to a set of communities $C_{u,C\text{Bin}(t)}$.

Similarly, this can also be mentioned as for item graph which item $i$ at time $t$ belongs to a set of item communities $C_{i,C\text{Bin}(t)}$. In this regard, $C\text{Bin}(t)$ identifies the community time bins for a time range $t$. In the following, we adopt the components of the baseline estimation.

**Baseline Estimation**

As we learned from TNSVD model, temporal user and item biases can face three types of drifts including time-independent, short-lived and gradual changes. Similarly, we may inject these effects
Applications of Overlapping Communities

for item and user communities, in other words, item and user communities can obtain these types of drifts. Hence, we rewrite the baseline estimation as follows:

\[
b_{ui}(t) = \mu + b_u + \alpha_u \cdot \text{dev}_u(t) + b_{u,t} + b_{i,\text{Bin}(t)} + \sum_{C \in C_u, \text{Bin}(t)} (b_C + b_{C,t}) \cdot m_{uC}
\]

\[
+ \sum_{C \in C_u} \alpha_C \cdot \text{dev}_C(t) \cdot m_{uC} + \sum_{C \in C_i, \text{Bin}(t)} (b_C + b_{C,\text{Bin}(t)}) \cdot m_{iC}.
\]

(6.24)

Here, \(b_C\) represents the time-independent bias of community \(C\) and \(b_{C,t}\) shows the community bias resulting from immediate effects regarding user communities and \(b_{C,\text{Bin}(t)}\) stands for the item community bias over time. Moreover, \(\alpha_C \cdot \text{dev}_C(t)\) shows the gradual drift regarding the user community biases; we considered linear drift for user communities. We aggregate these terms - community biases - and multiply them by membership degree of users and items to communities which are denoted by \(m_{iC}\) and \(m_{uC}\), respectively. Dynamic communities are employed to compute the time-independent and unexpected community effects; however, gradual drifts are calculated based on static communities.

Furthermore, users encounter daily, weekly, monthly and seasonal effects. For instance, daily effects might be related to a user that listens to different types of music during the day. On the other hand, we may observe seasonal effects for instance in shopping of clothes. In this regard, in our computations, the periodic effects are also considered. However, we limit them to weekly events while daily effects are not available in the dataset or there are not enough recurring seasonal effects.

Hence, we can write the extended baseline estimation considering seasonal effects as follows:

\[
b_{ui}(t) = \mu + b_u + \alpha_u \cdot \text{dev}_u(t) + b_{u,t} + b_{u,\text{Period}(t)} + b_{i,\text{Bin}(t)} + b_{i,\text{Period}(t)}
\]

\[
+ \sum_{C \in C_u, \text{Bin}(t)} (b_C + b_{C,t}) \cdot m_{uC} + \sum_{C \in C_u} \alpha_C \cdot \text{dev}_C(t) \cdot m_{uC}
\]

\[
+ \sum_{C \in C_i, \text{Bin}(t)} (b_C + b_{C,\text{Bin}(t)}) \cdot m_{iC},
\]

(6.25)

where \(b_{u,\text{Period}(t)}\) and \(b_{i,\text{Period}(t)}\) show the weekly user and item biases and the period subscript shows the weekly effect.

**Neighborhood Model**

We can extend the neighborhood model presented in the formula 6.17 and write it as follows:

\[
\hat{r}_{ui}(t) = |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} e^{-\beta_u \cdot |t-t_j|} ((r_{uj} - b_{uj}) w_{ij} + c_{ij}) + |I_{C_u}|^{-\frac{1}{2}} \sum_{j \in I_{C_u}} e^{-\beta_u \cdot |t-t_j|} d_{ij},
\]

(6.26)

where \(e^{-\beta_u \cdot |t-t_j|}\) is a decay function that shows the implicit feedback \(d_{ij}\) from items related to certain users of communities.
6.2. PROPOSED COMMUNITY-AWARE BASELINE PREDICTOR FOR ITEM RECOMMENDER ALGORITHM

Latent Factor Model

To apply the user community preference vector, we add the $t$ notation to Equation 6.21, hence, we can write as follows:

$$o_{Cu}(t) = \sum_{C \in C_u} o_C(t) \cdot m_{uC}, \quad (6.27)$$

where $o_{Cu}(t)$ shows the community preference vector at time $t$. Similar to user drifts, we may reflect the user community drifts including the short-lived, gradual and time-independent biases in the factor model. Hence, we write down the term regarding time-evolving user community preference vector $o_C(t)$ as follows:

$$o_{Ck}(t) = o_{Ck} + \alpha_{Ck} \cdot \text{dev}_C(t) + o_{Ck,t} \quad k = 1, \ldots, n, \quad (6.28)$$

which $o_{Ck}$ shows time-independent user community preference vector, $o_{Ck,t}$ and $\alpha_{Ck} \cdot \text{dev}_C(t)$ represent sudden and gradual drifts of community preferences. Similarly, these three terms are embedded as a component in the vector $o_C(t)^T = (o_{C1}(t), o_{C2}(t), \ldots, o_{Cn}(t))$. Now, we aggregate the community preference vector $o_{Cu}(t)$ and the user community drifts $o_C(t)$ as follows:

$$\hat{r}_{ui}(t) = (q_i + o\mathbf{c}_u)^T (p_u(t) + o_{Cu}(t) + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j + |I_{Cu}|^{-\frac{1}{2}} \sum_{j \in I_{Cu}} z_j). \quad (6.29)$$

As item characteristics do not change over time, item community characteristics are not reflected in the above formulations.

Combined Model and Parameter Learning

Now it is time to combine the Equations 6.25, 6.26, and 6.29 by aggregating the prediction values correspondent to the baseline estimation, the neighborhood and factor models. Hence, we can write as follows:

$$\hat{r}_{ui}(t) = \mu + b_u + \alpha_u \cdot \text{dev}_u(t) + b_u,t + b_u,\text{Period}(t) + b_i + b_i,\text{Bin}(t) + b_i,\text{Period}(t)$$

$$+ \sum_{C \in C_{u,C Bin}(t)} (b_C + b_{C,j}) \cdot m_{uC} + \sum_{C \in C_u} \alpha_C \cdot \text{dev}_C(t) \cdot m_{uC} + \sum_{C \in C_{1,C Bin}(t)} (b_C + b_{C,Bin}(t)) \cdot m_{uC}$$

$$+ |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} e^{-\phi_u \cdot |t-t_j|} \left( (r_{uj} - b_{uj}) w_{ij} + c_{ij} \right) + |I_{u,C Bin}(t)|^{-\frac{1}{2}} \sum_{j \in I_{u,C Bin}(t)} e^{-\psi_u \cdot |t-t_j|} d_{ij}$$

$$+ (q_i + \sum_{C \in C_{1,C Bin}(t)} \alpha_C \cdot m_{uC})^T \left( p_u + \alpha_u \cdot \text{dev}_u(t) + p_u,t \right) + \sum_{C \in C_{u,C Bin}(t)} (\alpha_C + \alpha_{C,t}) \cdot m_{uC}$$

$$+ \sum_{C \in C_u} \alpha_C \cdot \text{dev}_C(t) \cdot m_{uC} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j + |I_{Cu}|^{-\frac{1}{2}} \sum_{j \in I_{Cu}} z_j). \quad (6.30)$$
We apply stochastic gradient descent to minimize the following cost function:

\[
\begin{align*}
&\min_{b_u, \alpha, \phi, \psi, \omega, \epsilon, \gamma, \mu} \left( r_{ui} - \mu - b_u - \alpha_u \cdot \dev_u(t) - b_{u,t} - b_{u,\text{Period}(t)} - b_i \right) \\
&- b_{i,\text{Bin}(t)} - b_{i,\text{Period}(t)} - \sum_{C \in \mathcal{C}_{u,\text{CBin}(t)}} (b_C + b_{C,t}) \cdot m_{uC} - \sum_{C \in \mathcal{C}_u} \alpha_C \cdot \dev_C(t) \cdot m_{uC} \\
&- \sum_{C \in \mathcal{C}_{i,\text{CBin}(t)}} (b_C + b_{C,\text{Bin}(t)}) \cdot m_{iC} - |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} e^{-\phi_{u,t} \cdot |t-t_j|} ((r_{uj} - b_{uj})w_{ij} + c_{ij}) \\
&- |I_{u,\text{CBin}(t)}|^{-\frac{1}{2}} \sum_{j \in I_{u,\text{CBin}(t)}} e^{-\psi_{u,t} \cdot |t-t_j|} d_{ij} - (q_i + \sum_{C \in \mathcal{C}_{i,\text{CBin}(t)}} \alpha_C \cdot m_{iC})^T ((p_u + \alpha_u \cdot \dev_u(t) + p_{u,t}) + \sum_{C \in \mathcal{C}_{u,\text{CBin}(t)}} (b_C + b_{C,\text{Bin}(t)}) + \sum_{C \in \mathcal{C}_{u,\text{CBin}(t)}} \alpha_C \cdot m_{uC} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \\
&+ |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} z_j \right) + \lambda \left( \sum_{C \in \mathcal{C}_{u,\text{CBin}(t)}} \alpha_C^2 + \sum_{C \in \mathcal{C}_{i,\text{CBin}(t)}} (b_C^2 + b_{C,\text{Bin}(t)}^2) + \sum_{j \in I_u} (w_{ij}^2 + c_{ij}^2) + \phi_u^2 + \sum_{j \in I_{u,\text{CBin}(t)}} d_{ij}^2 + \psi_u^2 \\
&+ \|q_i\|^2 + \sum_{C \in \mathcal{C}_{i,\text{CBin}(t)}} \|\alpha_C\|^2 + \|p_u\|^2 + \|\alpha_u\|^2 + \|p_{u,t}\|^2 + \sum_{C \in \mathcal{C}_{u,\text{CBin}(t)}} (\|\alpha_C\|^2 + \|\alpha_u\|^2) \\
&+ \|\alpha_C\|^2 + \sum_{j \in I_u} \|y_j\|^2 + \sum_{j \in I_u} \|z_j\|^2 \right) \\
\end{align*}
\]

(6.31)

Similar to section 6.2.8, \(\lambda(\ldots)\) is a parameter to avoid biases in optimization term. The parameters of the model are iteratively updated as follows:

- \(b_u \leftarrow b_u + \gamma \cdot (e_{ui} - \lambda \cdot b_u)\)
- \(b_{u,t} \leftarrow b_{u,t} + \gamma \cdot (e_{ui} - \lambda \cdot b_{u,t})\)
- \(b_{u,\text{Period}(t)} \leftarrow b_{u,\text{Period}(t)} + \gamma \cdot (e_{ui} - \lambda \cdot b_{u,\text{Period}(t)})\)
- \(b_i \leftarrow b_i + \gamma \cdot (e_{ui} - \lambda \cdot b_i)\)
- \(b_{i,\text{Bin}(t)} \leftarrow b_{i,\text{Bin}(t)} + \gamma \cdot (e_{ui} - \lambda \cdot b_{i,\text{Bin}(t)})\)
- \(\alpha_u \leftarrow \alpha_u + \gamma \cdot (e_{ui} \cdot \dev_u(t) - \lambda \cdot \alpha_u)\)
- \(b_{i,\text{Period}(t)} \leftarrow b_{i,\text{Period}(t)} + \gamma \cdot (e_{ui} - \lambda \cdot b_{i,\text{Period}(t)})\)
6.2. PROPOSED COMMUNITY-AWARE BASELINE PREDICTOR FOR ITEM
RECOMMENDER ALGORITHM

• \( \forall C \in C_{u,CBin(t)} : \)
  \[ b_C \leftarrow b_C + \gamma \cdot (e_{ui} \cdot m_{uC} - \lambda \cdot b_C) \]

• \( \forall C \in C_{u,CBin(t)} : \)
  \[ b_{C,t} \leftarrow b_{C,t} + \gamma \cdot (e_{ui} \cdot m_{uC} - \lambda \cdot b_{C,t}) \]

• \( \forall C \in C_{i,CBin(t)} : \)
  \[ b_C \leftarrow b_C + \gamma \cdot (e_{ui} \cdot m_{iC} - \lambda \cdot b_C) \]

• \( \forall C \in C_{i,CBin(t)} : \)
  \[ b_{C,t} \leftarrow b_{C,t} + \gamma \cdot (e_{ui} \cdot m_{iC} - \lambda \cdot b_{C,t}) \]

• \( \forall C \in C_{u} : \)
  \[ \alpha_C \leftarrow \alpha_C + \gamma \cdot (e_{ui} \cdot dev_C(t) \cdot m_{uC} - \lambda \cdot \alpha_C) \]

• \( \forall j \in I_u : \)
  \[ w_{ij} \leftarrow w_{ij} + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} \cdot e^{-\psi_u|t-t_j|} \cdot (r_{uj} - b_{uj}) - \lambda \cdot w_{ij}) \]

• \( \forall j \in I_u : \)
  \[ c_{ij} \leftarrow c_{ij} + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} \cdot e^{-\psi_u|t-t_j|} - \lambda \cdot c_{ij}) \]

• \( \forall j \in I_{u,\text{CBin}(t)} : \)
  \[ d_{ij} \leftarrow d_{ij} + \gamma \cdot (e_{ui} \cdot |I_{u,\text{CBin}(t)}|^{-\frac{1}{2}} \cdot e^{-\psi_u|t-t_j|} \cdot -\lambda \cdot d_{ij}) \]

• \( \phi_{u} \leftarrow \phi_{u} + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} \cdot \sum_{j \in I_u} (|t_j - t_j|) e^{-\phi_u|t-t_j|}((r_{uj} - b_{uj})w_{ij} + c_{ij} - \lambda \cdot \phi_{u}) \]

• \( \psi_{u} \leftarrow \psi_{u} + \gamma \cdot (e_{ui} \cdot |I_{u,\text{CBin}(t)}|^{-\frac{1}{2}} \cdot \sum_{j \in I_{u,\text{CBin}(t)}} (|t_j - t_j|) e^{-\psi_u|t-t_j|}d_{ij} - \lambda \cdot \psi_{u}) \]

• \( q_{t} \leftarrow q_{t} + \gamma \cdot (e_{ui} \cdot (p_u(t) + \sum_{C \in C_{u,\text{CBin}(t)}} (o_C + o_{C,t}) \cdot m_{uC} + \sum_{C \in C_{u}} \alpha_C \cdot dev_C(t) \cdot m_{uC} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_{ij} + |I_{u,\text{CBin}(t)}|^{-\frac{1}{2}} \sum_{j \in I_{u,\text{CBin}(t)}} z_{j}) - \lambda \cdot q_{t}) \]

• \( \forall C \in C_{i} : \)
  \[ o_C \leftarrow o_C + \gamma \cdot (e_{ui} \cdot m_{iC} \cdot (p_u(t) + \sum_{C \in C_{i,\text{CBin}(t)}} o_C(t) \cdot m_{uC} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_{ij} + |I_{u,\text{CBin}(t)}|^{-\frac{1}{2}} \sum_{j \in I_{u,\text{CBin}(t)}} z_{j}) - \lambda \cdot o_C) \]

• \( p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot (q_{t} + \sum_{C \in C_{i,\text{CBin}(t)}} o_C \cdot m_{iC}) - \lambda \cdot p_u) \]

• \( p_{u,t} \leftarrow p_{u,t} + \gamma \cdot (e_{ui} \cdot (q_{t} + \sum_{C \in C_{i,\text{CBin}(t)}} o_C \cdot m_{iC}) - \lambda \cdot p_{u,t}) \]

• \( \alpha_u \leftarrow \alpha_u + \gamma \cdot (e_{ui} \cdot dev_u(t) \cdot (q_{t} + \sum_{C \in C_{i,\text{CBin}(t)}} o_C \cdot m_{iC}) \cdot -\lambda \cdot \alpha_u) \]

• \( \forall C \in C_{u,\text{CBin}(t)} : \)
  \[ o_C \leftarrow o_C + \gamma \cdot (e_{ui} \cdot m_{uC} \cdot (q_{t} + \sum_{C \in C_{i,\text{CBin}(t)}} o_C \cdot m_{iC}) - \lambda \cdot o_C) \]

• \( \forall C \in C_{u,\text{CBin}(t)} : \)
  \[ o_{C,t} \leftarrow o_{C,t} + \gamma \cdot (e_{ui} \cdot m_{uC} \cdot (q_{t} + \sum_{C \in C_{i,\text{CBin}(t)}} o_C \cdot m_{iC}) - \lambda \cdot o_{C,t}) \]

• \( \forall C \in C_{u,\text{CBin}(t)} : \)
  \[ o_C \leftarrow o_C + \gamma \cdot (e_{ui} \cdot dev_C(t) \cdot m_{uC} \cdot (q_{t} + \sum_{C \in C_{i,\text{CBin}(t)}} o_C \cdot m_{iC}) - \lambda \cdot o_C) \]
• $\forall j \in I_u$:
  $y_j \leftarrow y_j + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} \cdot (q_i + \sum_{C \in C_i, CBin(t)} o_C \cdot m_{iC}) - \lambda \cdot y_j)$

• $\forall j \in I_{Cu}$:
  $z_j \leftarrow z_j + \gamma \cdot (e_{ui} \cdot |I_{Cu}|^{-\frac{1}{2}} \cdot (q_i + \sum_{C \in C_i, CBin(t)} o_C \cdot m_{iC}) - \lambda \cdot z_j)$

6.2.10 Community-Aware NSVD for Faster Learning (CNSVD-Fast) Model

The time complexity of the learning process for the CNSVD model is too high, and this is mainly due to the high dimensionality of the problem considering both user and community level parameters. Therefore, we improve the learning running times by a little bit of sacrificing the precision. In other words, we only learn parameters regarding the community biases, feedbacks, preferences, and characteristics by avoiding learning at the level of individual user and item. Thus, the user and item community parameters will be updated on the specific sample belonging to that community. This setting leads to higher number of iterations and faster convergence of the model. Furthermore, we remove many of the membership multiplications and the aggregations because disjoint community detection techniques replace overlapping community detection algorithms. In the following, we rewrite and optimize the formulas to only consider community parameters.

Baseline Estimation

Formula 6.3 is employed as for the baseline; however, the user and item biases are replaced with community biases $b_{Cu}$ and $b_{Ci}$, respectively:

$$b_{ui} = \mu + b_{Cu} + b_{Ci}. \quad (6.32)$$

Neighborhood Model

Similarly, we adopt the neighborhood model presented in equation 6.4 to the following formula:

$$\hat{r}_{ui} = |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} ((r_{uj} - b_{uj}) w_{iC_j} + c_{iC_j}), \quad (6.33)$$

where $w_{ij}$ and $c_{ij}$ parameters are replaced with their community related terms $w_{C_{i,j}}$ and $c_{C_{i,j}}$, respectively.

Latent Factor Model

Regarding the latent factor model, the equation is quite straightforward, and we replaced the user preference vector and item preference vector of equation 6.5 with $o_{Cu}$ and $o_{Ci}$ which we also introduced in CNSVD model. Respectively, $y_{j}$ is replaced by $y_{C_{i,j}}$ and thus we obtain:

$$\hat{r}_{ui} = o_{Ci}^T (o_{Cu} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_{C_{i,j}}) \quad (6.34)$$
6.2. PROPOSED COMMUNITY-AWARE BASELINE PREDICTOR FOR ITEM RECOMMENDER ALGORITHM

Combined Model and Parameter Learning

Corresponding prediction terms regarding equations 6.32, 6.33 and 6.34 are summed and the following combined rating estimation is obtained:

\[ \hat{r}_{ui} = \mu + b_{Cu} + b_{Ci} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} ((r_{uj} - b_{uj})w_{iCj} + c_{iCj}) + o_{Cu}^T(o_{Cu} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_{Cj}). \]  

(6.35)

Similarly, we can find the parameters of the model by minimizing the following cost function embedded with regulation terms:

\[
\min_{b_{Cu}, w_{iCj}, c_{iCj}, o_{Cu}, y_{Cj}} \left( r_{ui} - \mu - b_{Cu} - b_{Ci} - |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} ((r_{uj} - b_{uj})w_{iCj} + c_{iCj}) - o_{Cu}^T(o_{Cu} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_{Cj}) \right)^2 + \lambda \left( b_{Cu}^2 + b_{Ci}^2 + \sum_{j \in I_u} (w_{iCj}^2 + c_{iCj}^2) \right). 
\]

(6.36)

Finally, parameters of the models are updated iteratively as follows:

- \( b_{Cu} \leftarrow b_{Cu} + \gamma \cdot (e_{ui} - \lambda \cdot b_{Cu}) \)
- \( b_{Ci} \leftarrow b_{Ci} + \gamma \cdot (e_{ui} - \lambda \cdot b_{Ci}) \)
- \( \forall j \in I_u : \)
  \( w_{iCj} \leftarrow w_{iCj} + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} \cdot (r_{uj} - b_{uj}) - \lambda \cdot w_{iCj}) \)
- \( \forall j \in I_u : \)
  \( c_{iCj} \leftarrow c_{iCj} + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} \cdot (r_{uj} - b_{uj}) - \lambda \cdot c_{iCj}) \)
- \( o_{Ci} \leftarrow o_{Ci} + \gamma \cdot (e_{ui} \cdot (o_{Cu} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_{Cj}) - \lambda \cdot o_{Ci}) \)
- \( o_{Cu} \leftarrow o_{Cu} + \gamma \cdot (e_{ui} \cdot o_{Ci} - \lambda \cdot o_{Cu}) \)
- \( \forall j \in I_u : \)
  \( y_{Cj} \leftarrow y_{Cj} + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} \cdot o_{Ci} - \lambda \cdot y_{Cj}) \)

6.2.11 Time and Community-Aware NSVD for Faster Learning (TCNSVD-Fast) Model

Similarly, we can formulate the TCNSVD-Fast model based on the same rules that we applied on CNSVD-Fast model. In this regard, we replace the individual user and item parameters with community-related terms. In the following, we elaborate the baseline estimation, neighborhood, and the factor models.
Applications of Overlapping Communities

Baseline Estimation

We employ the TNSVD baseline of equation 6.8 and reduce it to time-independent and gradual user and item community biases instead of individual user and item biases. The following formula describes the baseline estimation for TCNSVD-Fast:

\[ b_{ui} = \mu + b_{Cu} + \alpha_{Cu} \cdot \text{dev}_{Cu}(t) + b_{Cu,t} + b_{Ci} + b_{Ci,Bin(t)}. \] (6.37)

Neighborhood Model

For the neighborhood model the same strategy is employed and user related terms are replaced by community related expressions. In other words, \( w_{ij}, c_{ij} \) and \( \phi_u \) parameters of the Equation 6.10 are replaced with \( w_{Ci,j}, c_{Ci,j} \) and \( \phi_{Cu} \), respectively. We can rewrite the neighborhood model as follows:

\[ \hat{r}_{ui} = |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} e^{-\phi_{Cu}|t-t_j|}(r_{uj} - b_{uj})w_{ijCj} + c_{ijCj}). \] (6.38)

Latent Factor Model

Finally, we reduce the latent factor prediction rating term by replacing the user preference and item characteristic vectors by \( o_{Cu} \) and \( o_{Ci} \). In fact, Equations 6.11 and 6.12 from CNSVD and TCNSVD models are adapted, respectively. Furthermore, we adapt linear drift parameter \( \alpha_u \) to \( \alpha_{Cu} \) and implicit item feedback \( y_j \) to \( y_{Cj} \), respectively and thus we obtain the following latent factor model:

\[ \hat{r}_{ui} = o_{C_i}^T(o_{Cu} + \alpha_{Cu} \cdot \text{dev}_{Cu}(t) + o_{Cu,t} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_{Cj}). \] (6.39)

Combined Model and Parameter Learning

Baseline estimation from the Equation 6.37, the neighborhood model from the Equation 6.38 and the factor model from the Equation 6.39 are combined to yield the final rating prediction term as follows:

\[ \hat{r}_{ui} = \mu + b_{Cu} + \alpha_{Cu} \cdot \text{dev}_{Cu}(t) + b_{Cu,t} + b_{Ci} + b_{Ci,Bin(t)} \]
\[ + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} e^{-\phi_{Cu}|t-t_j|}((r_{uj} - b_{uj})w_{ijCj} + c_{ijCj}) \]
\[ + o_{C_i}^T(o_{Cu} + \alpha_{Cu} \cdot \text{dev}_{Cu}(t) + o_{Cu,t} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_{Cj}). \] (6.40)
Correspondingly, the cost function is as follows:

\[
\min_{\alpha_u, \phi_u, \omega_y, \phi_t} \left( r_{ui} - \mu - b_{Cu} + \alpha_{Cu} \cdot \text{dev}_{Cu}(t) - b_{Cu,t} - b_{Ci,t} - b_{Ci,\text{Bin}(t)} \right.
\]

\[- \frac{1}{2} \sum_{j \in I_u} e^{-\phi_{Cu,j}|t-t_j|} \left( (r_{uj} - b_{uj}) w_{ij} + c_{ij} \right) \]

\[- \alpha_{Cu}^T \left( \alpha_{Cu} \cdot \text{dev}_{Cu}(t) + \alpha_{Cu,t} + \frac{1}{2} \sum_{j \in I_u} y_{ij} \right) \left( \alpha_{Cu} + \alpha_{Cu,t} + \frac{1}{2} \sum_{j \in I_u} y_{ij} \right) \right)^2 \]

\[+ \lambda \left( b_{Cu}^2 + b_{Cu,t}^2 + b_{Ci}^2 + b_{Ci,\text{Bin}(t)}^2 + \sum_{j \in I_u} (\phi_{Cu,j}^2 + \phi_{Cu,t}^2 + \phi_{Cu,i}^2) \right) \]

\[+ \|\alpha_{Cu}\|^2 + \|\alpha_{Cu,t}\|^2 + \|\alpha_{Cu,i}\|^2 + \frac{1}{2} \sum_{j \in I_u} \|y_{ij}\|^2 \right). \]  

(6.41)

Finally, stochastic gradient descent is employed and the parameters of the models are iteratively updated until convergence, as follows:

- \( b_{Cu} \leftarrow b_{Cu} + \gamma \cdot (e_{ui} - \lambda \cdot b_{Cu}) \)
- \( \alpha_{Cu} \leftarrow \alpha_{Cu} + \gamma \cdot (e_{ui} \cdot \text{dev}_{Cu}(t) - \lambda \cdot \alpha_{Cu}) \)
- \( b_{Cu,t} \leftarrow b_{Cu,t} + \gamma \cdot (e_{ui} - \lambda \cdot b_{Cu,t}) \)
- \( b_{Ci} \leftarrow b_{Ci} + \gamma \cdot (e_{ui} - \lambda \cdot b_{Ci}) \)
- \( b_{Ci,\text{Bin}(t)} \leftarrow b_{Ci} + \gamma \cdot (e_{ui} - \lambda \cdot b_{Ci,\text{Bin}(t)}) \)
- \( \phi_{Cu} \leftarrow \phi_{Cu} + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} \cdot \sum_{j \in I_u} (e^{-\phi_{Cu,i} \cdot |t-t_j|} \cdot (r_{uj} - b_{uj}) - \lambda \cdot w_{ij}) \)
- \( \forall j \in I_u : w_{ij} \leftarrow w_{ij} + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} \cdot e^{-\phi_{Cu,i} \cdot |t-t_j|} \cdot (r_{uj} - b_{uj}) - \lambda \cdot w_{ij}) \)
- \( \forall j \in I_u : c_{ij} \leftarrow c_{ij} + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} \cdot e^{-\phi_{Cu,i} \cdot |t-t_j|} - \lambda \cdot c_{ij}) \)
- \( \alpha_{Ci} \leftarrow \alpha_{Ci} + \gamma \cdot (e_{ui} \cdot \alpha_{Cu} + \alpha_{Cu} \cdot \text{dev}_{Cu}(t) + \alpha_{Cu,t} + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_{ij} - \lambda \cdot \alpha_{Ci}) \)
- \( \alpha_{Cu} \leftarrow \alpha_{Cu} + \gamma \cdot (e_{ui} \cdot \alpha_{Ci} - \lambda \cdot \alpha_{Cu}) \)
- \( \alpha_{Cu,t} \leftarrow \alpha_{Cu,t} + \gamma \cdot (e_{ui} \cdot \text{dev}_{Cu}(t) \cdot \alpha_{Ci} - \lambda \cdot \alpha_{Cu,t}) \)
- \( \forall j \in I_u : y_{ij} \leftarrow y_{ij} + \gamma \cdot (e_{ui} \cdot |I_u|^{-\frac{1}{2}} \cdot \alpha_{Ci} - \lambda \cdot y_{ij}) \)
6.2.12 Evaluation Results

In this section, we demonstrate the evaluation settings, applied datasets and the discussion over the results. To identify the accuracy of the proposed community-aware recommender algorithms, we perform a couple of experiments measuring accuracy and computational costs of the algorithms. To employ the algorithms, we need to identify a couple of parameters. First is graph construction method which can be ratings-based or tags-based. Besides, similarity metrics for graph construction, community detection algorithm either DMID or Walktrap and finally the time bins to capture the community drift in TCNSVD model are also among the parameters which need to be predetermined. To evaluate the proposed CNSVD, TCNSVD, CNSVD-Fast and TCNSVD-Fast regarding accuracy and running times, we compare them with a couple of baselines. We extend the proposed models upon the benchmarks including NSVD, TNSVD. Also, we use the Weighted Regularized Matrix Factorization algorithm by [HKVo08], item-based $k$-nearest neighbors collaborative filtering and a baseline predictor that employs the item average as for the benchmark models. The prototypical implementation of this part of the research was supported by a master theses guided by the author of this work [Bart17].

Datasets

We apply Netflix and MovieLens dataset for the evaluation of the proposed recommendation algorithms. Some necessary information regarding the number of users, items, ratings and the metadata information such as user-user and item-item graphs are available in the Table 6.3. Koren applied the NSVD and TNSVD models on the Netflix dataset. This dataset has been used for the Netflix Price challenge\(^1\) that aimed to improve prediction accuracies. Netflix is a huge dataset comprising 480189 users and 17770 items with more than 100 million ratings. In this dataset, building the user-user and item-item graphs is possible using the rating information; however, tag metadata is not available. The other dataset which we employed to evaluate the algorithms is the MovieLens dataset which the data is crawled from the movie recommender Website\(^2\). This dataset not only comprises the rating information from users to items but also it contains metadata information such as tags and time points. The tag metadata can be applied to build the user-user and item-item graphs based on commonly used tags. Regarding MovieLens dataset, there are 138000 users and 27000 items with around 20 million ratings available.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Type</th>
<th>Users</th>
<th>Items</th>
<th>Ratings</th>
<th>User-User</th>
<th>Item-Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netflix Movies</td>
<td>Movies</td>
<td>480,189</td>
<td>17,770</td>
<td>100,480,507</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>MovieLens Movies</td>
<td>Movies</td>
<td>138,000</td>
<td>27,000</td>
<td>20,000,000</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

\(^1\)http://www.netflixprize.com/

\(^2\)https://movielens.org/
6.2. PROPOSED COMMUNITY-AWARE BASELINE PREDICTOR FOR ITEM RECOMMENDER ALGORITHM

Evaluation Protocol

We employed the whole datasets of MovieLens (ML) and Netflix (NF) to evaluate the recommender models on the nodes of RWTH compute cluster with 128 GB of memory; however, the evaluation could not finish on the whole datasets. In this regard, we had to reduce the volume of datasets through sub-sampling. As for this, we selected 40000 users from both ML and NF datasets, and we kept only the corresponding ratings. Table 6.4 indicates characteristics of the original and the sub-sampled datasets. We can infer that properties did not change. In other words, ML original and ML sub-sampled have 0.5400 and 0.6263 % as for data density. Moreover, average rating per user in ML original and sub-sampled are 144.41 and 143.50, respectively. The pattern is almost similar regarding the NF original and NF subsampled with 1.1776 and 1.1858 data density for each of them. Average rating per user is close as for NF original (209.25) and NF sub-sampled (210.60). Furthermore, we require applying a suitable evaluation protocol for all the algorithms. Evaluation of time-aware recommender algorithms may be challenging while we need to keep the time ordering of the ratings and thus using of cross-validation approaches may cause problems in this regard. As for this, we order the ratings based on the time stamp available in the datasets.

Moreover, the binning and splitting are applied on the whole datasets while a sliding window approach should be utilized instead of cross-validation. The classical cross-validation technique cannot be employed here while it disturbs the ordering of the ratings and thus the training samples might be more recent than the test samples. Regarding splitting the data and the time window strategy, we employ a couple of snapshots; each snapshot can be used either to train and test the models. Train and test sets of each snapshot are non-overlapping which test samples happen at slightly later times.

Although we can apply several iterations of evaluations on the whole dataset, only we use a portion of the data on each learning iteration. Figure 6.3 demonstrates the employed sliding window scheme for training and testing the models. To further clarify the sliding window scheme, Table 6.5 gives information regarding the five employed snapshots on the MovieLens dataset. Snapshot 1 is considered for 3500 days which 2800 days (2469032 samples) are for training and 700 days (770055 samples) for testing. To summarize, we apply the steps indicated in the Figure 6.2 as for the evaluation protocol.

Regarding item ranking, we employed Librec library strategy that uses user-specific target item set. It comprises all items that have received at least one rating from any user. All of the metrics including precision@$k$, recall@$k$ and NDCG measures are computed based on this strategy.

Results

We intend to evaluate the proposed CNSVD, TCNSVD and CNSVD-Fast and TCNSVD-Fast models with several benchmarks regarding rating prediction and item ranking accuracy. However, there are several parameters involved which we require identifying the best settings for the runs of the algorithms. In other words, we obtain a deeper understanding of the performance of the algorithm on MovieLens dataset.
Randomly select 40,000 users and keep their corresponding ratings.

Take five snapshots from the data; each snapshot contains 50 percent of all ratings.

Split each snapshot to 80 percent of training data and 20 percent of test data.

Average over all the snapshot runs; this is true for all the experiments.

**Figure 6.2:** This Figure shows the main steps to compute the rating.

**Table 6.4:** This Table indicates the comparison of the original and sub-sampled ML and NF datasets. This table shows the number of users, items, ratings as well as data density and average rating per user.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Users</th>
<th>Items</th>
<th>Ratings</th>
<th>Data Density</th>
<th>Avg. Ratings per User</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML original</td>
<td>138,493</td>
<td>26,744</td>
<td>20,000,263</td>
<td>0.5400%</td>
<td>144.41</td>
</tr>
<tr>
<td>ML subsampled</td>
<td>40,000</td>
<td>22,912</td>
<td>5,740,089</td>
<td>0.6263%</td>
<td>143.50</td>
</tr>
<tr>
<td>NF original</td>
<td>480,189</td>
<td>17,770</td>
<td>100,480,507</td>
<td>1.1776%</td>
<td>209.25</td>
</tr>
<tr>
<td>NF subsampled</td>
<td>40,000</td>
<td>17,760</td>
<td>8,424,080</td>
<td>1.1858%</td>
<td>210.60</td>
</tr>
</tbody>
</table>

**Table 6.5:** This Table shows the number of days and the sample size regarding MovieLens dataset. It indicates train and test sets for each snapshot for MovieLens dataset.

<table>
<thead>
<tr>
<th>Snapshot</th>
<th>Training Days</th>
<th>Training Samples</th>
<th>Test Days</th>
<th>Test Samples</th>
<th>Sample Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,800</td>
<td>2,469,032</td>
<td>700</td>
<td>770,055</td>
<td>76/24</td>
</tr>
<tr>
<td>2</td>
<td>2,800</td>
<td>2,723,942</td>
<td>700</td>
<td>609,454</td>
<td>82/18</td>
</tr>
<tr>
<td>3</td>
<td>2,800</td>
<td>2,666,220</td>
<td>700</td>
<td>565,258</td>
<td>83/17</td>
</tr>
<tr>
<td>4</td>
<td>2,800</td>
<td>2,537,520</td>
<td>700</td>
<td>416,326</td>
<td>86/14</td>
</tr>
<tr>
<td>5</td>
<td>2,800</td>
<td>2,145,875</td>
<td>700</td>
<td>355,127</td>
<td>86/14</td>
</tr>
</tbody>
</table>

For this part of the evaluation, one can include a reduced set of evaluation metrics as for the accuracy. Afterwards, it is possible to compare the algorithms with benchmarks and the state of the arts. We need to identify several parameters before running the algorithms. A complete comparison of these parameters would not be possible; however, we may change one parameter at a time and ob-
6.2. PROPOSED COMMUNITY-AWARE BASELINE PREDICTOR FOR ITEM RECOMMENDER ALGORITHM

Figure 6.3: This Figure indicates the sliding windows evaluation scheme which the timestamps of the earliest and the latest rating are denoted by $t_{\text{min}}$ and $t_{\text{max}}$, respectively. Ratings of each snapshot are split into training and test data at time point $t_i$.

serve the outcome. While community-aware algorithms require graph construction, we may require predetermining $k$ value in the $K$-NN approach.

Moreover, we should identify whether ratings or tags are employed to build user-user and item-item graphs. Besides, the index to find similarities among users and items need to be selected which can be Cosine, Pearson, and JMSD. Also as for community-aware algorithms such as CNSVD and TCNSVSD models, we require deciding which algorithms to choose between the two choices of DMID and Walktrap. Walktrap also requires identifying the number of steps. Finally, we need to determine community bins for TCNSVSD algorithm, which define the time slice for the considered community temporal effects. We selected a set of values as the parameters which Table 6.6 explains them.

Table 6.6: This Table shows the applied parameter values used for the evaluation after the initial investigation of the parameter space.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Community detection algorithm</td>
<td>DMID, Walktrap</td>
</tr>
<tr>
<td>Walktrap steps</td>
<td>2, 5</td>
</tr>
<tr>
<td>Graph construction method</td>
<td>Taggings, Ratings</td>
</tr>
<tr>
<td>$k$-NN neighbors</td>
<td>5, 10, 20</td>
</tr>
<tr>
<td>$k$-NN similarity</td>
<td>Cosine, Pearson, JMSD</td>
</tr>
<tr>
<td>Community time bins (TCNSVSD only)</td>
<td>1, 2, 4</td>
</tr>
</tbody>
</table>

First, we investigate the effect of graph construction on the algorithm performance. In this regard, we can refer to Figures 6.4 and 6.5 as for the number of $k$ in the $K$-NN method and the similarity index, respectively. Regarding the $k=5$, 10 and 20, RMSE values are quite close for all the algo-
Applications of Overlapping Communities

rithms. Moreover, regarding the precision and recall, there are further variations. For instance, CNSVD-Fast has a much better performance as for \( k=20 \) which is followed by \( k=5 \) and \( k=10 \), respectively. Hence, precision and recall even face more variations when we discuss fast versions of community-aware algorithms. In general, we cannot identify a parameter that achieves the best performance in this regard.

Figure 6.5 indicates the performance regarding different similarity metrics such as Cosine, Pearson and the JMSD. As we can observe, JMSD has better performance regarding precision and recall for all the models. Moreover, it has quite competitive performance as for CNSVD, CNSVD-Fast and TCNSVD-Fast and the only exception is TCNSVD. Overall, JMSD obtains the best performance in 7 cases and appears competitive in 2 cases. Afterwards, Cosine achieves the best performance in one case and two cases as competitive, and finally, Pearson does not surpass in any cases. In this regard, we cannot mention any of the measures as the best-performing in all the models; however, JMSD appeared satisfactory regarding the precision values. Table 6.7 gives further information regarding the best-performing parameters.

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE</th>
<th>Precision@10</th>
<th>Recall@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNSVD</td>
<td>( k=20 ), Pearson</td>
<td>( k=10 ), Cosine</td>
<td>( k=20 ), Cosine</td>
</tr>
<tr>
<td>TCNSVD</td>
<td>( k=20 ), Pearson</td>
<td>( k=10 ), Pearson</td>
<td>( k=10 ), Pearson</td>
</tr>
<tr>
<td>CNSVD-Fast</td>
<td>( k=5 ), Pearson</td>
<td>( k=20 ), JMSD</td>
<td>( k=20 ), JMSD</td>
</tr>
<tr>
<td>TCNSVD-Fast</td>
<td>( k=10 ), Cosine</td>
<td>( k=10 ), JMSD</td>
<td>( k=20 ), JMSD</td>
</tr>
</tbody>
</table>

Regarding the community detection approaches, we have two choices of DMID and Walktrap which as for the latter, we consider two choices of step size 2 and 5. Two types of graphs are available resulting from the rating and tags-based constructions, and thus we evaluate the community detection algorithms on both of them. Results are depicted in Figures 6.6 and 6.7. Regarding the ratings-based construction, Walktrap achieves the best performance compared to DMID. The only difference is the Rec@10 for TCNSVD model with a much higher performance of DMID. We may not identify the best-performing step size for Walktrap while the best winning one is ambiguous in this regard. The situation for DMID seems better as for the tags-based graph construction. DMID achieves better item ranking as for Prec@10 and Rec@10 as for TCNSVD model. Its performance is slightly better as for TCNSVD-Fast and Prec@10.

Moreover, DMID achieves slightly better performance in CNSVD, CNSVD-Fast and competitive performance as for TCNSVD-Fast regarding RMSE metric. However, there are cases which Walktrap surpassed DMID in the tags-based construction, for instance, we refer to CNSVD, CNSVD-Fast for such cases regarding Prec@10 and Rec@10. As for other cases, DMID is quite competitive, and the difference might be negligible. Overall, we can mention that Walktrap could achieve clear performance in comparison to DMID on ratings-based construction; however, DMID enjoyed better results as for tags-based construction even it was not the best in all the cases.

Another parameter that we need to figure out is the community time bin in TCNSVD model. Time bins identify the granularity which we need to consider the temporal community effects. Here,
three values of 1, 2 and four are employed which 4 indicates a finer granular analysis and 1 shows static community structures. The simulations are performed on ratings-based graph construction which is depicted in Figure 6.8 and on tags-based graph construction which is indicated in Figure 6.9. Regarding ratings-based graph construction, static community structures lead to lower RMSE and higher precision and recall values. In other words, CBins=1 generates better results, and thus the application of community drift does not improve the performance. However, regarding tags-based graph construction, TCNSVD could outperform with finer granular temporal bins that could achieve the lowest RMSE and highest Prec@10 and Rec@10. Thus, finer granular community temporal analysis may have better performance when we use tags-based graph construction.

Figure 6.4: This Figure indicates algorithms performance based on different $k$ values as the number of neighbors in the k-NN graph construction approach.

To discover the parameter space, we depicted the algorithm performance including RMSE, Prec@10, and Rec@10 versus number users and items per community. In other words, as for different parameters, we extract more information regarding the correlations among the precision and accuracy of item ranking and number of users and items per community. In this regard, we can observe the scattered plots in Figure 6.10. As we can see, here we also apply SLPA to obtain further implications regarding overlapping community detection algorithms. In this Figure, community detection algorithms including DMID, Walktrap (steps=2), Walkrap (steps=5) and SLPA are indicated with colors blue, red, brown and teal, respectively.

Moreover, similarity metrics including Cosine, Pearson, and JMSD are plotted with marker types of the square, triangle, and circle, respectively, and finally bigger marker sizes show more significant values for $k$. Furthermore, it can be observed that the number of users per community change from
Applications of Overlapping Communities

Figure 6.5: This Figure shows the algorithm performance using different similarity measures. Pearson correlation, cosine similarity, and Jaccard mean squared distance is applied to construct the graph using k-NN approach.

Table 6.8: This Table indicates the model parameters employed in evaluations. CBins identifies the community bins as for the TCNSVD model. Also, the steps parameter of the Walktrap algorithm is indicated with WT2 and WT5, respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>Graph Method</th>
<th>CD-Algo</th>
<th>k-NN Neighbors</th>
<th>Similarity</th>
<th>CBins</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNSVD</td>
<td>Ratings</td>
<td>WT5</td>
<td>20</td>
<td>Pearson</td>
<td>–</td>
</tr>
<tr>
<td>CNSVD</td>
<td>Tags</td>
<td>DMID</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>TCNSVD</td>
<td>Ratings</td>
<td>WT5</td>
<td>20</td>
<td>Pearson</td>
<td>1</td>
</tr>
<tr>
<td>TCNSVD</td>
<td>Tags</td>
<td>WT2</td>
<td>–</td>
<td>–</td>
<td>4</td>
</tr>
<tr>
<td>CNSVD-Fast</td>
<td>Ratings</td>
<td>WT2</td>
<td>5</td>
<td>Pearson</td>
<td>–</td>
</tr>
<tr>
<td>CNSVD-Fast</td>
<td>Tags</td>
<td>WT2</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>TCNSVD-Fast</td>
<td>Ratings</td>
<td>WT5</td>
<td>10</td>
<td>Cosine</td>
<td>–</td>
</tr>
<tr>
<td>TCNSVD-Fast</td>
<td>Tags</td>
<td>WT2</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

10 to 1700 and number of items per community range from 10 to 2300. If we consider DMID algorithm, we may notice that all of the blue points are concentrated almost near to one area. The precision of item ranking for DMID is rather low; however, observed stable results indicate that its performance does not depend on the selection of parameters. Regarding parameters with the best performance including Walktrap, Pearson or JMSD and k=20, the number of users per community is scattered; however, medium-sized and large number of objects per community are generated. In
6.2. PROPOSED COMMUNITY-AWARE BASELINE PREDICTOR FOR ITEM RECOMMENDER ALGORITHM

![Graphs showing performance metrics for community detection algorithms.]

Figure 6.6: This Figure shows the performance of community detection algorithms including DMID and Walktrap on rating-based constructed graphs. We used step parameters of 2 and five as for Walktrap (WT).

In this regard, we have observed large user communities ranging from 600 to 1100, small item communities ranging from 50 to 150 using JMSD and larger items per community even more than 2000.

The figure shows distributed data points along the $y$ axis as for SLPA algorithm which indicates a small number of users and items per community. In this regard, SLPA generates low and average performance in most of the cases and it achieves to the highest item ranking precision performance with JMSD. If we consider RMSE as for Walktrap, it can be observed that parameters do not affect performance when we consider items per community; however, there can be observed some better performing results with Pearson correlation. Regarding RMSE and user frequencies in each community, number of users per community are more scattered; however, the large number of users around 1000 generated better RMSE values with Pearson Correlation. Item ranking precision results with Prec@10 and Rec@10 are somewhat spread; however, we can observe high performance with JMSD and lower number of users per community. In this regard, Cosine similarity has the most inferior performance, and Pearson achieves average performance. Moreover, more significant number of neighbors in $k$-NN graph construction generates higher Prec@10 and Rec@10 as for Walktrap.

Up to this point, we analyzed the parameters of the algorithms and figured out the well-performing settings. Now, we intend to evaluate the algorithm performance compared to the baselines on MovieLens and Netflix datasets. In this regard, well performing parameter values are employed for this purpose - parameters in the Table 6.8. As for the benchmarks, we employed the NSVD and
Applications of Overlapping Communities

Figure 6.7: This Figure shows the performance of community detection algorithms including DMID and Walktrap on tags-based constructed graphs. We used step parameters of 2 and five as for Walktrap (WT).

Figure 6.8: This figure compares the performance of TCNSVD model using different community time bins. The results are based on the rating-based graph construction.

Figure 6.9: This figure compares the performance of TCNSVD model using different community time bins. The results are based on the tags-based graph construction.

TNSVD models [Kore08, Kore09]. CNSVD, CNSVD-Fast, TCNSVD and TCNSVD-Fast are the extensions to these models, and thus they appear to be suitable baselines. Furthermore, we also
6.2. PROPOSED COMMUNITY-AWARE BASELINE PREDICTOR FOR ITEM RECOMMENDER ALGORITHM

![Graphs showing RMSE, Prec@10, and Rec@10 for different average users and items per community.]

Figure 6.10: Each data point shows a different realization of the model parameters. Different colors represent different community detection algorithms, i.e., blue: DMID, red: Walktrap (steps=2), brown: Walktrap (steps=5), teal: SLPA. Similarity different marker types show metrics, i.e., square: cosine, triangle: Pearson, circle: JMSD. The size of the data points indicates the number of $k$-NN neighbors i.e. small: $k = 5$, medium: $k = 10$, large: $k = 20$.

employed the method proposed by Hu et al. named Weighted Regularized Matrix Factorization
Applications of Overlapping Communities

(WRMF) and Item k-Nearest-Neighbor Collaborative Filtering (ItemKNN) and ItemMean as for the baselines. In this regard, WRMF focuses on item ranking, and ItemMean computes the overall average rating for the targeted item.

First, we observe results in the Table 6.9 which indicates the rating prediction and the item ranking performance. On top of these tables, we can follow the metrics including MAE, RMSE, NDCG, Prec@10, and Rec@10. On the left side, we can see MovieLens and Netflix datasets, models and the graph construction including ratings-based and tags-based. Regarding MovieLens dataset, we can observe the superior rating prediction performance of ItemMean with around 0.72 and 0.94 values as for MAE and RMSE. Competitive models such as CNSVD based on tags (0.72650) and ratings (0.72788), TCNSVD-Fast based on ratings (0.72675) and NSVD (0.72444) follows ItemMean regarding MAE values. As for RMSE, the situation is similar to MAE, and NSVD, TCNSVD-Fast, and CNSVD are ranked after ItemMean.

To continue analyzing MovieLens regarding item ranking metrics, we can mention WRMF with highest performance values of 0.51121, 0.18846 and 0.0.02321 as for NDCG, Prec@10, and Rec@10, respectively. It is followed by ItemKNN with values of 0.50878, 0.16324 and 0.01432 regarding these metrics. If we consider precision, it is self-explanatory that TCNSVD achieves 0.08475 and 0.07183 as for ratings and tags-based construction. In fact, TCNSVD appeared to improve TNSVD model and also outperformed the baselines except for ItemKNN and WRMF. Although neighborhood and factor models are suitable for rating prediction, we can improve item ranking by employing community dimension when considering temporal effects. Regarding the Netflix dataset the situation is a bit different and CNSVD model could achieve the superior performance regarding MAE (0.80528) and RMSE (1.00641). Considering MAE, the followed up best performant algorithms are NSVD and CNSVD-Fast with 0.80659 and 0.81782 values. ItemMean is the next model with 0.84182 as for MAE. Regarding the item ranking and accuracy metrics, ItemKNN could not achieve the best performance for all the metrics. In fact, it reached 0.399929 as for NDCG which is the best among all the models; however, Prec@10 and Rec@10 reveal that TCNSVD could achieve the highest performance. In other words, TCNSVD obtained 0.18523 and 0.04392 as for Prec@10 and Rec@10 which is the best among others.

We can observe the superiority of temporal community effects on accuracy and item ranking metrics as for the TCNSVD-Fast models with 0.16453 and 0.03690 as for the precision and recall. What we can conclude from this table is the sheer positive effect of temporal community effects regarding accuracy and item ranking metrics. In some cases such as Netflix datasets, TCNSVD and TCNSVD-Fast models could defeat the best item ranking benchmarks such as WRMF. To obtain a fine-grained understanding of the community effect on the NSVD and TNSVD models, we refer to the Table 6.11. Similarly, on top of the table, you can see the rating prediction and item ranking evaluation metrics and on the left, you may look at the models, the respective baselines and the datasets. Regarding MovieLens dataset, the CNSVD achieves higher error values, i.e., +0.47 and +0.30 regarding MAE and RMSE; however, the situation changes as for accuracy and item ranking metrics. As for NDCG, Prec@10, and @Rec@10, CNSVD improves the basic NSVD model with around 1.65 %, 6.30 % and 14.61 %, respectively for each of the metrics.
6.2. PROPOSED COMMUNITY-AWARE BASELINE PREDICTOR FOR ITEM RECOMMENDER ALGORITHM

Similarly, CNSVD-Fast experiences a similar performance compared to CNSVD model. In other words, CNSVD made worse the rating prediction results; however, it improved accuracy and item ranking results. Still looking into MovieLens dataset, TCNSVD had lower MAE and RMSE performance compared to TNSVD; however, NDCG, Prec@10, and Rec@10 could be improved by 10.75%, 106.15% and 41.90%, respectively. The situation is much better regarding TCNSVD-Fast compared to TNSVD and it reduces MAE and RMSE errors 10.26% and 7.94%, respectively. Additionally, it also improves NDCG, Prec@10, and Rec@10 with 13.98%, 38.04% and 3.93%, respectively. Regarding Netflix dataset, the situation is a bit different, and here CNSVD could improve both the rating prediction error and accuracy. In other words, CNSVD surpasses NSVD with 0.16 and 0.03 percent improvement regarding MAE and RMSE. Moreover, it improves NDCG, Prec@10 and Rec@10 with respectively 2.64%, 46.25% and 55.15%. However, other models including CNSVD-Fast, TCNSVD, and TCNSVD-Fast could not achieve superior performance as for MAE and RMSE regarding Netflix dataset. On the contrary, these three models also improved item ranking precision with the maximum of 13.73%, 110.42% and 148.34% regarding NDCG, Prec@10 and Rec@10, respectively. Overall, TCNSVD-Fast and CNSVD could improve MAE, RMSE, NDCG, Prec@10 and Rec@10 regarding MovieLens and Netflix datasets. Moreover, we could observe the sheer improvement of item ranking performance compared to the baselines when community dimension is applied.

Finally, we investigate the running times of the models, shown in Table 6.10, to figure out the extent of improvements through CNSVD-Fast and TCNSVD-Fast models. While there might be some inconsistencies in the *init* time and the *learn* times, we only discuss regarding the total the CPU time here. These inconsistencies might be caused by the random initialization of the model parameters or waiting times to finish the snapshots. As we can observe, the fast models, CNSVD-Fast and TCNSVD-Fast could achieve higher performance as compared to CNSVD and TCNSVD models; however, they are unable to improve the running times compared to the baselines including NSVD and TNSVD. This observation is also true regarding the MovieLens dataset as for tags-based and ratings-based construction. There is only one exception, and it is the TCNSVD-Fast that cannot improve the TCNSVD model. To get a deeper understanding of the table, we first look at MovieLens dataset. In this regard, CNSVD-Fast obtains 2.635 and 3.850 which is faster than CNSVD with 21.200 and 12.110 minutes as for ratings and tags, respectively. Also, TCNSVD-Fast achieves 9.964 and 10.331 running time performance respectively regarding ratings and tags which is lower than TCNSVD with 13.003 and 80.836 minutes. However, they are far behind the NSVD and TNSVD with respectively 2.712 and 20.734 minutes.

Regarding Netflix dataset, the situation is quite similar with improvement from 8.562 to 4.935 which is corresponding to CNSVD and CNSVD-Fast; however, the exception is TNSVD-Fast with time deterioration from 12.310 (TNSVD) to 12.530 minutes. However, NSVD and TNSVD are still better with 2.503 and 6.220 minutes, respectively. Overall, we can mention that fast versions of algorithms could improve the proposed community-aware ones; however, they are still unable to improve the original models.
## Applications of Overlapping Communities

Table 6.9: This Table shows the evaluation results on MovieLens and Netflix datasets. Rating prediction and item ranking evaluation results are calculated and the best-performing metrics are made bold.

<table>
<thead>
<tr>
<th>DS</th>
<th>Model</th>
<th>Graph</th>
<th>MAE</th>
<th>RMSE</th>
<th>NDCG</th>
<th>Prec@10</th>
<th>Rec@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens</td>
<td>CNSVD</td>
<td>Ratings</td>
<td>0.72788</td>
<td>0.94548</td>
<td>0.41653</td>
<td>0.04403</td>
<td>0.00698</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tags</td>
<td>0.72650</td>
<td>0.94575</td>
<td>0.39887</td>
<td>0.03240</td>
<td>0.00480</td>
</tr>
<tr>
<td></td>
<td>TCNSVD</td>
<td>Ratings</td>
<td>0.81985</td>
<td>1.05163</td>
<td>0.41208</td>
<td>0.08475</td>
<td>0.01192</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tags</td>
<td>0.81727</td>
<td>1.05332</td>
<td>0.39872</td>
<td>0.07183</td>
<td>0.00994</td>
</tr>
<tr>
<td></td>
<td>CNSVD-Fast</td>
<td>Ratings</td>
<td>0.77181</td>
<td>0.99400</td>
<td>0.41754</td>
<td>0.06142</td>
<td>0.01009</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tags</td>
<td>0.74403</td>
<td>0.96785</td>
<td>0.42895</td>
<td>0.08283</td>
<td>0.01222</td>
</tr>
<tr>
<td></td>
<td>TCNSVD-Fast</td>
<td>Ratings</td>
<td>0.72675</td>
<td>0.94991</td>
<td>0.42408</td>
<td>0.05675</td>
<td>0.00873</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tags</td>
<td>0.74431</td>
<td>0.96875</td>
<td>0.41498</td>
<td>0.06920</td>
<td>0.01024</td>
</tr>
<tr>
<td></td>
<td>NSVD</td>
<td>–</td>
<td>0.72444</td>
<td>0.94262</td>
<td>0.40978</td>
<td>0.04142</td>
<td>0.00609</td>
</tr>
<tr>
<td></td>
<td>TNSVD</td>
<td>–</td>
<td>0.80981</td>
<td>1.03186</td>
<td>0.37207</td>
<td>0.04111</td>
<td>0.00840</td>
</tr>
<tr>
<td></td>
<td>WRMF</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td><strong>0.51121</strong></td>
<td><strong>0.18846</strong></td>
</tr>
<tr>
<td></td>
<td>ItemKNN</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>(0.77465)</td>
<td>(1.01116)</td>
</tr>
<tr>
<td></td>
<td>ItemMean</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td><strong>0.71458</strong></td>
<td><strong>0.93944</strong></td>
</tr>
<tr>
<td>Netflix</td>
<td>CNSVD</td>
<td>Ratings</td>
<td><strong>0.80528</strong></td>
<td><strong>1.00641</strong></td>
<td>0.34712</td>
<td>0.04060</td>
<td>0.00934</td>
</tr>
<tr>
<td></td>
<td>TCNSVD</td>
<td>Ratings</td>
<td>1.08952</td>
<td>1.38567</td>
<td>0.39885</td>
<td><strong>0.18523</strong></td>
<td><strong>0.04392</strong></td>
</tr>
<tr>
<td></td>
<td>CNSVD-Fast</td>
<td>Ratings</td>
<td>0.81782</td>
<td>1.01349</td>
<td>0.35650</td>
<td>0.07281</td>
<td>0.01495</td>
</tr>
<tr>
<td></td>
<td>TCNSVD-Fast</td>
<td>Ratings</td>
<td>0.88453</td>
<td>1.10569</td>
<td>0.39236</td>
<td>0.16453</td>
<td>0.03690</td>
</tr>
<tr>
<td></td>
<td>NSVD</td>
<td>–</td>
<td>0.80659</td>
<td>1.00675</td>
<td>0.33820</td>
<td>0.02776</td>
<td>0.00602</td>
</tr>
<tr>
<td></td>
<td>TNSVD</td>
<td>–</td>
<td>0.88183</td>
<td>1.08233</td>
<td>0.35069</td>
<td>0.08803</td>
<td>0.02174</td>
</tr>
<tr>
<td></td>
<td>WRMF</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td><strong>0.39929</strong></td>
<td>0.09293</td>
</tr>
<tr>
<td></td>
<td>ItemKNN</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>(0.86217)</td>
<td>(1.06189)</td>
</tr>
<tr>
<td></td>
<td>ItemMean</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td><strong>0.84182</strong></td>
<td><strong>1.05581</strong></td>
</tr>
</tbody>
</table>
### 6.2. PROPOSED COMMUNITY-AWARE BASELINE PREDICTOR FOR ITEM RECOMMENDER ALGORITHM

Table 6.10: This table shows the running times of the models on MovieLens and Netflix datasets. We reported the running times in minutes.

<table>
<thead>
<tr>
<th>DS</th>
<th>Model</th>
<th>Graph</th>
<th>Init</th>
<th>Learn</th>
<th>Test</th>
<th>Sum</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens</td>
<td>CNSVD Ratings</td>
<td>89</td>
<td>3,663</td>
<td>112</td>
<td>3,864</td>
<td>21,200</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNSVD Tags</td>
<td>69</td>
<td>1,967</td>
<td>115</td>
<td>2,151</td>
<td>12,110</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCNSVD Ratings</td>
<td>48</td>
<td>1,762</td>
<td>514</td>
<td>2,324</td>
<td>13,003</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCNSVD Tags</td>
<td>131</td>
<td>4,063</td>
<td>143</td>
<td>4,337</td>
<td>80,836</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNSVD-Fast Ratings</td>
<td>18</td>
<td>424</td>
<td>21</td>
<td>463</td>
<td>2,635</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNSVD-Fast Tags</td>
<td>24</td>
<td>651</td>
<td>23</td>
<td>698</td>
<td>3,850</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCNSVD-Fast Ratings</td>
<td>39</td>
<td>1,756</td>
<td>76</td>
<td>1,871</td>
<td>9,964</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCNSVD-Fast Tags</td>
<td>39</td>
<td>1,740</td>
<td>74</td>
<td>1,853</td>
<td>10,331</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NSVD – Ratings</td>
<td>24</td>
<td>494</td>
<td>14</td>
<td>532</td>
<td>2,712</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NSVD – Tags</td>
<td>31</td>
<td>2,303</td>
<td>64</td>
<td>2,398</td>
<td>20,734</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TNSVD – Ratings</td>
<td>0</td>
<td>119</td>
<td>1</td>
<td>120</td>
<td>620</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TNSVD – Tags</td>
<td>(29)</td>
<td>(0)</td>
<td>(4,890)</td>
<td>(4,919)</td>
<td>(30,397)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WRMF – Ratings</td>
<td>–</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WRMF – Tags</td>
<td>(4)</td>
<td>(0)</td>
<td>(3,308)</td>
<td>(3,312)</td>
<td>(29,017)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ItemKNN – Ratings</td>
<td>–</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ItemKNN – Tags</td>
<td>(29)</td>
<td>(0)</td>
<td>(4,890)</td>
<td>(4,919)</td>
<td>(30,397)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ItemMean –</td>
<td>–</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Netflix</td>
<td>CNSVD Ratings</td>
<td>41</td>
<td>818</td>
<td>480</td>
<td>1,339</td>
<td>8,562</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNSVD Tags</td>
<td>72</td>
<td>1,796</td>
<td>452</td>
<td>2,320</td>
<td>12,310</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNSVD-Fast Ratings</td>
<td>13</td>
<td>688</td>
<td>63</td>
<td>764</td>
<td>4,935</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNSVD-Fast Tags</td>
<td>28</td>
<td>2,078</td>
<td>275</td>
<td>2,381</td>
<td>12,530</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NSVD – Ratings</td>
<td>15</td>
<td>417</td>
<td>61</td>
<td>493</td>
<td>2,503</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NSVD – Tags</td>
<td>15</td>
<td>871</td>
<td>763</td>
<td>1,649</td>
<td>6,220</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TNSVD – Ratings</td>
<td>3</td>
<td>73</td>
<td>1</td>
<td>77</td>
<td>411</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TNSVD – Tags</td>
<td>(4)</td>
<td>(0)</td>
<td>(3,308)</td>
<td>(3,312)</td>
<td>(29,017)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ItemKNN – Ratings</td>
<td>–</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ItemKNN – Tags</td>
<td>(4)</td>
<td>(0)</td>
<td>(3,308)</td>
<td>(3,312)</td>
<td>(29,017)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ItemMean –</td>
<td>–</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.11: This Table compares the performance achieved by the community-aware models with the acquired performance of the baseline models including NSVD and TNSVD as for MovieLens and Netflix datasets.

<table>
<thead>
<tr>
<th>DS</th>
<th>Model</th>
<th>Basis</th>
<th>∆ MAE</th>
<th>∆ RMSE</th>
<th>∆ NDCG</th>
<th>∆ Prec@10</th>
<th>∆ Rec@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens</td>
<td>CNSVD</td>
<td>NSVD</td>
<td>+0.47%</td>
<td>+0.30%</td>
<td>+1.65%</td>
<td>+6.30%</td>
<td>+14.61%</td>
</tr>
<tr>
<td></td>
<td>CNSVD-Fast</td>
<td>NSVD</td>
<td>+6.54%</td>
<td>+5.45%</td>
<td>+1.89%</td>
<td>+48.29%</td>
<td>+65.68%</td>
</tr>
<tr>
<td></td>
<td>TCNSVD</td>
<td>TNSVD</td>
<td>+1.24%</td>
<td>+1.92%</td>
<td>+10.75%</td>
<td>+106.15%</td>
<td>+41.90%</td>
</tr>
<tr>
<td></td>
<td>TCNSVD-Fast</td>
<td>TNSVD</td>
<td>-10.26%</td>
<td>-7.94%</td>
<td>+13.98%</td>
<td>+38.04%</td>
<td>+3.93%</td>
</tr>
<tr>
<td>Netflix</td>
<td>CNSVD</td>
<td>NSVD</td>
<td>-0.16%</td>
<td>-0.03%</td>
<td>+2.64%</td>
<td>+46.25%</td>
<td>+55.15%</td>
</tr>
<tr>
<td></td>
<td>CNSVD-Fast</td>
<td>NSVD</td>
<td>+1.39%</td>
<td>+0.67%</td>
<td>+5.41%</td>
<td>+162.28%</td>
<td>+148.34%</td>
</tr>
<tr>
<td></td>
<td>TCNSVD</td>
<td>TNSVD</td>
<td>+23.55%</td>
<td>+28.03%</td>
<td>+13.73%</td>
<td>+110.42%</td>
<td>+102.02%</td>
</tr>
<tr>
<td></td>
<td>TCNSVD-Fast</td>
<td>TNSVD</td>
<td>+0.31%</td>
<td>+2.16%</td>
<td>+11.88%</td>
<td>+86.90%</td>
<td>+69.73%</td>
</tr>
</tbody>
</table>
### Table 6.12: An overview of the symbols used in the expert identification models’ formulation.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Ex_i$</td>
<td>Expertise value of node $i$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Adjusts the effect of inside and outside connections</td>
</tr>
<tr>
<td>$C_r$</td>
<td>Sample cover identified by any OCD algorithm</td>
</tr>
<tr>
<td>$a_i$</td>
<td>Authority value of node $i$</td>
</tr>
<tr>
<td>$h_i$</td>
<td>Hub value of node $i$</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Damping factor</td>
</tr>
<tr>
<td>$PR_i$</td>
<td>PageRank value of node $i$</td>
</tr>
<tr>
<td>$Outdeg_{intra}(j)$</td>
<td>Nodes which node $j$ refers to them and in the same community as node $j$ is</td>
</tr>
<tr>
<td>$Outdeg_{extra}(j)$</td>
<td>Nodes which node $j$ refers to them but in different community from node $j$ is</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of nodes in the network</td>
</tr>
</tbody>
</table>
6.3 Expert Identification

Classical ranking algorithms and their variations have been employed to identify experts. Their idea behind most of the algorithms is to construct a network among askers and answerers and apply HITS and PageRank (for ranking algorithms, see chapter 2). In this section, we propose community-aware ranking algorithms and use them to identify experts. We compare them with classical ranking algorithms such as HITS and PageRank. Afterwards, we introduce the datasets employed for the expert identification algorithms. Finally, we discuss the results. The prototypical implementation of this part of the research was supported by a master thesis guided by the author of this work [Pare15].

List of symbols we shall use for community-aware ranking algorithms are denoted in Table 6.17. We use the final rank value of node \( i \) as its expertise value, i.e., \( Ex_i \)

6.3.1 Proposed Community-Aware Ranking Algorithms for Expert Finding

To identify experts in our datasets, we suggest community-aware ranking methods based on classical HITS and PageRank. Community-aware concept values more to the shared knowledge among people in the same community than to the people in different communities. In other words, people working in the same cluster are more likely to have more commitment towards their shared interests.

Overlapping Community-Aware HITS

Similar to the classical HITS algorithm, we contemplate about applying hubs and authority vectors. Hubs are vectors pointing to some other nodes, on the contrary, authorities receive ranks. We initialize the hub and authority (Ex) vectors with \( \frac{1}{|N|} \) and update them as follows:

\[
\begin{align*}
  a_i &= \alpha \times \sum_{j,i \in C, j \in E} h_j + (1 - \alpha) \times \sum_{j,i \notin C, j \in E} h_j \\
  h_i &= \alpha \times \sum_{j,i \in C, j \in E} a_j + (1 - \alpha) \times \sum_{j,i \notin C, j \in E} a_j,
\end{align*}
\]

(6.42)

Where \( h_i \) and \( Ex_i \) are the hub and authority values for node \( i \). \( C_r \) is a sample cover identified by any OCD algorithm - DMD in this case. The parameter \( \alpha \) adjusts the effect of inside and outside connections. Finally, convergent \( Ex_i^* \) is considered to be the expertise value of node \( i \).
Overlapping Community-Aware PageRank

Similar to PageRank algorithm, we perform a random walk on the community-augmented graph. If we consider the expertise value of a node $i$ as $Ex_i$, this vector is initialized by $\frac{1}{|N|}$. The algorithm will repeatedly update it until convergence based on the following formula:

$$PR_i = \beta \times \left( \alpha \times \sum_{j, i \in C_r \cap j, i \in E_{ji}} \left( \frac{PR_j}{\text{Outdeg}_{\text{intra}}(j)} \right) + (1 - \alpha) \times \sum_{j, i \notin C_r \cap j \notin E_{ji}} \left( \frac{PR_j}{\text{Outdeg}_{\text{extra}}(j)} \right) + (1 - \beta) \times \left( \frac{1}{N} \right) \right).$$

(6.43)

Here $\alpha$ determines to what extent communities affect the random walk. Walks will be diverted towards inside and on the boards of communities when $\alpha$ is bigger than 0.5. $\beta$ is the teleporting parameter to avoid dead ends. Like to the previous case, we can use any OCD algorithm - DMID is the main one.

6.3.2 Evaluation Results

We applied Overlapping Community-Aware HITS DMID (OCAHD), Overlapping Community-Aware HITS SLPA (OCAHS), Overlapping Community-Aware PageRank DMID (OCAPD), Overlapping Community-Aware PageRank SLPA (OCAPS), HITS and PageRank (PR) on three question & answer forums including Fitness, Nature, and computer science forums.

Datasets

Stack exchange is a question and answer forum dedicated to discussing issues in particular domains. The topics range from social, political, technical and health issues. Physical fitness and computer science forums are among two topics that people share their opinions and innovations. In healthcare forum, people ask for health-related topics. In contrast, people target computer science forum for technical and theoretical issues in computer science domain. These two different contexts might cause some differences in research analysis. In addition to stack exchange forum, we received some data related to Nature. This data contains topics and discussions regarding the wildlife in Estonia. Information about these datasets including the number of posts, users, questions, answers and Table indicates the period 6.13.

Accuracy and Recall Values

To evaluate the ranking models for expert identification, MRR and MAP (see chapter 2) are computed. Figure 6.11 shows the MRR and MAP values for approximately a two year period from

---

3http://stackexchange.com/sites
4http://www.looduskalender.ee/forum/
### 6.3. EXPERT IDENTIFICATION

<table>
<thead>
<tr>
<th>Forum</th>
<th>Posts</th>
<th>Users</th>
<th>Questions</th>
<th>Answers</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Fitness</td>
<td>11522</td>
<td>7567</td>
<td>3602</td>
<td>7735</td>
<td>03.01.2011 - 03.05.2014</td>
</tr>
<tr>
<td>Computer Science</td>
<td>21731</td>
<td>22264</td>
<td>8955</td>
<td>12415</td>
<td>25.11.2008 - 08.03.2015</td>
</tr>
<tr>
<td>Nature</td>
<td>162325</td>
<td>1370</td>
<td>227</td>
<td>162098</td>
<td>01.01.2013 - 10.02.2015</td>
</tr>
</tbody>
</table>

Table 6.13: The number of posts, users, questions, answers and the period of stack exchange and Nature forums.

2013 to 2015. Community-aware ranking algorithms outperform for the two measures in comparison to its counterparts. Regarding MRR values, from 3.1.2013 to 7.1.2014, OCAHD has the best performance in comparison to others. After 7.1.2014, OCAHD and OCAHS have the superior performance concerning MAP values. It shows that community-aware ranking algorithms outperform others.

As for MRR, the situation is a bit different and OCAHS and HITS care about the position of the relevant items. We observe this dominance for approximately all of the time slots. Besides, in the beginning, approximately all of the algorithms face the cold start problem and are less able to generate highly accurate results.

![Figure 6.11](image1)

**Figure 6.11:** This figure shows the MAP and MRR values for Nature forum.

![Figure 6.12](image2)

**Figure 6.12:** This figure indicates the MAP and MRR values for Fitness forum.

To continue regarding the MRR and MAP values, we can look at Figure 6.12. The performance of the algorithms are quite similar in this dataset all the algorithms start from approximately low MAP.
values around 30% and reach MAP values of more than 67%. Respectively, OCAPD, OCAPS, PR, OCAHD, HITS and OCAHS obtain low precision levels. After 06.01.2012, they all reach more than 0.9. MAP and MRR values on Fitness forum also confirm the superiority of overlapping community aware algorithms. If one has a look at Figure 6.13, she can catch a similar understanding about the positive performance of community information on expert identification task.

Regarding MRR, OCAPD and OCAPS win over the other algorithms. For instance, at 06.01.2014, OCAPD and OCAPS obtain 93.83 and 92.49 respectively, which is better than PR (91.39), HITS (90.65), OCAHD (89.33) and OCAHS (87.74). Similarly, overlapping community-aware ranking algorithms yield better results in comparison to classical HITS and PR values regarding MRR measure.

**Spearman Correlation Values**

To figure out how the original HITS and PageRank algorithms correlate with community-aware ranking algorithms, we compute the Pearson correlation of these algorithms. Table 6.14 shows the correlation values for computer science forum. The bold values indicate higher values of correlation. As we can observe the PR, OCAPD and OCAPS have quite high similarity values ($\approx 0.999$). Classical HITS and community-aware HITS have quite high correlation values among themselves. On the contrary, HITS related algorithms have a lower correlation with PageRank related algorithms, for instance, OCAHD-PR (0.965), OCAHD-OCAPD (0.965). Table 6.16 also indicates very similar results to computer science forum with a bit lower cross-correlation among algorithms OCAHD-OCAPS (0.898), OCAHS-OCAPS (0.897), OCAHD-OCAPD (0.904), OCAHD-PR (0.902), OCAHS-PR (0.901), OCAHS-OCAPD (0.901) and so on. The correlation results are a bit different from Nature forum and cross-correlation values among HITS related and PageRank related algorithms slump. It is easily observable that PageRank related algorithms and HITS related algorithms have very much similar correlation among themselves. Moreover, HITS-PR (0.543), HITS-OCAPD (0.521), HITS-OCAPS (0.519), OCAHD-PR (0.558), OCAHD-OCAPD (0.537), OCAHD-OCAPS (0.534), OCAHS-PR (0.535), OCAHS-OCAPD (0.512) and OCAHS-OCAPS (0.513) obtain correlation values of $\approx 0.5$ which is much lower than the other two datasets. This observation indicates that similarity of algorithms behavior depends on the nature and structure of datasets.
6.4. COOPERATION AND DEFECTION: INDIVIDUAL AND COLLECTIVE PROPERTIES

Table 6.14: Pearson correlation among PageRank, HITS and community-aware HITS and PageRank algorithms on Computer Science forum dataset.

<table>
<thead>
<tr>
<th>dataset</th>
<th>PR</th>
<th>CA-PR-DMID</th>
<th>CA-PR-SLPA</th>
<th>HITS</th>
<th>CA-HITS-DMID</th>
<th>CA-HITS-SLPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR</td>
<td>1.000</td>
<td>0.999</td>
<td>0.999</td>
<td>0.964</td>
<td>0.965</td>
<td>0.963</td>
</tr>
<tr>
<td>CA-PR-DMID</td>
<td>0.999</td>
<td>1.000</td>
<td>0.999</td>
<td>0.964</td>
<td>0.965</td>
<td>0.963</td>
</tr>
<tr>
<td>CA-PR-SLPA</td>
<td>0.999</td>
<td>0.999</td>
<td>1.000</td>
<td>0.964</td>
<td>0.965</td>
<td>0.963</td>
</tr>
<tr>
<td>HITS</td>
<td>0.964</td>
<td>0.964</td>
<td>0.964</td>
<td>1.000</td>
<td>0.999</td>
<td>0.999</td>
</tr>
<tr>
<td>CA-HITS-DMID</td>
<td>0.965</td>
<td>0.965</td>
<td>0.965</td>
<td>0.999</td>
<td>1.000</td>
<td>0.999</td>
</tr>
<tr>
<td>CA-HITS-SLPA</td>
<td>0.963</td>
<td>0.963</td>
<td>0.963</td>
<td>0.999</td>
<td>0.999</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table 6.15: Pearson correlation among PageRank, HITS and community-aware HITS and PageRank algorithms on Nature forum dataset.

<table>
<thead>
<tr>
<th>dataset</th>
<th>PR</th>
<th>CA-PR-DMID</th>
<th>CA-PR-SLPA</th>
<th>HITS</th>
<th>CA-HITS-DMID</th>
<th>CA-HITS-SLPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR</td>
<td>1.000</td>
<td>0.997</td>
<td>0.990</td>
<td>0.543</td>
<td>0.558</td>
<td>0.535</td>
</tr>
<tr>
<td>CA-PR-DMID</td>
<td>0.997</td>
<td>1.000</td>
<td>0.993</td>
<td>0.521</td>
<td>0.537</td>
<td>0.512</td>
</tr>
<tr>
<td>CA-PR-SLPA</td>
<td>0.990</td>
<td>0.993</td>
<td>1.000</td>
<td>0.519</td>
<td>0.534</td>
<td>0.513</td>
</tr>
<tr>
<td>HITS</td>
<td>0.543</td>
<td>0.521</td>
<td>0.519</td>
<td>1.000</td>
<td>0.996</td>
<td>0.994</td>
</tr>
<tr>
<td>CA-HITS-DMID</td>
<td>0.558</td>
<td>0.537</td>
<td>0.534</td>
<td>0.996</td>
<td>1.000</td>
<td>0.990</td>
</tr>
<tr>
<td>CA-HITS-SLPA</td>
<td>0.535</td>
<td>0.512</td>
<td>0.513</td>
<td>0.994</td>
<td>0.990</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table 6.16: Pearson correlation among PageRank, HITS and community-aware HITS and PageRank algorithms on Fitness forum dataset.

<table>
<thead>
<tr>
<th>dataset</th>
<th>PR</th>
<th>CA-PR-DMID</th>
<th>CA-PR-SLPA</th>
<th>HITS</th>
<th>CA-HITS-DMID</th>
<th>CA-HITS-SLPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR</td>
<td>1.000</td>
<td>0.999</td>
<td>0.998</td>
<td>0.901</td>
<td>0.902</td>
<td>0.900</td>
</tr>
<tr>
<td>CA-PR-DMID</td>
<td>0.999</td>
<td>1.000</td>
<td>0.996</td>
<td>0.901</td>
<td>0.904</td>
<td>0.901</td>
</tr>
<tr>
<td>CA-PR-SLPA</td>
<td>0.998</td>
<td>0.996</td>
<td>1.000</td>
<td>0.897</td>
<td>0.898</td>
<td>0.897</td>
</tr>
<tr>
<td>HITS</td>
<td>0.901</td>
<td>0.901</td>
<td>0.897</td>
<td>1.000</td>
<td>0.999</td>
<td>0.997</td>
</tr>
<tr>
<td>CA-HITS-DMID</td>
<td>0.902</td>
<td>0.904</td>
<td>0.898</td>
<td>0.999</td>
<td>1.000</td>
<td>0.996</td>
</tr>
<tr>
<td>CA-HITS-SLPA</td>
<td>0.901</td>
<td>0.901</td>
<td>0.897</td>
<td>0.997</td>
<td>0.996</td>
<td>1.000</td>
</tr>
</tbody>
</table>

6.4 Cooperation and Defection: Individual and Collective Properties

Cooperation and defection problem has been applied to study correlation among motif frequencies and cooperativity level of motifs [SRJa10]. The similar problem is to compute the correlation between rank of nodes and their cooperativity level in complex networks. Moreover, we can also investigate cooperativity of overlapping communities and observe to what extent it depends on structural (overlapping) community properties. We can apply this investigation to several domains. For instance, it is essential to evaluate and predict cooperativity of experts in learning environments, or to estimate the probability of joining a developer to an open source software development project based on its rank; however, we require a reliable approach to rank the objects in complex networks. Also, we draw conclusions regarding community properties and their cooperativity levels. This issue becomes very important based on the context of a community. For instance, in a learning forum amount of cooperativity in different communities may play an essential role in learning rates of individuals.
To investigate cooperation and detection, we investigate two separate simulations. The first experiment shows the relation between cooperativity of nodes and their rank values. The latter, on the other hand, observes cooperativity of (overlapping) communities in different contexts.

Table 6.18: Payoff matrix for the basic PD game.

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>(R;R)</td>
<td>(S;T)</td>
</tr>
<tr>
<td>D</td>
<td>(T;S)</td>
<td>(P;P)</td>
</tr>
</tbody>
</table>

To investigate the correlation of cooperativity of nodes and ranking of nodes, we implemented the PD game adopted from Nowak and May [NoMa92, SRJa10]. Furthermore, we implemented classical centrality and rank strategies including PageRank, HITS, Betweenness (BW), Closeness (CL), Clustering Coefficients (CC) and Simple Degree (SD) (see chapter 2). Afterwards, three strategies are employed to evaluate the amount of correlation between cooperation and rank problem. As such, we first investigated the correlation of node ranks with cooperativity levels of nodes through Hamming Distance. In other words, each assigned rank value is mapped to a binary value and then compared with the binary cooperativity value obtained through PD game. In the second strategy, we considered the neighborhood of each node and computed the average real rank values of the node neighbors. Moreover, we can calculate the average cooperativity of the node neighborhood. Afterwards, we correlate these two vectors for the nodes through relative entropy. Finally, we did the same experiment on the neighborhood of nodes; however, the variance of rank values show correlation with the variance of cooperativity levels of the neighborhoods. In the second part of our simulations, we use prisoner’s dilemma based on replicator dynamic different temptation to defect parameter to investigate cooperativity of overlapping communities. On the one hand, we first detect
Table 6.19: Information regarding the datasets used for cooperation and defection simulation results.

<table>
<thead>
<tr>
<th>dataset</th>
<th># nodes</th>
<th># edges</th>
<th>avg</th>
<th>std</th>
<th>CC</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zachary</td>
<td>34</td>
<td>156</td>
<td>4.59</td>
<td>15.04</td>
<td>0.31</td>
<td>Social</td>
</tr>
<tr>
<td>Swamill</td>
<td>36</td>
<td>124</td>
<td>3.44</td>
<td>4.77</td>
<td>0.31</td>
<td>Social</td>
</tr>
<tr>
<td>Swamill Strike</td>
<td>24</td>
<td>76</td>
<td>3.17</td>
<td>1.88</td>
<td>0.4421</td>
<td>Social</td>
</tr>
<tr>
<td>Dolphin</td>
<td>62</td>
<td>318</td>
<td>5.13</td>
<td>8.74</td>
<td>0.26</td>
<td>Animal</td>
</tr>
<tr>
<td>Gnutella</td>
<td>6301</td>
<td>41554</td>
<td>6.6</td>
<td>72.94</td>
<td>0.01</td>
<td>Technological</td>
</tr>
<tr>
<td>Email</td>
<td>1133</td>
<td>10903</td>
<td>9.62</td>
<td>87.28</td>
<td>0.22</td>
<td>Communication</td>
</tr>
<tr>
<td>Netscience</td>
<td>1589</td>
<td>5485</td>
<td>3.45</td>
<td>12.04</td>
<td>0.64</td>
<td>Coauthorship</td>
</tr>
<tr>
<td>Protein</td>
<td>1706</td>
<td>6346</td>
<td>3.74</td>
<td>48.83</td>
<td>0.0059</td>
<td>Biological</td>
</tr>
<tr>
<td>Euro Roads</td>
<td>1174</td>
<td>2834</td>
<td>2.41</td>
<td>1.41</td>
<td>0.02</td>
<td>Infrastructure</td>
</tr>
</tbody>
</table>

the community structures out of various networks - here we used learning and open source software development forums. Moreover, we compute properties of detected (overlapping) communities such as size, density, average degree, etc., by using community detection algorithms such as SSK, SLPA, CLiZZ and so on. On the other hand, we calculate the cooperativity of every community with evolutionary games. In the continuation, we visualize and correlate cooperativity of community structures versus their respective properties. Our results show a lower amount of cooperation in forums of learning communities compared to open source software development communities. We also visualized the distribution of community properties versus their cooperativity levels.

In this sub-chapter, we first describe the cooperation and defection problem. Afterwards, we introduce the evolutionary cooperation process. Next, we discuss how to compute correlation among cooperativity values of nodes and their rank values.

### 6.4.1 Cooperation and Defection

Network researchers have been interested in understanding the dynamics of cooperation and selfish behaviors of individuals and groups. We may not separate interests and benefits of groups from individuals and vice versa. In other words, the cooperation of one person with others has reciprocal effects that need to be studied. In fact, the pursuit of selfish agents’ behavior might be detrimental for the whole community. These issues have been modeled with game theory approaches such as Prisoner’s Dilemma (PD) [RAB*14, RBH*14]. Merrill M. Flood and Melvin Dresher developed this game in around 1950’s that has been under much investigation in computational social and computer sciences. There are some variations to this game; however, the original game which is common is a two player game that each agent can take two strategies; either cooperation or non-cooperation. Non-cooperation is also known as defection. The underlying mindset behind this problem is that cooperation of both agents leads to higher lucrative outcomes while defection increases individual benefits [SpaPe05].

The original PD game is the story of two persons imprisoned which the prosecutor cannot convict them based on the existing evidence. Here the court specifies a bonus for the pairs, and they can behave as follows: both prisoners would serve one year in jail if both stay silent. If one of them betrays, he will be free, and the other must serve three years in prison and vice versa. Finally, if
Applications of Overlapping Communities

Figure 6.14: This figure shows the Hamming Distance of binary node ranks and their cooperativity levels for Dolphin, Zachary, Swamill, Science networks. All of the algorithms reach a stable cooperativity level after a few number of iterations.

Both of them betray (defect), they require serving two years in jail. Rationally speaking, pairwise cooperation helps more beneficial outcomes to both players and nonreciprocal defection is useful for each of them [EaK10]. To simulate the problem, we consider the PD game with a more formal definition. We investigate an evolutionary match played on a network of interactions among individuals or players. Each player can take a set of rational behaviors; named strategies. Here, we mention Cooperation (C) and Defection (D) as possible ones. Based on the selected strategy, she will obtain a payoff; higher payoff values are preferred. We denote the payoff by a matrix which Table 6.18 shows it. Elements of the payoff matrix are defined as follows:

- If both players cooperate then they both receive a payoff $R$.
- If one cooperates and the other defects then the later receives a $T$ and the former obtains $S$.
- If both players defect then $P$ is granted to both.
6.4. COOPERATION AND DEFECTION: INDIVIDUAL AND COLLECTIVE PROPERTIES

Figure 6.15: This figure shows the Hamming Distance of binary node ranks and their cooperativity levels for peer-to-peer, euro road, protein interaction and email networks. Rank values based on the HITS algorithm have the shortest distance to cooperativity vector.

Each agent plays with its neighbors in the network and receives the corresponding payoff (fitness). With $T > R > P > S$, the non-cooperation dynamic prevails and reaches a Nash Equilibrium. In contrast, the reciprocal cooperation of individuals prevails unilateral cooperation, and thus mutual cooperation generates higher payoff [Gint09].

**Evolutionary Cooperation Process**

In general, cooperative ($#C$) level of an evolutionary game on a network of agents interacting through PD game is the final number of cooperative agents. As we want to calculate the correlation between node cooperativity and its rank, we require knowing cooperation status of each agent. In this regard, we assign a binary variable as the cooperation status of an agent; 1 denotes cooperation, and 0 shows defection. We start the game with an equal number of cooperators and defectors.
Afterwards, in each evolutionary step of the PD game, each agent plays with its neighbors and its payoff value is updated based on the payoff matrix introduced in Table 6.18. We adopt the parameters for the payoff matrix proposed by Nowak and May, and thus we consider $T = b > 1, P = S = 0, R = 1$ which $T$ is the propensity to defect [NoMa92]. Synchronously, players update their payoffs; however, through the evolutionary steps of the game, players change their strategies. In other words, each player randomly looks at one of its neighbors and change its strategy with a probability given by the equation 6.44 only if its payoff is less than of its neighbor. In fact, regarding node $i$, it compares its payoff with one random neighbor $j$ and changes its strategy with the following probability:

$$P_{i \Rightarrow j} = \frac{PO_j - PO_i}{b \times \text{maximum}(\#d_i, \#d_j)}.$$ (6.44)

The pseudo code better shows the process.

**Algorithm 1** Evolutionary Prisoner’s Dilemma Game on Complex Networks

1: $T \leftarrow b$
2: $P, S \leftarrow 0$
3: $R \leftarrow 1$
4: $\#\text{Cooperators} \leftarrow N/2$
5: $\#\text{Defectors} \leftarrow N/2$
6: while Repetition of the Evolutionary PD Game $\leq$ Threshold do
7:     while Iterations $\leq$ Time Window do
8:         for each node $i$ in the set of all nodes do
9:             for each node $j$ as neighbor of node $i$ do
10:                if $\text{strategy}(i) == C, \text{strategy}(j)==C$ then
11:                   payoffNew($i$) $\leftarrow$ payoffOld($i$)+$R$
12:                end if
13:                if $\text{strategy}(i)==D, \text{strategy}(j)==C$ then
14:                   payoffNew($i$) $\leftarrow$ payoffOld($i$)+$T$
15:                end if
16:         end for
17:         $j \leftarrow$ random neighbor of node $i$
18:         if payoff($i$) $\leq$ payoff($j$) then
19:             strategy($i$) $\leftarrow$ strategy($j$) with $P_{i \Rightarrow j} = \frac{PO_j - PO_i}{b \times \text{maximum}(\#d_i, \#d_j)}$.
20:         end if
21:     end for
22: end while
23: Average over the game realizations
24: end whilereturn Cooperativity Level
6.4. COOPERATION AND DEFECTION: INDIVIDUAL AND COLLECTIVE PROPERTIES

6.4.2 Ranking and Cooperation

In this section, we present our experiments to investigate cooperativity of nodes and their rank values.

Node Rank and Cooperation

To compute the cooperativity level of each node, we consider binary values for each of the ranking algorithms introduced in section 2. After normalizing each ranking vector $R$ to a real value between 0 and 1, it is mapped to binary values by considering a threshold value equal to mean value of the rank vector $R$. For each of the time windows, we correlate the binary rank vector $R$ with binary cooperation vector $C$. The correlation is merely the Hamming Distance which is the number of binary elements that needs to be changed to reach from binary rank vector to cooperation vector.

Neighbor Rank and Cooperativity

To observe how cooperativity of a node is related to its rank value, we consider the nodes’ neighbors. In other words for node $i$, we compute the average of rank values of neighbors of node $i$ as follows:

$$R_{avg}^{i} = \frac{\sum_{i=1}^{\#d_i} R_i}{\#d_i},$$  \hspace{1cm} (6.45)

where $R_{avg}^{i}$ is the average rank value of node $i$. Similarly, variance of rank values of neighbors of node $i$ can be computed as follows:

$$R_{var}^{i} = \frac{\sum_{i=1}^{\#d_i} (R_i - R_{avg}^{i})^2}{\#d_i},$$  \hspace{1cm} (6.46)

where $R_{var}^{i}$ is the variance rank values of neighbors of node $i$. Similarly, the cooperativity level of neighbors of node $i$ can be calculated as the average of cooperation strategies of nodes $i$. We denote average cooperativity level of neighbors of node $i$ with $\#C_{d_i}^{a_i}$. We calculate as follows:

$$\#C_{d_i}^{a_i} = \frac{\sum_{j=1}^{\#d_i} \#C_j}{\#d_i},$$  \hspace{1cm} (6.47)

Finally, we correlate $R_{avg}^{i}$ and $R_{var}^{i}$ vectors with $\#C_{d_i}^{a_i}$ vector based on relative entropy or kullback-leibler divergence (KL divergence). KL divergence is usually applied to compute dissimilarities of two random variables. KL divergence can be computed as follows:

$$KL(p, q) = \sum_{k=1}^{K} p_k \log\left(\frac{p_k}{q_k}\right),$$  \hspace{1cm} (6.48)
Applications of Overlapping Communities

which $p$ and $q$ are two vectors that we need to compute their relative information. $KL$ divergence can be rewritten by entropy terms as follows:

$$KL(p, q) = \sum_{k=1}^{K} p_k - \sum_{k=1}^{K} p_k \log(q_k) = -H(p) + H(p, q), \quad (6.49)$$

where $H(p, q)$ is named the cross entropy. We can show that when exactly $p = q$ then $KL(p, q)$ equals zero and always $KL(p, q) > 0$ [Murp12]. Here in the evaluations, we compute the relative entropy of average and variance neighbor rank and cooperation.

**Evaluation Results**

To simulate the evolutionary process, we run the PD game for a $TimeWindow = 500$. To obtain more stable results, we iterate the PD game ten times and average over their output. To observe the correlation between rank values of nodes and their cooperativity level, we employ a couple of real-world datasets as listed in the Table 6.19. One can find all of the datasets on the Web. The corresponding table provides some information regarding the number of nodes, the number of edges, average degree, the standard deviation of node degrees and clustering coefficients of these networks.

We apply the three strategies on a couple of datasets and plot the correlation versus the time window of the PD game. For the first strategy, we plot the Hamming Distance of the binary rank vectors and the binary cooperation vector versus the number of iterations in the PD game which we set to 500. Ranking algorithms include Simple Degree (SD), PageRank, Closeness Centrality (CL), Betweenness centrality (BW), Clustering Coefficient (CC) and HITS. We can observe legends of each algorithm besides its figure. Figures 6.14 and 6.15 indicate the correlation among node ranks and cooperativity level of each node using Hamming Distance. As we can observe, the distance is normalized between 0 and 1. Here the higher values indicate less correlation and lower values the reverse. As for Dolphin network, almost for all the ranking algorithms, the correlation values have a smooth horizontal line. This observation indicates that after a few iterations the PD game on the network convergences to a steady state with a fixed cooperativity level. Moreover, we can observe the highest correlation for HITS algorithm regarding cooperativity. In other words, rank values obtained by HITS have the most top similarity with cooperativity of nodes - mostly less than 0.1 value. In comparison to HITS, additional ranking strategies including CL, PageRank, CC, SD, and BW obtain less similarity, respectively; however, their difference is not high. In this regard, BW, SD, and CC obtain similar distance values of rank and cooperativity.

Similar to Dolphin network, we observe the same pattern as for Zachary - HITS showing the highest correlation between node rank and cooperativity. Regarding other ranking algorithms, the pattern might be a bit different - SD has the greatest Hamming Distance, and BW has the lowest. Additionally, CL, CC, and PageRank retain similar correlations. As for Swamill social network, we approximately observe a similar pattern which HITS shows the highest similarity and others show the least. Regarding Swamill, these correlation values remain between 0.4 and 0.55.

Now we observe bigger networks with higher complexities. Although we can find some fluctuations, the pattern is still kept the same for network Science data which HITS has the highest
correlation of node rank and cooperativity. In this dataset, CC and CL have higher similarity values in comparison to the rest and BW produces least correlation values. Regarding the Europe road and email networks, the pattern is smooth and almost the same with HITS reaching the highest similarity of rank and cooperation for nodes. Moreover, other ranking strategies obtain fewer correlations in the range of 0.4 to 0.75. As for Protein-Protein interaction and Gnutella peer-to-peer networks, the correlations show more fluctuations due to variations of some nodes between cooperation and defection strategies. However, the pattern is kept similar to the previous simulations with HITS obtaining the highest correlation between rank and cooperation.

Table 6.20: LFR Parameters for cooperation and defection simulations.

<table>
<thead>
<tr>
<th>Name</th>
<th>networks N</th>
<th>k max k</th>
<th>mu t1 t2</th>
<th>on om</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFR 160</td>
<td>50</td>
<td>160</td>
<td>3.4</td>
<td>80</td>
<td>0.1 2.4</td>
</tr>
<tr>
<td>LFR 380</td>
<td>50</td>
<td>380</td>
<td>11.8</td>
<td>190</td>
<td>0.1 2.4</td>
</tr>
</tbody>
</table>

Table 6.21: This table shows the basic statistics about open source software development and learning forums. Cooperativity of community structures is investigated on these datasets.

<table>
<thead>
<tr>
<th>Name</th>
<th>networks avg n</th>
<th>avg m</th>
<th>avg D</th>
<th>avg deg</th>
<th>avg std deg</th>
<th>avg CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSS</td>
<td>70</td>
<td>123,0</td>
<td>328,7</td>
<td>0,0511</td>
<td>4,5801</td>
<td>11,3185</td>
</tr>
<tr>
<td>JMOL Releases</td>
<td>13</td>
<td>235,3</td>
<td>1670,6</td>
<td>0,0348</td>
<td>13,3446</td>
<td>0,8694</td>
</tr>
<tr>
<td>BioJava Releases</td>
<td>10</td>
<td>289,7</td>
<td>1670,6</td>
<td>0,0350</td>
<td>12,6486</td>
<td>0,6008</td>
</tr>
<tr>
<td>BioPerl Releases</td>
<td>18</td>
<td>634,7</td>
<td>5744,8</td>
<td>0,0279</td>
<td>23,4578</td>
<td>0,6537</td>
</tr>
<tr>
<td>BioPython Releases</td>
<td>29</td>
<td>309,0</td>
<td>2088,0</td>
<td>0,0432</td>
<td>13,3446</td>
<td>0,8694</td>
</tr>
</tbody>
</table>

To extend our observations, we also apply neighbor mean and variance correlation strategies on these datasets. Figures 6.17, 6.16 and 6.18 show KL divergence of the mean strategy on the left and KL divergence of the variance on the right. We might notice that KL divergence is a value larger than zero and can be up to infinite amounts; therefore, very high values are correlations are omitted for plotting. When KL divergence approaches zero, it means the neighbor rank and neighbor cooperativity have the highest correlation - this is true for both average and variance values in this regard. Let us first observe smaller datasets, for example, Figure 6.17. Regarding Dolphin network and the mean strategy, the highest correlation can be observed for CL and CC followed by SD, HITS, PageRank and BW, respectively. In other words, if one intends to evaluate the cooperativity level of a node’s neighbor with the average rank of the node’s neighbor, CL and CC show the highest correlations. As for Swamill and the Mean strategy, HITS and SD have the most prominent correlations followed by CL, PageRank, BW; therefore the pattern is a bit different. Regarding Swamill Strike, CL, CC and SD obtain the highest correlation values near to zero, which indicates the similarity of mean neighbor rank and mean neighbor cooperation strategies. Here
again, CC and CL are among highest correlations and also SD joins to the previous set. Still looking into small datasets, Swamill Strike shows the highest correlation for CC followed by CL and SD. Here HITS, BW and PageRank have lower correlations; however, CC and CL were among the top correlation values. To extend our observation regarding the correlation of mean neighbor rank and mean neighbor cooperation, we investigate bigger datasets to check whether CL, CC, and SD show the same correlation patterns.

Regarding Network science dataset, more or less, we can observe that BW and SD obtain the highest similarity followed by CL, HITS, PageRank and CC. As for Europe road network, we can observe almost similar correlations for all the algorithms except HITS with lower similarities. Regarding Email network, the pattern is similar to Europe road network and HITS also shows close similarities like other algorithms. Although HITS showed the highest correlation in node level, it has the lowest correlation for the neighborhood strategy. As for Gnutella network, the correlation values are somehow increasing which it depends on the internal structure of the network. Here BW and HITS have the highest correlation values followed by SD, PageRank, BW, and CL. Finally, regarding the Protein network, some correlations were too low that is not plotted in this diagram and the highest correlation can be observed for CL and SD. Overall, we can perceive that CL, SD, and CC show the highest mean neighbor rank and mean neighbor cooperation correlations; this is also true for big datasets such as Email network and Network science. By investigating Table 6.19, we figure out that Europe road and Science networks and the small datasets such as Swamill, Swamill Strike, and Dolphin have approximately low variance degrees; however, datasets such as Gnutella, Email and Protein-Protein interactions have high variance degrees which their ranking might be different with different strategies. All the datasets have an approximately similar average degree for the nodes, but the variance degrees can be observed quite differently which indicates various internal structures.

Regarding neighbor variance strategy, we intend to observe whether CC, CL, and SD are among the highest correlation values. As for Dolphin network, SD, HITS, and CL have the highest similarities among others. Regarding Swamill, the pattern is a bit different, and we can observe the highest correlation for CL followed by PageRank and BW. Here Dolphin has double variance degree in comparison to Swamill which leads into different correlation values. Regarding Zachary, the pattern is smooth, and SD, BW, and CC obtain the highest correlations among others. Here Zachary has high variance degree in comparison to other datasets. As for Swamill Strike, we can observe CC, SD, CL, PageRank and HITS among algorithms with high correlation values. Swamill Strike has low variance degree and approximately high clustering coefficient of 0.44 in this regard. As for Europe road network, SD neighbor variance rank vector is entirely similar to cooperation neighbor variance vector which is followed by BW. In this sense, the variance degree for Europe road is quite low. As for Email network, we observe the highest variance correlation for SD, CL and CC followed by PageRank, HITS, and BW. Correspondingly, Email network has approximately high variance degree of 87.28.

As for Protein, Email, Science and Europe networks, we observe SD as the KL divergence of near to zero which shows that variance of the simple degree of neighbors of a node has a high correlation with the variance of neighbors’ cooperativity. Regarding Protein network, the pattern is a bit distinctive, and all other algorithms do not correlate with the cooperativity of the neighbors with the variance strategy. This issue might be because of the low clustering coefficients observed in the protein networks with 0.0059 value. Regarding Gnutella with high variance degree, the
correlations values are a bit unstable but reaching a steady state with PageRank, SD, and CL as the most top correlations.

Table 6.22: This table shows the effect of the $T$ parameter on average and standard deviations of cooperativity, wealth and convergence iterations in URCH dataset. The cooperativity values in URCH is somewhat low; and the amount of cooperativity decreases as the temptation to defect parameter increases.

<table>
<thead>
<tr>
<th>data</th>
<th>Game</th>
<th>CC</th>
<th>CD</th>
<th>DC</th>
<th>DD</th>
<th>Dynamic</th>
<th>$\Delta(N)$</th>
<th>$std(\Delta(N))$</th>
<th>$P(N)$</th>
<th>$std(P(N))$</th>
<th>avg iter.</th>
<th>std iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>average</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.20</td>
<td>0.00</td>
<td>REP</td>
<td>0.2184</td>
<td>0.1578</td>
<td>0.2184</td>
<td>0.1578</td>
<td>1026.6250</td>
<td>23.8084</td>
</tr>
<tr>
<td>1</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.30</td>
<td>0.00</td>
<td>REP</td>
<td>0.3645</td>
<td>0.3247</td>
<td>0.3645</td>
<td>0.3247</td>
<td>1034</td>
<td>102.4523</td>
</tr>
<tr>
<td>2</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.40</td>
<td>0.00</td>
<td>REP</td>
<td>0.3217</td>
<td>0.3241</td>
<td>0.3217</td>
<td>0.3241</td>
<td>1042</td>
<td>130.8496</td>
</tr>
<tr>
<td>3</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.50</td>
<td>0.00</td>
<td>REP</td>
<td>0.1969</td>
<td>0.2260</td>
<td>0.1969</td>
<td>0.2260</td>
<td>1041</td>
<td>146.3767</td>
</tr>
<tr>
<td>4</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.60</td>
<td>0.00</td>
<td>REP</td>
<td>0.1295</td>
<td>0.1207</td>
<td>0.1295</td>
<td>0.1207</td>
<td>1023</td>
<td>104.5257</td>
</tr>
<tr>
<td>5</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.70</td>
<td>0.00</td>
<td>REP</td>
<td>0.1027</td>
<td>0.0867</td>
<td>0.1027</td>
<td>0.0867</td>
<td>1004</td>
<td>28.0703</td>
</tr>
<tr>
<td>6</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.80</td>
<td>0.00</td>
<td>REP</td>
<td>0.0780</td>
<td>0.0655</td>
<td>0.0780</td>
<td>0.0655</td>
<td>1002</td>
<td>19.9497</td>
</tr>
<tr>
<td>7</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.90</td>
<td>0.00</td>
<td>REP</td>
<td>0.0600</td>
<td>0.0465</td>
<td>0.0600</td>
<td>0.0465</td>
<td>1000</td>
<td>0</td>
</tr>
</tbody>
</table>

6.4.3 Cooperativity of Community Structures

We test cooperation and defection simulations on several real-world networks. We interpret every dataset as a separate network. We perform some preprocessing and remove multiple edges as well as loops. Furthermore, edges of the graph are considered in-directed. Moreover, in our simulations, nodes with at least one neighbor are found only; therefore we remove nodes with no incident edge. Additionally, two main forum datasets are used for our simulations including open source software development and learning forums that primary statistics about these datasets are shown in Table 6.21. We can observe that open source datasets are based on software projects of the open bioinformatics foundation. Open source software development datasets contain the data crawled from the mailing list interactions of four projects JMOL, BioJava, BioPerl, and BioPython. In this regard, JMOL project is related to molecular modeling of chemical structures; however, other projects provide frameworks for processing of biological data in programming languages, i.e., Java, Perl and Python [HaKl13]. Networks are fabricated based on communications among participants of the mailing list, which we considered one node for every participant who posts a question or answer. An edge is created between two nodes if they have at least one edge contact in common. Data contains release periods’ information so that we could build the networks corresponding to each respective timeline. As for learning forums, we use URCH and STDOCTOR datasets. The URCH as an English language learning forum provides a bed to discuss admission tests [PKKl11].

For our experiments, we only consider years 2003 and 2005, which we produced monthly networks. Similarly to open source software development forums, we connect participants who have at least one mail contact in common and thus we build a network of interactions for each respective month. We also used the Lancichinetti Fortunato Radicchi benchmark [LaFo09] introduced in chapter 2, to generate synthetic networks. Previously, these networks have been used to evaluate community detection algorithms; however, here we use them as spatial structures for cooperation and defection. We generated two sets of LFR networks with their corresponding ground-truth community structures. We intend to apply our method against networks with a close approximation to the forums.
Applications of Overlapping Communities

Table 6.23: This table shows the effect of the $T$ parameter on average and standard deviations of cooperativity, wealth and convergence iterations in STDOCTOR dataset. The STDOCTOR is a low cooperative dataset compared to URCH, and the temptation to defect parameter has a strong effect on the amount of cooperativity in this network.

<table>
<thead>
<tr>
<th>data</th>
<th>Game</th>
<th>CC</th>
<th>CD</th>
<th>DC</th>
<th>DD</th>
<th>Dynamic</th>
<th>$\Delta(N)$</th>
<th>$std(\Delta(N))$</th>
<th>$P(N)$</th>
<th>$std(P(N))$</th>
<th>avg iter.</th>
<th>std iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0370</td>
<td>0.0121</td>
<td>0.0370</td>
<td>0.0121</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>1 PD</td>
<td></td>
<td>1.00</td>
<td>0.00</td>
<td>1.20</td>
<td>0.00</td>
<td>REP</td>
<td>0.0615</td>
<td>0.0370</td>
<td>0.0615</td>
<td>0.0370</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>2 PD</td>
<td></td>
<td>1.00</td>
<td>0.00</td>
<td>1.30</td>
<td>0.00</td>
<td>REP</td>
<td>0.0476</td>
<td>0.0253</td>
<td>0.0476</td>
<td>0.0253</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>3 PD</td>
<td></td>
<td>1.00</td>
<td>0.00</td>
<td>1.40</td>
<td>0.00</td>
<td>REP</td>
<td>0.0384</td>
<td>0.0219</td>
<td>0.0384</td>
<td>0.0219</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>4 PD</td>
<td></td>
<td>1.00</td>
<td>0.00</td>
<td>1.50</td>
<td>0.00</td>
<td>REP</td>
<td>0.0344</td>
<td>0.0221</td>
<td>0.0344</td>
<td>0.0221</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>5 PD</td>
<td></td>
<td>1.00</td>
<td>0.00</td>
<td>1.60</td>
<td>0.00</td>
<td>REP</td>
<td>0.0310</td>
<td>0.0186</td>
<td>0.0310</td>
<td>0.0186</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>6 PD</td>
<td></td>
<td>1.00</td>
<td>0.00</td>
<td>1.70</td>
<td>0.00</td>
<td>REP</td>
<td>0.0288</td>
<td>0.0190</td>
<td>0.0288</td>
<td>0.0190</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>7 PD</td>
<td></td>
<td>1.00</td>
<td>0.00</td>
<td>1.80</td>
<td>0.00</td>
<td>REP</td>
<td>0.0277</td>
<td>0.0181</td>
<td>0.0277</td>
<td>0.0181</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>8 PD</td>
<td></td>
<td>1.00</td>
<td>0.00</td>
<td>1.90</td>
<td>0.00</td>
<td>REP</td>
<td>0.0266</td>
<td>0.0163</td>
<td>0.0266</td>
<td>0.0163</td>
<td>1000</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.20 gives further information regarding the generated LFR networks with their corresponding parameters. The prototypical implementation of this part of the research was supported by a bachelor thesis guided by the author of this work [Kram17].

### 6.4.4 Evaluation Protocol

To find overlapping communities in several forums, we use community detection algorithms including InfoMap, SLPA, SSK, and CLiZZ. Similar to node rank cooperativity, we adopt the commonly-used approach by Nowak and May [NoMa92]. The parameters in this method are $R = 1$, $S = P = 0$ and $T = b > 1$, which this approach can be considered as a simplified version of the Prisoner’s Dilemma (PD) with only one parameter. As for update dynamic, we use the replicator dynamic (RP). In RP, updating rule depends on the tendency to imitate others ($b$ parameter) that we set 1.5. We consider a population structure, which we consider each node as a player. We randomly initialize the players with strategies, afterward, they play the game against all their neighbors. The payoff a player receives is the aggregation of each payoff through each of the games. Finally, the strategy of all agents is updated based on a global update rule. Here, we run our simulation for at least 1000 iterations. Afterwards, we check whether players have reached a stationary state. Therefore, we compute the standard deviation of the network cooperativity values of the last 200 iterations. We consider the conditions of a stable state as fulfilled whenever the standard deviation of the last 200 iterations is less or equal to $1/\sqrt{N}$. We let the system run for another 200 iterations if this is not the case. The simulation run until the system reaches the maximum of 9000 iterations.

As the system reaches a steady state, we compute the network cooperativity and the agents’ cooperativities for the simulation. We calculated the network cooperativity as the average cooperativity value of the network over the last 200 iterations. Also, we considered the cooperativity value for an agent, which it can be computed based on the number of time the agent played cooperation within the last 200 iterations divided by 200. To prevent biases in our simulations, we run every simulation 200 times and compute the average cooperativity values. The cooperativity $\Delta(N)$ of Network N is the average network cooperativity of the 200 simulations. The same condition is met for agents’ cooperativity. Similarly, we calculate cooperativity value of a community. The cooperativity value of a community $C$ is the average cooperativity of its members, in other words, we can compute it as:
6.4. COOPERATION AND DEFECTION: INDIVIDUAL AND COLLECTIVE PROPERTIES

Table 6.24: This table contains correlations based on the 70 release-based OSS networks as well as 48 monthly-based learning forums. Most of the network properties have the negative correlation with the amount of cooperativity.

<table>
<thead>
<tr>
<th>Measure</th>
<th>covariance</th>
<th>pearson</th>
<th>spearman</th>
<th>kendall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release Networks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>0.0067</td>
<td>0.1646</td>
<td>0.2460</td>
<td>0.1745</td>
</tr>
<tr>
<td>D</td>
<td>-0.0208</td>
<td>-0.4782</td>
<td>-0.4677</td>
<td>-0.3060</td>
</tr>
<tr>
<td>avg deg</td>
<td>-0.0156</td>
<td>-0.3298</td>
<td>-0.1651</td>
<td>-0.1135</td>
</tr>
<tr>
<td>std deg</td>
<td>-0.0157</td>
<td>-0.3765</td>
<td>-0.2236</td>
<td>-0.1478</td>
</tr>
<tr>
<td>CC</td>
<td>-0.0355</td>
<td>-0.5868</td>
<td>-0.3310</td>
<td>-0.2616</td>
</tr>
<tr>
<td>Learning Forums</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>-0.0241</td>
<td>-0.3797</td>
<td>-0.5070</td>
<td>-0.3307</td>
</tr>
<tr>
<td>D</td>
<td>-0.0165</td>
<td>-0.3822</td>
<td>-0.3662</td>
<td>-0.2429</td>
</tr>
<tr>
<td>avg deg</td>
<td>-0.0414</td>
<td>-0.6430</td>
<td>-0.7141</td>
<td>-0.5195</td>
</tr>
<tr>
<td>std deg</td>
<td>-0.0277</td>
<td>-0.4262</td>
<td>-0.4077</td>
<td>-0.3404</td>
</tr>
<tr>
<td>CC</td>
<td>-0.0712</td>
<td>-0.7987</td>
<td>-0.8108</td>
<td>-0.6277</td>
</tr>
</tbody>
</table>

\[
\Delta(C) = \sum_{n \in C} \frac{\Delta n}{|C|}.
\]

We classify a community as cooperative if \( \Delta(C) > 0.5 \), otherwise as defective. To detect communities, we use one of the (overlapping) community detection algorithms, i.e., DMID, SLPA, etc. We define the wealth, which characterizes the final benefit based on to the same principle. The wealth \( P(n) \) of an agent is the average payoff he received during the last 200 iterations. We define the wealth \( P(N) \) of a network with \( N \) nodes as the average wealth of all agents of a network. Moreover, we describe the wealth \( P(C) \) of a Community \( C \) as the average wealth of all players that are the member of the Community \( C \).

Experiments

We first calculate the cooperativity values of the whole networks and correlated them with network properties. We also find correlations among cooperativity of communities and their internal properties. To do so, we used the correlation metrics introduced in chapter 2 for these experiments.

Network Cooperativity and Temptation to Defect

As for prisoner’s dilemma with the replicator dynamic, we changed the game Parameter \( T \), which is the temptation to defect, between 1.1 and 1.9. It means as we increase the temptation for defection, we expect less cooperativity in the test networks. However, we want to figure out to what extent \( T \) parameter affects on cooperativity of learning and OSS forums. As such, we can see in Tables 6.25, 6.26 and 6.27, 6.28, information such as average and standard deviation of cooperativity as well
Table 6.25: This table shows the effect of the $T$ parameter on average and standard deviations of cooperativity, wealth and convergence iterations in BioJava dataset. As the temptation to detect parameter increases, the amount of network cooperativity decreases.

<table>
<thead>
<tr>
<th>data</th>
<th>Game</th>
<th>CC</th>
<th>CD</th>
<th>DC</th>
<th>DD</th>
<th>Dynamic</th>
<th>$\Delta(N)$</th>
<th>std($\Delta(N)$)</th>
<th>$P(N)$</th>
<th>std($P(N)$)</th>
<th>avg iter.</th>
<th>std iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>average</td>
<td></td>
<td>0.8488</td>
<td>0.0629</td>
<td>3.3369</td>
<td>0.1592</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1038.33</td>
<td>24.0156</td>
</tr>
<tr>
<td>1 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.10</td>
<td>0.00</td>
<td>REP</td>
<td>0.9500</td>
<td>0.0868</td>
<td>3.6110</td>
<td>0.2881</td>
<td>1008</td>
<td>44.1104</td>
<td></td>
</tr>
<tr>
<td>2 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.20</td>
<td>0.00</td>
<td>REP</td>
<td>0.9025</td>
<td>0.1585</td>
<td>3.4522</td>
<td>0.5455</td>
<td>1011</td>
<td>45.7105</td>
<td></td>
</tr>
<tr>
<td>3 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.30</td>
<td>0.00</td>
<td>REP</td>
<td>0.8903</td>
<td>0.1527</td>
<td>3.4317</td>
<td>0.4763</td>
<td>1012</td>
<td>51.6657</td>
<td></td>
</tr>
<tr>
<td>4 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.40</td>
<td>0.00</td>
<td>REP</td>
<td>0.8803</td>
<td>0.1580</td>
<td>3.4147</td>
<td>0.4726</td>
<td>1038</td>
<td>112.3347</td>
<td></td>
</tr>
<tr>
<td>5 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.50</td>
<td>0.00</td>
<td>REP</td>
<td>0.8365</td>
<td>0.1803</td>
<td>3.3013</td>
<td>0.5231</td>
<td>1038</td>
<td>99.0203</td>
<td></td>
</tr>
<tr>
<td>6 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.60</td>
<td>0.00</td>
<td>REP</td>
<td>0.8284</td>
<td>0.2029</td>
<td>3.2840</td>
<td>0.6010</td>
<td>1041</td>
<td>100.8457</td>
<td></td>
</tr>
<tr>
<td>7 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.70</td>
<td>0.00</td>
<td>REP</td>
<td>0.8105</td>
<td>0.2108</td>
<td>3.2574</td>
<td>0.5856</td>
<td>1064</td>
<td>138.5641</td>
<td></td>
</tr>
<tr>
<td>8 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.80</td>
<td>0.00</td>
<td>REP</td>
<td>0.7980</td>
<td>0.2423</td>
<td>3.2156</td>
<td>0.6984</td>
<td>1067</td>
<td>146.3492</td>
<td></td>
</tr>
<tr>
<td>9 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.90</td>
<td>0.00</td>
<td>REP</td>
<td>0.7430</td>
<td>0.2730</td>
<td>3.0645</td>
<td>0.8207</td>
<td>1066</td>
<td>154.1519</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.26: This table shows the effect of the $T$ parameter on average and standard deviations of cooperativity, wealth and convergence iterations in JMOL dataset. As the temptation to detect parameter increases, the amount of network cooperativity decreases.

<table>
<thead>
<tr>
<th>data</th>
<th>Game</th>
<th>CC</th>
<th>CD</th>
<th>DC</th>
<th>DD</th>
<th>Dynamic</th>
<th>$\Delta(N)$</th>
<th>std($\Delta(N)$)</th>
<th>$P(N)$</th>
<th>std($P(N)$)</th>
<th>avg iter.</th>
<th>std iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>average</td>
<td></td>
<td>0.4779</td>
<td>0.1051</td>
<td>0.4779</td>
<td>0.1051</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1014</td>
<td>6,5955</td>
</tr>
<tr>
<td>1 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.10</td>
<td>0.00</td>
<td>REP</td>
<td>0.6435</td>
<td>0.3805</td>
<td>0.6435</td>
<td>0.3805</td>
<td>1015</td>
<td>56.4885</td>
<td></td>
</tr>
<tr>
<td>2 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.20</td>
<td>0.00</td>
<td>REP</td>
<td>0.6309</td>
<td>0.3802</td>
<td>0.6309</td>
<td>0.3802</td>
<td>1014</td>
<td>61.8313</td>
<td></td>
</tr>
<tr>
<td>3 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.30</td>
<td>0.00</td>
<td>REP</td>
<td>0.5385</td>
<td>0.3921</td>
<td>0.5385</td>
<td>0.3921</td>
<td>1018</td>
<td>70.0036</td>
<td></td>
</tr>
<tr>
<td>4 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.40</td>
<td>0.00</td>
<td>REP</td>
<td>0.4971</td>
<td>0.3984</td>
<td>0.4971</td>
<td>0.3984</td>
<td>1020</td>
<td>66.4990</td>
<td></td>
</tr>
<tr>
<td>5 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.50</td>
<td>0.00</td>
<td>REP</td>
<td>0.4177</td>
<td>0.4055</td>
<td>0.4177</td>
<td>0.4055</td>
<td>1026</td>
<td>80.9740</td>
<td></td>
</tr>
<tr>
<td>6 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.60</td>
<td>0.00</td>
<td>REP</td>
<td>0.4223</td>
<td>0.4073</td>
<td>0.4223</td>
<td>0.4073</td>
<td>1005</td>
<td>31.3033</td>
<td></td>
</tr>
<tr>
<td>7 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.70</td>
<td>0.00</td>
<td>REP</td>
<td>0.3873</td>
<td>0.4115</td>
<td>0.3873</td>
<td>0.4115</td>
<td>1008</td>
<td>39.2902</td>
<td></td>
</tr>
<tr>
<td>8 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.80</td>
<td>0.00</td>
<td>REP</td>
<td>0.3864</td>
<td>0.4162</td>
<td>0.3864</td>
<td>0.4162</td>
<td>1009</td>
<td>46.1481</td>
<td></td>
</tr>
<tr>
<td>9 PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.90</td>
<td>0.00</td>
<td>REP</td>
<td>0.3772</td>
<td>0.4216</td>
<td>0.3772</td>
<td>0.4216</td>
<td>1011</td>
<td>45.7105</td>
<td></td>
</tr>
</tbody>
</table>

as average and standard deviation of obtained wealth (payoff) in OSS networks. In these tables, last columns show average and standard deviation of iterations for convergence. On the one hand, results show that amount of cooperativity, $\Delta(N)$, decreases in OSS networks and the standard deviations increase for them. Similar to this trend, the wealth value, $P(N)$, also decreases for OSS networks. Besides, networks need more iterations to reach the stationary state when the $T$ value increases. Moreover, the decrement rate in the amount of cooperativity is more in BioJava, BioPerl and BioPython compared to Jmol dataset. On the other hand, the decrement in cooperativity in learning forums is a stronger effect than in the OSS Networks. Thus, we can observe that OSS networks possess more cooperativity and players receive higher payoff values (benefits - wealth) even with different scales of $T$ values. Overall, Bio-OSS networks are mostly cooperative. Among OSS networks, JMOL has lower cooperativity values. With varying amounts of temptation, URCH has an average cooperativity value of around 0.2, which is higher than average cooperativity value of STDOCTOR dataset (0.03). Even with low $T$ values, the cooperativity level is below 20% in learning forums as we can observe in Tables 6.22 and 6.23.

We also used two groups of synthetic networks, which approximate the structure of real-world networks, i.e., forums. As such, Tables 6.29 and 6.30 show results of cooperativity values. LFR160 networks are cooperative; however, LFR380 networks are almost defective. The temptation parameter has a strong effect on LFR380 networks compared to LFR160.
Table 6.27: This table shows the effect of the $T$ parameter on average and standard deviations of cooperativity, wealth and convergence iterations in BioPython dataset. As the temptation to detect parameter increases, the amount of network cooperativity decreases.

<table>
<thead>
<tr>
<th>data</th>
<th>Game</th>
<th>CC</th>
<th>CD</th>
<th>DC</th>
<th>DD</th>
<th>Dynamic</th>
<th>$\Delta(N)$</th>
<th>std($\Delta(N)$)</th>
<th>$P(N)$</th>
<th>std($P(N)$)</th>
<th>avg iter.</th>
<th>std iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>average</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.10</td>
<td>0.00</td>
<td>REP</td>
<td>0.9400</td>
<td>0.2379</td>
<td>0.9400</td>
<td>0.2379</td>
<td>1000</td>
<td>0.00</td>
</tr>
<tr>
<td>1</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.20</td>
<td>0.00</td>
<td>REP</td>
<td>0.8963</td>
<td>0.3040</td>
<td>0.8963</td>
<td>0.3040</td>
<td>1001</td>
<td>14.1421</td>
</tr>
<tr>
<td>3</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.40</td>
<td>0.00</td>
<td>REP</td>
<td>0.8621</td>
<td>0.3409</td>
<td>0.8621</td>
<td>0.3409</td>
<td>1002</td>
<td>19.9497</td>
</tr>
<tr>
<td>5</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.60</td>
<td>0.00</td>
<td>REP</td>
<td>0.8072</td>
<td>0.3845</td>
<td>0.8072</td>
<td>0.3845</td>
<td>1003</td>
<td>28.0703</td>
</tr>
<tr>
<td>7</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.80</td>
<td>0.00</td>
<td>REP</td>
<td>0.7436</td>
<td>0.4206</td>
<td>0.7436</td>
<td>0.4206</td>
<td>1004</td>
<td>41.9500</td>
</tr>
<tr>
<td>9</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>2.00</td>
<td>0.00</td>
<td>REP</td>
<td>0.6732</td>
<td>0.4014</td>
<td>0.6732</td>
<td>0.4014</td>
<td>1005</td>
<td>51.3033</td>
</tr>
</tbody>
</table>

Table 6.28: This table shows the effect of the $T$ parameter on average and standard deviations of cooperativity, wealth and convergence iterations in BioPerl dataset. As the temptation to detect parameter increases, the amount of network cooperativity decreases.

<table>
<thead>
<tr>
<th>data</th>
<th>Game</th>
<th>CC</th>
<th>CD</th>
<th>DC</th>
<th>DD</th>
<th>Dynamic</th>
<th>$\Delta(N)$</th>
<th>std($\Delta(N)$)</th>
<th>$P(N)$</th>
<th>std($P(N)$)</th>
<th>avg iter.</th>
<th>std iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>average</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.10</td>
<td>0.00</td>
<td>REP</td>
<td>0.9446</td>
<td>0.1549</td>
<td>0.9446</td>
<td>0.1549</td>
<td>1043</td>
<td>97.9745</td>
</tr>
<tr>
<td>2</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.20</td>
<td>0.00</td>
<td>REP</td>
<td>0.8006</td>
<td>0.1819</td>
<td>0.8006</td>
<td>0.1819</td>
<td>1069</td>
<td>133.8979</td>
</tr>
<tr>
<td>4</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.40</td>
<td>0.00</td>
<td>REP</td>
<td>0.7641</td>
<td>0.2206</td>
<td>0.7641</td>
<td>0.2206</td>
<td>1143</td>
<td>243.4076</td>
</tr>
<tr>
<td>6</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.60</td>
<td>0.00</td>
<td>REP</td>
<td>0.7345</td>
<td>0.2380</td>
<td>0.7345</td>
<td>0.2380</td>
<td>1229</td>
<td>392.5935</td>
</tr>
<tr>
<td>8</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.80</td>
<td>0.00</td>
<td>REP</td>
<td>0.6847</td>
<td>0.2621</td>
<td>0.6847</td>
<td>0.2621</td>
<td>1248</td>
<td>421.1685</td>
</tr>
<tr>
<td>10</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>2.00</td>
<td>0.00</td>
<td>REP</td>
<td>0.6134</td>
<td>0.2753</td>
<td>0.6134</td>
<td>0.2753</td>
<td>1254</td>
<td>391.2151</td>
</tr>
</tbody>
</table>

Table 6.29: This table shows the effect of $T$ parameter on average and standard deviations of cooperativity, wealth and convergence iterations in LFR160 synthetic networks. The temptation to defect parameter has a weaker effect on the network cooperativity in comparison to LFR380 networks.

<table>
<thead>
<tr>
<th>data</th>
<th>Game</th>
<th>CC</th>
<th>CD</th>
<th>DC</th>
<th>DD</th>
<th>Dynamic</th>
<th>$\Delta(N)$</th>
<th>std($\Delta(N)$)</th>
<th>$P(N)$</th>
<th>std($P(N)$)</th>
<th>avg iter.</th>
<th>std iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>average</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.10</td>
<td>0.00</td>
<td>REP</td>
<td>0.85</td>
<td>0.17</td>
<td>0.85</td>
<td>0.17</td>
<td>1007.0</td>
<td>36.85</td>
</tr>
<tr>
<td>2</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.20</td>
<td>0.00</td>
<td>REP</td>
<td>0.83</td>
<td>0.20</td>
<td>0.83</td>
<td>0.20</td>
<td>1005.0</td>
<td>31.30</td>
</tr>
<tr>
<td>4</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.40</td>
<td>0.00</td>
<td>REP</td>
<td>0.76</td>
<td>0.24</td>
<td>0.76</td>
<td>0.24</td>
<td>1008.0</td>
<td>39.29</td>
</tr>
<tr>
<td>6</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.60</td>
<td>0.00</td>
<td>REP</td>
<td>0.73</td>
<td>0.26</td>
<td>0.73</td>
<td>0.26</td>
<td>1011.0</td>
<td>45.71</td>
</tr>
<tr>
<td>8</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.80</td>
<td>0.00</td>
<td>REP</td>
<td>0.66</td>
<td>0.28</td>
<td>0.66</td>
<td>0.28</td>
<td>1020.0</td>
<td>66.50</td>
</tr>
<tr>
<td>10</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>2.00</td>
<td>0.00</td>
<td>REP</td>
<td>0.60</td>
<td>0.31</td>
<td>0.60</td>
<td>0.31</td>
<td>1011.0</td>
<td>49.91</td>
</tr>
</tbody>
</table>

Correlations of Network Properties and Network Cooperativity

We further compute the correlations between network properties and their cooperativity values. We considered 70 OSS and 48 learning networks from Table 6.21, which each of these networks belong to a specific time span. We used covariance, Pearson, Spearman and Kendall values to compute the correlations. As for network properties, we used size, density (D), average degrees, standard deviations of degrees and clustering coefficient values. Table 6.24 shows correlation values for both
Table 6.30: This table shows the effect of $T$ parameter on average and standard deviations of cooperativity, wealth and convergence iterations in LFR380 synthetic networks. The LFR380 networks have the low amount of cooperativity especially when the temptation to defect parameter increases.

<table>
<thead>
<tr>
<th>data</th>
<th>Game</th>
<th>CC</th>
<th>CD</th>
<th>DC</th>
<th>DD</th>
<th>Dynamic</th>
<th>$\Delta(N)$</th>
<th>std($\Delta(N)$)</th>
<th>$P(N)$</th>
<th>std($P(N)$)</th>
<th>avg iter.</th>
<th>std iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.22</td>
<td>0.23</td>
<td>0.22</td>
<td>0.23</td>
<td>1009.44</td>
<td>9.95</td>
</tr>
<tr>
<td>1</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.10</td>
<td>0.00</td>
<td>REP</td>
<td>0.63</td>
<td>0.28</td>
<td>0.63</td>
<td>0.29</td>
<td>1026.0</td>
<td>73.15</td>
</tr>
<tr>
<td>2</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.20</td>
<td>0.00</td>
<td>REP</td>
<td>0.50</td>
<td>0.31</td>
<td>0.50</td>
<td>0.31</td>
<td>1020.0</td>
<td>66.50</td>
</tr>
<tr>
<td>3</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.30</td>
<td>0.00</td>
<td>REP</td>
<td>0.38</td>
<td>0.32</td>
<td>0.38</td>
<td>0.33</td>
<td>1020.0</td>
<td>69.45</td>
</tr>
<tr>
<td>4</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.40</td>
<td>0.00</td>
<td>REP</td>
<td>0.195</td>
<td>0.28</td>
<td>0.19</td>
<td>0.28</td>
<td>1005.0</td>
<td>31.30</td>
</tr>
<tr>
<td>5</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.50</td>
<td>0.00</td>
<td>REP</td>
<td>0.15</td>
<td>0.23</td>
<td>0.14</td>
<td>0.23</td>
<td>1009.0</td>
<td>46.15</td>
</tr>
<tr>
<td>6</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.60</td>
<td>0.00</td>
<td>REP</td>
<td>0.08</td>
<td>0.14</td>
<td>0.08</td>
<td>0.14</td>
<td>1003.0</td>
<td>24.37</td>
</tr>
<tr>
<td>7</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.70</td>
<td>0.00</td>
<td>REP</td>
<td>0.04</td>
<td>0.08</td>
<td>0.04</td>
<td>0.08</td>
<td>1002.0</td>
<td>19.95</td>
</tr>
<tr>
<td>8</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.80</td>
<td>0.00</td>
<td>REP</td>
<td>0.01</td>
<td>0.051</td>
<td>0.01</td>
<td>0.05</td>
<td>1000.0</td>
<td>0.0</td>
</tr>
<tr>
<td>9</td>
<td>PD</td>
<td>1.00</td>
<td>0.00</td>
<td>1.90</td>
<td>0.00</td>
<td>REP</td>
<td>0.01</td>
<td>0.03</td>
<td>0.01</td>
<td>0.03</td>
<td>1000.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

OSS and learning forums. Regarding OSS networks, size of the network is the only property, which has the positive correlation with network cooperativity; however, other properties such as density, average and standard deviation of degrees and clustering coefficient have negative correlations with cooperativity values. As for learning forums, different correlation metrics show somehow various correlation directions; however, clustering coefficient has the positive correlation with cooperativity values. This observation can be because of the strong effect of structural properties and connections on the whole networks compared to OSS networks.

To obtain a clear view of the correlations, we also plot the actual network property values versus their corresponding cooperativity values for OSS and learning forums, respectively. These visualizations are shown in Figures 6.19 and 6.20. Compliant with the results in Table 6.21, OSS networks’ properties show negative correlations with cooperativity. In other words, low values of these metrics have high levels of cooperativity. This observation is compliant with studies of [YZXL09, SRJa10], that they showed a reverse relation between the size of communities and significance levels of motifs with cooperativity. We can observe exceptions for JMOL dataset, which even for the high value of the properties, i.e., average degree, degree deviation, and clustering coefficient, average values of cooperativity around 0.5 can be observed. In other words, for other networks, i.e., Bio-OSS, the relation among network properties and cooperativity values is the reverse.

Regarding learning forums, we can observe that the results are a bit scattered, which the correlation results for URCH (2003, 2004, 2005) and STDOKCTOR (2009) are somehow different. Concerning STDOKCTOR, high values of properties, in most cases lead to low values of cooperativity. On the contrary, for URCH dataset the correlation seems to become more positive through time, which the year 2005 has higher correlation values. To put this another way, structural properties and density of connections in URCH have a stronger effect in forming cooperative networks compared to STDOKCTOR. We do not know what specific contextual properties may cause such effects in these two forums; however, we can figure out that context of forums and nature of communications has a substantial impact on cooperativity of networks. In other words, STDOKCTOR forum is dedicated to medical students, which discuss and share issues related to exams, classes, etc. In contrast, URCH is a forum for discussions about exams such as TOEFL, GRE, etc. [Kens15]. So, we can observe language learners have a higher amount of cooperativity compared to those discussing medical issues. Interestingly structure of URCH network as a language learning forum show the positive
correlation among average cooperativity values of nodes.

**Community Cooperativity**

We consider different community detection algorithms such as InfoMap (non-overlapping), SSK, CLiZZ and SLPA (overlapping). We computed the correlations of community properties by these algorithms with their respective cooperativity values. To avoid biases, we excluded unrealistic communities with only one member or communities as large as the size of the whole network. For the remaining communities, we computed their properties, i.e., size, density, etc., and correlated them with their respective cooperativity values. For every community, we created a sub-graph, which internal nodes are connected; however, the sub-graph does not contain edges to external nodes of the community. The correlation values are computed based on the 70 release-based OSS networks as well as 48 monthly-based learning forums.

Table 6.31: This table contains information regarding correlations of community properties and their cooperativity values using the InfoMap algorithm. The values are computed based on the 70 release-based OSS networks as well as 48 monthly-based learning forums. In learning forums, most of the correlation values are negative; however, only for density, we can observe negative correlations in OSS forums.

<table>
<thead>
<tr>
<th>Measure</th>
<th>covariance</th>
<th>pearson</th>
<th>spearman</th>
<th>kendall</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Release Networks</strong></td>
<td><strong>Communities: 1837</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>0.0006</td>
<td>0.0721</td>
<td>0.1862</td>
<td>0.1332</td>
</tr>
<tr>
<td>density</td>
<td>-0.0123</td>
<td>-0.1598</td>
<td>-0.1874</td>
<td>-0.1342</td>
</tr>
<tr>
<td>avg Deg</td>
<td>0.0021</td>
<td>0.1036</td>
<td>0.1581</td>
<td>0.1110</td>
</tr>
<tr>
<td>std Deg</td>
<td>0.0031</td>
<td>0.1914</td>
<td>0.2347</td>
<td>0.1708</td>
</tr>
<tr>
<td><strong>Learning Forums</strong></td>
<td><strong>Communities: 1367</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>-0.0020</td>
<td>-0.1004</td>
<td>-0.2233</td>
<td>-0.1756</td>
</tr>
<tr>
<td>density</td>
<td>-0.0106</td>
<td>-0.1642</td>
<td>-0.1376</td>
<td>-0.0985</td>
</tr>
<tr>
<td>avg Deg</td>
<td>-0.0042</td>
<td>-0.2810</td>
<td>-0.3909</td>
<td>-0.2855</td>
</tr>
<tr>
<td>std Deg</td>
<td>-0.0009</td>
<td>-0.0449</td>
<td>0.0618</td>
<td>0.0433</td>
</tr>
</tbody>
</table>

Tables 6.31, 6.32, 6.33 and 6.34 show the results of correlations for InfoMap, SLPA, SSK and CLiZZ algorithms. As we can observe in all of these tables, community detection algorithms show negative correlations in learning forums for all the properties. This observation indicates that for bigger property values, we observe smaller cooperativity values; however, smaller values of community properties show high levels of cooperativity. This trend, more or less, can be observed in Figures 6.25, 6.22, 6.24 and 6.27.

First, we demonstrate our findings concerning InfoMap algorithm on learning and OSS networks through Table 6.31 and Figures 6.23 and 6.22. On the one hand, through Table 6.31, we observe that InfoMap algorithm identifies 1837 and 1367 communities in OSS and learning forums. Concerning OSS networks, for all the metrics, i.e., covariance, Pearson, Spearman, Kendall, correlation values are respectively 0.0006, 0.0721, 0.1862 and 0.1332, which are positive. We as well observe positive values for average and standard deviation of degrees. Regarding average degree, correlation values are 0.0021, 0.1036, 0.1581, and 0.1110 for covariance, Pearson, Spearman and Kendall distance.
Applications of Overlapping Communities

Table 6.32: This table contains information regarding correlations of community properties and their cooperativity values using the SLPA algorithm. The values are computed based on the 70 release-based OSS networks as well as 48 monthly-based learning forums. Similar to the InfoMap algorithm, in learning forums, most of the correlation values are negative; however, only for density, we can observe negative correlations in OSS forums.

<table>
<thead>
<tr>
<th>Measure</th>
<th>covariance</th>
<th>pearson</th>
<th>spearman</th>
<th>kendall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release Networks</td>
<td>Communities: 1454</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>0.0011</td>
<td>0.0880</td>
<td>0.3435</td>
<td>0.2505</td>
</tr>
<tr>
<td>density</td>
<td>-0.0343</td>
<td>-0.3461</td>
<td>-0.3574</td>
<td>-0.2531</td>
</tr>
<tr>
<td>avg Deg</td>
<td>0.0044</td>
<td>0.1329</td>
<td>0.1724</td>
<td>0.1192</td>
</tr>
<tr>
<td>std Deg</td>
<td>0.0059</td>
<td>0.2053</td>
<td>0.2809</td>
<td>0.2053</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measure</th>
<th>covariance</th>
<th>pearson</th>
<th>spearman</th>
<th>kendall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Forums</td>
<td>Communities: 1303</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>-0.0014</td>
<td>-0.0748</td>
<td>-0.4080</td>
<td>-0.3391</td>
</tr>
<tr>
<td>density</td>
<td>-0.0124</td>
<td>-0.2105</td>
<td>-0.0910</td>
<td>-0.0670</td>
</tr>
<tr>
<td>avg Deg</td>
<td>-0.0050</td>
<td>-0.3476</td>
<td>-0.5479</td>
<td>-0.4319</td>
</tr>
<tr>
<td>std Deg</td>
<td>-0.0016</td>
<td>-0.0768</td>
<td>-0.0226</td>
<td>-0.0177</td>
</tr>
</tbody>
</table>

Table 6.33: This table contains information regarding correlations of community properties and their cooperativity values using the SSK algorithm. The values are computed based on the 70 release-based OSS networks as well as 48 monthly-based learning forums. SSK detects 349 and 739 communities in OSS and learning forums, respectively. All the community properties in OSS and learning forums have negative correlation values with community cooperativity (except density in OSS forums).

<table>
<thead>
<tr>
<th>Measure</th>
<th>covariance</th>
<th>pearson</th>
<th>spearman</th>
<th>kendall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release Networks</td>
<td>Communities: 349</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>-0.0246</td>
<td>-0.4167</td>
<td>-0.3999</td>
<td>-0.2584</td>
</tr>
<tr>
<td>density</td>
<td>0.0134</td>
<td>0.2754</td>
<td>0.4186</td>
<td>0.2841</td>
</tr>
<tr>
<td>avg Deg</td>
<td>-0.0080</td>
<td>-0.2374</td>
<td>-0.1861</td>
<td>-0.1225</td>
</tr>
<tr>
<td>std Deg</td>
<td>-0.0126</td>
<td>-0.2831</td>
<td>-0.1680</td>
<td>-0.1245</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measure</th>
<th>covariance</th>
<th>pearson</th>
<th>spearman</th>
<th>kendall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Forums</td>
<td>Communities: 739</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>-0.0112</td>
<td>-0.1863</td>
<td>0.0199</td>
<td>-0.0185</td>
</tr>
<tr>
<td>density</td>
<td>-0.0264</td>
<td>-0.4470</td>
<td>-0.5435</td>
<td>-0.3687</td>
</tr>
<tr>
<td>avg Deg</td>
<td>-0.0352</td>
<td>-0.5908</td>
<td>-0.6783</td>
<td>-0.4854</td>
</tr>
<tr>
<td>std Deg</td>
<td>-0.0250</td>
<td>-0.3953</td>
<td>-0.3350</td>
<td>-0.2237</td>
</tr>
</tbody>
</table>

respectively. As such, Spearman correlation obtains the highest correlation values compared to the other correlation coefficients. The only exception is density, which the correlation values are negative. For density, Spearman correlation also has the most negative correlation of -0.1874. Figure 6.23 shows the actual distribution of community properties versus community cooperativity values. First of all, the number of points (number of detected communities) are higher in comparison to other algorithms. Second, for size, average and standard deviation of degrees, the points are piled up at the bottom of the diagram. However, for density, the points are scattered through the chart, which the picture does not show any specific finding to explain the negative correlation between density and positive values for size, avg and std degrees.
Table 6.34: This table contains information regarding correlations of community properties and their cooperativity values using the CLiZZ algorithm. The values are computed based on the 70 release-based OSS networks as well as 48 monthly-based learning forums. Implicit community structures detected by CLiZZ algorithm have the negative correlation with the amount of community cooperativity.

<table>
<thead>
<tr>
<th>Measure</th>
<th>covariance</th>
<th>pearson</th>
<th>spearman</th>
<th>kendall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release Networks</td>
<td>Communities: 368</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>-0.0287</td>
<td>-0.4738</td>
<td>-0.4749</td>
<td>-0.3352</td>
</tr>
<tr>
<td>density</td>
<td>0.0131</td>
<td>0.3178</td>
<td>0.5325</td>
<td>0.3573</td>
</tr>
<tr>
<td>avg Deg</td>
<td>-0.0075</td>
<td>-0.1904</td>
<td>-0.1278</td>
<td>-0.0669</td>
</tr>
<tr>
<td>std Deg</td>
<td>-0.0146</td>
<td>-0.3513</td>
<td>-0.2728</td>
<td>-0.1925</td>
</tr>
<tr>
<td>Learning Forums</td>
<td>Communities: 574</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>-0.0068</td>
<td>-0.1224</td>
<td>0.0499</td>
<td>0.0021</td>
</tr>
<tr>
<td>density</td>
<td>-0.0317</td>
<td>-0.5120</td>
<td>-0.6347</td>
<td>-0.4251</td>
</tr>
<tr>
<td>avg Deg</td>
<td>-0.0354</td>
<td>-0.5774</td>
<td>-0.6511</td>
<td>-0.4541</td>
</tr>
<tr>
<td>std Deg</td>
<td>-0.0215</td>
<td>-0.3485</td>
<td>-0.3041</td>
<td>-0.1931</td>
</tr>
</tbody>
</table>

On the other hand, the lower part of Table 6.31 shows the correlation values for learning forums. In contrast to the InfoMap algorithm on OSS forum, we observe that size, avg and std deviation of degrees also have the negative correlation with community cooperativity values. For instance, size of the detected communities have negative correlation values of respectively -0.0020, -0.1004, -0.2233 and -0.1756 for different metrics. If we look at Figure 6.22, we can observe the power-law-like distribution of points concerning community properties versus community cooperativity values. This observation explains the effect why the properties have now negative correlations; because the actual distributions of points have a power-law-like distribution.

Secondly, we analyze the results of SLPA algorithm on OSS and learning forums through Table 6.32 and Figures 6.21 and 6.24. In total, SLPA detects 1454 and 1303 communities for OSS and learning forums, respectively. On the one hand, the number of communities is approximately as high as the InfoMap algorithm, and we can observe to some extent the same pattern concerning the correlation values. In other words, community density values only have the negative correlation with community cooperativity values. However, size, average and standard deviation of degrees have positive correlations with cooperativity values of communities. This finding is complaint with the InfoMap algorithm on OSS datasets. To further explore the SLPA on OSS forums, we can have a look at Figure 6.21. Similarly, the actual points are scattered either in the whole Figure (avg deg, std deg) or are piled up at the bottom of the diagram, which makes it difficult to extract some firm conclusions.

On the other hand, SLPA results on learning forums are similar to InfoMap results with negative correlations for all the metrics. Moreover, Spearman correlation obtains the highest values for correlation. By looking at Figure 6.24, we figure out that the distribution of actual points for community properties, i.e., size, avg deg, std deg have a power-law-like distribution versus correlation values. This finding can be a reason for the negative correlation observed in Table 6.32.

Furthermore, we demonstrate the results by the SSK algorithm on OSS and learning forums through Table 6.33, and Figures 6.25 and 6.26. In total, SSK identifies 349 and 739 communities for OSS
and learning forums, respectively. The correlation values are a bit different than InfoMap and SLPA algorithm, which we observe negative correlations both for OSS and learning forums. The only exception is the density property, which has positive values as for the correlations. If we consider Figures 6.26 and 6.25, we figure out that the actual distribution of the points are power-law-like for both OSS and learning forums. This observation as well holds for almost all of the community properties, i.e., size, density, avg deg, std deg. As we obtain large communities, they tend to be more sparse, and they have a wider variety of degrees for the nodes. CLiZZ algorithm obtains the similar output to SSK algorithm. We can observe the correlation results in Table 6.34 and the actual distribution of community properties versus cooperativity in Figures 6.28 and 6.27.

Similar to SSK, CLiZZ also detects smaller community sizes, i.e., 368 and 574, for OSS and learning forums, respectively. Moreover, we can see that all the correlation values for all the community properties are negative. There is only an exception for density in OSS forums. Complaint with other Tables, Spearman correlation, has the highest values of correlation compared to other metrics, i.e., Pearson, Covariance, and Kendall. Concerning the distribution of community properties versus cooperativity, we can observe that CLiZZ has a power-law-like distribution in OSS networks. However, there is also an exception for learning forums, which the points are scattered over the diagram. We can conclude that for SLPA and InfoMap algorithms, we observed the higher number of communities with negative correlations only for learning forums. In contrast, SSK and CLiZZ detected the fewer number of communities with negative correlations in both OSS and learning forums. Hence, we can conclude that when the size of implicit community structures detected by the algorithms is big, we may have sparser communities with the wider variety of degrees for the nodes. This finding may better shows the actual relationships among community properties and their cooperativity values, which is a negative correlation.

To summarize, the correlation values are a bit different for OSS datasets. On the one hand, CLiZZ and SSK show negative correlation values except for density. On the other hand, InfoMap and SLPA show positive correlations except for density. We can ascribe it to the nature of community detection algorithms. In other words, CLiZZ works based on identifying of influential nodes, and InfoMap and SLPA might have some information propagation property. When we look at Figures 6.26, 6.23, 6.21, and 6.28, we figure out that the number of detected communities are different as well as the distribution of community property values over the plots. For instance, Figures 6.26 and 6.28 show that number of detected communities are lower compared to SLPA and InfoMap. Moreover, communities are scattered all over the plots. For instance, we can observe all range of values for density as well as cooperativity. In contrast, SLPA and InfoMap detect a slightly higher number of communities. This observation indicates that the communities have a smaller size with lower average degrees. Therefore, communities pile up on the lower side of the diagram, i.e., lower values show lower cooperativity, and higher values show higher cooperativity compared to the other cases. However, there is only an exception for density.

Additionally, for all community detection algorithms, we see that community properties show negative correlations with respective cooperativity values; however, correlation values for OSS datasets showed either positive or negative. In other words, different community detection algorithms detected different community structures in OSS forums, and this may cause bipolar amounts in this regard. Another explanation for this observation is the property of the datasets. We can observe in Table 6.21 that properties of OSS and learning forums are the same; however, Figure 6.19 and 6.20 show that learning forums have bigger networks and thus larger communities. Moreover, learning
forums have the property of a forum with the characteristics of question and answers; however, OSS networks are mailing lists, so this also may cause some difference in the structural property of the networks.

Our results, in general, are compliant with findings from [YZXL09], that they showed a reverse relation between size and cooperativity of communities.
Applications of Overlapping Communities

Figure 6.16: This figure shows KL Divergence of neighbor rank, and neighbor cooperativity for mean and variance approaches. We evaluated it on three datasets of network science, Europe road, and Email Networks. There is no single metric having the highest correlation, but we can observe CC, SD, CL, and BW in several cases with the highest values of correlation with cooperativity.
6.4. COOPERATION AND DEFECTION: INDIVIDUAL AND COLLECTIVE PROPERTIES

(a) Dolphin Neighbor Mean KL Divergence.
(b) Dolphin Neighbor Variance KL Divergence.
(c) Swamill Neighbor Mean KL Divergence.
(d) Swamill Neighbor Variance KL Divergence.
(e) Zachary Neighbor Mean KL Divergence.
(f) Zachary Neighbor Variance KL Divergence.

Figure 6.17: This figure shows KL divergence of neighbor rank and neighbors cooperativity for mean and variance approaches. We examined it on three datasets including Dolphin, Swamill, and Zachary. There is no single metric having the highest correlation, but we can observe SD, CC, and CL with the highest values of correlation with cooperativity.
Applications of Overlapping Communities

(a) Gnutella Neighbor Mean KL Divergence.

(b) Gnutella Neighbor Variance KL Divergence.

(c) Swamill Strike Neighbor Variance Mean KL.

(d) Swamill Strike Neighbor Variance KL.

(e) Protein Neighbor Variance Mean KL Divergence.

(f) Protein Neighbor Variance KL Divergence.

Figure 6.18: This figure shows the correlation of neighbor rank with neighbor cooperativity for mean and variance approaches. We applied it on three datasets of Gnutella, Swamill Strike and Protein-Protein Networks. The amount of cooperativity is somewhat unstable in several of the figures, and some metrics have the minimal correlations, so we did not plot them due to big KL divergence values.
6.4. COOPERATION AND DEFECTION: INDIVIDUAL AND COLLECTIVE PROPERTIES

Figure 6.19: These plots show network properties, i.e., size, average degree, density, degree deviation and clustering coefficient, versus cooperativity values in OSS forums. In many cases, network properties have reverse relation with the amount of cooperativity.
Figure 6.20: These plots show network properties, i.e., size, average degree, density, degree deviation and clustering coefficient, versus cooperativity values in learning forums. The plots indicate that network properties, in many cases, have the reverse relationship with the amount of cooperativity.
Figure 6.21: These figures show properties of identified communities by SLPA algorithm, i.e., size, density, average degree and standard deviation of degrees, versus cooperativity of their respective communities on OSS forums. We can observe that the distribution of community properties concerning density, average degree, and standard deviation of degrees are piled up at the bottom, and the distribution of the size of communities versus cooperativity is flattened on the bottom of the diagram.

Figure 6.22: These figures show properties of detected communities by the InfoMap algorithm, i.e., size, density, average degree and standard deviation of degrees, versus cooperativity of their respective communities on learning forums. We can observe that average degree, size, density and standard deviation of degrees have to some extent a power-law-like relation to the amount cooperativity values.
Applications of Overlapping Communities

Figure 6.23: These figures show properties of identified communities by the InfoMap algorithm, i.e., size, density, average degree and standard deviation of degrees, versus cooperativity of their respective communities on OSS forums. We can observe that average degree, size and standard deviation of degrees are piled up at the bottom of the diagrams; however, points concerning density versus cooperativity are scattered through the diagram.

Figure 6.24: These figures show properties of detected communities by SLPA algorithm, i.e., size, density, average degree and standard deviation of degrees, versus cooperativity of their respective communities on learning forums. Similar to InfoMap on learning forums, we can observe that average degree, size, density and standard deviation of degrees have to some extent a power-law-like relation to the amount cooperativity values.
6.4. COOPERATION AND DEFECTION: INDIVIDUAL AND COLLECTIVE PROPERTIES

Figure 6.25: These figures show properties of detected communities by SSK algorithm, i.e., size, density, average degree and standard deviation of degrees, versus cooperativity of their respective communities on learning forums. We can observe that the distribution of community properties, i.e., size, density, avg deg, std deg, are power-law-like. The result is compliant with the output of InfoMap and SLPA algorithm on learning forums.

Figure 6.26: These figures show properties of detected communities by SSK algorithm, i.e., size, density, average degree and standard deviation of degrees, versus cooperativity of their respective communities on OSS forums. We can observe that the distribution of community properties, i.e., size, density, avg deg, std deg, are scattered over the diagrams.
Applications of Overlapping Communities

Figure 6.27: These figures show properties of detected communities by the CLiZZ algorithm, i.e., size, density, average degree and standard deviation of degrees, versus cooperativity of their respective communities on learning forums. The community properties have a power-law-like distribution similar to SSK algorithm. CLiZZ identifies the fewer number of communities (bigger communities), so we obtain more sparse communities with the broader variety of node degrees.

Figure 6.28: These figures show properties of detected communities by the CLiZZ algorithm, i.e., size, density, average degree and standard deviation of degrees, versus cooperativity of their respective communities on OSS forums. The distribution of actual community properties are scattered through the diagrams; however, CLiZZ obtains negative correlations concerning the metrics.
6.5 Conclusion

In this chapter, we addressed applications of overlapping communities. First, we proposed recommender models using temporal dynamics of community structures. In this regard, one of our suggested models, i.e., TCNSVD, achieved superior performance regarding accuracy and item ranking metrics in comparison to the-state-of-the-art methods. However, the proposed model has high time complexity. As such, we proposed faster models, i.e., TCNSVD-Fast and CNSVD-Fast, using dynamics of community structures. Fast models make a compromise between accuracy and running time, which in many cases obtain satisfactory performance. We also identified the best parameters for the models, i.e., similarity metrics, number of neighbors in graph construction. Additionally, we proposed ranking algorithms using implicit community structures. Our experiments showed better performance of community-aware ranking models compared to classical ranking algorithms, i.e., HITS and PageRank. Dynamics of community structures improve expert identification. To learn about the willingness of nodes to cooperate, we considered cooperativity through game theoretic approaches, i.e., prisoner’s dilemma. To consider cooperativity of nodes, we implemented different ranking algorithms and the necessary prisoner’s dilemma game and evaluated such correlations. Results showed that HITS ranking algorithm has the highest correlation with cooperativity value of a node; however, centrality metrics such as simple degree, closeness centrality, and clustering coefficient have higher correlation values with cooperativity when considering the neighborhood of a node. Regarding proximity of nodes and their cooperativity values results were more scattered, which may require more investigation.

Last but not least, we could reveal some findings regarding cooperativity of learning and open source software development forums. We not only focused on cooperativity and payoff values but also we considered cooperativity in community structures of these forums. One main finding was that open source software development forums possess a much higher amount of cooperativity compared to learning forums. We verified this by considering different realizations from our simulations as well as scaling temptation to defect parameter in our models. The second finding is related to properties of these networks. Cooperation showed mainly reverse correlation with network properties in both learning and OSS forums. We also learned that context in forums is a primary drive in specifying the dynamic, as such, URCH as a platform for language learners showed higher levels of cooperation compared to STDOCTOR forum for discussing medical topics. Last but not least, we investigated community structures in OSS and learning forums. Interestingly, our results were compliant with previous works, which detected community properties showed a negative correlation with cooperativity of communities.
Applications of Overlapping Communities
Chapter 7

WebOCD Software Framework

7.1 Introduction

We introduced several networks and community analytic tools in chapter 3, and we found that they are somewhat scattered and users (researchers) still encounter challenges while using them. For instance, some of these tools do not provide a suitable graphical interface for the person working with them. Moreover, the research community has a few time to do reverse engineering to understand the code. Secondly, the tools do not provide a complete functionality expected from a community analytic framework. In other words, sometimes there is no evaluation measure or synthetic network generator to support the users. Or even preprocessing and postprocessing functionalities are missing. Additionally, most of the tools are not Web-based, and one needs to develop them via a suitable Integrated Development Environment (IDE), and often several libraries may be missing. Furthermore, most of the tools are not open source and are not possible to be easily extended by the research community. Additionally, one has presumed potential use cases for most of the tools, and it is hard to deploy them in real applications. Last but not least, analytic tools are based on different programming languages, which this makes it difficult to use and integrate them consistently.

These problems and challenges motivated us to devise an integrative approach to develop a community analytic framework also known as WebOCD. In this regard, we have considered principal requirements in the design and implementation of WebOCD framework. One of the primary needs was integrating different community detection algorithms. As such, we have either incorporated or implemented algorithms from scratch, which can serve for analytic purposes. We further supported WebOCD with evaluation metrics that users can compare different algorithms on different datasets. Besides, performing some preprocessing tasks on the datasets are possible, examples are making the network undirected or analyzing a network through visualization. The framework has a Web client, which users can simultaneously enter the system and perform analytic tasks on their datasets. We have considered additional primary requirements in the WebOCD framework that facilitate the analysis of community structures. Besides the possibility to upload different network types, it is also possible to generate synthetic networks that simulate real-world networks, and thus one can evaluate community detection algorithms on them. This capability gives much flexibility to perform more extensive and multifaceted analysis through the framework. Other requirements
are as well considered for WebOCD, for instance, developing an open source software, possibility of extension and adding new algorithms, metrics, having a RESTful design, etc. are among them.

To answer the research question four (RQ4) raised in chapter 1, we used the WebOCD in several use cases, which we facilitate its usage in different contexts and scenarios. The first use case was the Learning Layers project. It was a project running for four years that partners from various universities aimed at developing applications and tools to support informal learning at construction and medical clusters. In this project, WebOCD has been used to detect communities of learners to increase trust and support users with targeted recommendations. The second application is students of a seminar at RWTH Aachen university that used WebOCD to write a collaborative book on community detection topic. In this seminar, WebOCD has been used to apply available algorithms on different networks, and perform a multifaceted analysis through diagrams, e.g., Spider Web diagram. In other words, the students not only should present the algorithms but also they had to apply the algorithms on new datasets to generate additional comparisons and results. Without the WebOCD framework, writing such a book with new results out of the algorithms might be more difficult, which students should have handled technical troubles in algorithmic levels to run the algorithms on new datasets. However, WebOCD provided them the opportunity to use available functionalities to generate new content. Finally, we used WebOCD in an online evaluation session, which master and Ph.D. computer science students evaluated it. The results of the online survey showed that WebOCD could facilitate working with OCD algorithms as well as performing analytic tasks. The students mentioned that they could assume to use the framework for their future needs.

7.2 WebOCD

The WebOCD framework meets three essential functional requirements. First of all, we implemented different OCD algorithms for identifying covers. Secondly, we provided various metrics for measuring the quality of detected covers. Finally, the service can create visualizations of the resulting covers, especially for small-scale networks. Besides, we considered some non-functional requirements. We implemented the service in the Java programming language. Moreover, it follows a RESTful design and has a Web client to interact with the system. Additionally, the framework is easily extensible to allow the integration of further OCD algorithms. Figure 7.1 shows WebOCD service. It consists of two services named OCD service and Viewer service. OCD service tackles signed, unsigned networks and content-enriched networks such as Question & Answer forums. Moreover, preprocessing, postprocessing, different algorithms, benchmarks and evaluation metrics are supported by the OCD service. The viewer service handles the visualization of the graphs and covers. Both of these services communicate with each other, which is developed based on Las2peer. WebOCD service has a Web client and even communicates with other Web applications. The prototypical implementation of this part of the research was supported by bachelor theses [Krot14, Hafe16, Li2016] guided by the author of this work. In the following, we explain the basic components of this service.
7.2. WEBOCD

The core of the implemented framework is formed by the so-called OCD service, which is in charge of all generated metrics, benchmarks and OCD tasks like running algorithms, metrics and benchmarks. A list of all OCD algorithms is given in Table 7.2 (the algorithms are explained in details in chapters 3 and 4). Implemented metrics include extended modularity [NMCM09], combined modularity [DaVi12], extended NMI [LFKe09], omega index [CoDe88], signed modularity [GJAr09], Frustration [CaHa56] and execution time. Finally, benchmarks include LFR networks proposed by Lancichinetti [LaFo09], Girvan-Newman Model [GiNe02] and the signed LFR adapted by Liu et al. [LiLJ14]. Please see chapter 2 for the detailed description of metrics and benchmarks.

One can see the base structure of OCD service in Figure 7.2. All packages in OCD service, as well as the class forming the service interface, reside under the auspices of the ocd package which is a
super package. _ocd_ package transfers every incoming request to its respective classes. The _graphs_ package contains all necessary classes responsible for the data structures of graphs and communities. Its core class is _CustomGraph_, which extends yFiles’ _Graph2D_. It includes additional graph metadata like graph name. Moreover, it provides access to metadata on nodes and edges such as node names or edge weights. We store the data in respective _CustomNode_ and _CustomEdge_ objects, that a listener class can manage them. These classes are extensible for other development purposes. Besides, a _Cover_ object holds the community structure of a graph obtained by a community detection algorithm. It also consists of several instances of the class _Community_, which define the node membership values. We embed different OCD tasks regarding e.g. algorithms, benchmarks and metrics, in the packages _algorithms_, _benchmarks_ and _metrics_, respectively. We also provide some accessory tasks with the _utils_ package, which does not require any package binding, e.g., simple input/output and persistence tasks. These tasks are handled through the _RequestHandler_. Additionally, We created _ThreadHandler_ that moderates the creation and cancellation of jobs. Finally, we created the _adapters_ package to deal with different formats of input/output graphs/cover. Different formats are handled by _adapters_ package, i.e., the XML-based graph formats provided by I/O classes from yFiles, edge lists, etc. It is also possible to keep node names and edge weights as well as membership degrees.

### 7.2.2 Viewer Service and Web Client

The viewer service is responsible for the visualization of graphs and covers, which the developed Web client renders it in the WebOCD framework. Figure 7.3 describes the base structures. The viewer has few binding with OCD service; however, some parts from _ocd_ package are reused here as well. The _adapters_ package as well as the _graphs_ package with minor changes, for instance,
7.2. WEBOCD

Figure 7.3: This figure shows Viewer Service Architecture, which the most critical package relationships are illustrated.

are also reused in the viewer service. Additionally, we include certain parts of the utils package as well as a few base classes from the metrics package. Compared to the OCD service, we base the ocdViewer package as the super package, which contains all viewer-specific packages as well as the service interface. Regarding layouts, we implemented only one layout, which is suitable for small networks and it works based on the "organic layout" from the yfiles library. Layouters package manages all the layout-related functionalities, which provides classes for defining graph layouts, i.e., the placement of nodes and edges. We created painters package to deal with community colors and visualizations. The utils package in the viewer service provides an extended version of RequestHandler from the OCD service as well as a LayoutHandler, which moderates the tasks of the layouters and painters package. Additionally, it provides some other layouts functionalities such as defining node sizes and colors. Last but not least, to handle visualization output, we integrated as well the slightly modified adapters package from ocd package. We developed the Web client because all the OCD algorithms, metrics and benchmarks should be accessible to end users. In other words, those users who intend to apply one of the available algorithms on a data set can easily perform their evaluations on data. Furthermore, the Web client has a simple graphical user interface and does not require to be installed, and thus facilitates the access of end users.

Overall, a user can log in to the WebOCD service and access its different functionalities. It is possible for her to upload a network data, i.e., undirected, directed, weighted or even upload a sample cover to compute several metrics. When we upload a graph, the user can apply one of the algorithms in Table 7.2 and calculate some metrics. Even she can generate a synthetic network based on different parameters and apply one of the algorithms. These set of functionalities provided the opportunity to use the WebOCD in a seminar class to collaboratively write a book on OCD topic (see page 175).
Figure 7.4: A simplified view of the *algorithms* package. Classes from the package are presented in gray while the packages of other related classes are explicitly indicated.

### 7.2.3 WebOCD is Extensible

One of the main requirements which we respected for WebOCD service is the possibility of adding further algorithms and more functionalities. Figure 7.4 shows the basic structure and main classes of "algorithms" package and other related classes. *ConcreteAlgorithm* indicates any class that implements a community detection algorithm. To create such a new algorithm, we need to implement the *OcdAlgorithm* interface, which defines all the methods for integration with the framework. The method *detectOverlappingCommunities()* is responsible for executing the algorithm by receiving a graph and returning the detected communities. In addition, the method *compatibleGraphTypes()* specifies the graph types the algorithm is compatible with i.e. directed or weighted graphs. The method *getAlgorithmType()* returns a descriptor object, which is used e.g. for the algorithm instantiation. Furthermore, the *OcdAlgorithm* interface extends two further interfaces. On the one hand, *Parameterizable* interface contains two methods, which provide the possibility to set and to get an object’s attributes based on a parameter name mapping. *CoverCreationMethod* interface, on the other hand, specifies the algorithm’s ability to create a cover and it does not require any additional method.

We can create an algorithm instance using *OcdAlgorithmFactory*, and thus we can create complex objects. A generic interface named *ConditionalParameterizableFactory* is implemented by *OcdAlgorithmFactory* for any factory creating objects of any given *Parameterizable* subclass. A descriptor object with required parameters is sent to the factory to determine which product to create for each
instantiation. \textit{isInstantiatable()} method, besides, specifies if a descriptor defines an object that is a valid product for the factory. For instance regarding \textit{OcdAlgorithmFactory}, the product is any \textit{ConcreteAlgorithm}, which implements the \textit{OcdAlgorithm} interface. We can instantiate an algorithm by enumerating a simple constant of the enumeration class \textit{CoverCreationType}. Such a constant has a field of the type \textit{Class<? extends CoverCreationMethod>} and thus it can be a valid descriptor if this class object represents a subclass of \textit{OcdAlgorithm}. According to the \textit{Parameterizable} interface, the algorithm instance is customized through the default constructor. Additionally, an algorithm is executed by the \textit{OcdAlgorithmExecutor}, which handles other tasks such as adapting the input graph to the algorithm’s graph type compatibilities and dividing the graph into its weakly connected components before passing it to the algorithm. In summary, to integrate a new algorithm with the OCD service, we need to take care of three aspects. First, the interface \textit{OcdAlgorithm} should be implemented. Second, we should implement the default constructor. Finally, we should define a corresponding \textit{CoverCreationType}. It is not only possible to integrate further algorithms but also to extend other parts of the framework. It is because we need to take care of the possibility for extension in all steps of design. Table 7.1 gives more details about extensible parts of the service.

<table>
<thead>
<tr>
<th>Extendable Superclass</th>
<th>Factory</th>
<th>Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoverInputAdapter</td>
<td>CoverInputAdapterFactory</td>
<td>CoverInputFormat</td>
</tr>
<tr>
<td>CoverOutputAdapter</td>
<td>CoverOutputAdapterFactory</td>
<td>CoverOutputFormat</td>
</tr>
<tr>
<td>CoverPainter</td>
<td>CoverPainterFactory</td>
<td>CoverPaintingType</td>
</tr>
<tr>
<td>GraphInputAdapter</td>
<td>GraphInputAdapterFactory</td>
<td>GraphInputFormat</td>
</tr>
<tr>
<td>GraphLayouter</td>
<td>GraphLayouterFactory</td>
<td>GraphLayoutType</td>
</tr>
<tr>
<td>GraphOutputAdapter</td>
<td>GraphOutputAdapterFactory</td>
<td>GraphOutputFormat</td>
</tr>
<tr>
<td>GroundTruthBenchmark</td>
<td>OcdBenchmarkFactory</td>
<td>GraphCreationType</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CoverCreationType</td>
</tr>
<tr>
<td>OcdAlgorithm</td>
<td>OcdAlgorithmFactory</td>
<td>OcdCreationType</td>
</tr>
<tr>
<td>OcdMetric</td>
<td>OcdMetricFactory</td>
<td>OcdMetricType</td>
</tr>
</tbody>
</table>

Table 7.1: Extensible System Components.

### 7.2.4 Las2peer

We implemented OCD service and viewer service with Las2peer\textsuperscript{1,2}, which is a software with a particular focus on security and act as a Peer-to-Peer platform as for online social communities. The underlying architecture contains a couple of loosely connected nodes that each of them can comprise one or more agents. These agents can be either user agents or service agents. Users can control user agents; however, service agents render services to other agents in the Peer-to-Peer network. Figure 7.5 indicates an example of Las2peer network with a user agent. As for the service development, we used Las2peer while it provides access to outside network through a Web connector.

\textsuperscript{1}https://Las2peer.org/

\textsuperscript{2}https://github.com/rwth-acis/Las2peer
Moreover, services can distribute across different nodes inside the Peer-to-Peer network [KRLJ16]. Hence, development and deployment of services with Las2peer are quite straightforward, which it is employed to develop the WebOCD service.

Figure 7.5: In Las2peer, nodes communicate through Peer-to-Peer messages with each other. A Web client and a Web application can access the service using the Web connector.

In addition to Web connector, REST Mapper plays an essential role in Las2peer while it maps the HTTP requests to Java methods. The HTTP method and the URI are passed to the Mapper to identify the called service method. Besides, the body of the post HTTP requests together with input variables is extracted by the Mapper to be processed. In the source code, specific annotations are employed to identify the methods and parameters.

7.2.5 Software and Libraries

For the implementation of WebOCD service, we have used a list of libraries, which are mentioned as follows:

Libraries related to databases & Persistence

- Apache Derby

  Apache Derby ³ is a relational database based on the Java programming language. Derby offers Network server, which enables the database access in client/server mode via

³http://db.apache.org/derby/
Table 7.2: List of evaluation metrics, synthetic generators and OCD algorithms developed in the WebOCD framework.

<table>
<thead>
<tr>
<th>Type</th>
<th>Author</th>
<th>Network Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>Clizz Zhang et al. [LZL*12]</td>
<td>unsigned</td>
</tr>
<tr>
<td>Algorithm</td>
<td>DMID Shahriari et al. [ShKK15]</td>
<td>unsigned</td>
</tr>
<tr>
<td>Algorithm</td>
<td>SLPA Xie et al [XSLi11]</td>
<td>unsigned</td>
</tr>
<tr>
<td>Algorithm</td>
<td>LC Ahn et al [ABLc10]</td>
<td>unsigned</td>
</tr>
<tr>
<td>Algorithm</td>
<td>MONC Havemann et al. [HHSG11]</td>
<td>unsigned</td>
</tr>
<tr>
<td>Algorithm</td>
<td>SSK Stanoev et al. [SSKo11]</td>
<td>unsigned</td>
</tr>
<tr>
<td>Algorithm</td>
<td>DOCA Nguyen et al. [NDTN11]</td>
<td>unsigned</td>
</tr>
<tr>
<td>Algorithm</td>
<td>CFOCA Shahriari et al. [ShHK16]</td>
<td>Q &amp; A</td>
</tr>
<tr>
<td>Algorithm</td>
<td>TCMA Shahriari et al. [ShHK16]</td>
<td>Q &amp; A</td>
</tr>
<tr>
<td>Algorithm</td>
<td>MEA Liu et al. [LiJ14]</td>
<td>signed</td>
</tr>
<tr>
<td>Algorithm</td>
<td>SPM Chen et al. [CWYT14]</td>
<td>signed</td>
</tr>
<tr>
<td>Algorithm</td>
<td>SDMID Shahriari et al. [ShLK16]</td>
<td>signed</td>
</tr>
<tr>
<td>Metric</td>
<td>Modularity Nicosia et al. [NMCM09]</td>
<td>signed</td>
</tr>
<tr>
<td>Metric</td>
<td>Omega Index Collins and Dent [CoDe88]</td>
<td>unsigned</td>
</tr>
<tr>
<td>Benchmark</td>
<td>LFR synthetic networks Lancichinetti and Fortunato [LaFo09]</td>
<td>unsigned</td>
</tr>
<tr>
<td>Benchmark</td>
<td>Signed LFR Liu et al. [LiJ14]</td>
<td>signed</td>
</tr>
<tr>
<td>Benchmark</td>
<td>Girvan-Newman Network Newman and Mark [GiNe02]</td>
<td>unsigned</td>
</tr>
</tbody>
</table>

the TCP/IP protocol. Apache Derby is employed as for the WebOCD while it is open source and one can embed it in programs developed in Java.

- **Java Persistence API**

Regarding the persistence of data into the databases, we employ the Java Persistence API (JPA) as an application programming interface (API) to handle relational databases in Java applications. We use Java annotations to map classes to entities. In fact, they consider each entity as a database table. Other annotations can handle relations between different entities, the persisted class attributes, and other configurations. We use JPA for WebOCD because it provides the independence of back-end from database implementations.

**Graph Operations & Visualization**

yFiles\(^4\) as a commercial library provides functionalities for different layout and visualization of graphs and covers. yFiles’ Java-based API\(^5\) makes it possible to integrate it with different frameworks - as such easily, we integrate it with the WebOCD. There are three main functionality available in yFiles including the **Basic**, **Layout** and the **Viewer**. Implementation of graph data types and a couple of algorithms, graph layouts, and visualization of graphs are provided in these components, respectively.

---

\(^4\) [http://www.yworks.com/de/products_yfiles_about.html](http://www.yworks.com/de/products_yfiles_about.html)

Linear algebra libraries

Another library for linear algebra is la4j, which is quite efficient and suitable for matrix multiplication, holding sparse matrices and operations on them as well as eigenvalue and eigenvector computations. la4j is open source, which we could use it to perform algebraic operations in the WebOCD.

Libraries for development of WebOCD client

WebOCD client is developed based on front-end programming languages such as JavaScript, HTML, CSS and jQuery. The communication with the backend, i.e., the OCD service, is done through HTTP requests by using Asynchronous JavaScript and XML (Ajax). Other technologies are also integrated such as XML, the Document Object Model (DOM) and the XMLHttpRequest (XHR) API.

7.2.6 Initial User Tests of the WebOCD Service

In parallel to its development, the WebOCD system has been pretested with online users, mostly recruited among master and Ph.D. students in computer science. Subjects were invited to perform a couple of tests with the WebOCD service. We reasonably explained the main elements and functionalities in the WebOCD to the participants. They were supposed to import some graphs, run some algorithms, compute modularity and NMI measures and observe the visualization. To ask about their feedback, we devised some Web-based questioners and handed them out online among participants. We consider questions with possible answers on a five-point scale ranging from strongly agree to disagree strongly. Moreover, they were required to accomplish the task within 20 minutes which all of the participants succeeded. The survey attendees did not receive any remunerations for all their efforts. The WebOCD framework, as we see in Table 7.3, received satisfactory feedback and almost all of the questions obtained the mean value more than four. However, only one item received a three amount. It is about the error handling of the framework. Participants commented out at the end that they neither agree nor disagree with it, or they did not face any problem which happen when using the service. Other aspects of the design of the framework also obtained a fair assessment. It was very nice to acquire a positive response regarding utilizing the framework for future. There are still opportunities to add further algorithms and improve the client. The service supports different algorithms and metrics; however, test results by end users showed that the Web client needs to offer more explanations about the OCD algorithms. Especially this should be considered when users are not familiar with details of OCD algorithms.

In the next section, we explain the use cases for the WebOCD framework.
7.3 Use Cases

In this section, we report on three scenarios where the developed framework has been applied to real cases. They include Learning Layers project, a seminar course and a graph database named ArangoDB.

7.3.1 WebOCD in Learning Layers Project

Following a series of earlier European large-scale research projects on technology-enhanced learning, the EU Integrated Project Learning Layers (2012-2016) aimed at a layered infrastructure for technology-enhanced learning which one can embed a great variety of learning technologies and content domains, and strong learner communities could organize themselves. The major sample application communities include vocational learning groups of construction and health workers that are particularly hesitant to take learning technologies. Learning Layers consortium comprised an informal association of independent partners (9 universities, one research organization, one large company; Small and Medium Enterprises, including three research-intensive ones). Moreover, the project contained a clear strategy of how to scale adoption of the technologies in the two clusters with more than 1000 end-users through the involvement of professional associations and other partners. By building on recent advances in contextualized learning, these layers provided a meaningful learning context when people interact with people, digital and physical artifacts for their informal education, thus making learning faster and more efficient.

In Learning Layers, project partners established Social Semantic Server (SSS) to create the social relations among the users and entities in the system and track the tiny pieces of learners’ activities. It uses a social semantic network that supports users in situated and personalized learning [KDT*13]. SSS has been integrated with various applications and tools to organize the orchestration of different technologies. SSS can manage tags, time and other contextual information by this full-fledged framework [LLK*15, LLK*15]. Moreover, various types of support, i.e., the recommendation of

---

9http://results.learning-layers.eu/
items such as videos, tags, and recommendation of users such as experts, and identification of overlapping communities is possible. In Learning Layers project, experts can increase trust in the network of learners. In other words, learners can find an answer to a question via experts in specific domains. Experts can offer reliable learning materials. Moreover, we can identify communities in networks of learners by employing WebOCD service. To clarify the problem, let us consider a simple example scenario depicted in Figure 7.6. Here, we showed users with direct connections toward each other by double-line edges and relationships of users with events such as Build Stuff or How To Team are denoted by single-line edges. Users 5, 6 and seven are both attending in teaming and building events; therefore they have information of both domains. In fact, they are overlapping among the communities of people who visit these two classes. An overlapping community detection algorithm identifies these overlapping members between these two clusters. Correspondingly, expert 1 and expert 2 have expertise in Build Stuff and How To Team domains that are identified by an expert identification algorithm. This example scenario indicates the importance of communities and experts in an informal learning scenario.

Figure 7.6: Use case for community detection and expert identification algorithms in the context of Learning Layers project.

In Learning Layers, OCD algorithms support meaningful learning. As to scale up informal learning across communities of learners, we should identify overlapping nodes. Identifying boundary spanners in learning environments and especially informal learning ecosystems is a significant task because they are useful in expanding the communities and circulating information among different disciplines. Furthermore, finding experts in communities depends on OCD algorithms. In fact, experts need to be aware of overlapping nodes that can scale up informal learning across communities of learners. In Layers project, OCD algorithms can play different roles. First, they detect overlapping nodes to expand community borders and increase information flow across communities. Secondly, they enhance knowledge about groups and enable internal tagging to reach a shared understanding. Moreover, identifying hidden relationships among groups of learners contributes to targeted recommendations of experts and learning materials. As we can observe in Figure 7.7,
7.3. USE CASES

WebOCD is integrated with SSS through RESTful communications. As such, users of different tools can benefit from the support offered by the SSS. Among these devices, we can mention Bits and Pieces, Living Documents, Discussion Tool, etc. For instance, with Bits and Pieces, a user can organize and collect the smallest pieces of information grasped in an informal learning scenario or with Aachso!, they can capture videos and annotate the clips as for future use. We can track activities of the users with the SSS capabilities. In this figure, TagRec is a tag recommendation service at the heart of SSS that supports different algorithms.

![Diagram of WebOCD integration with SSS](image)

Figure 7.7: This figure shows how community detection service is integrated with the social semantic server in the Learning Layers project.

7.3.2 WebOCD in Collaborative Writing at RWTH Aachen University

Currently, our life is affected by societal and socio-technological innovations, as such, we can mention the effect of technological innovation on teaching and learning activities. We, at RWTH Aachen University, designed and innovated a seminar course on the topic of overlapping community detection, which students not only can learn the basics of research on community detection algorithm design, but also they have the opportunity to learn scientific writing through collaborative editing and writing of a book. Investigation of OCD algorithms by the students can provide them the chance to compare it with other existing ones from the literature. They can remix the content and the values mentioned in the papers and analyze them from different perspectives and come up with aspects and shortcomings related to the topics. This capability paves the way for the students to
come up with not only the shortcomings of an algorithm but also to re-think about different criteria for community detection algorithm design. Rendering a multi-faceted analysis by various groups, e.g., content and animation teams, would open the perspective to think over the algorithm design deeply and would help the students to follow up and understand computer science related research. Furthermore, comparing their topic with other topics in the course facilitates this process and thus provides the bed for boundary spanning discussions over the groups. Last but not least, this seminar trains the students to research network and data science domain as well as to grasp the basics related to research, design, technical writing, presentation, problem formulation and so on.

Figure 7.8: In the seminar on overlapping community detection, students should have participated in two essential teams horizontal and vertical teams. In the horizontal team, WebOCD team was responsible for using the WebOCD framework through the Web-based client to calculate evaluation metrics in this domain on new datasets.

In the OCD seminar, two teams are considered as for organizing the tasks: vertical and horizontal teams. The former is responsible for editing and providing the content for the book, i.e., the primary text. The later, on the other hand, is responsible for visualization, content design and layout design, evaluation, peer review and documentation of the process. As we can see in Figure 7.8, one of the teams is the WebOCD team that is responsible for applying OCD algorithms on new datasets and calculate the evaluation metrics and render them through suitable diagrams, e.g., spider Web diagram. In other words, the WebOCD team and visualization team cooperate to apply the available algorithms in the WebOCD on new datasets to generate original content and evaluation results on this topic.

This course is organized to assimilate various technological innovations to improve the learning
Figure 7.9: This figure shows that WebOCD has been used to collaboratively write a seminar book on the topic of overlapping community detection. WebOCD has been used by seminar students to test algorithms with new datasets, remix the content to generate results and render it with spiderweb diagrams.

process of students. As we can observe in Figure 7.9, the collaborative writing process is moderated with Git technology as a version control system for the source materials while editing the Jekyll Web pages. Finally, the book is deployed online as a clickable and navigable text book. As it is depicted, students send data to the WebOCD service and calculate the metrics and render it through suitable diagrams.

### 7.3.3 DMID Algorithm from WebOCD Implemented with Pregel in ArangoDB Database

Recently, several distributed computational frameworks, e.g., Map-Reduce, etc., have been designed, which automatically handle processing of large-scale data through partitioning of the data to frames and do the actual processing on clusters of computers. However, only a few of these systems are suitable for processing of large-scale graphs. As such, a distributed processing framework like Map-Reduce is not ideal for handling of large graphs, which practitioners implemented distributed graph processing frameworks such as Pregel. Correspondingly, frameworks like Pregel enables developers to implement algorithms for large-scale processing of networks.

Pregel works based on message passing in a vertex-oriented manner. It provides an API to handle distributed graph processing - Google developed it in around 2010, and Google and Facebook use it for graph algorithms. Pregel is known as a vertex-oriented framework that each node performs as a separate processing unit, which communicates with other nodes through message passing. On
each vertex, there is a function that has access to outgoing connections of the vertex, which can
send messages to other nodes. Moreover, the function knows about the state of a vertex. The
worker executes the function to change the state of the vertex or to send messages to other workers
[MAB*10]. Supersteps perform the computations, which when a superstep is finished, the messages
are used for the next one. As Pregel keeps no global state and it employs messages to send data to
other vertexes, it resists against getting locked. Figure 7.10 shows a schema of a vertex computation.
Map-reduce, on the other hand, consists of a mapper and a reducer function. Mapper function has
key-value pairs, which generates intermediate results. The generated output from mappers is then
reduced to produce the final output [ChSc08]. Map-reduce, however, is not suitable for bulky graph-
like processing, which Pregel was developed as for such an alternative that supports vertex-oriented
message passing.

As for graph processing frameworks like Pregel, one should handle the burden of complicated se-
tups. Moreover, they are only suitable for offline processing of large-scale graphs, which one needs
to convert the graph data to an appropriate format readable by, e.g., Pregel. These challenges cause
several problems such as the delay and the cost in transferring between systems and maintaining
different software stacks. It is efficient that one builds a Pregel-like graph processing system on
top of a distributed database to reduce the workload and costs. By integrating the distributed graph
processing functionalities on top of a distributed database, software and hardware can be reused
efficiently and thus reduce the complexity and operating costs [Graz17].

Among distributed databases, ArangoDB\textsuperscript{10} offers to store large-scale graphs; however, it does not
support network algorithms. As such, integration of a Pregel-like graph processing system on top
of ArangoDB enhances capabilities of the database, and thus graph algorithms can be run paralleled
near to the graph data. Hence, ArangoDB administrators decided to improve the functionality of the
system and its usability for application developers by integrating a graph processing framework on
top. They developed a processing system, which offers a general purpose API to implement graph
algorithms. They have selected several graph problems and algorithms, which their system can use
the optimized implementation. As such, they chose graphs algorithms like shortest path, centrality
metrics, connected components, ranking algorithms, e.g., HITS and PageRank, and community de-
tection algorithms. Among the community detection algorithms, they have chosen DMID because
they could not find any Pregel-like implementation for available overlapping community detec-
tion algorithms (see chapter 4). DMID supports parallel cascading of information, which makes it suitable for parallel programming models like Pregel. The Pregel-like implementation of DMID
outperforms the original implementation.

\footnote{\url{https://www.arangodb.com/}}
7.4 Conclusion

We implemented a service for the detection of overlapping communities, which provides various functionalities. WebOCD supports not only directed, weighted and undirected unsigned graphs but also it is suitable for content-enriched networks as well as networks with both positive and negative connections. It provides preprocessing, postprocessing, different evaluation metrics, and benchmarks and it has a Web client. These features made it possible to use it in two use cases. The first use case is the Learning Layers project to find the overlapping communities in informal learning networks at medium-sized enterprises. The second one is a seminar class on the topic of overlapping community detection that students remixed the content and applied algorithms to new datasets and collaboratively write an online book. The seminar class, in specific, proved that we could improve the interactions of users with an analytics software. The initial user tests of WebOCD service showed that it got acceptable feedback from real users, and they presumed to use such a service in future.
Chapter 8

Conclusion and Outlook

In this chapter, we discuss what we have learned and to which extent we have reached the research goals. We also mention the impact of our work by referring to use cases. We finally mention limitations of our methods as well as opportunities to research in future.

8.1 Conclusion

We selected overlapping communities as a way to describe complex networks in addition to other fundamental properties such as small-world-ness, power-law degree distribution, etc. In this regard, we have proposed structural overlapping community detection algorithms. We identified to what extent dynamics such as degree mixing and information diffusion improve identifying of overlapping community structures compared to other methods. We cannot judge the suitability of an OCD algorithm by using only one metric and thus multifaceted experiments have been performed. As such, rendering the results through spider Web diagram often helps to decide if an algorithm is useful concerning a specific aspect. Moreover, modularity is not always the prevailing metric to judge the performance of OCD algorithms. Other metrics such as time complexity as well as other community properties such as size, number of overlapping nodes, etc., can reveal informative information regarding community properties and community detection methods. Besides, application of overlapping communities to various domains such as expert identification and item recommender systems provided a better view to judge an algorithm’s performance. Furthermore, we do not need always to consider complex dynamics and scenarios, but simple somewhat effective dynamics can in many cases improve the situation and prediction accuracy in networks.

Besides, we applied the OCD algorithms to predict mixing patterns and community structures. We identified the significant community properties in network evolution. As such, we identified the size of communities as the most essential feature for all the events, i.e., dissolve, merge, split and survive. Different overlapping community detection algorithms lead to different community evolution prediction accuracies. Moreover, prediction of communities might depend on the context of dynamics in the network. Without knowing the dynamics of community formation, it might not be possible to predict communities by using only structural properties. Furthermore, we recognized
Conclusion and Outlook

overlapping communities as an important property in complex networks, which overlapping members might have higher levels of information in comparison to other nodes. Overlapping nodes have the role of boundary spanners and scale up the borders of communities; however, online social software does not employ this significant effect. As such, we used the impact of overlapping nodes to the sign prediction problem. Our experiments showed that overlapping nodes competitively predict the edge signs in comparison to intra and extra node types. Overlapping community structures can also contribute to improving different applications on different network types.

Additionally, we applied dynamics of (overlapping) community structures to the expert identification and item recommender systems. We identified to what extent community structures improve online applications. As such, community structures can improve the evaluation metrics for recommender systems as well as expert identification. To do so, we investigated item recommender systems and extended the neighborhood and factor models. Two models named TCNSVD and CNSVD were proposed, which outperform the-state-of-art recommender approaches. We acquired the knowledge that models may even generate high accuracy values; however, they might not be practical in real application scenarios. Thus, we extended these models to faster versions, which make a compromise between accuracy and running time. We found TCNSVD a proper model for the evaluation of community detection algorithms in the recommender domain. As such, we used DMID and Walktrap to detect overlapping communities in the constructed user-user and item-item networks; however, it is possible to evaluate and compare the performance of other community detection algorithms. Additionally, we extended classical ranking algorithm, i.e., HITS and PageRank, to identify experts. Our experimental results showed that the extended community-aware ranking algorithms have high correlation values with their respective baselines, e.g., community-aware PageRank has similar behavior with PageRank.

We applied prisoner’s dilemma on learning and open source software development forums. We found that OSS forums have more cooperativity compared to learning forums. In this regard, cooperativity of community structures showed reverse correlation with their properties such as size, density, etc., especially in learning forums. Moreover, among learning forums, we observed different levels of cooperativity values, i.e., URCH was more cooperative than STDOCTOR. Furthermore, cooperation in a language learning forum is more than cooperation among users discussing medical topics. We as well performed simulations to find relations among rank values of nodes and their cooperativity. As such, we found correlations of ranking methods and their cooperativity values through prisoner’s dilemma game. As such, HITS showed the highest correlation for cooperativity of the nodes. Other ranking strategies such as simple degree, closeness centrality, and clustering coefficient show high values of correlation with cooperativity of the neighborhood of a node.

The developed algorithms and services in this thesis have already shown its impact in real use cases. WebOCD previously supported collaborative writing in a seminar class at RWTH Aachen University. In such a scenario, students in groups used WebOCD service to run the algorithms on various datasets and calculated evaluation metrics such as modularity, running times, etc., and rendered the results through a spider Web diagram. Thus, WebOCD contributes to remix and generate online content. Moreover, WebOCD was used in a European project named Learning Layers running for four years.
8.2 Outlook

We solved several research questions; however, there are still opportunities to do further research in this domain. Yet overlapping communities algorithms may need to be improved. First, we may propose adaptive (overlapping) community detection algorithms, which leverage time in an adaptive manner. Also, algorithms for overlapping community detection might be context-specific. In other words, we may offer algorithms to specific environments. Each environment has its processes and dynamics, and thus it may be better to propose context-specific algorithms. Furthermore, there are many OCD algorithms in the literature; however, for a given dataset, we do not yet know which algorithm is more appropriate and provides the highest accuracies. As such, we can assume to use meta-learning to such algorithmic selection and automate the procedure.

The invention of GPS functionalities and location-based social networks provide the location of users and more enriched sources of information about individuals. Tracking the individuals and mapping their activities help us to mine their future intentions, and thus even better predict the evolution of communities. Applying contextual features besides the structural properties can better predict the community structures. Moreover, using of explicit community structures as well as different community detection algorithms can be assumed for future directions. Another exciting direction may be to figure out the correlation between patterns of users’ data generation and their community structures, for instance, there may exist patterns between the location of users and their community belonging. Altogether combining the location of users and their communities may enhance the performance of recommender systems.

Regarding expert identification, considering task management and automatic extraction of experts’ properties can be assumed as for directions for research. As such, cooperativity of experts is an important aspect, which may require more attention. We may investigate contextual and structural information from the expert network to construct prediction models for such an expert task. Also, we can extend the link prediction problem to the expert cooperativity prediction problem. In other words, we require prediction models to predict whether an expert will help a user or not. Among the features, we can consider the personality of the experts, workload, rank and task management. Regarding cooperativity of community structures, it would be yet necessary to apply other game theory strategies and different updating dynamics. Moreover, we may compute cooperativity in explicit community structures rather than communities detected by a community detection algorithm. Additionally, we still do not know which factors may enhance cooperativity in learning and open source software development forums. In this thesis, we performed our experiments on networks with several thousands of nodes and edges; however, we can extend the simulation protocol to large-scale networks with millions of nodes and edges as well as networks with more complex spatial structures.

Regarding item recommender systems, we may select suitable platforms, which real users can evaluate complex models, e.g., models proposed in this thesis. This way, end users can better state strong and weak aspects of a recommendation, and thus we may better judge the performance of a recommender model. In our experiments, we selected the optimum learning rate and regularization based on RMSE. Future work needs to assess whether optimization, e.g., concerning NDCG, can further improve the models. Although TCNSVD considers temporal dynamics of overlapping com-
Conclusion and Outlook

munity structures, modeling the effect of community life cycles, i.e., birth, death, atrophy, grow, split and merge, on recommendation systems still need to be studied. Moreover, we may examine the impact of time bins as well as the speed of community changes in the proposal of a recommender system. Additionally, more community detection algorithms need to be investigated with our models, to identify best performing ones. Finally, we can examine the effect of explicit community structures as the current ones are based on implicit structures and already achieving remarkable results.

Regarding signed networks, the primary challenging problem is the time complexity of overlapping community detection algorithms. We believe overlapping community detection and analysis in signed networks are worth to research on due to the importance of trust in these types of social networks. There is a few research works considering temporal aspects in signed networks, and researchers have studied most of the methods on static graphs. In this regard, we can investigate community evolution prediction as well as expert finding and recommender issues in signed networks in more details. Still, we know few regarding cooperativity of users and experts in signed networks. Especially, when trust and distrust information is available, expert finding applications become more complex. Finally, it is possible to extend and experiment the outcomes of this thesis to other domains such as computational biology and other aspects of computational computer and social sciences. Finally, regarding WebOCD service, we can equip and integrate further community detection algorithms, other evaluation metrics as well as additional preprocessing and postprocessing functionalities. Moreover, adding an overlapping community detection, e.g., based on different programming languages, through the Web client can be assumed as for fast and flexible extensions. Also, WebOCD needs to provide various evaluation protocol, which researchers can use it for their experiments and write their research papers.
Bibliography


time-aware recommender system based on dependency network of items. The Com-

[DeKa11] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-
144. Springer, 2011.

[DKLy08] Hongbo Deng, Irwin King, and Michael R. Lyu. Formal models for expert finding
on dblp bibliography data. In Eight IEEE International Conference on Data Mining:

[DKWW11] Yuxiao Dong, Qing Ke, Bai Wang, and Bin Wu. Link prediction based on local
information. In 2011 International Conference on Advances in Social Networks
Analysis and Mining (ASONAM), pages 382–386, 2011.

[DLLL12] Dongsheng Duan, Yuhua Li, Ruixuan Li, and Zhengding Lu. Incremental k-clique
147, 2012.

[DoMr96] Patrick Doreian and Andrej Mrvar. A partitioning approach to structural balance.

[DoMr09] Patrick Doreian and Andrej Mrvar. Partitioning signed social networks. Social Net-

[Doma15] Marven von Domarus. Pregel: Parallel implementation of overlapping community
detection algorithms. Bachelor’s thesis, RWTH Aachen University, Aachen, Ger-
many, 2015.

[Dori04] Patrick Doriean. Evolution of human signed networks. Metodološki zvezki - Ad-

[DuSt98] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of small-world net-

and relationship. In Extraction et gestion des connaissances (EGC’2012), pages

[EsHa09] Ernesto Estrada and Naomichi Hatano. Communicability graph and commu-
nity structures in complex networks. Applied Mathematics And Computation,


[EvLa10] Tim S. Evans and Renaud Lambiotte. Line graphs of weighted networks for over-


[LKFa07] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graph evolution: Densi-
ification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.


[LiLJ14] Chenlong Liu, Jing Liu, and Zhongzhou Jiang. A multiobjective evolutionary algo-

[LNDe16] Panagiotis Liakos, Alexandros Ntoulas, and Alex Delis. Scalable link community

[LiKl03] David Liben-Nowell and Jon M. Kleinberg. The link prediction problem for social

[LPTo08] Leslie Luthi, Enea Pestelacci, and Marco Tomassini. Cooperation and community


List of Figures

2.1 Community Evolution Events. ........................................... 12
2.2 Theory of balance in signed networks. ............................. 16
3.1 An overview of the related work. ................................. 32
4.1 OCD algorithms on LFR synthetic networks. ...................... 57
4.2 Running times of OCD algorithms on synthetic networks. .... 58
4.3 Performance overview of MEA, SDMID and SPM in signed LFR networks with different network size. ............................. 62
4.4 Performance Overview of MEA, SDMID and SPM in signed LFR networks with different average node degree. ......................... 63
4.5 Performance overview of MEA, SDMID, and SPM in signed LFR networks with different maximum node degree. ....................... 64
4.6 Performance overview of MEA, SDMID and SPM in signed LFR networks with a different fraction of edges sharing with other communities. ................................. 65
4.7 Performance overview of MEA, SDMID and SPM in signed LFR networks with different maximum community size. ....................... 66
4.8 Performance overview of MEA, SDMID and SPM in signed LFR networks with different number of nodes in overlapping communities ......................... 67
4.9 Performance overview of OCD algorithms in signed synthetic networks ................................. 68
4.10 Performance Overview with Different Fractions of Positive Connections between Communities. ........................................... 69
4.11 Comparison of SDMID, SPM, and MEA algorithms based on modularity, frustration, number of standalone and overlapping nodes and time complexity on Wiki-Elec dataset. ........................................... 70
4.12 Evaluation of algorithms through radar diagram ...................... 71
5.1 Comparison of features through HeatMap plot. .................... 79
5.2 Sign prediction errors for Wiki-Elec dataset. ....................... 83
LIST OF FIGURES

5.3 Sign prediction errors for Wiki-RfA dataset. ............................................. 85
6.1 Steps to compute the ratings. ................................................................. 97
6.2 This Figure shows the main steps to compute the rating. ....................... 114
6.3 Sliding window evaluation scheme. ......................................................... 115
6.4 Results of different neighbors in knn graph construction. ...................... 117
6.5 Comparison of similarity metrics .......................................................... 118
6.6 Comparison of different community detection algorithms using rating-based graph construction ................................................... 119
6.7 Comparison of different community detection algorithms using tags-based graph construction .................................................... 120
6.8 Comparison of different time bin values on the tags-based graph construction using TCNSVD ......................................................... 120
6.9 Comparison of different time bin values on the tags-based graph construction using TCNSVD ......................................................... 120
6.10 Prediction accuracy, item ranking precision and recall versus structural properties .......................................................... 121
6.11 Accuracy metrics for the Nature forum. ................................................. 129
6.12 Accuracy metrics for Fitness forum. ....................................................... 129
6.13 Accuracy metrics for computer science forum ...................................... 130
6.14 Hamming distance of binary node rank and cooperativity for Dolphin, Zachary, Swamill and network science datasets. ......................... 134
6.15 Hamming distance of binary node rank and cooperativity for road, Gnutella, protein interaction and email datasets. ................................. 135
6.16 Neighbor rank and cooperativity for mean and variance on network science, Europe road and Email networks. ................................. 152
6.17 Neighbor rank and cooperativity for mean and variance on Dolphin, Swamill and Zachary ................................................................. 153
6.18 Neighbor rank and cooperativity for mean and variance on Gnutella, Swamill Strike and Protein-Protein networks. ................................. 154
6.19 Network Properties versus Cooperativity for OSS forums .................... 155
6.20 Network Properties versus Cooperativity for learning forums ................ 156
6.21 Properties of detected communities by SLPA algorithm versus their respective cooperativity values on OSS Forums ............................. 157
6.22 Properties of detected communities by InfoMap algorithm versus their respective cooperativity values on Learning Forums ....................... 157
6.23 Properties of detected communities by InfoMap algorithm versus their respective cooperativity values on OSS Forums ............................. 158
6.24 Properties of detected communities by SLPA algorithm versus their respective co-operativity values on Learning Forums .................................................. 158
6.25 Properties of detected communities by SSK algorithm versus their respective co-operativity values on Learning Forums .................................................. 159
6.26 Properties of detected communities by SSK algorithm versus their respective co-operativity values on OSS Forums .................................................. 159
6.27 Properties of detected communities by CLiZZ algorithm versus their respective co-operativity values on Learning Forums ........................................... 160
6.28 Properties of detected communities by CLiZZ algorithm versus their respective co-operativity values on OSS Forums .................................................. 160

7.1 Overlapping Community Detection Framework (WebOCD). ............................ 165
7.2 OCD Service Architecture .............................................................................. 166
7.3 Viewer Service Architecture .......................................................................... 167
7.4 Simplified view of the algorithms package. ................................................... 168
7.5 Example Las2peer network ............................................................................ 170
7.6 Learning Layers Project Use Case .................................................................. 174
7.7 Learning Layers Project Services ................................................................. 175
7.8 Seminar Team Organization .......................................................................... 176
7.9 WebOCD in A Seminar Course ..................................................................... 177
7.10 Pregel programming model ........................................................................ 179
List of Tables

3.1 A list of community detection algorithms .............................................. 39
4.1 OCD algorithms symbols overview ...................................................... 53
4.2 Basic statistics of datasets used for evaluation of OCD algorithms ............. 59
4.3 Modularity values for different OCD algorithms on real-world datasets. ........ 59
4.4 Running time of OCD algorithms on real world networks. ....................... 59
5.1 Features for community evolution prediction. ......................................... 75
5.2 Properties of overlapping community detection algorithms. ..................... 76
5.3 List of real-world networks used to extract overlapping community detection properties. ................................................................. 77
5.4 Prediction accuracy on Facebook, Enron and DBLP ................................. 80
5.5 Sign prediction symbols overview ......................................................... 81
5.6 Three sets of features used for sign prediction. ....................................... 82
5.7 Information about datasets in signed networks. ....................................... 86
5.8 Prediction accuracy of classifiers on Wiki-Elec dataset. ............................ 87
5.9 Prediction accuracy of Classifiers on Wiki-RfA dataset. ........................... 87
5.10 Community-based HITS coefficients for Wiki-RfA ................................. 88
5.11 Community-based PageRank coefficients for Wiki-RfA ........................... 89
6.1 Recommender systems symbols overview .............................................. 93
6.2 Overview of Recommender Models ..................................................... 97
6.3 Datasets comparison ............................................................................. 112
6.4 Dataset statistics .................................................................................. 114
6.5 MovieLens dataset snapshot statistics ................................................... 114
6.6 Employed parameter values .................................................................. 115

207
LIST OF TABLES

6.7 Parameter values with best performance of $k$-NN graph construction ........ 116
6.8 Employed model parameters for the evaluation ................................. 118
6.9 Rating prediction accuracy and item ranking precision results. ............... 124
6.10 Running time evaluation results. .................................................. 125
6.11 Performance comparison of the community-aware models with baseline models. 125
6.12 Expert Identification Symbols overview ....................................... 126
6.13 The number of posts, users, questions, answers and the period of stack exchange and Nature forums. .................................................. 129
6.14 Correlation among ranking algorithms on Computer Science forum. ........... 131
6.15 Correlation among ranking algorithms on Nature forum. ....................... 131
6.16 Correlation among ranking algorithms on Fitness forum. ...................... 131
6.17 Cooperation and defection symbols overview ................................... 132
6.18 Payoff matrix for the basic PD game. ............................................ 132
6.19 Datasets used for cooperation and defection simulations. ....................... 133
6.20 LFR Parameters for cooperation and defection simulations ................... 139
6.21 This table shows the basic statistics about open source software development and learning forums. Cooperativity of community structures is investigated on these datasets. .................................................. 139
6.22 URCH Cooperativity Values ....................................................... 141
6.23 STDOCTOR Cooperativity Values ................................................ 142
6.24 Correlations of Network Cooperativity and Network Properties .............. 143
6.25 BioJava Cooperativity Values ..................................................... 144
6.26 JMOL Cooperativity Values ....................................................... 144
6.27 BioPython Cooperativity Values .................................................. 145
6.28 BioPerl Cooperativity Values ..................................................... 145
6.29 LFR160 Cooperativity Values ...................................................... 145
6.30 LFR380 Cooperativity Values ..................................................... 146
6.31 Correlations of community properties and their cooperativity values using InfoMap algorithm .......................................................... 147
6.32 Correlations of community properties and their cooperativity values using SLPA algorithm .......................................................... 148
6.33 Correlations of community properties and their cooperativity values using SSK algorithm .......................................................... 148
6.34 Correlations of community properties and their cooperativity values using CLiZZ algorithm .......................................................... 149
7.1 Extensible System Components. ............................................. 169
7.2 List of algorithms, metrics and measures in OCD service .......... 171
7.3 Responses to the system evaluation survey. .......................... 173
Appendices
Appendix A

Own Publications

Book Contributions and Journal Publications


Conference Publications


Project Deliverables


Other Publications


Appendix B

Curriculum Vitae

Name: Mohsen Shahriari
Birthday: July 30, 1988
Birth Place: Birjand, Iran
Address: Im Johannistal 2a
D-52064 Aachen, Germany
Phone: +4917643469765
Email: shahriari@dbis.rwth-aachen.de

Language Skills: Persian (native), English (fluent), German (basic)

Professional Experience:
PhD Research & Project Work at Chair of Computer Science 5 (Databases & Information Systems), RWTH Aachen University
01/2014 - 01/2017: Project worker in EU FP7 IP Learning Layers
Research Assistant at Networked Information Processing Lab (NIPL), Computer Engineering Department, Sharif University of Technology
04/2011 - 06/2013

Academic Education:
09/2010 - 09/2012: Information Technology at Computer Engineering Department, Sharif University of Technology (MSc)
09/2006 - 09/2010: Information Technology at Engineering Department, University of Birjand (BSc)

Teaching Experience:
Teaching Assistant of Web Science Seminar: Summer 2014, Summer 2015, Summer 2016