COMMUNITY DETECTION IN GRAPHS

Zheng Gao

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in the Department of Information and Library Science Department,

Indiana University Bloomington

May 2020

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

Xiaozhong Liu, Ph.D. (Principle Advisor)

Xiaofeng Wang, Ph.D.

Yong-Yeol Ahn, Ph.D.

Kahyun Choi, Ph.D.

Date of Defense: 05/15/2020

ACKNOWLEDGMENT

Thank you letter.

Zheng Gao

COMMUNITY DETECTION IN GRAPHS

Community detection has always been one of the fundamental research questions in graph mining. As a type of unsupervised or semi-supervised approach, community detection aims to explore graphs hidden patterns merely relying on graph topological structure. By grouping similar nodes or edges into the same community while separate dissimilar ones apart to different communities, graph structure can be revealed in a coarser resolution. It can be beneficial for numerous applications such as user shopping recommendation and advertisement in e-commerce, protein-protein interaction prediction in biomedical domain, and literature recommendation or scholar collaboration in citation analysis.

However, identifying communities is an ill-defined problem. Due to the No Free Lunch theorem, there is neither gold standard to represent perfect community partition nor universal methods which are able to detect satisfied communities for all community detection tasks in various types of graphs. To have a global view of this research question, I summarize state-of-art community detection methods by categorizing them based on graph types, research tasks and utilized method tracks. As academic researches for community detection grows rapidly in recent years, I hereby particularly focus on the works published in the recent decade, which may leave out some classic models published decades ago.

Meanwhile, three subtle community detection tasks are proposed and assessed in this dissertation as well. First, apart from general models which only consider graph structure for community partition, personalized community detection considers user need as auxiliary information. In the end, there will be higher resolution communities for nodes better matching user needs while coarser resolution communities for those less relevant nodes. Second, graphs always suffer from sparse connectivity issue. Leveraging conventional models directly on such graphs may hugely distort the quality of generate communities.

To tackle such problem, cross-graph techniques are involved to propagate external graph information as a support for current graph community detection. Third, as a type of unsupervised or semi-supervised learning method, community partition results are lack of explanations. The final output is only nodes/edges clusters with no further in-depth interpretation. By involving natural language processing techniques, a joint learning model is applied to not only detect node communities but also generate a short descriptions for node intrinsic characteristics.

The contribution of this dissertation is threefold. First, a decent amount of recent works are summarized under a well defined taxonomy. Existing works about methods, evaluation and applications are all addressed in a proper order. Second, three novel community detection tasks are proposed and their associated models are evaluated by comparing with alternative baselines under various datasets. Third, the limitations of current works and several future tracks with potential academic and industrial value are also discussed in the final chapter.

TABLE OF CONTENTS

Curriculum Vitae

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

focus on community detection. introduce the overall development in the recent decade and particularly address three subtle tasks.

summary or empirical studies. graph types: heterogeneous, directed, weighted, sparse, temporal(time involving), hyper graph,multi-layer,large scale different tasks: math proof (threshold), overlapping, number of communities, top-k community search// subgraph, Enhancing community detection, semi-supervised community detection applications: biology, citation analysis, transportation, social network methods: Modularity, Spectral, Stochastic block model, deep learning, nmf,Multiscale (hierarchical), multi-view, with enriched information, community recovery, flow based(random walk), link based, motif, others challenges: resolution limit, unbalanced groups, evaluation, software, datasets

so far community detection is more like a statistic/ physics questions instead of computer science and arificial intelligence question. modularity -¿ sbm -¿ deep learning. modularity and sbm are more clear tasks. deep learning methods are fuzzy. spectral clustering all the time

future tracks: from computer science perspective, deep learning, random graph? hyper graph? multiscale?

findings: biology has delay, not easy to get a lot citations in short time.

concept level difference between subgraph, community, motif, cliques: community is from node side, others are from graph side. it is a part of community search.

overlapping has huge correlation with nmf spetral clustering and sbm are highly correlated

some tracks are huge: modularity (newman), stochastic block model (newman) and overlapping (Jure) try my best to distinguish each category, but there are still some overlaps between them. Some algorithms are both for overlapping community detection using deep learning techniques.

mention other types of approaches a bit, such as link community detection,

summarize some trend, well known professor/teams, by years. which venue tends to be more popular.

I manually code five hundred most influential papers in community detection domain. [1]

1.1 Motivation

why I do community detection? Network Science, which is also called graph mining or complex network analysis, is one of the fundamental research questions in artificial intelligence.

1.2 Problem Definition

what is community detection, wrap up in mathematical formulas

1.3 Research Questions

raise the three sub topics mentioned in the later experiements.

1.4 Contribution

explain the contribution of this thesis such as categorizing the community detection into several forms, solve the three research questions, etc.

1.5 Thesis Structure

each chapter tells what.

CHAPTER 2
RELATED WORKS

[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12],[13],[14],

2.1 Graph Types

2.1.1 Heterogeneous & Multi-layer Graph

hetero: [15],[16],[17],[18],[19]incomplete graph, [20],[21], multi-layer: [12] survey,[22],[23],[24],[25]

2.1.2 Sparse Graph

[26],[27],[28],[29],[30],[31],[32],

2.1.3 Dynamic Graph

[33],[34],[35],[36],[37],[38],[39],[40],[41],[42],[43],[44],[45],[46],[47],

2.1.4 Large Graph

[4](survey),[48],[49],[50],[51],[52],[53],[54],[55],[56],[57],[58],[59],[60],[61],[62],[63]

2.1.5 Attribute Graph

[64],[65],[66],[67],[68],[69],[70],[71],[72],[73],[74]

2.1.6 Summary

2.2 Tasks

2.2.1 Overlapping Community Detection

[13] survey,[9], [75], [76],[15],[77],[78],[79],[80],[81],[82],[83],[84],[85],[86],[87],[88]

2.2.2 Number of Communities

[89],[90],[91],[92]

2.2.3 Community Search

[93],[94],[95],[96],[97],[98],[41],[99],[100],[101],[102]

2.2.4 Community Detection Enhancement

[103],[104],[105],[106],[107],[108],[109],[110],[111]

2.2.5 Semi-Supervised Community Detection

[65](duplicate with attribute graph),[112],[113],[114],[115],[116],[117],[118],[119],[120]

2.2.6 Summary

2.3 Main Track Methods

spectral clustering and stochastic block model relationship. some other tracks such as multi-objective, multi-scale community detection, joint training methods.

2.3.1 Modularity

[121],[122],[123],[124],[125],[126],[127],[128],[129],[130],[131],[132],[133],[134],

2.3.2 Spectral Clustering

[8](survey),[135],[136],[137],[26],[138],[139],[140],[141],[142],[143],[144],[145],[146],[63]

2.3.3 Stochastic Block Model

[5] (survey),[147],[18],[148],[149],[150],[151],[152], [153],[154],[155],[156][157],[158], [159],[160],[161],[162],[163],[164],[165],[166],[167],

2.3.4 Deep Learning

[168],[169],[170],[171],[172],[173],[174],[175],[176],[177],[55],[124],[178],[65](also for attribute graph),[64],(also for attribute graph)

2.3.5 Matrix Factorization

[179],[180],[76],[181],[182],[183], [184],[185],[186],[187],[188],[189], [190],[117],[191],[192],[193]

2.3.6 Flow-Based

random walk can solve ocal search community problem. some potts model, heat kernel can also be categorized here. [194],[195],[196],[197],[198],[199], [200],,[201],[202],[203], [204],[205],[206],[207],[208],

2.3.7 Summary

2.4 Applications

2.4.1 Interdisciplinary Supports

Social Media

[209],[210],[211],[212],[213],[214],[215],[216],[217],[218],[219],[220],[221],[222],[223],[224],

Miscellaneous Domains

[225],[226],[227],[228],[229] ,[230],[231],[232],[233],[234],

2.4.2 Datasets and Packages

[235] [236],[237],[238],[239],[240],[241],[242],[243],[244],[245],

Even we understand how those state-of-art algorithms work, it is still not enough for us to apply those algorithms as they are usually too complicated to implement in a short time.

Hence, it is necessary to have some open source softwares to make it easier for us to run some baseline algorithms or use those algorithms to better understand our networks. Github is one of the largest code repository so far in the world, and many free softwares always have either their own websites or deposit their code on Github. Based on our empirical study, we find some of the softwares are reliable to support solving community detection problems.

For those packages and softwares, embedded algorithm is one key factor and visualization is another key factor to evaluate whether a package is good or not. Because as we know that the network is unstructured and messy. Sometimes even though we get a community partition result, we still can't tell whether the result is good or bad intuitively. Hence, if there is a way to visualize the community partition result, we can better judge whether our model is capable or not from our eyes directly. In this section, all softwares introduced her either have a Graphical User Interfaces (GUIs), or are written scripting/ programming languages. The GUI based packages are easier to learn, whereas the scripting tools are more powerful and extensible. Well-known and well-documented GUI packages include NetMiner, UCINet, Pajek , GUESS, ORA, Cytoscape, muxViz (opensource) and Gephi. Popularly used scripting tools include R statistical programming language, NetMiner with Python scripting engine, igraph (packages for R, Python, C/C++), the NetworkX library for Python, and the Social Network Analysis Package (SNAP) for large-scale network analysis in C++ and Python. These tools are also well-documented. Even though some of these packages have much functionality and features than others, they are more difficult to learn compared with GUIs.

Graphical User Interfaces

Pajek

Pajek [240] is a public network analysis program particularly for analysis and visualization of very large graphs. It is free and can be downloaded publicly from its official website

[1] for non-commercial use. It contains tools for analysing and visualizing various of graphs networks from different domains, such as visualizing the collaboration relationship between academic authors (collaboration networks), social relationship between online users (Internet networks), checking for related academic works (citation networks), showing the protein-receptor interaction for healthcare usage and many. In its GUI, it offers some default datasets which can be used for testing and visualizing those networks.

In Pajak, it aims to divide a large graph into several smaller modules where nodes with closer distances can be located near each other. After that, many advanced and complicated methods can be further utilized on smaller components which manually or automatically detected by users via Pajak . What's more, It helps users with less technical background to understand and analysis networks with complex structure. Many advanced and powerful visualization tools and graph analysis methods are all implemented in the package.

It allows up to run many detailed analysis to explore how the graphs look like in terms of many important metrics, such as the number of triangles in the graph, the modularity coefficient of the graph, etc. By detecting community structures and offering many modularity or block model based methods. It can also help to identify the node importance in graphs via Page Rank models. The methods offered in Pajek is especially good for large sparse networks.

Many visualization based models or metrics are also embedded in the packages. It can help to better render the graph into a 2-D plot so that to directly visualize the graph to users. Moreover, it offers some methods to calculate basic metrics to assess the fitness or closeness of the graph structure as well as community structure. For example, in one of its embedded algorithm, it finds communities in the network using Louvain method and Visualization Of Similarities (VOS). Here we show the interface of the Pajek software in Figure **??**.

NetMiner

[1]http://vlado.fmf.uni-lj.si/pub/networks/pajek/

Cyram NetMiner [241] is a software tool for exploratory analysis and visualization of network data by integrating standard social network analysis (SNA) methodologies with modern visualization network techniques. It is more or less similar with Pajek software but has more interactive function than Pajek. Via this software, it is easier to help users better the graph underlying pattern and network structure in social network. Unfortunately, it is a paid service. In its interactive interface, Cyram NetMiner can be used for social network analysis, teaching and presentation. It can be used under different business related scenarios where its network analysis tools can help to extract and understand the business factors. Hence, it is more business oriented software so that companies can use this software to testify their commercial dataset. Overall, the Cyram NetMiner has the following features [2]:

- It supports large network analysis. Networks with thousands of nodes can be easily handled by the software.
- The network can be explained by the embedded network metrics such as degree-related coefficients.
- It can also help to make some predictions on the change of networks. Such as how the communities are changed associated with the edge dynamic changes.
- It can also support exploratory network analysis, meaning to visualize the networks into 2-D plots with various of ways.
- Some of the classic models and measurement metrics have been embedded in the software already.
- Robust data management
- The workflow of the whole network analysis is easy to reproduce

[2]http://www.netminer.com/main/main-read.do

- The interface is user friendly
- The analytics is Interactive, user can get the updated results immediately by changing some of the parameter settings
- It also offer some built-in statistical charts to help summarize the analysis results.

NetMiner implements five modularity and imformation propagation algorithms for discovering community structure such as Edge betweenness, Blondel, Eigen vector, Label propagation and Modularity. All of these methods are introduced in the previous section. A brief screenshot to illustrate the software can be found at Figure **??**.

CFinder

CFinder is a graphic tool for network cluster (community) detection based on Clique Percolation Method (CPM) [**kumpula2008sequential**]. It is a tool particularly good for detecting overlapping and Top-K communities. CFinder [3] uses the Clique Percolation Method for detecting k-clique communities [**palla2005uncovering**]. Hence, it focuses on a particular sub-domain of the whole community detection algorithms. K-clique partition is also a key topic in the community detection domain. And a k-clique community is a combination of all k-cliques that can be reached from each other via a series of adjacent k-cliques [242]. In k-clique algorithms, whole network can be disjoint and separated into several blocks. And CFinder detects the same communities of a subgraph that whether the subgraph is linked to a large network or not. For heterogeneous network, it divides it into homogeneous subnetworks and applies the method to them separately [**bartaonline**]. For visualizing graphs, CFinder uses Spring layouts. By adjusting the visualization settings in the Tools menu, the visualization interface provides different views of the network such as the cliques, basic statistics of the graph and the community detection results of the graph. In a word, the software is also good at its interactive functions with users, which makes it widely used by researchers and have a good reputation. A broad view of the software

[3]http://www.cfinder.org/

interface can be found here in Figure **??**.

UCINet

UCINET [4] for Windows [244] is a software package for the analysis of social network data, which was developed by Lin Freeman, Martin Everett and Steve Borgatti. It comes with the NetDraw network visualization tool. It only has a Windows version of the software, but it still keeps updating now. Users can download either the 32-bit or 64-bit version of UCINET. The 32-bit version is the standard version and runs on both 32-bit and 64-bit Windows. The 64-bit version is limited in that it does not have all of the functions of the 32-bit version. From its official website, it is said the 64-bit version often crashes. Therefore, it is suggested to use the 32-bit version. It is not a free software so that users need to purchase it if they are willing to use. UCINET is a comprehensive package for the analysis of social network data. It can generate many statistics of the graphs such as the mean and variance on node degrees. It can handle various kind of data format as the input such as Excel files. From its official announcement, it claims that it is able to handle a maximum of 32,767 nodes (with some exceptions) although practically speaking many procedures get too slow around 5,000 - 10,000 nodes. Social network analysis methods embedded in the software include centrality measures, subgroup identification, role analysis, elementary graph theory, and permutation-based statistical analysis. In addition, the package has strong matrix analysis routines, such as matrix algebra and multivariate statistics. Moreover, it is able to draw graphs into 2-D plots as well. But based on the free trials on the software, it does not perform well and is not easy to handle as other software such Pajek. The Figure **??** shows the interface of this software.

GUESS

GUESS [**adar2007softguess**], as mentioned in its official website[5], is an exploratory data analysis and visualization tool for graphs and networks. The software also contains a script language called Gython to handle large graph data. Gython, or named as Jpython,

[4]https://sites.google.com/site/ucinetsoftware/home
[5]http://graphexploration.cond.org/

is a connector between Java and Python, which allows to use both Java and Python to run the data analysis functions and visualize the graph result in Java Applet. An interactive interface is also offered to generate community detection results. The interface of GUESS visualization a window can be zoomed in and out to adapt graph size for a better visualization experience. The visualization part is based on Piccolo [6], a visualization tool designed by University of Maryland.

GUESS also supports dynamic and time sensitive graphs to check the dynamic changes of graphs. The input format of the software can be various, such standard graph formats (Pajek, GML). It can also export the results to various of formats as images (GIF, PNG, EPS, PDF, JPG, SVG...)

By taking the advantage of JUNG, as well as other software (HSQLDB, RServe , etc), GUESS support various layout algorithms and graph analysis functions run in terminal. Hence, it has various kinds of scripting language versions. The original version of the software is written in Java and the lasted version is in 2007. An overview and brief tutorial of the software can be found on YouTube [7]. However, the software is no longer update, which causes fewer people to use it anymore. But it still offer some good insights of community detection visualization and analysis, and is particularly friendly to users with few technical background. People need to type related commands to the terminals, and the graph interface will be automatically popped out. Figure **??** shows the layouts of the GUI.

ORA-LITE

ORA-LITE [245] is a graph evaluation tookit good use for dynamic graphs as well as a network analysis package developed by CASOS at Carnegie Mellon University [8]. It is a software or a tool mainly used for dynamic networks, which means the networks they analyze changes over time or other attributes. As it claims in the official webpage, ORA-LITE can handle multi-mode, multi-plex, multi-level networks.It can not only find out

[6]http://www.cs.umd.edu/hcil/piccolo/

[7]https://www.youtube.com/watch?time_continue=227&v$\bar{\text{T}}$WMW8CZqAX8

[8]http://www.casos.cs.cmu.edu/projects/ora/

the community structure of those dynamic networks but also other basic patterns of the networks. Although the calculation time complexity is usually high in dynamic networks, this software is able to handle super large scale networks. Based on the announcement from the official website, the ORA-LITE is itself limited to a maximum of 2,000 nodes per entity class. But the total number of nodes can be up to a million, which is pretty large compared with other software. One good part of this software is that it offers multi-language introduction tutorial page and even have google groups in which users can ask questions and exchange ideas. Another advantage of this software is that the version of it keeps updating. The latest version is in 2019. But it only has Windows version of the software. A screen shot of a demo graph generated via the software can be seen here in Figure **??**.

Cytoscape

Cytoscape [236] is an open source software platform for visualizing molecular interaction networks. It is a very famous network analysis tools for healthcare data with really powerful functions [9]. Although Cytoscape was originally designed for biological research, now it is a general platform for complex network analysis and visualization. Inside its software, there are a basic set of features for data analysis and visualization. Moreover, it offers many plugins added to the basic version software. It is originally written in Java and JS, which allows to add extra plugins to make better visualization with various types of layouts for graph analysis. It offers very complex data flow analysis on graphs and detect node pattern and relationships with embedded functions such as link prediction and node community detection .It can also connected with databases remotely and visualized in html via their designed Cytoscape.js. Cytoscape even has an App Store where you can download those plugins. It has Mac version and is developed based on Java. This software is pretty active now and keeps updating all the time. Based on its official website recorded statistics, Cytoscape downloads in 2017 is average 15,600 per month (512 per day) and in 2018 is

[9]http://www.cytoscape.org/

average 17,600 per month (586 per day). A screenshot of the software is shown in Figure **??**.

muxViz

MuxViz [237] is a framework for the multilayer analysis and visualization of networks [10]. It allows an interactive visualization and exploration of multilayer networks, i.e., graphs where nodes connecetd with multi-type relationships or with multi-type attributes simultaneously. Hence, the most outstanding feature of this software is that it can handle multiple relationship at the same time while most of the other softwares are only capable to analyze homogeneous networks. It is suitable for the analysis of social networks exhibiting relationships of different type (e.g., social relationship and professional relationships), interactions on different platforms (Twitter, Facebook, etc), or biological networks characterized by different type of interactions (e.g., electric, chemical, etc. Another advantage of this software is that MuxViz is based on R and GNU Octave, running on Windows, Linux and Mac OS X. It is open-source and free. A demo of this software is in Figure **??**.

Gephi

Gephi [11] is a leading visualization and exploration tool for social networks. The goal of Gephi is to make better analysis to find patterns, separate structure to uncover graph structures and visualize the result in an intuitive way to make it easier for users to understand. In its GUI, it has a visualization window along with a bunch of functional buttons. By clicking on the buttons or change the values inside the related boxes, it can directly change the visualization layout appeared in the canvas window. Actually it also has a Python library associated with it. However, empirically using the software directly is more convenient and a better choice for users. Moreover, one advantage of the software is that dynamic networks can be also conveniently explored in Gephi. In Gephi, the default mode for network layout is 2-D, even though it uses 3-D rendering engine [243].

In Gephi, communities can be detected by Louvain method. i.e., modularity method.

[10]http://muxviz.net/gallery.php
[11]https://gephi.org/

There are also a bunch of other algorithms embedded in this software. Users can even have a choice to try different rendering ways and choose one of them as the ideal one. In addition to supporting various graph layouts, it has Antialiasing, which is a visualization choice which models the edges looks smoother. The software is pretty new and until now, it is still updated frequently, which makes it the most popular visualization and analysis tools for graph mining and community detection. The following Figure **??** shows the interface of the Gephi software in a vivid way.

Visone

Visone [**baur2001visone**] is a tool for analyzing and visualizing social networks [12]. It is software for the visual creation, analysis, transformation, exploration, and representation of network data [**brandes2004analysis**]. It is written in Java and can be run from terminal. Visone implements many algorithms as well, such as Louvain clustering for discovering communities.The visualization layouts supported by Visone are Spring, Circular, Random and Spectral. It defines that the features inside Visone:

- It is an interactive graphical user interface, which particularly designed for social network analysis.
- It has many version written in Java for Windows, Linux, and MacOS.
- The input format of graph data is in standard format including Pajek, GraphML or Excel.
- The generated imges can be saved in various fomats including JPEG, PDF, SVG, Metafile, and others.

The visualized output and analysis result of the software can be directly used for academic paper writing, which is pretty convenient for researchers with limited programming background. From its official website, the latest version of this software is in 2016. So

[12]https://visone.info/

although it has some comprehensive functions, the algorithms embedded are a little bit out of date. However, for some baseline algorithm test or some basic metrics for the network in order to retrieve some overall impressions of the graph, it is still a good tool to use. For a demo of this software, a figure is shown in Figure **??**.

Summary

In this section, we introduce almost all available GUI software for network analysis. Some of them are free to use while the rest are paid services. We introduce eight software in total. Among all of them, we suggest to use Gephi and Pajek for use. If you are a biological domain researcher, we suggest Cytoscape instead. These recommended ones are all free software with powerful functions. They are enough to solve community detection problems on large scale graph with thousands of nodes. Moreover, Pajek is also a well known graph format and Gephi has a Python package to use. Empirically, the network figures draw by Gephi is better than the rest GUI tools. The manual of recommended softwares are more detailed than the rest software as well. Therefore, Gephi and Pajek are the two recommended GUI to serve as network community detection tools. While for other graph related analysis, some other graph related methods such as Page Rank or graph statistic analysis metrics such as node degree distribution are also good to use.

Script Language Tools

Besides softwares, actually there are more packages for community detection. As compared with software, scripting languages are more flexible and able to generate better visualization results as well as to run the algorithms faster. Hence, in this section, we include and overview some of the widely used scripting languages in community detection. Most of the packages are written in R, Python, Java and C, which are most basic and hottest programming languages used in nowadays.

R packages

R is the most popular programming language in statistics as there are many complex models are written in R language. However, one drawback of R language is that it can't handle as huge data input as Java or C. And the running speed is much slower than other programming languages mentioned above. However, if your dataset is not that large (hundreds of nodes for instance), many R packages can actually perform really well.

modMax [13] is a good R packages that contains many algorithms related to Modularity. The algorithms implemented here are used to detect the community structure of a network. These algorithms follow different approaches, but are all based on the concept of modularity maximization. Algorithms inside include genetic algorithm, fast greedy, simulated annealing, spectral optimization and local modularity methods.

networktools [14] is another new R package that uses for network analysis. The functions inside include not only basic community detection algorithms but also have visualization functions within.

RANN [15] finds the k nearest neighbours for every point in a given dataset in $O(NlogN)$ time using Arya and Mount's ANN library (v1.1.3).

igraph

igraph [239] is one of the most widely used packages for community detection[16]. Actually this package is particularly designed for solving community detection problem. Another reason why igraph is that popular is that the package have all R, Python and C version. So no matter what kind of programming language users are familiar with, there is always a version of igraph suitable for users. This package used to be frequently updated. However, in recent years, it seems quite silent. The latest version of igraph is in 2015. But there are still a bunch of widely used community detection algorithms inside the package. So far, algorithms like Optimal Modularity, Edge-Betweenness, Leading Eigenvector, Fast-Greedy, Multi-Level, Walktrap, Label Propagation, and InfoMAP are all in the packages.

[13]https://cran.r-project.org/web/packages/modMax/modMax.pdf
[14]https://cran.r-project.org/web/packages/networktools/networktools.pdf
[15]https://github.com/jefferis/RANN
[16]http://igraph.org/redirect.html

NetworkX

NetworkX [238] is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks[17]. It is not particularly designed for community detection and actually it is a general framework for social network analysis. It is more like a foundation package of the graph mining and many other community detection packages are built based on that. However, within the package, it still has some community detection related functions. Inside the package, there are sub-components named 'centrality','clique','clustering' in which those functions all talk about community detection related algorithms or evalution metrics.

SNAP

SNAP [**sosic2015large**] is a general purpose network analysis and graph mining library[18]. It is written in C++ and easily scales to massive networks with hundreds of millions of nodes. So far the package has both C++ and Python version. But the Python version is just like an interface to call the C++ functions. Hence, it is so far the quickest packages for community detection in terms of running speed. Another advantage of this package is that the algorithms inside keep updating all the time. So far most of the state-of-art community detection algorithms are inside the package. Within the package, the community detection algorithms include overlapping communities, dynamic network community detection, and Modularity based community detection. There are tens of algorithms within the official website[19].

Summary

In this section, we introduce some of the script langauge packages used for community detection. Actually besides these packags, some GUI softwares also offer script language packages as well such as Gephi and Cytoscape. One strong advantage of script languages

[17]https://networkx.github.io/
[18]http://snap.stanford.edu/index.html
[19]http://snap.stanford.edu/snap/description.html

is they are more flexible to use compared to GUIs. Although GUIs do not require strong technical background and what users do with softwares is just clicking buttons. However, that pattern is standard and strictly limited by the software design itself. Therefore, it is hard to create some personalized layouts using GUIs. While script languages are more easier to combine with other programs to generate results more suitable for personal desire. Among all introduced packages, we can see they are written in three programming languages. All R based package are written in R; igraph and networkX are written in Python and SNAP is written in C++. NetworkX is one of the fundamental packages in network analysis and we suggest it is a required package to use if your are a Python user.

2.4.3 Summary

2.5 Evaluation

Once the community partition result is retrieved by taking a certain type of model, a subsequent approach is to evaluate model performance and run comparisons among different models. Therefore, how to carry out comparative analysis and what metrics should be reported as benchmarks are two imperative factors to be considered. In the following sections, I will address these two questions separately by introducing several highly cited papers about comparative analysis and a number of best known metrics widely used in industry and academia.

2.5.1 Comparative Studies

[246] compares 13 community scoring functions and employ them on 230 graph datasets with various topics (social, collaboration, and purchasing graphs, etc.) and different sizes (from hundreds of thousand to hundreds of millions of nodes and edges in graphs) to generate communities. The 13 scoring functions lie in four categories including Internal Connectivity, External Connectivity, the combination of Internal & External Connectivity and Network Model (Modularity). To assess these scoring functions' community adequacy,

four metrics are thereafter to considered, including separability, density, cohesiveness and clustering coefficient (all these metrics will be explained in the later section). There are significant performance differences among all scoring functions where Conductance (it measures the ratio of edges linking outside of the community) and Triad participation ratio (the ratio of nodes in the same cluster forms a triad) achieves the best performance, while Modularity score is with poor performance.

[247] argues that current community detection methods totally ignore the topological nature of the communities as two methods may have similar evaluation performance but totally different community distributions. It claims that adding community topological metrics such as community-wise density, average distance, internal transitivity, and hub dominance may give more supportive understandings about the community partition result. Moreover, by testing on several real-world graph datasets as well as synthetic graphs using different types of approaches (e.g., Modularity-based and diffusion-based approaches), it also empirically shows that there is no clear correlation between the method performance metrics (e.g., rand index and normalized mutual information) and community topological metrics.

[248] questions the appropriateness to simply use the node metadata information (e.g., node category or membership) as the criteria to build up the ground truth. By running 11 different community detection models on 16 different graph datasets, it claims that there is a trivial correlation between the communities constructed by graph metadata and model-detected communities. Thus, it concludes that metadata groups are not able to fully infer graph topological structure as they are separate views for the graphs. Instead, graph metadata can be taken as either auxiliary information to enhance community partition performance or another perspective to interpret graphs.

By employing 8 different models on Lancichinetti-Fortunato-Radicchi (LFR) benchmark graph, [249] summarizes how well these models can handle graph scale, community size recovery problems. All model performance and computing time are also assessed and compared with each other. In the end, the paper offers some insights about how to choose the proper model given graph descriptive parameters like LFR mixing parameter.

[250] analyzes communities by taking account of a broadspectrum of structural properties. The analysis reveals nu-ances of the structure of real and extracted communities

We extract dierent classes of communities thatcan be grouped into two categories: intrinsically-dened andextrinsically-dened communities.

a large set of structural properties and ten dierentcommunity detection procedures to produce examples of dif-ferent structural classes.

[251]

2.5.2 Metrics

[10](survey),[252],[253],[254],[255],[256],[257],[258],[259],[260],

In this chapter we discuss how to evaluate the generated community quality, which is also called community evaluation [10]. Typically, a network can be divided into several subgraphs with explicit or implicit relationships. Users need to decide whether the generated community is of good quality and is the most appropriate as a clustering result. Typically, there are some widely accepted metrics such as clustering accuracy are used for community evaluation. [**lancichinetti2008benchmark**] offers a bunch of graph algorithms as well as metrics to be used ad benchmark in community detection model comparison. However, in some cases, not all widely accepted metrics are equally important to evaluate the community quality because the research questions are different [6]. For example, to solve the dynamic network problems, time related indicator should be more important for model assessment. And for solving large scale network community problems, the performance of an algorithm in terms of time complexity is also a crucial factor.

Since a bunch of community algorithms have been proposed for both overlapping and non-overlapping networks, indicators used to solve the community detection problems on two different types of graphs are calculated in the different ways. In other words, we should separately discuss the metrics used in terms of different types of networks. In most of the cases, we have to know the real community structure (ground truth) for tested dataset, so that we can compare our model results with other baseline methods, which is the widely applied approach for model evaluation.

In this chapter, although there are many taxonomies for metric categorization, such as metrics for undirected or directed networks, we classify and report the evaluation metrics as overlapping metrics and non-overlapping metrics [**schaeffer2007graph**, 6]. While some of the metrics are common in the two scenarios such as the number of triangles in community [261, 262], we still think it is the best way to separate metrics in those categories.

In the following two sections, we discuss the metrics used to measure the similarity between the detected and the ground-truth community structures for either non-overlapping or overlapping community detection approaches. In fact, most of existing communities focus on non-overlapping community detection, as it is a simpler scenario, some studies do point out the significance of overlapping community evaluation, mostly in a series of papers published in SNAP [20]. Within those papers, many overlapping metrics are introduced.

Non-overlapping Metrics

In this section, we discuss the metrics used to measure the similarity between the detected and the ground-truth community structures for non-overlapping community structure.

We define, given a graph $G(V, E)$, $\Omega = \{w_1, w_2, ...w_k\}$ is the set of the detected communities users generated with some non-overlapping community detection models. And $C = \{c_1, c_2, ..., c_k\}$ is the set of the ground truth communities given in advance. $N = |V| = \sum_{k \in K} |w_k| = \sum_{j \in J} |c_j|$ is the total number of the nodes in the original graph. $|N_{c_j}| = |c_j|$ and $|N_{w_i, c_j}| = |w_i \cap c_j|$ refers to the number of nodes in each community as well as the number of common nodes detected in the two communities from both graph partition C and Ω.

[20]http://snap.stanford.edu/index.html

Metric	**Expression**
Purity	$\frac{1}{N}\sum_k \text{argmax}_j \lvert w_k, c_j \rvert$
F-measure	$\frac{2*Purity(\Omega,C)*Purity(C,\Omega)}{*Purity(\Omega,C)+Purity(C,\Omega)}$
Rand index	$\frac{TP+TN}{TP+FP+FN+TN}$
F_β, precision P, recall R	$\frac{(\beta^2+1)PR}{\beta^2(P+R)}$; $P = \frac{TP}{TP+FP}$;$R = \frac{TP}{TP+FN}$
Adjusted rand index	$\frac{\sum_{ij}\binom{N_{w_i c_j}}{2}-\sum_i\binom{N_{w_i}}{2}\sum_j\binom{N_{c_j}}{2}/\binom{N}{2}}{1/2*\sum_i\binom{N_{w_i}}{2}+\sum_j\binom{N_{c_j}}{2}--\sum_i\binom{N_{w_i}}{2}\sum_j\binom{N_{c_j}}{2}}$
Entropy	$H(\Omega) = -\sum_k \frac{\lvert w_k \rvert}{N} log \frac{w_k}{N}$
Normalized mutual information	$\frac{I(\Omega,C)}{[H(\Omega)+H(C)]/2}$; $I(\Omega, C) = \sum_k \sum_j \frac{\lvert w_k \rvert \cap c_j}{N} log \frac{N \lvert w_k \cap c_j \rvert}{\lvert w_k \rvert \lvert c_j \rvert}$
Variation of information	$\sum_{i,j} r_{i,j}[log(r_{ij}/p_i) + log(r_{ij}/q_j)]$= $H(\Omega) + H(C) - 2I(\Omega, C)$
Modified purity	$\sum_i \sum_{u \in w_i} \frac{w_u}{w} Purity(u, \Omega, C)$
Modified ARI	$\frac{\sum_{ij} W(w_i \cap W(c_j)) - \sum_j W(w_i)W(c_j)/W(V)}{\frac{1}{2}(\sum_i W(w_i + \sum_j W(c_j))) - \sum_i W(w_i)\sum_j W(W(c_j))/W(s)}$

Table 2.1: Evaluation metrics for non-overlapping community detection results

After we define some basic symbols to represent the graph community detection results, we can use the following evaluation metrics to judge whether our predicted community partition is good or bad to estimate the ground truth community results. The whole introduced evaluation metrics for non-overlapping community detection can be seen from Table 2.1. In the following paragraphs, we start to introduce each metric one by one.

Separability measures the ratio between the internal and the external number of edges of node set S: $g(S) = \frac{m_S}{c_S}$

Density builds on intuition that good communities are well connected. One way to capture this is to characterize the fraction of the edges (out of all possible edges) that appear between the nodes in S, $g_S = \frac{m_S}{n_S(n_S-1)/2}$.

Clustering coefficient is based on the premise that network communities are manifestations of locally inhomogeneous distributions of edges, because pairs of nodes with common neighbors are more likely to be connected with each other.

Cohesiveness characterizes the internal structure of the community. Intuitively, a good community should be internally well and evenly connected, i.e., it should be relatively hard to split a community into two sub communities. We characterize this by the conductance of the internal cut. Formally,$g(S) = max_{S' \in S}\phi(S)$ where $\phi(S')$ is the conductance of

$S^{'}$measured in the induced subgraph by S. Intuitively,conductance measures the ratio of the edges in $S^{'}$ that point outside the set and the edges inside the set $S^{'}$. A good community should have high cohesiveness (high internal conductance) as it should require deleting many edges before the community would be internally split into disconnected components

Purity

Purity [**schutze2008introduction**, **lin2005foundations**] is introduced where each detected community is assigned to the ground-truth label which is most frequent in the community. It is calculated by the ratio of nodes that are correctly assigned labels in the community. The formula to calculate it can be defined as:

$$Purity(\Omega, C) = \frac{1}{N}\sum_{k} max|w_k, c_j|$$

The purity score is in range from 0 -1. 1 means the detected community is perfectly matched with the ground truth label. And the 0 means the opposite. From the definition, it is easy to find that the fomula is not symetric. Which means that $Purity(\Omega, C)$ and $Purity(C, \Omega)$ is different. In practice, $Purity(\Omega, C)$ is usually used and called as the simple purity. While the $Purity(C, \Omega)$ is called the reverse purity [**artiles2007semeval**, 263]. As the ground truth label is always known in advance for evaluation, hence the previous one makes more sense as we use the known labels to assign for predicted communities. However, purity is really sensitive to the number of communities for the original graph. If the number of communities increases, the purity score is more likely to be higher. The extreme case is that purity will be 1 if every node is assigned to the community itself only. While the case of inverse purity is the opposite. The most extreme case happens when all the nodes are assigned to the same community. Hence, purity iteself is not an ideal evaluation metric to judge whether the community partition is good or bad, especially when the number of clusters/ communities is unknown or undecided.

F-Measure

To solve this problem, Newman [**newman2004finding**] introduced an additional constraint generally adopted in cluster analysis rather consists in processing the F-Measure, which is the harmonic mean of both the versions of the purity.

$$F = \frac{2 * Purity(\Omega, C) * Purity(C, \Omega)}{*Purity(\Omega, C) + Purity(C, \Omega)}$$

This formula considers both of the purity and the inverse purity, which can balance the two metrics at one time.

Rand index

We can also regard the community detection problem as a classification problem [**hubert1985comparin**
A true positive (TP) decision assigns two same-labeled nodes to the same community; a true negative (TN) decision assigns two different labeled nodes to different communities. There are two types of errors we can commit. A (FP) decision assigns two different labeled nodes to the same community. A (FN) decision assigns two same labeled nodes to different communities. Hence, based on the definition of the clustering analysis, we can still use the rand index calculated for evaluating clustering result:

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

Precision, Recall, F_β

Derived from the methods mentioned above, we suddenly find some basic classification evaluation metrics can be natually used for evaluating community detection result. Hence, we can still use precision, recall and F_β. As RI gives equal wieght to FP and FN based on its definition, we can use the F_β to measure the same metrics in a more flexible way by involving one more parameter β in the formula. We can use the F_β to penalize FNs more strongly than FPs by selecting a value $\beta > 1$. Thus, the defined precision, recall and F_β can be calculated as:

$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2(P+R)}$$
$$P = \frac{TP}{TP+FP}$$
$$R = \frac{TP}{TP+FN}$$

Adjusted Rand Index

Based on the previous discussion, we can see that simple rand index is not able to capture enough community detection information and get a relevant reliable feedbacks for the partition result. Hence, Adjusted Rand Index (ARI) is also raised [**hubert1985comparing**]. It is less sensitive to the number of communities [**nguyen2009information**]. The chance correction is based on the general formula defined for any measure M, which is a typical standardization function widely used in current days.

$$M_c = \frac{M - E(M)}{M_{max} - E(M)}$$

where M_c is the chance-corrected measure, M_{max} is the maximal value that M can reach, and E(M) is the value expected for some null model. Under the assumption that the partitions are generated randomly given a community number, the original function of rand index can be standardized by using the function mentioned above as:

$$ARI = \frac{\sum_{ij}\binom{N_{w_i c_j}}{2} - \sum_i \binom{N_{w_i}}{2}\sum_j \binom{N_{c_j}}{2}/\binom{N}{2}}{1/2 * [\sum_i \binom{N_{w_i}}{2} + \sum_j \binom{N_{c_j}}{2}] - \sum_i \binom{N_{w_i}}{2}\sum_j \binom{N_{c_j}}{2}/\binom{N}{2}}$$

This metric has an upper bound of 1. And if the value is below 0, it means that the partition result is even worse than a randomly generated community partition.

Entropy

Entropy probabily is the most important metric to evaluate whether the prediction result is good or bad in machine learning tasks. Community detection also borrows idea from entropy to judge the partition. Lower entropy score refers to a more stable status. And the definition of it is shown as:

$$H(\Omega) = -\sum_k \frac{|w_k|}{N} log \frac{w_k}{N}$$

Normalized Mutual Information and Mutual Information

NMI [**schutze2008introduction**, **strehl2002cluster**, **fortunato2009community**] is another quite popular evaluation metric in community detection. And it is defined as

$$NMI = \frac{I(\Omega, C)}{[H(\Omega) + H(C)]/2}$$

While $I(\Omega, C)$ refers to the mutual information of the predicted community result with the ground truth result. It calculates how much similar the two results look like. The calculation function is the same as the normal MI function used in traditional machine learning. NMI is always a number between 0 and 1. A major problem of NMI is that it is not a true metric, i.e., it does not follow triangle-inequality thereon. MI function is calculated as below:

$$I(\Omega, C) = \sum_k \sum_j \frac{|w_k \cap c_j}{N} | log \frac{N|w_k \cap c_j|}{|w_k||c_j|}$$

Variation of information

NMI has a very serious drawback called selection bias problem. The value is biased to the number of generated communities. Larger community size pretend to obtain a higher NMI score. To overcome the drawbacks of NMI, Variation of Information (VI) [**meilua2007comparing**, **kraskov2005hierarchical**] is defined which is able to calculate the shared information distance obeys the triangle inequality. It is defined as:

$$VOI = \sum_{i,j} r_{i,j} [log(r_{ij}/p_i) + log(r_{ij}/q_j)]$$

or

$$VOI = H(\Omega) + H(C) - 2I(\Omega, C)$$

where $p_i = \frac{w_i}{N}$, $q_j = \frac{c_i}{N}$ and $r_{ij} = \frac{w_i \cap c_j}{N}$.

Modified purity

[**labatut2015generalised**, 247] argue that the traditional metrics consider a community structure and ignore the network topology. They add the weights information of a node to calculate purity score. First, as the same Purity function mentioned above, they represent the function use another term:

$$Purity(u, \Omega, C) = \delta(C_j | s.t. N_{w_i c_j} \text{ is maximum})$$

After they add link weights to the original purity function, it terms to be:

$$MP = \sum_i \sum_{u \in w_i} \frac{w_u}{w} Purity(u, \Omega, C)$$

where $W = \sum_v w_v$ is the sum of all the weights of the nodes in a community or in the whole graph. This function can keep the modified purity result in a range of 0 to 1.

Modified ARI

Taking the similar idea from modified purity function, since Rand Index is based on pairwise comparisons, it is not possible to isolate the individual effect of each node, previous studies propose to use the product of the two corresponding nodal weights: $w_u w_v$. Then for any subset of nodes S, it can be translated as: $W(S) = \sum_{u,v \in S} w_v w_u$. And the modified ARI turns into:

$$ARI = \frac{\sum_{ij} W(w_i \cap W(c_j)) - \sum_j W(w_i) W(c_j)/W(V)}{\frac{1}{2}(\sum_i W(w_i + \sum_j W(c_j))) - \sum_i W(w_i) \sum_j W(W(c_j))/W(s)}$$

Summary

In this section, we introduce twelve basic evaluation metrics in non-overlapping community detection methods. Those metrics are almost all metrics used among current researches to evaluate the non-overlapping community quality. Some of the metrics are previously raised, which were classic metrics used for years, such as Mutual information and Rand index. However, in recent years, researchers argue those metrics do not consider the normalization issue, which causes bias towards the evaluation. Therefore, many refined metrics such Adjusted rand index and Normalized mutual information start to be used more often. Based on experience, we suggest to use the normalized metrics for model evaluation, which reduces the negative effect on community numbers or biased community size distribution. Precision, Recall and F measures are the three basic metrics for non-overlapping community detection, we suggest to at least report F measure for academic papers. As for other metrics, they can be chosen based on the evaluation task requirements.

Overlapping Metrics

In overlapping community detection evaluation, the same as the non-overlapping community detection, We define, given a graph $G(V,E)$, $\Psi = \{\psi_1, \psi_2, ...\psi_k\}$ is the set of the detected communities generated with some community detection models. And $C = \{c_1, c_2, ..., c_j\}$ is the set of the ground truth communities given in advance. $N = |V| = \sum_{\psi \in \Psi} |\psi_k| = \sum_{j \in J} |c_j|$ is the total number of the nodes in the original graph where $|N_{c_j}| = |c_j|$ and $|N_{\psi_i, c_j}| = |\psi_i \cap c_j|$. Hereby, we will introduce four kinds of basic metrics widely used in overlapping community detection. We can see that the metrics are sort of derived from non-overlapping community detection and there are only small changes on the non-overlapping community detection metrics one. The calculation function of the four metrics in shown in Table 2.2.

The metrics include Overlapping normalized mutual information, Omega index , Generalized external index and F1-score. As precision, recall and F1-score are calculated in the same track, the difference of F1-score between overlapping and non-overlapping community detection is the same as the difference between precision and recall in the two community scenarios. Although it seems that the calculation is pretty complex, after the introductions listed in the following paragraphs, we can see that all metrics are of pretty

Metric	Expression
Overlapping normalized mutual information	$ONMI(X\|Y) = 1 - \frac{H(X\|Y)+H(Y\|X)}{2}$
Omega index	$\frac{Omega_u(\Psi,C)-Omega_e(\Psi,C)}{1-Omega_e(\Phi,C)}$ $Omega_u = \frac{1}{M}\sum_{j=1}\text{argmax}(\|\Psi\|,\|C\|)\|t_j(\psi_i)\cap t_j(c_j)\|$ $Omega_e = \frac{1}{M^2}\sum_{j=1}\text{argmax}(\|\Psi\|,\|C\|)\|t_j(\psi_i)\cdot t_j(c_j)\|$
Generalized external index	$a_G(i,j) = min\{\alpha_\Psi(i,j),\alpha_C(i,j)\} + min\{\beta_\Psi(i),\beta_C i\} + min\{\beta_\Psi(j),\beta_C j\}$ $a_G(i,j) = abs\|\alpha_\Psi(i,j) - \alpha_C(i,j)\| + abs\|\beta_\Psi(i)-\beta_C i\|+abs\|\beta_\Psi(j)-\beta_C j\|$ $GEI(\Psi,C) = \frac{a)G}{a_G+d_G}$
F1-score	$\frac{1}{2}(\frac{1}{\|\Psi\|}\sum_{\psi_i\in\Psi}F1(\psi_i,C_{g_i}) + \frac{1}{\|C\|}\sum_{c_i\in C}F1(\psi_{g'(i)},C_i))$

Table 2.2: Evaluation metrics for overlapping community detection results

naive and straight forward thoughts on community evaluation. In the following paragraphs, we will introduce them one by one.

Overlapping normalized mutual information

The conditional entropy of a cluster ψ_k given C_l is defined as $H(\psi_k|C_l) = H(\psi_k, C_l) - H(C_l)$. The entropy of ψ_k with respect to the entire vector C is based on the best matching between ψ_k and any component of C given by vector C is based on the best matching between ψ_k and any component of C given by:

$$H(\psi_k|C) = min_l(\psi_k|C_l)$$

And the normalized conditional entropy ofΨ to C is naturally calculated as:

$$H(\Psi|C) = \frac{1}{|C|}\sum_k \frac{H(\psi_k|C)}{H(\Psi_k)}$$

Similarly, we can define $H(C|\Psi)$. And the overlapping normalized mutual information

(ONMI) [264] is defined as:

$$ONMI(\Psi|C) = 1 - \frac{H(\Psi|C) + H(C|\Psi)}{2}$$

If the overlapping part is 0, this function is also able to represent non-overlapping NMI score. In ONMI, it calculated the conditional entropy from both sides ($\Psi \rightarrow C$ and $C \rightarrow \Psi$). Higher ONMI score refers to the higher correlation between ground truth communities and generated communities.

Omega index

Omega index [252] can be regarded as the overlapping version of the Adjusted Rand Index [**collins1988omega**, 252], which is another critical indicator to measure the overlapping community detection performance. Omega index is defined in the following way [265, 266]:

$$\Omega = \frac{Omega_u(\Psi, C) - Omega_e(\Psi, C)}{1 - Omega_e(\Phi, C)}$$

The unadjusted Omega index $Omega_u$ is defined as:

$$Omega_u = \frac{1}{M} \sum_{j=1} \text{argmax}(|\Psi|, |C|) |t_j(\psi_i) \cap t_j(c_j)|$$

And the expected Omega index in the null model $Omega_e$ is given by,

$$Omega_e = \frac{1}{M^2} \sum_{j=1} \text{argmax}(|\Psi|, |C|) |t_j(\psi_i) \cdot t_j(c_j)|$$

Omega index refers to how much the generated community has overlapped components with ground truth over than the null model (random overlapping communities).

Generalized external index

[**campello2007fuzzy**, **campello2010generalized**] introduce another evaluation metric called Generalized external index (GEI) for comparing the predicted community and ground truth community result. It includes the following information:

- $\alpha_\Psi(i,j)$: Number of communities shared by nodes i and j in partition Ψ
- $\alpha_C(i,j)$: Number of communities shared by nodes i and j in partition C.
- $\beta_\Psi(i,j)$: Number of communities to which node i belongs in Ψ minus 1.
- $\beta_C(i,j)$: Number of communities to which node i belongs in C, minus 1.

Based on the definitions above, the agreements a_G and disagreements d_G associated are defined as:

$$a_G(i,j) = min\{\alpha_\Psi(i,j), \alpha_C(i,j)\} + min\{\beta_\Psi(i), \beta_C i\} + min\{\beta_\Psi(j), \beta_C j\}$$
$$d_G(i,j) = abs|\alpha_\Psi(i,j) - \alpha_C(i,j)| + abs|\beta_\Psi(i) - \beta_C i| + abs|\beta_\Psi(j) - \beta_C j|$$

And the final GEI formula is:

$$GEI(\Psi, C) = \frac{a_G}{a_G + d_G}$$

Instead of calculation the community similarity on community level, GEI evaluate the performance of generated communities on pairwise node level. From the extremely complicated formulas mentioned above, we can see the fundamental idea is to check whether their relationships keep the same on each pair of nodes in the graphs. While the calculation takes much more time complexity then the rest indicators mentioned above.

F1-score

[76] uses average F1-score to measure the equivalence of two overlapping partitions. It is defined to be the average of the F1-score of the best matching ground-truth community to each detected community. The calculation formula is defined as:

$$F1 = \frac{1}{2}(\frac{1}{|\Psi|}\sum_{\psi_i\in\Psi} F1(\psi_i, C_{g_i}) + \frac{1}{|C|}\sum_{c_i\in C} F1(\psi_{g'(i)}, C_i))$$

Summary

In this section, we introduce four different metrics for overlapping community evaluation. We can see there are some connections on those metrics with non-overlapping community detection evaluation metrics but with much more complex considerations. They either evaluate from the community level similarity or pairwise node level similarity, which offers more potential tracks for overlapping community detection. However, as overlapping community is only a part of the whole community detection family, more future works need to work on both model side and evaluation side.

2.5.3 Summary

CHAPTER 3
PERSONALIZED COMMUNITY DETECTION

3.1 Introduction

Community detection is an important topic in graph mining. By learning node community labels on the graph, we are able to detect node hidden attributes as well as explore the closeness between nodes [6]. Conventional methods are mostly user-independent to detect communities solely relying on graph topological structure [2], generate semi-supervised communities with node constraints [65], or select top-K sub graphs as user-centric communities [101]. These approaches are no longer enough to satisfy users with a pursuit of personalization, which makes involving user need into community detection to become an inevitable task. Specifically, from a user-centric viewpoint, the ideal communities should provide a high-resolution partition in areas of the graph relevant to the user need and a coarse manner partition on the remaining areas so as to best depict user need (*we also call it "query" in the rest of this paper*) in concentrated areas while fuzz irrelevant areas.

For instance, in Figure 3.1, two different scholars in *education* and *data mining* domains may consume the same scholarly graph differently because they may need more detailed community exploration in their own domains while generalized community information in other irrelevant domains (e.g., the *data mining* scholar needs more detailed communities such as *Deep Learning*, *Graph Mining*, and *Bayesian Analysis*. While an *education* scholar may need to generalize those communities as *Computer Science* or just *Science*).

As aforementioned in the Related Work section , current investigations are still with limited scope. First, user-independent approaches solely consider graph topological structure without user need. Second, semi-supervised approaches detect communities restricted by pre-selected seed nodes. As different user needs refer to different seeds, each individ-

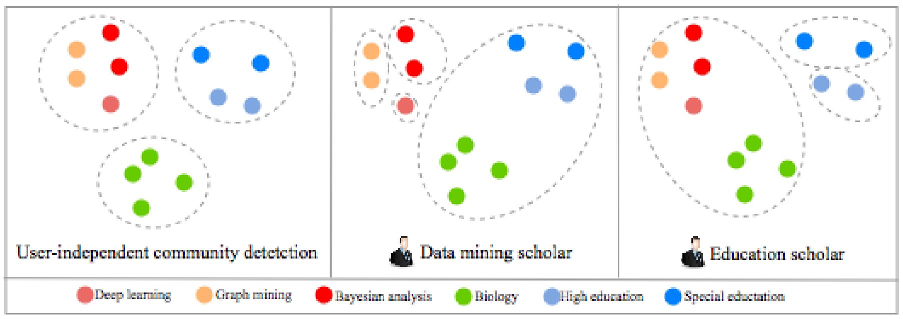


Figure 3.1: An example of personalized community detection on a scholarly graph

ual user requires a separate process to run the whole model completely to get personalized communities, which is inapplicable in real cases. Third, sub-graph selection approaches only generate communities from the partial graph instead of the whole one.

To detect personalized communities on the whole graph, I propose a **g**enetic **P**ersonalized **C**ommunity **D**etection (gPCD) model with an offline and an online step. Specifically, in the offline step, I convert the user-independent graph community to a binary community tree which is encoded with binary code. Subsequently, a deep learning method is utilized to learn low-dimensional embedding representations for both user need and nodes on the binary community tree. In the online step, I propose a genetic tree-pruning approach on the tree to detect personalized communities by maximizing user need and minimizing user searching cost simultaneously. The whole genetic approach runs in an iterative manner to simulate an evolutionary process and generate a number of partition candidates which are regarded as "chromosomes" in each genetic generation. Through the selection, cross-over and mutation process, successive chromosomes are bred as better personalized community partitions to meet with user need.

The contribution of this study is threefold.

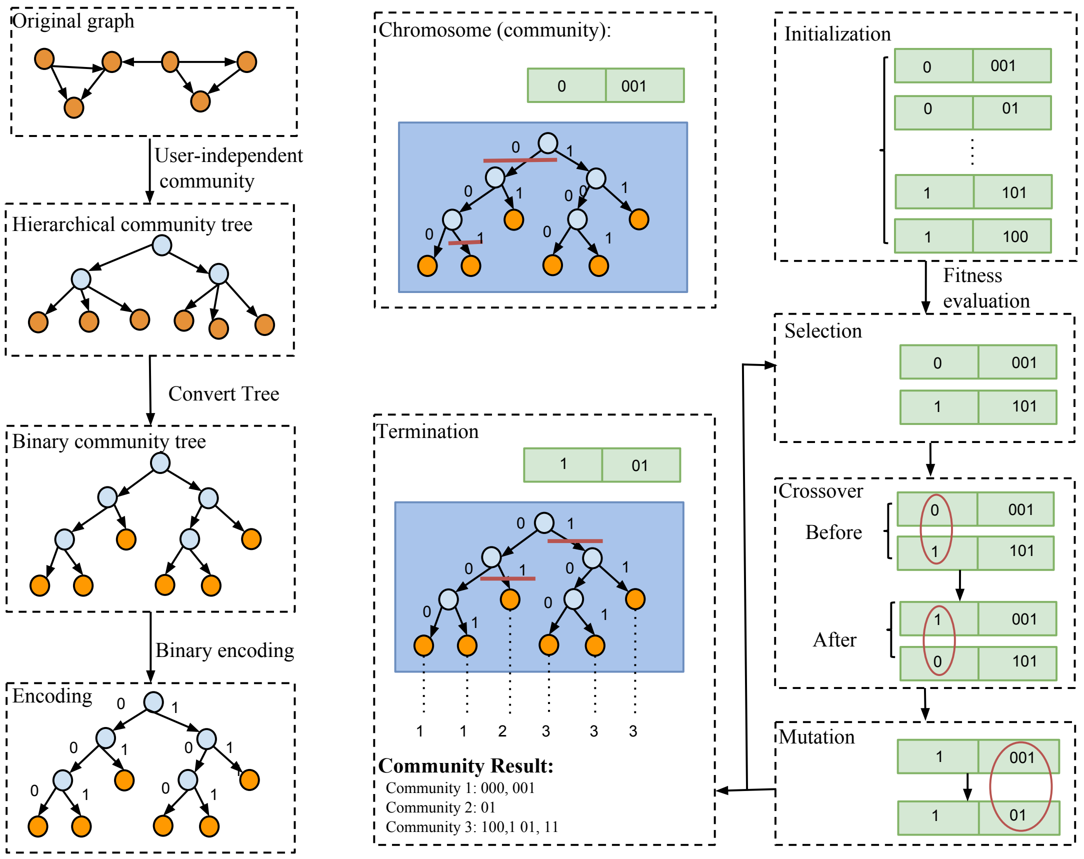
- I address a novel personalized community problem and propose a model to generate different-resolution communities associated with user need.

- The proposed model contains an offline and an online step. The offline step takes charge of most calculation to enable an efficient online step: The construction of binary community tree has a time complexity of $O(\|V\|^2)$ in the worst case where $\|V\|$ denotes the number of vertices in the graph; Representation learning on both binary community tree and user need has the same time complexity as Node2vec [267]. The online genetic pruning step running under the parallel environment achieves $O(\frac{2^d KP}{M})$ time complexity where d denotes the depth of the tree, K denotes the community number, P denotes the initialized population size in the genetic approach and M denotes the number of Mappers/ Reducers in Hadoop Distributed File System (HDFS).

- I evaluate the proposed model on a scholarly graph and a music graph. In my model, the offline step is separately calculated and keeps unchanged once constructed, while the online step guides the personalized community detection. Hence I only compare the online step results with baselines' performance. Extensive experiments shows my model outperforms in terms of both accuracy and efficiency.

3.2 genetic Personalized Community Detection

My proposed gPCD model contains an offline and an online step. Figure 3.2 shows the pipeline of the whole framework. The offline step first encodes the user-independent binary community tree (*Section 3.1*), and subsequently learns embedding representations for both user need and nodes on the binary community tree (*Section 3.2*). The online step introduces the genetic personalized community detection approach (*Section 3.3*). To accelerate the running speed, a distributed version of gPCD model is also deployed on HDFS (*Section 3.4*). To disambiguate the notations mentioned in this section to better explain the gPCD model, some commonly used notations can be found in Table 3.1.



(a) Offline:
binary community tree

(b) Online:
genetic pruning (Community size K = 3)

Figure 3.2: The framework of gPCD model. (a) refers to the offline construction step on the original graph and (b) refers to the online genetic pruning step to generate personalized communities.

3.2.1 Offline Community Tree Index

One challenge of solving personalized community detection problem is the computational cost due to the complexity of personalization and graph structure. In order to reduce the online workload, most of computation cost is put into the one-time offline step whose time cost can be excluded from the online personalized community detection step. Thus, I first convert the graph into a binary community tree offline to retain user-independent community information.

I employed the Infomap algorithm [194] to generate the user-independent communities

Notations	**Descriptions**
$G(V, E)$	Original graph G with vertex set V and edge set E
$T_c(N^c, L^c)$	The hierarchical community tree generated from graph $G(V, E)$ with node set N^c and link set L^c. Each node $N_k^c \in N^c$ denotes a group of vertices belonging to V.
$T_b(N^b, L^b)$	The binary community tree reconstructed from $T_c(N^c, L^c)$. Each node $N_k^b \in N^b$ denotes a group of vertices belonging to V.
B	The binary codebook for $T_b(N^b, L^b)$. Particularly, $B_k \in B$ denotes the binary code of both $N_k^b \in N^b$ and $L_k^b \in L^b$ where L_k^b is the link points to node N_k^b.

Table 3.1: Commonly used notations in gPCD model

solely based on the graph *G(V, E)*. Infomap algorithm simulates a random walker wandering on the graph and indexes the description length of his random walk path via multilevel codebooks. By minimizing the description length based on the map equation below, community structures are formed for the graph.

$$L(M) = q_{\curvearrowright} H(\mathcal{Q}) + \sum_{i=1}^{m} p_{\circlearrowright}^{i} H(\mathcal{P}^i) \tag{3.1}$$

where $L(M)$ is the description length for a random walker in the current community M. $q_{\curvearrowright}$ and $p_{\circlearrowright}^{i}$ are the jumping rates between communities and within the i_{th} community. $H(\mathcal{Q})$ is the frequency-weighted average length of codewords in the global index codebook and $H(\mathcal{P}^i)$ is frequency-weighted average length of codewords in the i_{th} community codebook. Followed by this equation to partition communities into sub-communities , a hierarchical community tree $T_c(N^c, L^c)$ is constructed from the original graph $G(V, E)$.

In $T_c(N^c, L^c)$, each parent node can have multiple child nodes which can be regarded as a community partition on the parent node. For instance, a node $N_k^c \in N^c$ from $T_c(N^c, L^c)$ represents a community of vertices. Its m child nodes $\{N_{k_1}^c, N_{k_2}^c, ..., N_{k_m}^c\}$ represent m sub-communities of vertices from $G(V, E)$ where we have $\bigcap_{i=1}^{m} N_{k_i}^c = \varnothing$ and $\bigcup_{i=1}^{m} N_{k_i}^c = N_k^c$.

In order to achieve an efficient personalized community detection in the following online step, I convert the hierarchical community tree $T_c(N^c, L^c)$ to a binary community tree $T_b(N^b, L^b)$ for index. Specifically, for m child nodes of a parent node N_k^c, a bottom-up approach is proposed to merge a selected pair of sibling nodes as a new node in an iterative manner. The approach runs until all m child nodes merged together to form the parent node N_k^c. To avoid an unbalanced tree where small communities are always left to merge with huge communities in the end, I first select the node with the smallest community size among all sibling nodes in each merging step. It is merged with its sibling node with the largest normalized linked weight (Please refer to Figure 3.2(a)). The normalized linked weight function $w(\cdot)$ between two nodes N_i^c and N_j^c is defined as:

$$w(N_i^c, N_j^c) = \frac{N_i^c \odot N_j^c - \frac{\mathcal{D}(N_i^c)\cdot\mathcal{D}(N_j^c)}{2\|E\|}}{\|N_i^c\|\|N_j^c\|} \tag{3.2}$$

where $N_i^c \odot N_j^c$ denotes the number of edges linked between vertices in node N_i^c and N_j^c, which can be interpreted as the linkage strength between them; $\|N_i^c\|$ is the number of vertices inside node N_i^c; $\mathcal{D}(N_i^c)$ is the out-degree of node N_i^c (the total number of edges linked to other nodes) and $\|E\|$ is the total number of edges in the original graph $G(V, E)$. $\frac{\mathcal{D}(N_i^c)\cdot\mathcal{D}(N_j^c)}{2\|E\|}$ denotes the random linkage strength between node N_i^c and N_j^c. The $w(\cdot)$ function calculates how much that two nodes are better connected beyond random connection and is normalized by node size. Given the node N_i^c with the smallest community size and all its sibling node set S, The merging step can be formulated as:

$$\begin{aligned} N_j^c &\Leftarrow \operatorname*{argmax}_{N_j^c \in S} w(N_i^c, N_j^c) \\ N_*^c &= N_i^c \bigcup N_j^c \end{aligned} \tag{3.3}$$

The bottom-up process will stop until all child nodes are merged together to form the parent node. In the end, the hierarchical community tree $T_c(N^c, L^c)$ is fully converted to a binary community tree $T_b(N^b, L^b)$ with user-independent community information. The node size $\|N^b\|$ as well as the link size $\|L^b\|$ in $T_b(N^b, L^b)$ is at most $2\|V\|$ which is smaller than the size of original graph $G(V, E)$. If I consider to form the binary community tree with only k levels, the size of $T_b(N^b, L^b)$ can be even smaller.

For running time analysis, calculating normalized linked weight takes constant time. In each merging step, node pair selection takes linear time. Therefore, in the worst case, the time complexity of binary community tree construction is $O(\|V\|^2)$ where the depth of the hierarchical community tree $T_c(N^c, L^c)$ is 1 and each vertex in $G(V, E)$ forms a single-vertex community.

To encode the nodes and links on $T_b(N^b, L^b)$ as binary code, the root node is encoded as 'null' first. For a parent node N_k^b with its left child node $N_{k_l}^b$ and right child node $N_{k_r}^b$, the binary code of a child node and the related link defined in the Notation Table 3.1 is calculated as:

$$B_{k_i} = \begin{cases} B_k + \text{“0”}, & i = \text{“}l\text{”} \\ B_k + \text{“1”}, & i = \text{“}r\text{”} \end{cases} \tag{3.4}$$

For instance, if the node N_k^b is with binary code "00," its left child node's binary code is "000" while the right child node's binary code is "001." The link L_k^b that points to N_k^b also has the binary code "00".

3.2.2 Community and User Need Representation

Node2vec [267] helps to learn fixed-length embeddings for both user need and communities. It simulates random walks on the graph $G(V, E)$ and learns the vertex embedding by optimizing the sequential relationships from random walk paths. In the end, each vertex V_k in graph $G(V, E)$ has a vector representation as $\vec{V_k}$. Each node N_k^b on the binary community tree $T_b(N^b, L^b)$ refers to a vertex community C_k in the graph $G(V, E)$. Its representation $\vec{C_k}$ is calculated as the averaged embedding of all vertices inside the community. In the end, the binary community tree $T_b(N^b, L^b)$ represents the hierarchical community partition of Graph $G(V, E)$. Each node N_k^b on the tree is indexed with three attributes: a group of vertices from graph $G(V, E)$, a binary code B_k, and an embedding representation $\vec{C_k}$.

On the other hand, User need (query) I can also be represented by a combination of t different vertices $\{V_1, V_2 ... V_t\}$ in the graph $G(V, E)$. In this study, two different scenarios for user need representation are offered:

Vertex-based Query. User need can be directly represented by the vertices based on the generation probability $P(V_k|I)$ between them. Hence the user need representation $\vec{I}$ is calculated as:

$$\vec{I} = \sum_{k=1}^{t} P(V_k|I) \cdot \vec{V_k} \tag{3.5}$$

For instance, in a music sharing network, each vertex V_k denotes a music and a user listing history can be used to reflect the user need I. $P(V_k|I)$ therefore can be regarded as the probability that a music being listened by the user.

Text-based Query. Under this scenario, user need I is represented as a text query, and each vertex V_k in the graph $G(V, E)$ also contains textual content. From language model viewpoint, each vertex importance weight is the query likelihood $P(I|V_k)$, and the user need can is the weighted average of vertex embedding:

$$\vec{I} = \frac{\sum_{k=1}^{t} P(I|V_k) \cdot \vec{V_k}}{\sum_{k=1}^{t} P(I|V_k)} \tag{3.6}$$

In either case, user need is conceptualized as an embedding with the same dimension as the node embeddings on the binary community tree. It enables very efficient online personalized community detection in later steps. And running Node2vec takes most of the time in this step.

3.2.3 Online Genetic Pruning

The whole process, as the Figure 3.2 shows, is to generate communities by pruning the constructed binary community tree. After each cut on a link, the original tree will be separated into two sub-trees. After a specific number of cuts to the links on the tree, a fixed number of communities with different resolutions are detected. By applying genetic selection, crossover, and mutation steps, the model converges to the optimized solution efficiently with a clear-defined fitness function. The details are shown in the following paragraphs.

Genetic Representation

A chromosome is formed by a set of genes $\{g_1, g_2, ..., g_{K-1}\}$, and each gene g_i holds a cut link L_i^b in the binary community tree $T_b(N^b, L^b)$. Since communities can be created by cutting links on the offline tree, a chromosome can be represented as a generated community partition of the original graph $G(V, E)$ in this way. To constrain a chromosome so that it can be decoded to a fixed number of communities, four **Cutting Rules** are necessarily to be applied:

- **Rule 1**: If a link L_i^b is picked to cut on the binary community tree $T_b(N^b, L^b)$, its pointing node N_i^b will be retrieved and all the vertices within it form a community.
- **Rule 2**: If a link L_i^b and its ancestor link L_j^b are stored in the same chromosome, all vertices in L_i^b's related node N_i^b are a subset of vertices in L_j^b's related node N_j^b. In this case, the two cut links generate two communities where community C_i is all vertices in N_i^b and community C_j is the remaining vertices in N_j^b but not in N_i^b. It can be formulated as $C_i = \bigcup_k \{V_k | (V_k \in N_i^b)\}$ and community $C_j = \bigcup_k \{V_k | (V_k \in N_j^b) \cap (V_k \notin N_i^b)\}$.
- **Rule 3**: Sibling links can't be stored in the same chromosome, and it is not allowed to store duplicated links in a chromosome.
- **Rule 4**: The depth's upper bound is set to be d, which means all eligible cut links should be located in the first d depth on the binary community tree. It avoids to generate super tiny communities and hugely reduces the genetic searching scope on cut links.

By applying the cutting rules to the online pruning process, I ensure a K community partition can be retrieved from a chromosome with $K-1$ cut links.

Initialization

Initially, the model generates a given number P chromosomes as the seed "chromosome population". And each iteration in the genetic approach breeds a new "generation" of chromosome population. In order to ensure a chromosome is an encoder of a K community partition, $K-1$ links will be randomly picked (on the binary community tree) following the cutting rules and stored in the related genes of a chromosome.

Fitness Function

As each chromosome can be decoded as a community partition, it is important to measure the quality of each generated chromosome (how well the generated communities can satisfy user need). The measurement is hosted in a fitness function.

In the proposed model, the fitness function simulates the user searching behavior on the graph given the community partition. For instance, a user can be more likely to pick the most relevant communities while avoiding the redundant information already selected. With the help of the offline step, the relevance score of node N_i^b (community C_i) towards user need I can be calculated with the cosine similarity $cos(\vec{I}, \vec{C_i})$, and the information redundancy can be $\sum_{C_j \in S_c} cos(\vec{C_j}, \vec{C_i})$ where S_c is the set of communities that the user have already picked from the communities decoded from the target chromosome. Following this, I use a greedy selection approach to iteratively rank and pick communities given a chromosome (community partition) until all communities are picked:

$$\underset{C_i}{\operatorname{argmax}}\, \lambda \cdot cos(\vec{I}, \vec{C_i}) - (1-\lambda) \cdot \frac{\sum_{C_j \in S_c} cos(\vec{C_j}, \vec{C_i})}{\|S_c\|} \tag{3.7}$$

where C_i is the candidate community to be picked and $\|S_c\|$ is the number of communities already been picked. λ is a parameter controls whether user prefers to obtain new useful information or to avoid redundant information.

For chromosome quality evaluation, a query-generated vertex ranking list l_q is first created by retrieving top n vertices relevant to the query (user need) with the largest cosine similarities on embeddings of graph $G(V, E)$. I store the top n vertex ranking label $R(l_q) = \{1, 2, ..., n\}$ as the pseudo ground truth. On the other hand, given the k_{th} chromosome ch_k in the current chromosome generation, I can also retrieve the community-generated ranking of each vertex $V_k \in l_q$ from the sequentially selected communities decoded by the chromosome. I assign the ranking label on each vertex V_k based on the following formula:

$$\sum_{V_j \in l_q} \Phi(\delta(V_j) < \delta(V_k)) + 1 \tag{3.8}$$

V_j refers to all vertices in l_q. $\delta(V_j)$ shows the ranking (selection sequence) of the community which V_j belongs to. Φ is a binary operator to determine whether V_j satisfy the condition $\delta(V_j) < \delta(V_q)$. This formula helps to construct the community-generated ranking label $R(l_c)$. For instance, when $n = 3$, I have a query-generated ranking list $l_q = \{V_1, V_2, V_3\}$ and its related ranking label $R(l_q) = \{1, 2, 3\}$. Given a chromosome where the community of V_1 and V_2 is the same and selected before V_3, I can generate the related community ranking label $R(l_c) = \{1, 1, 3\}$ with the same vertex sequence of l_q.

Then, I define the fitness function $f(\cdot)$ to evaluate the chromosome ch_k. As I have the query-generated ranking label $R(l_q)$ (ground truth) and community-generated ranking label $R(l_c)$ from ch_k, I calculate their Kendall's τ correlation coefficient as the fitness score $f(ch_k)$ of chromosome ch_k where higher score means the chromosome ch_k can generate better personalized communities to meet with user need.

$$f(ch_k) = 1 - \frac{\sum_{i=1}^{n} R(l_{ci}) \cdot R(l_{qi})}{\sum_{i=1}^{n} R(l_{ci})^2 \cdot \sum_{i=1}^{n} R(l_{qi})^2} \tag{3.9}$$

where $R(l_{ci})$ is the i_{th} vertex ranking in community-generated ranking label $R(l_c)$ and $R(l_{qi})$ is the i_{th} vertex ranking in query-generated ranking label $R(l_q)$. Kendall's τ is a widely used metric to evaluate the correlation between two lists where higher score means stronger correlation. Thus, higher fitness score reflects that the generated community ranking ($R(l_c)$) can better meet with user need ($R(l_q)$).

Moreover, it is clear that the fitness function aims to separate all top n vertices in different communities to get the optimal case. It matches the research goal to generate high resolution communities on vertices which are more relevant to user need. As the number of community is a given number K, it also leads to a coarser manner partition on the remaining less relevant vertices. On the other hand, the binary community tree $T_b(N^b, L^b)$ and the Cutting Rule 4 naturally preserve the community structure and unite the most relevant vertices in the same community. Hence the whole genetic approach is a gambling process. The final chromosome result is the equilibrium case to detect communities both contain graph topological structure and meet with user need.

Selection

I select the superior chromosomes from current chromosome population based on their fitness scores. The probability that the k_{th} chromosome ch_k is picked can be calculated via the Softmax normalization function $p(ch_i) = \frac{exp(f(ch_k))}{\sum_{i=1}^{P} exp(f(ch_i))}$. Then, the Fitness Proportionate Selection method is applied to randomly select P chromosomes into chromosome pairs based on probability distribution. In order to enhance optimization efficiency, I also use elitism selection to ensure the best chromosome in the current generation will always be selected to the next generation.

Crossover

To reach global optimum community partition efficiently, given a pair of chromosomes, the crossover operation can randomly exchange part of the genes in both chromosomes to produce a new pair of chromosomes with a certain crossover rate.

In order to make sure that the newly generated chromosomes meet the cutting rules, an

Exchange Rule is defined to restrict gene exchange: If gene g contains link L_g^b, g can't do crossover process with genes that contain either link L_g^b or its sibling link $L_g^{b'}$. This rule can help avoid having duplicated links or sibling links stored together in the newly generated chromosome (To satisfy Cutting rule 3).

After m random numbers are selected from $\{1, 2, ..., K-1\}$ as exchanged gene position indexes, genes located in the chosen positions of two chromosomes will exchange the stored link restricted by the Exchange Rule.

Mutation

Mutation operation is applied to avoid local optimization. If a chromosome is chosen to mutate, a gene within the chromosome will be randomly picked, and its stored link will be changed to another link restricted by the Exchange Rules. An example is illustrated in Figure 3.2(b) where the link stored in the second gene is changed from "001" to "01".

Termination

After T iterations, the whole process stops and the current best chromosome is retrieved as the final result. Choosing the number of T is dependent on the task. In order to decode the final chromosome to the related community partition, all genes in the chromosome are sorted in an ascending order based on the binary code of their stored cut link. Vertices whose binary codes start with the same cut link's binary code will be assigned to the same community label. And its later assigned community label can overwrite the previous assigned community label. For instance, if there are a vertex with binary code "0011" and two cut links with binary code "00" and "001", the vertex will be assigned to a community label "00" first, and its community label is overwritten by "001" afterwards. The Termination step in Figure 3.2(b) also illustrates a vivid example. In this way, the binary code of the binary community tree can help to decode the final chromosome into communities in an efficient way.

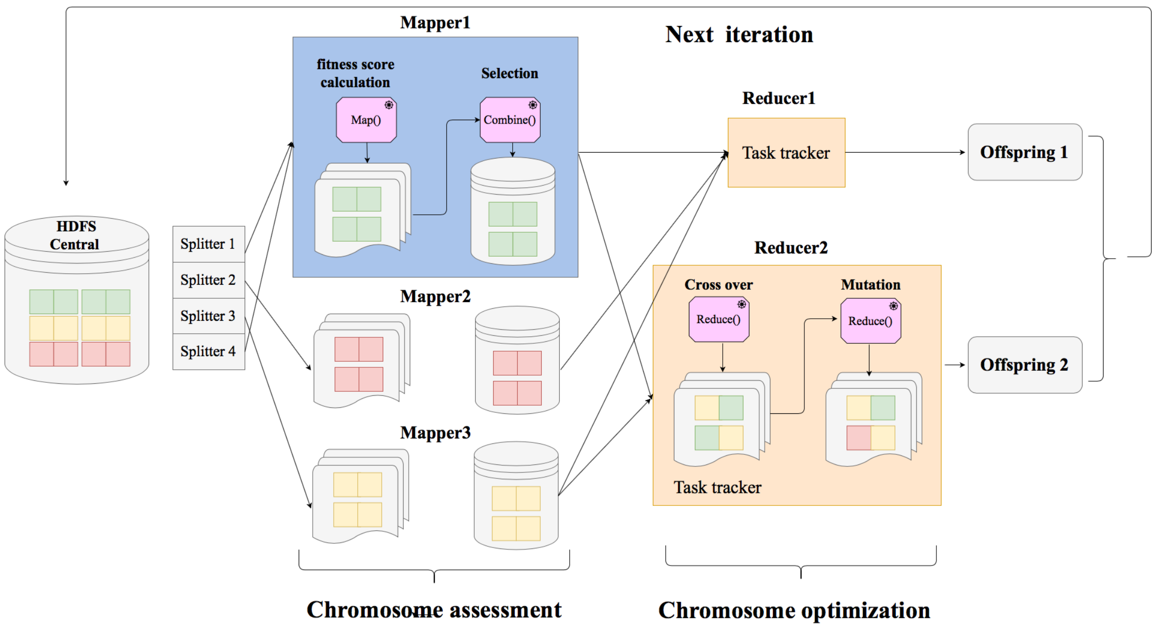


Figure 3.3: Online parallel computing process on Hadoop Distributed File System (HDFS)

3.2.4 Distributed gPCD

To enhance the online step efficiency, a MapReduce framework is utilized to enable the distributed genetic evolution. Figure 3.3 depicts the personalized community detection under a MapReduce framework. The chromosome collection is either originally initialized from binary community tree or obtained from the last generation. It contains the whole chromosome population in the central depository. In its first "Splitter" process, all chromosomes are split into M groups based on their hash values and sent out to related M Mappers to calculate the "Fitness" scores. In the same Mapper, after all chromosomes are assigned fitness scores, based on their scores, a Combiner groups all chromosomes together and random select equal number of chromosomes with duplicated as the "Selection" step. All the selected chromosomes are sent to R reducers (I set $R = M$ arbitrarily in order to better represent time complexity) to form pairs for the "Crossover" and "Mutation" step, calculate new chromosome offsprings for the next generation and store them back to the central repository.

The complexity of the proposed algorithm is $O(2^d KP)$ without parallel computing and

$O(\frac{2^d KP}{M})$ with parallel computing, where d denotes the upper bound where the cut links are restricted in the top d depth of the binary community tree $T_b(N^b, L^b)$; K denotes the community number; P denotes the initialized population size of the genetic algorithm and M denotes the number of Mappers/Reducers in parallel environment. As all the parameters are considerably small (compared with the node/edge size in the original graph), the whole process runs very fast to retrieve the final community partition.

3.3 Experiments

3.3.1 Datasets

Datasets Description

Dataset	**Node description**		**Edge description**	
	Type	Size	Type	Size
Scholarly	paper	166,170	citation	750,181
Music	song	145,203	co-listening	1,172,525

Table 3.2: Dataset Description

Table 4.1 shows the statistics of the two datasets. The scholarly graph is unweighted and directed, while the music graph is a weighted and undirected.

Scholarly Graph: It contains academic publications with metadata extracted from ACM Digital Library. From the dataset, I build the experimental graph via paper citation relationship. Each vertex in the graph represents a paper, and if a paper cites another paper, there will be an edge linking the two. My model aims to detect personalized communities on the scholarly graph for authors. For each author in the dataset, I represent his/her need in two ways: their previous publications (text-based query) and cited paper history (vertex-based query).

Music Graph: It contains user listening histories and user-generated playlists from an online music streaming service, Xiami. I create a music graph with songs as the vertices and co-listening relationship as the edges. If two songs appear in the same user's listening history, there will be an edge linking them two. For users in the music dataset, I represent their music tastes (user need) from the songs in their listening history.

Ground Truth Construction

The ground truth for the two datasets are generated based on each user's publishing/ citing/ listening history:

For the scholarly dataset, the references of 112 random sampled papers are manually annotated from their literature reviews where authors summarize previous works. Different paragraphs (or sub-sections) in the literature review typically focus on separate but coherent topics while the same paragraph talks about the same topic. Based on this assumption, the papers cited in the same paragraph/sub-section naturally form a community with high resolution. To ensure each paper's cited papers form enough communities and each community contains enough papers, only papers with no fewer than three topics and all of whose communities have at least five papers are kept. After applying all these filters, 101 papers are left for evaluation.

For the music dataset, each user has several self-generated playlists. The songs in each playlist should contain a coherent theme. To avoid the playlists sharing mutually exclusive themes with other playlists, a Jaccard similarity check is applied on any pair of playlists created by the same user. If a user has at least two highly correlated playlists (Jaccard coefficient between them is above 0.5), I remove one of the playlists. Each playlist forms a separate community. Furthermore, to ensure the number of playlist and playlist size are both large enough, only users with at least three playlists and each playlist contains at least five songs are kept. In the end, there are 117 users who meet the above criteria.

In this paper, as all communities constructed in the ground truth are relevant communities with high resolution for users, my task is generating personalized communities to reconstruct the ground truth on two different datasets with vertex- and text-based user need.

3.3.2 Settings

Metrics & Parameter Settings

F1-score (F1), Rand index (Rand), Jaccard index (Jaccard) and running time are reported as the evaluation metrics in this paper. Based on empirical studies, population size P is 100. Crossover rate is 95%. Mutation rate is 1%. The maximum depth of binary community tree d is 10. The number of iteration T is 30. User searching preference λ is 0.6. Community size K is 50. The number of Mappers/ Reducers for parallelization M is 50. The number of top vertices to construct pseudo ground truth n is 10. Parameters in Infomap and Node2vec are both the default settings in their original papers.

Baselines

Considering both efficacy and efficiency, I select eight widely used user-independent community detection models. Ideally, to achieve personalized community detection, user-independent models should run on each user separately by assigning higher weights on user related edges. Thus, their time complexity should be only compared with my online step time complexity as my offline step is independent with user numbers. In this paper, to run baselines within acceptable time, I report their user-independent community results as the average performance.

- **Spinglass**: Spinglass constructs communities by minimizing the Hamiltonian score on signed graphs.[1].
- **Fast Greedy** (FG): Fast Greedy is a greedy search method to get the maximized modularity for community detection.[2].

[1]https://github.com/antoine-lizee/SG
[2]https://github.com/kjahan/community

- **Louvain**: Louvain is an agglomerative method to construct communities in a bottom-up manner guided by modularity.[3].

- **Walktrap**: Walktrap detects communities based on the fact that a random walker tends to be trapped in dense part of a network.[4].

- **Infomap**: Infomap generates communities by simulating a random walker wandering on the graph and indexing the description length of his random walk path via multilevel codebooks.[5].

- **Bigclam**: Bigclam generates overlapping communities via a non-negative matrix factorization approach.[6].

- **DeepWalk**: DeepWalk generates node embeddings via random walks and utilizes K-means on node embeddings to detect communitis.[7].

- **Node2vec**: Node2vec is an extended version of DeepWalk with a refined random walk strategy.[8].

3.4 Discussion

3.4.1 Evaluation Results

There are two scenarios to construct user need. For the scholarly graph, including a citation (vertex-based query) model and a keyword (text-based query) model. In the citation model, for each user, we first extract all the papers he/she cited before, then use their centroid embedding as user need vector $\vec{I}$. In the keyword model, for each user (author),

[3]https://github.com/taynaud/python-louvain
[4]https://www-complexnetworks.lip6.fr/ latapy/PP/walktrap.html
[5]http://www.mapequation.org/code.html
[6]https://github.com/snap-stanford/snap/tree/master/examples/bigclam
[7]https://github.com/phanein/deepwalk/tree/master/deepwalk
[8]https://github.com/aditya-grover/node2vec

we first extract all keywords he/she used in all previous papers to form a text query. Then we retrieve the top 100 relevant papers given the query based on probability language model with Dirichlet smoothing. Finally, we average those retrieved papers' vectors as the user need vector $\vec{I}$.

Model	**Scholarly Graph**			**Music Graph**		
	F1	Rand	Jaccard	F1	Rand	Jaccard
Spinglass	0.4294	0.4149	0.3593	0.4282	0.5317	0.2823
FG	0.4290	0.3852	0.3735	0.4645	0.4100	0.3070
Louvain	0.4417	0.4546	0.3627	0.1832	0.4201	0.1174
Walktrap	0.4304	0.3777	0.3777	0.3999	0.3507	0.3490
Infomap	0.4436	0.4165	0.3606	0.2147	**0.6074**	0.1344
Bigclam	0.2314	0.2572	0.1348	0.1499	0.2078	0.1227
DeepWalk	0.3904	0.3237	0.3234	0.3535	0.3253	0.3001
Node2vec	0.4001	0.3472	0.3433	0.4122	0.4101	0.3087
gPCD-Citation	**0.5351***	**0.4551***	**0.4086***	-	-	-
gPCD-Keyword	0.5069	0.4114	0.3708	-	-	-
gPCD-Listening	-	-	-	**0.5188***	0.5865	**0.3550***

Note: "*" means the p-value through a pairwise t-test is smaller than 0.001.

Table 3.3: Personalized Community Evaluation on gPCD and Baselines.

For the music task, text information is not available. The centroid embedding of the songs listened to by a target user are taken as user need.

Community Accuracy: I compare the average performance of my model on all testing users with baselines. Table 3.3 shows the detailed metrics. When running on the Scholarly graph, both Citation model and Keyword model can achieve around 10% increase on F1-

score compared with all baselines. Citation model also performs the best in Rand Index and Jaccard Index. For Music graph, my Listening model also has a significant improvement on F1-score and Jaccard Index. Although it has similar performance on Rand Index compared with Infomap, I believe my model in fact works much better due to the Infomap's poor performance on the rest two metrics. Moreover, I apply pairwise t-tests for all metrics on all testing users. All metrics' p-values in Citation model and the p-values of F1-score and Jaccard Index in Listening model are all smaller than 0.001, which means the improvements of my model performance are significant compared with baselines.

Running Time Comparison: Table 3.4 shows both the theoretical time complexity and real running time. To represent baseline algorithms' time complexity, "V" refers to the vertex number and "E" refers to the edge number in the graph $G(V, E)$. For some models (FG, Walktrap, and Infomap.), their specific time complexities are officially mentioned in the original papers. The time complexity of Louvain and Bigclam are roughly estimated in the original papers as well but those papers don't mention specific numbers. For Spinglass, DeepWalk and Node2vec, I can't find the exact time complexity in existing studies. Hence in this paper, I arbitrarily assign labels based on the their real running speed. Considering the running time, all the baseline algorithms run relatively fast except for the Spinglass algorithm. However, compared with all other models, the distributed gPCD always performs the fastest. Its real running time of is less than one-tenth of the fastest baseline's running time.

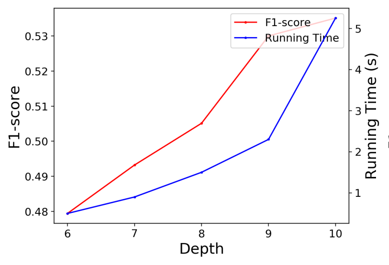
3.4.2 Parameter Analysis

I show how three parameters can affect my gPCD model performance in this section. They are the depth of the binary community tree d, genetic iteration number T and user searching preference λ. Figure 3.4 show the overall impacts of all tuned parameters.

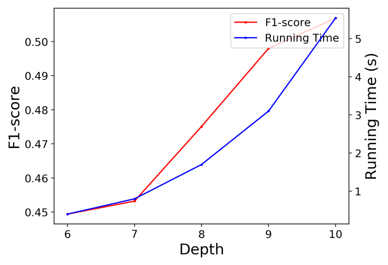
Depth on the Tree: Figure 3.4(a) to Figure 3.4(c) show how the depth of the binary community tree affects the model performance in accuracy and efficiency. From the figures,

Model	**Time Complexity**	**Scholarly Graph** (s)	**Music Graph** (s)
Spinglass	very slow	12548.68	10372.17
FG	$O(\|V\|log^2\|V\|)$	280.60	272.34
Louvain	linear	80.01	63.02
Walktrap	$O(\|V\|^2log\|V\|)$	638.44	503.24
Infomap	$O(\|V\|(\|V\|+\|E\|))$	501.79	425.63
Bigclam	linear	57.01	112.43
DeepWalk	fast	720.56	688.32
Node2vec	slow	3508.44	3100.12
gPCD	$O(\frac{2^dKP}{M})$	**5.25**	**6.50**

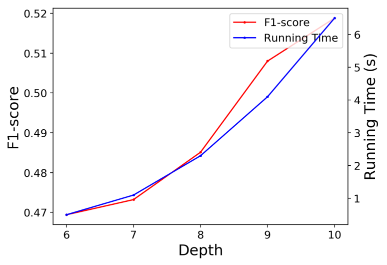
Table 3.4: Running time analysis on gPCD and all baselines in seconds (s).



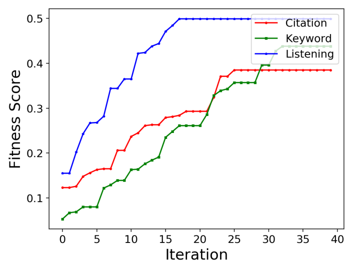
(a) Depth d in Citation model



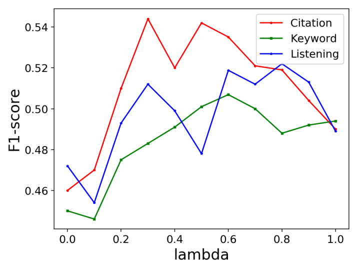
(b) Depth d in Keyword model



(c) Depth d in Listening model



(d) Iteration T in all models



(e) User searching preference λ in all models

Figure 3.4: Parameter effects on model performance

larger depth leads to a better personalized community detection result, while causes an exponential running time increase at the same time. Based on empirical studies, the upper bound of the depth is set to be 10 in this paper. While the depth selection may varies based on different graph sizes.

Convergence Analysis: I observe the best chromosome updates in 40 iterations. From Figure 3.4(d), I can see the fitness score start to be stable after the 30 iterations, which means the best chromosome is no longer changed after around 30 iterations. Thus, I set $T = 30$ as the default iteration number in my approach.

Searching Preference: In Figure 3.4(e), λ reflects the user searching preference whether he/she wants to explore new information or avoid redundant information. By selecting λ from 0 to 1, I find the F1-score are not very stable or have a clear correlation with λ. Based on the empirical experiments, I achieve the best performance on three models when $\lambda = 0.6$.

CHAPTER 4
CROSS-GRAPH COMMUNITY DETECTION

4.1 Introduction

Community detection is an essential task for cyberspace mining, which has been successfully employed to explore users' resemblance for retrieval/recommendation enhancement and user behavior analysis. Taking social media and e-commerce as examples, the complex, and often heterogeneous, relations among users and other objects, e.g., products, reviews, and messages, can be encapsulated as bipartite graphs, and the topology can help to synthesize and represent users with a coarser and broader view.

While a graph is well-connected, conventional methods, e.g., modularity-based approach [121], spectral approach [8], dynamic approach [33] and deep learning approach [168], are able to estimate the internal/external connectivity and generate high-quality communities directly on nodes [2].

For a vulnerable graph with sparse connectivity, however, prior community detection algorithms can hardly probe enough information to optimize the community structure. Unfortunately, in the real cyberspace this can be a very common problem, i.e., while a handful of giant players (like Google, Facebook and Amazon) maintaining high-quality graphs, thousands of apps are suffering from graph cold start problem. If many users are isolated because of data sparseness, I can hardly tell any community information.

In order to cope with this challenge, in this paper, I propose a novel research problem – Cross Graph Community Detection. The idea is based on the fact that an increasing number of small apps are utilizing the user identity information inherited from giant providers, i.e., users can easily login a large number of new apps by using Facebook and Google ID. In such ecosystem, the main large graph can provide critical information to enlighten the

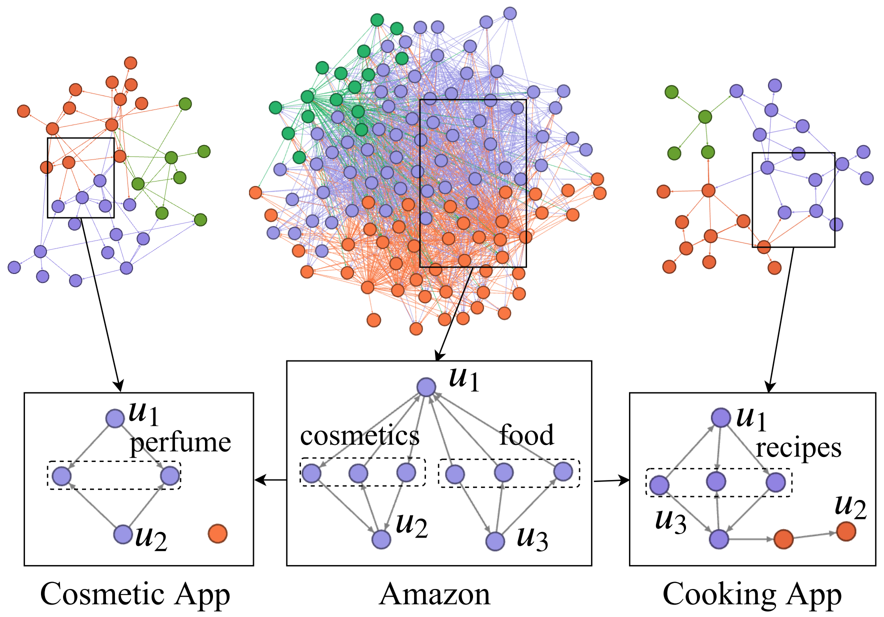


Figure 4.1: The Amazon graph is a main shopping graph while the Cosmetic App and the Cooking App are two small sparse graphs. Those graphs share mutual users in which three of them are selected to demonstrate their behaviors. Node colors indicate their related communities.

community detection on many small sparse graphs. Figure 4.1 depicts an example, where the Cooking or Cosmetic Apps inherits important topological information from the main Amazon graph for their enhanced community detection.

Note that, while the small sparse graphs can engage with a local field (like cooking or cosmetic in the example), the main graph can be quite comprehensive and noisy. As Figure 4.1 shows, not all the connections in Amazon (shopping graph) can be equally important for the two candidate app graphs. Three mutual users are selected where u_1 and u_2 mainly share similar shopping interests on cosmetics and u_1 and u_3 mainly share similar shopping interests on food products in Amazon. Then, with deliberate propagation from main graph, in the Cosmetic graph, u_1 and u_2 have a better chance to be grouped together, while u_1 and u_3 are more likely to be assigned the same community ID in the Cooking graph. Therefore, the proposed model should be able to differentiate various kinds of information from the main graph for each candidate sparse graph to enhance its local community detection performance.

As another challenge, small sparse graphs often suffer from training data insufficiency, e.g., the limited connections in these graphs can hardly tell the community residency information. In this study, I employed a novel data augmentation approach - cross graph pairwise learning. Given a candidate user and an associated user triplet, the proposed model can detection the community closeness superiority by leveraging main graph and the sparse graph simultaneously. Moreover, the proposed pairwise learning method can cope with the main graph heterogeneity issue and reduce noisy information by taking care of graph local structure. Theoretically, I can offer at most $\mathcal{O}(N^3)$ user triplets to learn graph community structure while conventional community detection methods by default can only be applied on $\mathcal{O}(N)$ users (N is the number of users in the sparse graph).

Inspired by aforementioned discussion, I propose an innovative *Pairwise Cross-graph Community Detection* (PCCD) model for enhanced sparse graph user community detection. Specifically, given user u_i and its associated triplet $\langle u_i, u_j, u_k \rangle$, I aim to predict their pairwise community relationship, e.g., compared with user u_k, user u_j should have closer, similar or farther community closeness to user u_i. The contribution of this paper is fourfold:

- I propose a novel problem - Cross Graph Community Detection, which can be critical for thousands of small businesses or services if they enable external user account login.

- Unlike conventional community detection methods, I explore community structure from a pairwise viewpoint. In this way, I can efficiently deal with graph heterogeneous information and solve cold-start problem for users with no behaviors in sparse graphs.

- The proposed model is trained in an end-to-end manner where a two-level filtering module locates the most relevant information to propagate between graphs. A Community Recurrent Unit (CRU) subsequently learns user community distribution from

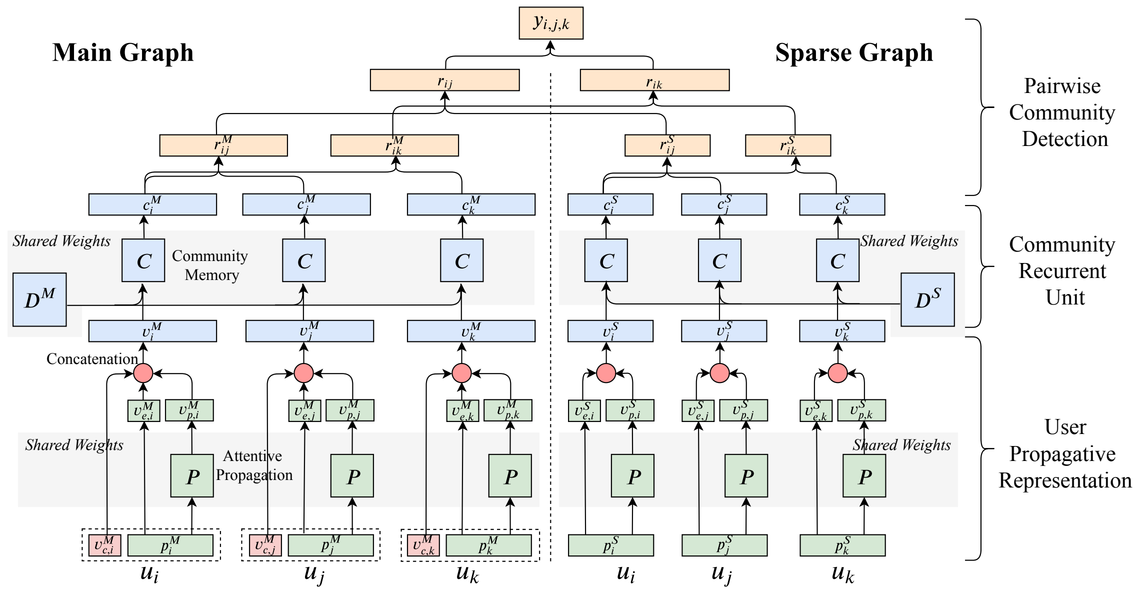


Figure 4.2: The overall architecture of our proposed PCCD model. It contains three major modules mentioned in the right part of the figure. Each training instance is a mutual user triplet $\langle u_i, u_j, u_k \rangle$. Compared with the sparse graph, the main graph involves raw user community as part of model input.

the filtered information.

- Extensive experiments on two real-world cross-graph datasets validate my model superiority. I also evaluate my model robustness on graphs with varied sparsity scales and many other supplementary studies.

4.2 Method

4.2.1 Task Overview

As aforementioned, conventional methods suffer from graph sparseness problem. In this study, I propose a Pairwise Cross-graph Community Detection (PCCD) model that particularly aims at detecting user pairwise community closeness in sparse graphs by involving cross-graph techniques.

Particularly, a well connected graph is called ”main graph” in this paper, corresponding to the targeted sparse graph. In general, as users may visit multiple cyber domains within a short time period, these mutual users co-occurred in both the main and sparse graph are taken as the bridge to connect the two graphs. Therefore, the relevant information from the main graph can be propagated to the sparse graph to support its user community detection.

Specifically, my proposed model (showed in Figure 4.2) is trained on mutual user triplets to learn three types of pairwise community relationship including "*similar*", "*closer*" and "*farther*". The goal of this study can be reformulated as the following pairwise learning task: Given a sparse graph S and a main graph M where N^C denotes their mutual user collection, for a mutual user triplet $\langle u_i, u_j, u_k \rangle \in N^C$ (**Model Input**), I aim to predict the relationship of its pairwise community closeness in graph S (**Model Output**). In this paper, "*similar*" relationship means that u_j and u_k are either in the same community or different communities with u_i. "*closer*" relationship means that u_j and u_i are in the same community, while u_k and u_i are in different communities. "*farther*" relationship means that u_j and u_i are in different communities, while u_k and u_i are in the same community.

For a mutual user triplet, my proposed model detects communities firstly on separate graphs to obtain all user raw community representations (Section 4.2.2). Their propagative representations are learned through a community-level and a node-level filter (Section 4.2.3). Both representations are jointly utilized for predicting user community affiliation in each graph via a Community Recurrent Unit (Section 4.2.4). In the end, I integrate the community affiliation distributions to identify the pairwise community relationship of the triplet (Section 4.2.5). Particular training strategies are introduced in the end (Section 4.2.6).

Note that even though my model is only trained on mutual users, it is still functional to detect pairwise community closeness on those users solely appeared in either the main or sparse graph. Further experiments in Section 4.4.3 will verify its effectiveness on different types of users.

4.2.2 Raw Community Representation

As the user behavior is enriched in the main graph M, detecting user communities in it would offer auxiliary community features to potentially enhance my model performance. Therefore, I decide to involve user communities in the main graph as a part of model input. The same information from the sparse graph is intentionally omitted because its sparse graph structure is not able to offer reliable community partition results.

Considering all possible community detection methods listed in Section 4.3.2, Infomap is empirically selected to detect user communities in the main graph M. Infomap method simulates a random walker wandering on the graph for m steps and indexes his random walk path via a two-level codebook. Its final goal aims to generate a community partition with the minimum random walk description length, which is calculated as follows:

$$L(\pi) = \sum_{i}^{m} q_{\curvearrowright}^{i} H(\mathcal{Q}) + \sum_{i=1}^{m} p_{\circlearrowright}^{i} H(\mathcal{P}^{i}) \tag{4.1}$$

where $L(\pi)$ is the description length for a random walker under current community partition π. $q_{\curvearrowright}^{i}$ and $p_{\circlearrowright}^{i}$ are the jumping rates between communities and within the i_{th} community in each step. $H(\mathcal{Q})$ is the frequency-weighted average length of codewords in the global index codebook and $H(\mathcal{P}^{i})$ is frequency-weighted average length of codewords in the i_{th} community codebook.

In the end, given a user u_i, I obtain its one-hot community representation $v_{c,i}^{M}$ in the main graph M, which is regarded as part of user representations for the model input.

4.2.3 User Propagative Representation

To better convey the mutual user insights in both main and sparse graph, their representations are learned from both direct transformation and weighted information propagation. The optimization processes are similar in the main and sparse graph. In the beginning, a user u_i can be represented as a multi-hot embedding p_i^M in the main graph M (left part

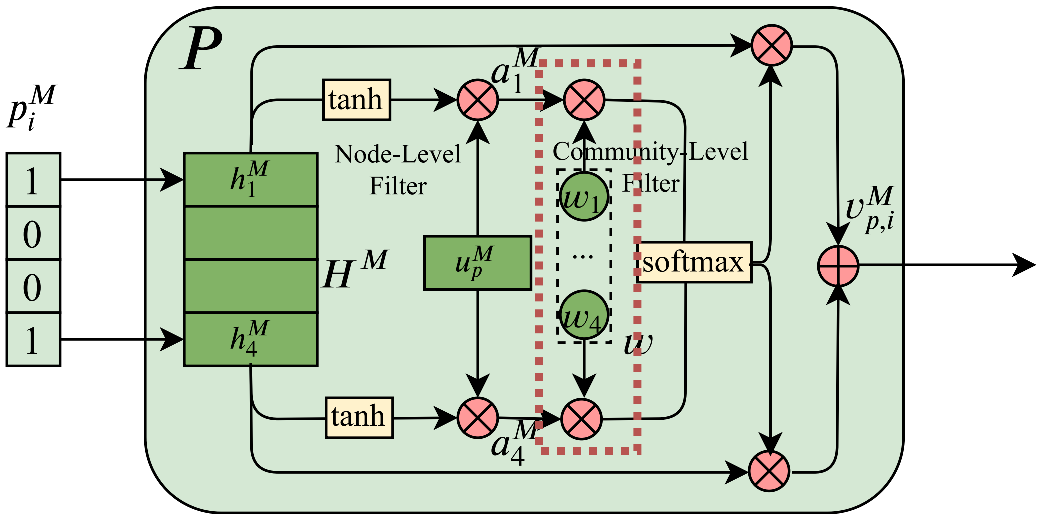


Figure 4.3: The two-level filter to support user propagative representation estimation in the main graph M. Sparse graph S does not contain the community-level filter (wrapped in the red rectangle).

in Figure 4.3) and p_i^S in the sparse graph S. Each dimension in the embedding refers to a unique object with which the user connects in the related graph.

The direct transformation from user multi-hot embedding learns a dense vector via one hidden layer so as to retrieve the compressed information from user connected objects directly, which is calculated as follows:

$$v_{e,i}^G = \tanh(W_e^G p_i^G + b_e^G) \tag{4.2}$$

where $G \in \{M, S\}$ denotes either the main or the sparse graph. W_e^G and b_e^G denote the weight matrix and bias respectively. $v_{e,i}^G$ is the learned direct transformation representation for user u_i.

On the other hand, from information propagation perspective, each object carries information, which can be propagated to its connected users as a type of representation. However, as such information are noisy and heterogeneous, not all of them are equally important to users. Therefore, a filtering module is proposed to automatically select appropriate information to propagate from both community level and node level. The overall flow of the filtering module is illustrated in Figure 4.3.

Community-Level Filter: As the raw detected communities in the sparse graph is not reliable because of graph sparseness, I only apply the community-level filter in the main graph M. According to the raw community partition π calculated in Section 4.2.2, each object node n_i is assigned to a community $\pi(n_i)$. The community-level filter propagate object information in a coarser manner by considering its community importance weight $w_{\pi(n_i)}$. Throughout this filter, I aim to learn a weight vector w where each dimension of w denotes the importance score of a related community.

Node-Level Filter: Node-level filter assesses the propagated information with a fine resolution. In the main graph M (a user-object bipartite graph), I aim to learn object representations $H^M = \{h_1^M, ..., h_n^M\}$ where h_n^M denotes the n_{th} object representation. Similarly, H^S denotes the object representations in the sparse graph S. The weight of the j_{th} object node decides the amount of its information propagating to its connected user nodes, which is calculated in an attentive means:

$$a_j^G = (u_p^G)^\intercal \tanh(W_p^G h_j^G + b_p^G) \tag{4.3}$$

where $G \in \{M, S\}$ denotes either the main or the sparse graph. u_p^G denotes the project vector in the related graph to calculate object attentive weights. W_p^G and b_p^G denote the corresponding weights and bias, respectively.

I combine the community-level and node-level weights together to quantify the information propagation in the main graph M. While only the node-level weights is considered in the sparse graph S. Normalized by the $\mathrm{softmax}(\cdot)$ function, the related representation of user u_i in both graphs are calculated as follows:

$$\begin{aligned} v_{p,i}^M &= \sum_{n_j \in \mathcal{N}^M(u_i)} \mathrm{softmax}(a_j^M w_{\pi(n_j)}) h_j^M \\ v_{p,i}^S &= \sum_{n_j \in \mathcal{N}^S(u_i)} \mathrm{softmax}(a_j^S) h_j^S \end{aligned} \tag{4.4}$$

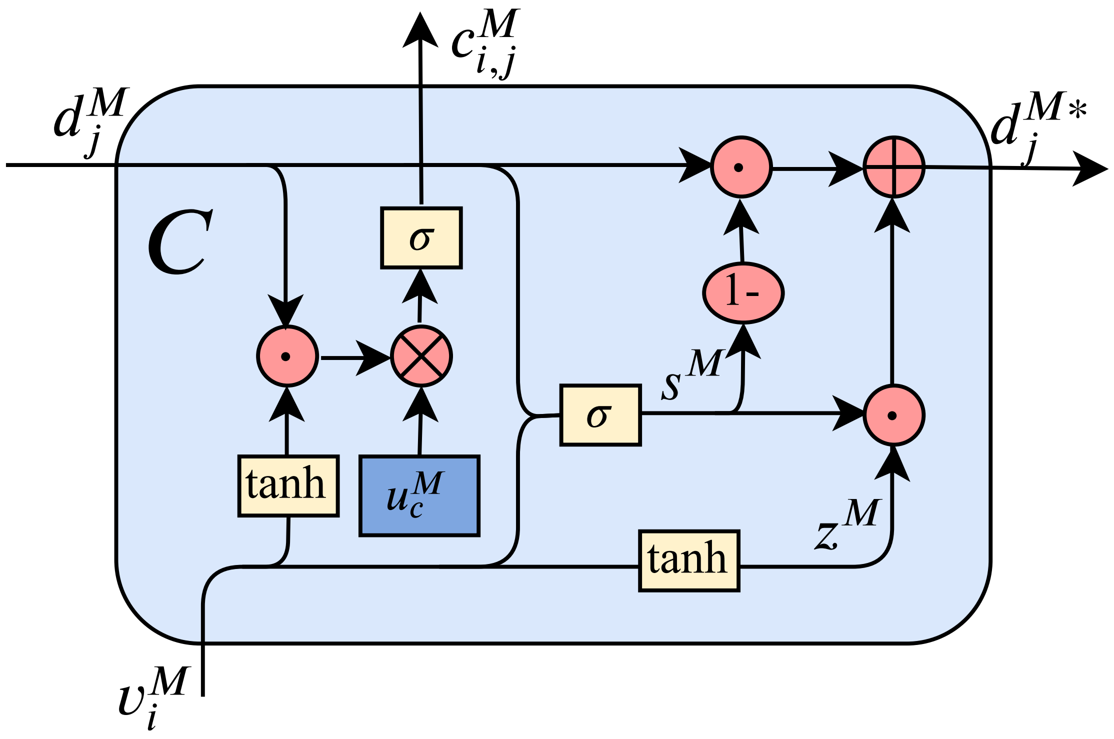


Figure 4.4: The flow of Community Recurrent Unit in the main graph M. The left part refers to the community affiliation gate and the right part is the community update gate.

Both $v^M_{p,i}$ and $v^S_{p,i}$ are the weighted sum of the neighbour object representations. $\mathcal{N}^S(n_i)$ and $\mathcal{N}^M(n_i)$ denote user connected objects in both graphs.

Finally, given a user u_i, its raw community representation, direct transformation representation and two-level filtered representation construct its propagative representation in the main graph M. While its direct transformation representation and node-level filtered representation construct its propagative representation in the sparse graph S. To avoid gradient vanishing, a $\mathrm{batch\_norm}(\cdot)$ function is applied on top of the concatenated representations in both graph:

$$\begin{aligned} v^M_i &= \mathrm{batch\_norm}([v^M_{c,i}, v^M_{e,i}, v^M_{p,i}]) \\ v^S_i &= \mathrm{batch\_norm}([v^S_{e,i}, v^S_{p,i}]) \end{aligned} \tag{4.5}$$

4.2.4 Community Recurrent Unit

After previous steps, each user of the mutual user triplet $\langle u_i, u_j, u_k \rangle$ is associated with a propagative representation. In this section, its corresponding community affiliation scores c_i^G are further calculated in related graphs $G \in \{M, S\}$ through a designed Community Recurrent Unit (CRU), which is showed in Figure 4.4. The CRU contains an affiliation gate to calculate the user affiliation score for each community and an update gate to update community self representations. Within this unit, a community memory $D^G = \{d_1^G, ..., d_K^G\}$ is designated to store K community representations. Particularly, community representation is required to be with the same dimension as user propagative representation in both graphs.

Community Affiliation Gate: The affiliation gate helps to generate the affiliation score of a user u_i towards each community in both graphs, which forms a K-dimensional vector $c_i^G = \{c_{i,1}^G, ..., c_{i,K}^G\}$ where $G \in \{M, S\}$. The user u_i's affiliation score $c_{i,j}^G$ towards the j_{th} community in the graph G is calculated as follows:

$$c_{i,j}^G = \sigma((u_c^G)^\intercal (\tanh(v_i^G) * d_j^G)) \tag{4.6}$$

The dot product between transformed user propagative representation $\tanh(v_i^G)$ and the j_{th} community representation d_j^G indicates their potential correlation, which further turns to a scalar affiliation score $c_{i,j}^G$ between 0 to 1 with the help of a projection vector u_c^G and normalization function $\sigma(\cdot)$.

Community Update Gate: When calculating u_i's community affiliation, its information can help to update community representations in return. The updated representation is relied on both previous step community representation and current user propagative representation. Therefore, to better embrace the two representations, I use a delicate RNN variant, where the update process is calculated as follows:

$$
\begin{aligned}
s^G &= \sigma(W_s^G[v_i^G, d_j^G] + b_s^G) \\
z^G &= \tanh(W_z^G v_i^G + b_z^G) \\
d_j^{G*} &= (1 - s^G) * d_j^G + s^G * z^G
\end{aligned}
\tag{4.7}
$$

where $G \in \{M, S\}$. s^G denotes the update rate and z^G is transformed user information to be updated in current community. d_j^{G*} denotes the updated representation of the j_{th} community in graph G, which is the weighted sum on previous community and current user representations. W_s^G and W_z^G, denote related weight matrices, while b_s^G and b_z^G, denote related biases.

Community Constraint: To obtain the communities with less overlapping information in graph $G \in \{M, S\}$, the community representations ideally should be independent with each other. In my model, cosine similarity $\cos(\cdot)$ is taken as the criteria to measure the community similarity where higher score indicates stronger correlation. The averaged cosine similarities of all possible community representation pairs is calculated as a community loss to minimize:

$$
\mathcal{L}_c^G = \frac{1}{2K^2} \sum_{i,j} \cos(d_i^G, d_j^G) \tag{4.8}
$$

where K is the number of communities in each graph.

4.2.5 Pairwise Community Detection

Similar to RankNet [**burges2010ranknet**], given a mutual user triplet $\langle u_i, u_j, u_k \rangle$, as three types of pairwise label are considered including "*closer*", "*similar*" and "*farther*", label $y_{i,j,k}$ is calculated as follows:

$$
y_{i,j,k} = \frac{1}{2}(1 + S_{jk}) \tag{4.9}
$$

In terms of the community closeness for u_i, if u_j is closer than u_k, $S_{jk} = 1$; if u_j is farther than u_k, $S_{jk} = -1$; if u_j and u_k are similar, $S_{jk} = 0$. In this way, I convert the pairwise ranking task to a regression task.

To optimize this task, first of all, I calculate the correlation representation r_{ij}^G between u_j and u_i in single graph $G \in \{M, S\}$:

$$r_{ij}^G = W_r^G(c_i^G - c_j^G) + b_r^G \tag{4.10}$$

where W_r^G and b_r^G are related weight and bias, respectively.

To construct the information from both graphs, I concatenate the correlation representations from both graphs:

$$r_{ij} = \tanh([r_{ij}^S, r_{ij}^M]) \tag{4.11}$$

Similarly, the cross-graph correlation representation between u_i and u_k is calculated as r_{ik}.

After that, I predict the community relationship between u_j and u_k towards u_i as follows:

$$\hat{y}_{i,j,k} = \sigma(W_o(r_{ij} - r_{ik}) + b_o) \tag{4.12}$$

where W_o and b_o are the related weight and bias. $\sigma(\cdot)$ denotes the sigmoid activation function. In the end, the final loss $\mathcal{L}_{total}$ is the weighted sum of the optimization loss calculated via cross entropy and the community constraint losses where $\mathcal{L}_c^S$ is for the sparse graph and $\mathcal{L}_c^M$ is for the main graph:

$$\mathcal{L}_{total} = \underbrace{-y_{i,j,k} log \hat{y}_{i,j,k} - (1 - y_{i,j,k}) log(1 - \hat{y}_{i,j,k})}_{optimization} + \underbrace{\alpha(\mathcal{L}_c^S + \mathcal{L}_c^M)}_{constraint} \tag{4.13}$$

4.2.6 Training Strategy

Although my model is trained solely on mutual user triplets, it is also able to detect pairwise community closeness among users only appeared in either sparse graph or main graph.

Two training tricks are particularly employed in our model to improve model robustness, including shared weights and masked training.

First, as my model should be insensitive to the sequence of input user triplet, the weight matrices, biases and project vectors in both Section 4.2.3 and Section 4.2.4 should be shared among the three users. Moreover, in Section 4.2.4, the community memory D^G where $G \in \{M, S\}$ is updated by taking the average of the three updated community representations (calculated in Eq. 4.7) from input triplets.

Second, to alleviate the negative effects of excessive connections in main graph M, I employ a masked training strategy by randomly remove a small ratio ρ of connected objects from users in the main graph during each training batch. $\rho = 0$ means users remain their original multi-hot embeddings while $\rho = 1$ means users lose all their connections on objects.

4.3 Experiments

4.3.1 Dataset

As one of the largest e-commerce ecosystems, Alibaba owns multiple online businesses, where users could try different services via the same login ID. For this experiment, I employ three services from Alibaba including an online shopping website, a digital news portal and a cooking recipe website. The online shopping website is regarded as the main graph because it is Alibaba core service having the largest number of active users. By contrast, the other two services are regarded as sparse graphs. In the online shopping website, each object refers to a product for sale, and links are user purchase behaviors on products. In the digital news portal, each object refers to a news article, and links are user reads on news. In the cooking recipe website, each object refers to a cooking recipe, and links mean that users follow recipe instructions.

To prove my model robustness with respect to the graph size, two cross-graph datasets are constructed in different scales, which are showed in Table 4.1. The SH-NE dataset is the cross-graph dataset constructed by the shopping graph and news graph, which is ten times larger than the SH-CO dataset constructed by the shopping graph and recipe graph. Users are categorized into three different types including *MU* (mutual users appeared in both graphs), *MO* (users only appeared in the main graph), and *SO* (users only appeared in the sparse graph). Note that, only mutual user triplets are used for training.

Dataset	**#MU**	**Main Graph**			**Sparse Graph**		
		#MO	**#Object**	**#Link**	**#SO**	**#Object**	**#Link**
SH-NE	105,702	392,549	135,320	17,352,482	26,133	9,377	881,721
SH-CO	1,029	1,543	23,881	1,161,293	719	3,739	147,266

Table 4.1: Statistics of Two cross-graph datasets.

Both datasets are built by taking a week of user behaviors from 09/20/2018 to 09/26/2018. In this paper, for two sparse graphs, I aim to predict user pairwise community labels (calculated from enriched seven-day user behaviors) using sparse user behaviors (the first four-day user behaviors, arbitrarily defined in this paper).

To construct my ground truth data, I run all baseline models (will be introduced in Section 4.3.2) on both seven-day sparse graphs which are assumed to contain sufficient information for detecting user communities. Only mutual user communities agreed by all seven baselines are kept, from which mutual user triplets are subsequently selected. In this paper, I define three types to label pairwise community closeness for a mutual user triplet, including "*closer*", "*similar*", and "*farther*" (detailed definition is in Section 4.2.1). Equal number of triplets are randomly selected for each label type. In total, there are 207,291 triplets in SH-NE dataset and 20,313 triplets in SH-CO dataset for testing propose.

Training data generation is the same as ground truth data construction except using first four-day user behaviors. In total, there are 2,072,905 triplets in SH-NE dataset and 203,123 triplets in SH-CO dataset, from which 90% of the triplets are used for training and 10% for validation.

In both training and testing process, my model input is user triplets with first four-day behavior information. Their difference is that the pseudo labels for training are generated from four-day user behaviors, while the ground truth labels for testing are generated from whole seven-day user behaviors in sparse graphs.

4.3.2 Baselines and Settings

As there is neither reliable nor open-sourced studies on pairwise cross-graph community detection, I have no direct comparative models. In this paper, I select one random model and six compromised but state-of-art baselines which are originally designed for whole graph community detection. In all seven baselines, after user communities are detected, I calculate user triplet labels via Eq. 4.9. As all baselines are trained on the sparse graph with first four-day user behavior and the main graph, they bring no information loss compared with my model. Here is the details of how these baselines calculate user communities:

- **random**: users are randomly assigned to a given number of communities.
- **Infomap**: A random walk based method to detect user communities by minimizing description length of walk paths[1].
- **DNR**: A novel nonlinear reconstruction method to detect communities by adopting deep neural networks on pairwise constraints among graph nodes[2].
- **LSH**: A random walk based method for community detection in large-scale graphs[3].
- **node2vec**: A skip-gram model to generate node emebeddings based on guided random walks on graphs. And K-means method subsequently calculates communities from the generated embeddings[4].
- **BigClam**: A non-negative matrix factorization method to detect overlapping communities in large scale graphs[4].

[1]https://github.com/mapequation/infomap
[2]http://yangliang.github.io/code/
[3]https://github.com/melifluos/LSH-community-detection
[4]https://github.com/snap-stanford/snap/

- **WMW**: A fast heuristic method for hierarchical community detection inspired by agglomerative hierarchical clustering[5].

The result is reported with best hyper-parameters from grid search. Empirically, a mini-batch (size = 200) Adam optimizer is chosen to train my model with learning rate as 0.01. Community number on both graphs is $K = 50$ (Eq. 4.8). Community constraint weight is $\alpha = 0.1$ (Eq. 4.13). Masked training rate in the main graph is $\rho = 0.05$. The model is trained for 1000 epochs in order to get a converged result.

Model performance is evaluated from both classification and retrieval viewpoints. From classification aspect, Accuracy (ACC), F1-macro (F1) and Matthews Correlation Coefficient (MCC) are reported metrics. From retrieval aspect, Mean Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain (NDCG) and Mean Average Precision (MAP) are reported metrics. All of them are fundamental metrics widely used for model evaluation.

4.4 Discussion

4.4.1 Performance Comparison

All model performance are evaluated from both classification and retrieval viewpoints. Each baseline method is run on three graphs including the main graph, the sparse graph, and their combined graph. While our model can only be tested by utilizing both graphs. Table 4.2 shows the overall model performance under three graphs. Particularly, we run our model ten times and report the average performance result. To verify our model's superiority, we calculate the performance differences between our model and the best baseline on each metric for all the runs, and apply a pairwise t-test to check whether the performance difference is significant.

Our model output is a predicted score in the range from 0 to 1. Its associated label is assigned to be one of "*closer*", "*similar*" or "*farther*" based on which third the predicted

[5]https://github.com/eduarc/WMW

score lies in (associated with label definition in Section 4.2.5). For example, a user triplet $\langle u_i, u_j, u_k \rangle$ with score 0.3 will be labelled as "*farther*", which means compared with u_k, u_j is has a farther community closeness to u_i. After that, from classification perspective, ACC, F1 and MCC scores are calculated by comparing predicted labels with the ground truth labels. While from retrieval perspective, MRR, NDCG and MAP are calculated for the pairwise ranking within user triplets.

Table 4.2 shows that all baseline performances in the combined graph achieve the best among three types of graphs. And performances in the main graph are always better than in the sparse graph. It demonstrates that main graph holds beneficial information to reveal user potential communities in sparse graphs. The random baseline is with similar performance among all three graphs, which makes sense because the random community partition is regardless of graph structures. It can be used as the default model to compare with in each graph. For the rest six baselines, random walk based baselines (Infomap and LSH) do not perform well in all three graphs. Their evaluation results are just a bit better than random model performance. WMW model performs the best among all baselines. Actually, its results are always the best in three graphs of both datasets. However, by taking the pairwise t-test, it statistically proves that our model still significantly outperforms the WMW model in terms of all metrics.

It is also interesting to see that the model evaluation results are consistent in all datasets. For example, DNR model always performs better than LSH model. The only exception is that BigClam performs better than node2vec in SH-NE dataset, but worse in SH-CO dataset.

4.4.2 Ablation Study

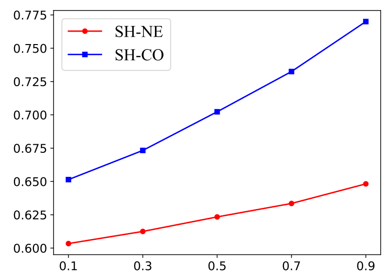
In this section, we aim to explore whether all components of our proposed model have positive effect on final model performance and which component has the largest impact. There are six components that can be disassembled from the main model, including Raw

Community Representation in the main graph (RCR), Direct Transformation Representation (DTR), Node-level filter (NF), Community-level filter (CF), Community Constraint (CC) and Masked Training (MT). We iteratively remove each component while keep the rest firmed to demonstrate their performance difference compared to the original model. All comparison results are showed in Table 4.3.

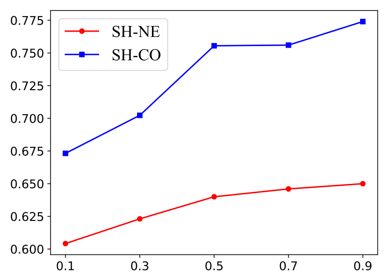
Among the six components, if we remove node-level filter, the performance drop dramatically, even worse than some of baseline models. It indicates that node-level information are excessive and noisy. Without appropriate filtering on the propagated information, it would vitally pollute the quality of user representations. Similarly, community-level filter also has a huge impact on model performance, meaning filtering propagated information from a coarser view also has a positive effect on learning user representations. By contrast, the two extra representations (RCR and DTR) do not have strong influence in our model. Having these information can slightly improve model performance as they still involve extra information. However, as one-hot embeddings, the amount of new information that RCR can offer is very limited. Similarly, DTR is only one-layer transformation on multi-hot embeddings. The information it can offer is also limited. By adding extra constraint, CC also has a little positive effect to enhance model performance. But as we always set its weight with a small value in training process, its influence is also indifferent. MT has the least impact on model performance as randomly masking user behaviors in the main graph has an overlapped effect with attentive weights calculated from the node-level filter in Section 4.2.3.

4.4.3 User-Type Analysis

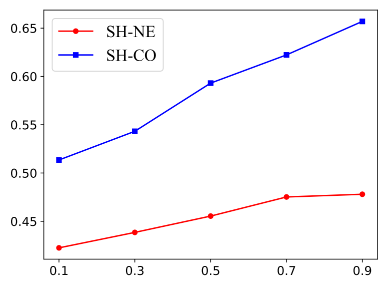
As aforementioned in Section 4.3.1, there are three types of users in our cross-graph datasets including *MU* (mutual users appeared in both graphs), *MO* (users only appeared in the main graph), and *SO* (users only appeared in the sparse graph). Unlike our training data which is constructed only with mutual users, our testing data can be with all types of users.



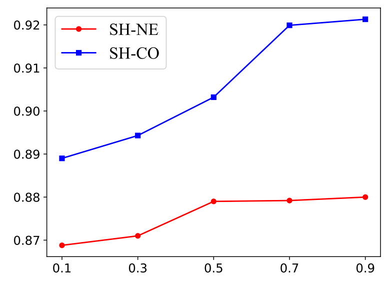
(a) ACC



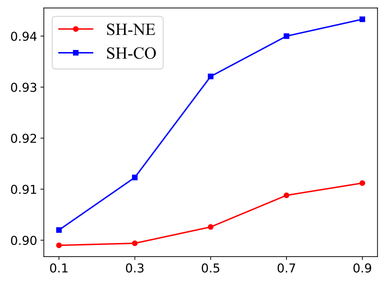
(b) F1



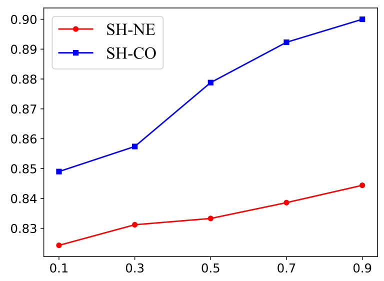
(c) MCC



(d) MRR



(e) NDCG



(f) MAP

Figure 4.5: Our model performance on all metrics under different graph sparsity scales. X-axis is the ratio of links preserved in the sparse graph. Y-axis is related metric score.

To examine our model performance on each type of users, user triplets with only single type of users are screened, i.e., an eligible user triplet only contains three *MO* type users. In fact, the performance on *MO* user triplets reflects the capability of our model to solve cold-start problem as these users have no behaviors in the sparse graphs. The result showed in Table 4.4 demonstrates that the label of *MU* triplets can be best identified, which makes sense as the information of mutual users come from both graphs. While performance results on *MO* are better than *SO* user triplets, which might because that users are likely to have more enriched behaviors in the main graph compared with the sparse graph. Even that, the performance of our model on *SO* user triplets is still better than most of baselines running on the combined graph.

4.4.4 Graph Sparsity Influence

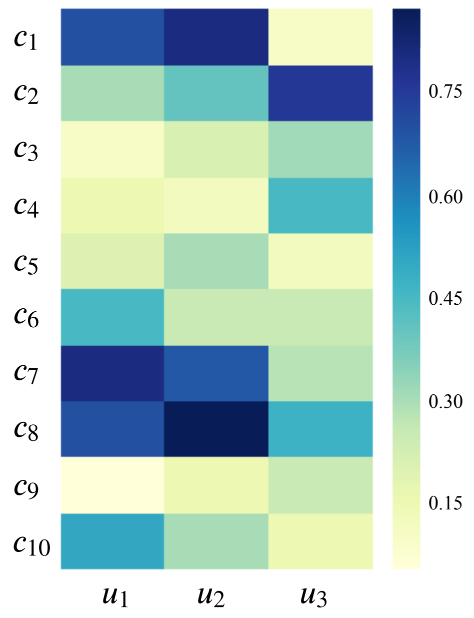
To explore our model robustness on graph sparsity, we randomly keep δ ratio of the total links in the sparse graph, where $\delta = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. Our model is trained on the whole main graph and each truncated sparse graph. The reported evaluation metrics under all sparse graphs with varied scales are visualized in Figure 4.5. All metrics in both SH-NE and SH-CO datasets are in a rising trend with more links preserved in related sparse graphs. Compared with SH-NE dataset, SH-CO can benefit more from the sparse graph as it has a larger increase in all reported metrics.

With more links are preserved, all evaluation metrics first grows rapidly. While the increasing speed slows down when 70%-90% links are preserved. This phenomenon can be seen in almost all plots in Figure 4.5. It can be explained by the law of diminishing marginal utility, meaning that the marginal utility to bring new information derived from each additional link is declined.

Although classification related metrics significantly increase with more links preserved (i.e., F1 score in SH-CO dataset increases 10% when involving sparse graph), the retrieval related metrics (MRR, NDCG and MAP) do not change much in the mean time. One possible reason is that the information from the main shopping graph already contains enough information for the pairwise ranking in user triplets. For example, without considering sparse graph information, our model has already achieved high MRR score as 0.87 in SH-NE dataset and 0.89 in SH-CO dataset, which is already better than most baseline results running on the combined graph.

4.4.5 Case Study

In order to testify whether our proposed Community Recurrent Unit is able to accurately calculate user affiliation scores in each community, we choose three users (labelled from user u_1 to user u_3) and calculate their affiliation scores of ten selected communities (labelled from community c_1 to community c_{10}) in the main shopping graph. The detailed result is visualized in Figure 4.6. In the left part of the figure, darker color indicates higher affiliation scores in the range from 0 to 1. Besides, in the shopping graph, we also extract all products purchased by each user, and manually select the most representative ones



User	Keywords
u_1	maxi dress, long sleeve, pullover hoodie, oversized sweatshirt, jeans
u_2	hoodie, linen blazer, leather jacket, sportswear, legging short
u_3	window shades, light, kitchen countertops, toilet, ceiling fan

Figure 4.6: The left part shows ten community affiliation scores of three selected users in the shopping graph. The right part shows the keywords of their purchased products.

summarized as keywords in a table, which is demonstrated in the right part of Figure 4.6.

From the right-part table, u_1 and u_2 share similar shopping interests in clothing products. u_2 is more like a girl fond of sports. And u_1 more tends to be a fashion girl. While the shopping interests of u_3 is much far away from the other users. All purchased products by u_3 is furnishing stuffs. It seems s/he is decorating her/his living space. Referring to the left part of Figure 4.6, the affiliation distribution among ten communities between u_1 and u_2 are very similar. They both have a high weight in community c_1, c_7 and c_8. While the community distribution of u_3 is totally different. u_3 is heavily affiliated with community c_2. The results of our calculated community affiliation distribution is consistent with actual user behaviors, which indicates our Community Recurrent Unit (CRU) is functional well to reflect user community information.

Graph	Model	SH-NE Dataset						SH-CO Dataset					
		ACC	**F1**	**MCC**	**MRR**	**NDCG**	**MAP**	**ACC**	**F1**	**MCC**	**MRR**	**NDCG**	**MAP**
Main	random	0.3324	0.1677	-0.0001	0.7719	0.8373	0.7299	0.3302	0.1700	0.0027	0.7842	0.8444	0.7401
	Infomap	0.3700	0.2731	0.1201	0.8334	0.8421	0.7299	0.3811	0.3723	0.1000	0.7888	0.8419	0.7430
	DNR	0.4210	0.4022	0.1789	0.8000	0.8366	0.7473	0.5194	0.5122	0.2712	0.8099	0.8773	0.7592
	LSH	0.3798	0.3813	0.0888	0.8002	0.8433	0.7328	0.4000	0.4123	0.1277	0.7812	0.8329	0.7570
	node2vec	0.4888	0.5014	0.2521	0.8229	0.8744	0.7832	0.5959	0.5832	0.3895	0.8422	0.8936	0.8222
	BigClam	0.5555	0.5399	0.3349	0.8421	0.8890	0.8024	0.5301	0.5334	0.3290	0.8335	0.8573	0.8005
	WMW	0.6032	0.6158	0.4111	0.8632	0.8988	0.8256	0.6489	0.6661	0.5001	0.8789	0.9032	0.8499
Sparse	random	0.3333	0.1785	0.0003	0.7815	0.8438	0.7419	0.3301	0.1720	0.0030	0.7801	0.8400	0.7338
	Infomap	0.3455	0.1929	0.0031	0.7843	0.8399	0.7437	0.3676	0.3211	0.0422	0.7905	0.8446	0.7401
	DNR	0.4433	0.4407	0.2020	0.8313	0.8678	0.7742	0.5273	0.5289	0.2787	0.8293	0.8765	0.7980
	LSH	0.3889	0.3917	0.0942	0.8008	0.8528	0.7343	0.4093	0.4177	0.1138	0.7833	0.8438	0.7404
	node2vec	0.4849	0.4958	0.2467	0.8219	0.8664	0.7774	0.5732	0.5746	0.3898	0.8412	0.8823	0.8146
	BigClam	0.5277	0.5355	0.3332	0.8441	0.8750	0.8005	0.5282	0.5280	0.3339	0.8298	0.8399	0.8033
	WMW	0.5814	0.5900	0.3988	0.8666	0.9001	0.8210	0.5911	0.6212	0.4347	0.8599	0.8890	0.8133
Main + Sparse	random	0.3337	0.1788	-0.0008	0.7815	0.8387	0.7387	0.3291	0.1699	0.0027	0.7739	0.8331	0.7310
	Infomap	0.3625	0.2567	0.0908	0.7864	0.8423	0.7437	0.3930	0.3988	0.0916	0.7951	0.8488	0.7528
	DNR	0.4383	0.4204	0.1817	0.8101	0.8599	0.7686	0.5273	0.5322	0.2957	0.8391	0.8812	0.8000
	LSH	0.3992	0.4052	0.0993	0.8020	0.8538	0.7599	0.4180	0.4237	0.1309	0.8021	0.8539	0.7600
	node2vec	0.5072	0.5124	0.2611	0.8339	0.8774	0.7943	0.6015	0.6055	0.4045	0.8677	0.9023	0.8324
	BigClam	0.5580	0.5626	0.3412	0.8511	0.8901	0.8135	0.5398	0.5383	0.3389	0.8475	0.8874	0.8095
	WMW	0.6168	0.6205	0.4281	0.8702	0.9042	0.8353	0.6682	0.6711	0.5041	0.8879	0.9173	0.8561
	PCCD	**0.6524***	**0.6551***	**0.4808***	**0.8802***	**0.9116***	**0.8470***	**0.7740***	**0.7758***	**0.6627***	**0.9244***	**0.9442***	**0.9005***

Table 4.2: All model performances for three different graph combinations in two datasets. Symbol '*' highlights the cases where our model significantly beats the best baseline with $p < 0.01$.

Dataset	**Model**	**ACC**	**F1**	**MCC**	**MRR**	**NDCG**	**MAP**
SH-NE	– RCR	-2.61	-2.51	-3.76	-0.73	-0.54	-0.96
	– DTR	-1.32	-1.49	-2.96	-1.01	-0.14	-0.57
	– NF	-17.31	-17.01	-25.92	-5.35	-3.88	-6.05
	– CF	-7.94	-8.48	-11.71	-2.61	-1.92	-3.01
	– CC	-2.15	-2.05	-3.08	-0.49	-0.35	-0.65
	– MT	-2.43	-2.55	-4.82	-0.78	-0.82	-1.24
SH-CO	– RCR	-2.13	-2.45	-3.07	-1.65	-0.58	-0.79
	– DTR	-3.45	-2.89	-1.95	-2.22	-0.43	-0.88
	– NF	-26.13	-25.83	-39.97	-9.81	-7.50	-19.35
	– CF	-21.12	-5.53	-8.45	-2.03	-1.50	-2.50
	– CC	-4.32	-4.34	-6.57	-1.56	-1.24	-1.91
	– MT	-1.32	-1.24	-2.66	-0.33	-0.21	-0.45

Table 4.3: Performance differences on all evaluation metrics between each ablated model and the original model. “-” refers to remove related component from our model. Results are scaled in percentage (%).

Dataset	**User**	**ACC**	**F1**	**MCC**	**MRR**	**NDCG**	**MAP**
SH-NE	MU	0.7132	0.7133	0.6012	0.9112	0.9324	0.8704
	MO	0.6356	0.6477	0.4780	0.8739	0.9002	0.8418
	SO	0.6009	0.6117	0.4324	0.8600	0.8988	0.8393
SH-CO	MU	0.7852	0.7780	0.6724	0.9300	0.9474	0.9015
	MO	0.7334	0.7565	0.6601	0.9012	0.9400	0.8874
	SO	0.6823	0.6915	0.6844	0.8990	0.9222	0.8789

Table 4.4: Three types of user performance in two datasets.

CHAPTER 5
SEMI-SUPERVISED EXPLAINABLE COMMUNITY DETECTION IN DYNAMIC GRAPHS

5.1 Introduction

Understanding product aspect-sentiments and tracking its changes in a timely manner can support better decision-making for commercial purposes, such as enlightening online retailers to make timely sales plans. Therefore, the *Dynamic Summarization on Product Aspects* task is of great importance. It not only indicates products' dynamic aspect-sentiment changes, but also depicts the changes into readable contexts for easier interpretation.

Prior investigations on another similar topic, review summarization, mainly follow Natural Language Generation (NLG) approaches, such as using new Recurrent Neural Network (RNN) variant that uses gated connections to construct a character-level text generation mode or designing a multi-task model to predict rating and generate review summarization simultaneously using pairwise user-product relationship. Memory network is also used for review summarization generation. However, all these models are originally designed for static review summarization and take product reviews as model input. They are vulnerable to depict product sentiment changes because of (real-time review) data sparsity. For instance, after investigating 2.16 billion products sold by *Taobao*, a world-leading online shopping website owned by Alibaba, only 0.05% of products are able to gather more than 100 reviews within a three-day window. Thus, review-based approaches are not feasible for dynamic summarizations in large scope because of the lack of instant reviews.

On the other hand, user behavior offers an alternative to address sentiment dynamics. Based on the statistics of the *Taobao* collection, more than 2.53% of products can receive more than 100 multi-type user behaviors (e.g., *'Click'* or *'Purchase'*) within a three-day

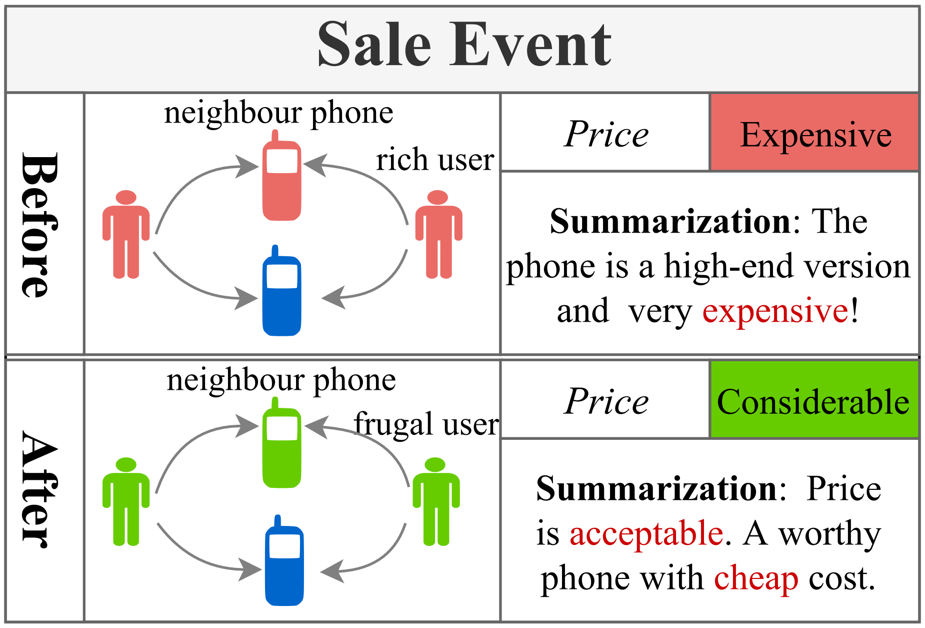

Sale Event
Before
neighbour phone
rich user
Price
Expensive
Summarization: The phone is a high-end version and very expensive!
After
neighbour phone
frugal user
Price
Considerable
Summarization: Price is acceptable. A worthy phone with cheap cost.

Figure 5.1: An example to illustrate how to dynamically select neighbour products (red phone → green phone) for depicting current product (blue phone) sentiment change before & after a sale event from user behaviors.

window where the coverage is 50 times greater than the review scope. Rational Choice Theory, on the theory side, proves that user shopping behavior rationality has a coherent relationship with the product peculiarity. As Figure 5.1 depicted, when a sale event on a high-end phone brings frugal users' instant *clicks*, behavior-based algorithms can immediately consume this information and locate updated neighbour products (red phone → green phone), whose sufficient reviews help to update the product (blue phone) summarization. For review-based approaches, accumulating enough reviews to characterize this dynamic change may take a longer time.

In this paper, instead of review summarization, I aim to generate product aspect summarization in a dynamic manner. They are similar topics but still with huge differences in terms of concept definition and generated summary context. Conceptually, review summarization reflects customer subjective and personalized expressions on products. While aspect summarization contains objective descriptions only on restricted product aspects.

Contextually, review summarization contains more emotional and general terms in a free format, such as ‘I love its color so much’. While aspect summarization generates more formal and descriptive expressions, which only focus on specific aspects such as ‘price is more expensive than expected’.

Motivated by all mentioned above, I propose a **B**ehavior based **D**ynamic **S**ummarization (BDS) model to accommodate user behavior for dynamic product aspect summarization. The user shopping preference is stable in a relatively long-term period, which offers us the theoretical feasibility to learn product behavior representation from user dynamic behavior and consistent shopping preference. The learned representation supports neighbor product selection from a group of seed products with abundant instant reviews (**Task 1**) and meanwhile implicitly helps to generate aspect summarization from product own descriptive phrases and neighbour products’ filtered sentimental phrases (**Task 2**). As both user behavior and seed products’ instant reviews are changed across time, the selected neighbour products and generated summarizations are associated with changes as well. The contribution of this work is threefold:

- To the best of my knowledge, this is the first effort to leverage user behavior for dynamic summarization on product aspects. This work pioneers behavior-based summarization investigation.
- In my model, a reinforcement learning approach learns the sampling strategy on seed products with rewards from both sentiment (calculated from behavior-to-sentiment prediction) and semantic (calculated from summarization generation) viewpoints. When generating product aspect summarization, my model does not require target product’s reviews as model input, which is able to solve review sparseness, even zero-review, problem.
- Experiments on a large E-commerce dataset show that my proposed model significantly outperforms the baselines from both automatic and human perspectives. Ex-

tensive studies also prove the efficacy of each model input component.

5.2 BDS Model

5.3 Experiments

5.4 Discussion

CHAPTER 6
CONCLUSION

6.1 Contributions

6.2 Limitation

Existing methods limitation: [268],[269],[270],[271] ,[272],

6.3 Future Work

REFERENCES

[1] Mark EJ Newman. "Communities, modules and large-scale structure in networks". In: *Nature physics* 8.1 (2012), pp. 25–31.

[2] Santo Fortunato and Darko Hric. "Community detection in networks: A user guide". In: *Physics Reports* 659 (2016), pp. 1–44.

[3] Fragkiskos D Malliaros and Michalis Vazirgiannis. "Clustering and community detection in directed networks: A survey". In: *Physics Reports* 533.4 (2013), pp. 95–142.

[4] Steve Harenberg et al. "Community detection in large-scale networks: a survey and empirical evaluation". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 6.6 (2014), pp. 426–439.

[5] Emmanuel Abbe. "Community detection and stochastic block models: recent developments". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6446–6531.

[6] Santo Fortunato. "Community detection in graphs". In: *Physics reports* 486.3-5 (2010), pp. 75–174.

[7] Michele Coscia, Fosca Giannotti, and Dino Pedreschi. "A classification for community discovery methods in complex networks". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 4.5 (2011), pp. 512–546.

[8] Maria CV Nascimento and Andre CPLF De Carvalho. "Spectral methods for graph clustering–a survey". In: *European Journal of Operational Research* 211.2 (2011), pp. 221–231.

[9] Jierui Xie, Stephen Kelley, and Boleslaw K Szymanski. “Overlapping community detection in networks: The state-of-the-art and comparative study”. In: *Acm computing surveys (csur)* 45.4 (2013), p. 43.

[10] Tanmoy Chakraborty et al. “Metrics for community analysis: A survey”. In: *ACM Computing Surveys (CSUR)* 50.4 (2017), p. 54.

[11] Muhammad Aqib Javed et al. “Community detection in networks: A multidisciplinary review”. In: *Journal of Network and Computer Applications* 108 (2018), pp. 87–111.

[12] Jungeun Kim and Jae-Gil Lee. “Community detection in multi-layer graphs: A survey”. In: *ACM SIGMOD Record* 44.3 (2015), pp. 37–48.

[13] Alessia Amelio and Clara Pizzuti. “Overlapping community discovery methods: a survey”. In: *Social Networks: Analysis and Case Studies*. Springer, 2014, pp. 105–125.

[14] Punam Bedi and Chhavi Sharma. “Community detection in social networks”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 6.3 (2016), pp. 115–135.

[15] Mingqing Huang et al. “Overlapping community detection in heterogeneous social networks via the user model”. In: *Information Sciences* 432 (2018), pp. 164–184.

[16] Yizhou Sun et al. “Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7.3 (2013), p. 11.

[17] Manish Gupta, Jing Gao, and Jiawei Han. “Community distribution outlier detection in heterogeneous information networks”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2013, pp. 557–573.

[18] Dongxiao He et al. “A Stochastic Model for Detecting Heterogeneous Link Communities in Complex Networks.” In: *AAAI*. 2015, pp. 130–136.

[19] Yizhou Sun, Charu C Aggarwal, and Jiawei Han. “Relation strength-aware clustering of heterogeneous information networks with incomplete attributes”. In: *Proceedings of the VLDB Endowment* 5.5 (2012), pp. 394–405.

[20] Filippo Radicchi. “Detectability of communities in heterogeneous networks”. In: *Physical Review E* 88.1 (2013), p. 10801.

[21] Yang Zhou and Ling Liu. “Social influence based clustering of heterogeneous information networks”. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, pp. 338–346.

[22] Marya Bazzi et al. “Community detection in temporal multilayer networks, with an application to correlation networks”. In: *Multiscale Modeling & Simulation* 14.1 (2016), pp. 1–41.

[23] Manlio De Domenico et al. “Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems”. In: *Physical Review X* 5.1 (2015), p. 11027.

[24] Toni Valles-Catala et al. “Multilayer stochastic block models reveal the multilayer structure of complex networks”. In: *Physical Review X* 6.1 (2016), p. 11036.

[25] Ling Huang, Chang-Dong Wang, and Hong-Yang Chao. “A harmonic motif modularity approach for multi-layer network community detection”. In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2018, pp. 1043–1048.

[26] Florent Krzakala et al. “Spectral redemption in clustering sparse networks”. In: *Proceedings of the National Academy of Sciences* 110.52 (2013), pp. 20935–20940.

[27] Yudong Chen, Sujay Sanghavi, and Huan Xu. “Clustering sparse graphs”. In: *Advances in neural information processing systems*. 2012, pp. 2204–2212.

[28] Arash A Amini et al. “Pseudo-likelihood methods for community detection in large sparse networks”. In: *The Annals of Statistics* 41.4 (2013), pp. 2097–2122.

[29] Jess Banks et al. “Information-theoretic thresholds for community detection in sparse networks”. In: *Conference on Learning Theory*. 2016, pp. 383–416.

[30] Peter Chin, Anup Rao, and Van Vu. “Stochastic block model and community detection in sparse graphs: A spectral algorithm with optimal rate of recovery”. In: *Conference on Learning Theory*. 2015, pp. 391–423.

[31] Olivier Guédon and Roman Vershynin. “Community detection in sparse networks via Grothendieck’s inequality”. In: *Probability Theory and Related Fields* 165.3-4 (2016), pp. 1025–1049.

[32] Atieh Mirshahvalad et al. “Significant communities in large sparse networks”. In: *PloS one* 7.3 (2012).

[33] Tiago P Peixoto and Martin Rosvall. “Modelling sequences and temporal networks with dynamic community structures”. In: *Nature communications* 8.1 (2017), p. 582.

[34] Myunghwan Kim and Jure Leskovec. “Nonparametric multi-group membership model for dynamic networks”. In: *Advances in neural information processing systems*. 2013, pp. 1385–1393.

[35] J-C Delvenne, Sophia N Yaliraki, and Mauricio Barahona. “Stability of graph communities across time scales”. In: *Proceedings of the National Academy of Sciences* (2010).

[36] Amir Ghasemian et al. “Detectability thresholds and optimal algorithms for community structure in dynamic networks”. In: *Physical Review X* 6.3 (2016), p. 31005.

[37] Kevin Xu. “Stochastic block transition models for dynamic networks”. In: *Artificial Intelligence and Statistics*. 2015, pp. 1079–1087.

[38] Marta Sarzynska et al. “Null models for community detection in spatially embedded, temporal networks”. In: *Journal of Complex Networks* 4.3 (2016), pp. 363–406.

[39] Laetitia Gauvin, André Panisson, and Ciro Cattuto. “Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach”. In: *PloS one* 9.1 (2014).

[40] Kevin S Xu and Alfred O Hero. “Dynamic stochastic blockmodels for time-evolving social networks”. In: *IEEE Journal of Selected Topics in Signal Processing* 8.4 (2014), pp. 552–562.

[41] Xin Huang et al. “Querying k-truss community in large and dynamic graphs”. In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 2014, pp. 1311–1322.

[42] Lei Tang, Huan Liu, and Jianping Zhang. “Identifying evolving groups in dynamic multimode networks”. In: *IEEE Transactions on Knowledge and Data Engineering* 24.1 (2011), pp. 72–85.

[43] Tanja Hartmann, Andrea Kappes, and Dorothea Wagner. “Clustering evolving networks”. In: *Algorithm Engineering*. Springer, 2016, pp. 280–329.

[44] Yudong Chen, Vikas Kawadia, and Rahul Urgaonkar. “Detecting overlapping temporal community structure in time-evolving networks”. In: *arXiv preprint arXiv:1303.7226* (2013).

[45] Chang-Dong Wang, Jian-Huang Lai, and S Yu Philip. “NEIWalk: community discovery in dynamic content-based networks”. In: *IEEE transactions on knowledge and data engineering* 26.7 (2013), pp. 1734–1748.

[46] Chang-Dong Wang, Jian-Huang Lai, and Philip S Yu. “Dynamic community detection in weighted graph streams”. In: *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM. 2013, pp. 151–161.

[47] Francesco Folino and Clara Pizzuti. “An evolutionary multiobjective approach for community discovery in dynamic networks”. In: *IEEE Transactions on Knowledge and Data Engineering* 26.8 (2013), pp. 1838–1852.

[48] Yixuan Li et al. “Uncovering the small community structure in large networks: A local spectral approach”. In: *Proceedings of the 24th international conference on world wide web*. 2015, pp. 658–668.

[49] Lucas GS Jeub et al. “Think locally, act locally: Detection of small, medium-sized, and large communities in large networks”. In: *Physical Review E* 91.1 (2015), p. 12821.

[50] Pasquale De Meo et al. “Mixing local and global information for community detection in large networks”. In: *Journal of Computer and System Sciences* 80.1 (2014), pp. 72–87.

[51] Kathy Macropol and Ambuj Singh. “Scalable discovery of best clusters on large graphs”. In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), pp. 693–702.

[52] George Kollios, Michalis Potamias, and Evimaria Terzi. “Clustering large probabilistic graphs”. In: *IEEE Transactions on Knowledge and Data Engineering* 25.2 (2013), pp. 325–336.

[53] Liaoruo Wang et al. “Detecting community kernels in large social networks”. In: *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE. 2011, pp. 784–793.

[54] David Hallac, Jure Leskovec, and Stephen Boyd. “Network lasso: Clustering and optimization in large graphs”. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2015, pp. 387–396.

[55] Vandana Bhatia and Rinkle Rani. “Dfuzzy: a deep learning-based fuzzy clustering model for large graphs”. In: *Knowledge and Information Systems* 57.1 (2018), pp. 159–181.

[56] Tiago P Peixoto. “Model selection and hypothesis testing for large-scale network models with overlapping groups”. In: *Physical Review X* 5.1 (2015), p. 11033.

[57] Charalampos E Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. “Scalable motif-aware graph clustering”. In: *Proceedings of the 26th International Conference on World Wide Web*. 2017, pp. 1451–1460.

[58] Yakun Li et al. “Efficient community detection with additive constrains on large networks”. In: *Knowledge-Based Systems* 52 (2013), pp. 268–278.

[59] Arnau Prat-Pérez, David Dominguez-Sal, and Josep-Lluis Larriba-Pey. “High quality, scalable and parallel community detection for large real graphs”. In: *Proceedings of the 23rd international conference on World wide web*. 2014, pp. 225–236.

[60] Joyce Jiyoung Whang, Xin Sui, and Inderjit S Dhillon. “Scalable and memory-efficient clustering of large-scale social networks”. In: *2012 IEEE 12th International Conference on Data Mining*. IEEE. 2012, pp. 705–714.

[61] Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan. “Local graph sparsification for scalable clustering”. In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 2011, pp. 721–732.

[62] Daniel A Spielman and Shang-Hua Teng. “A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning”. In: *SIAM Journal on computing* 42.1 (2013), pp. 1–26.

[63] Jialu Liu et al. “Large-scale spectral clustering on graphs”. In: *Twenty-Third International Joint Conference on Artificial Intelligence*. 2013.

[64] Xiaotong Zhang et al. “Attributed graph clustering via adaptive graph convolution”. In: *arXiv preprint arXiv:1906.01210* (2019).

[65] Di Jin et al. “Graph convolutional networks meet Markov random fields: Semi-supervised community detection in attribute networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 152–159.

[66] Jaewon Yang, Julian McAuley, and Jure Leskovec. “Community detection in networks with node attributes”. In: *Data Mining (ICDM), 2013 IEEE 13th international conference on*. IEEE. 2013, pp. 1151–1156.

[67] Dongxiao He et al. “Joint identification of network communities and semantics via integrative modeling of network topologies and node contents”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[68] Xiao Wang et al. “Semantic community identification in large attribute networks”. In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[69] Guo-Jun Qi, Charu C Aggarwal, and Thomas Huang. “Community detection with edge content in social media networks”. In: *2012 IEEE 28th International Conference on Data Engineering*. IEEE. 2012, pp. 534–545.

[70] Xin Huang, Hong Cheng, and Jeffrey Xu Yu. “Dense community detection in multi-valued attributed networks”. In: *Information Sciences* 314 (2015), pp. 77–99.

[71] Yu Han and Jie Tang. “Probabilistic community and role model for social networks”. In: *Proceedings of the 21th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. 2015, pp. 407–416.

[72] Zhiqiang Xu et al. “A model-based approach to attributed graph clustering”. In: *Proceedings of the 2012 ACM SIGMOD international conference on management of data*. 2012, pp. 505–516.

[73] Simon Pool, Francesco Bonchi, and Matthijs van Leeuwen. “Description-driven community detection”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 5.2 (2014), pp. 1–28.

[74] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. “Clustering large attributed graphs: An efficient incremental approach”. In: *2010 IEEE International Conference on Data Mining*. IEEE. 2010, pp. 689–698.

[75] Michele Coscia et al. “Demon: a local-first discovery method for overlapping communities”. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012, pp. 615–623.

[76] Jaewon Yang and Jure Leskovec. “Overlapping community detection at scale: a nonnegative matrix factorization approach”. In: *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM. 2013, pp. 587–596.

[77] Hongyi Zhang, Irwin King, and Michael R Lyu. “Incorporating Implicit Link Preference Into Overlapping Community Detection.” In: *AAAI*. 2015, pp. 396–402.

[78] Hongyi Zhang et al. “Modeling the Homophily Effect between Links and Communities for Overlapping Community Detection.” In: *IJCAI*. 2016, pp. 3938–3944.

[79] Xiaofeng Wang, Gongshen Liu, and Jianhua Li. “Overlapping Community Detection Based on Structural Centrality in Complex Networks”. In: *IEEE Access* 5 (2017), pp. 25258–25269.

[80] Di Jin, Bogdan Gabrys, and Jianwu Dang. “Combined node and link partitions method for finding overlapping communities in complex networks”. In: *Scientific reports* 5 (2015), p. 8600.

[81] Joyce Jiyoung Whang, David F Gleich, and Inderjit S Dhillon. “Overlapping community detection using seed set expansion”. In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2013, pp. 2099–2108.

[82] Jaewon Yang and Jure Leskovec. “Community-affiliation graph model for overlapping network community detection”. In: *2012 IEEE 12th international conference on data mining*. IEEE. 2012, pp. 1170–1175.

[83] Prem K Gopalan and David M Blei. “Efficient discovery of overlapping communities in massive networks”. In: *Proceedings of the National Academy of Sciences* 110.36 (2013), pp. 14534–14539.

[84] Di Jin et al. “Detect overlapping communities via ranking node popularities”. In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[85] Joyce Jiyoung Whang, David F Gleich, and Inderjit S Dhillon. “Overlapping community detection using neighborhood-inflated seed expansion”. In: *IEEE Transactions on Knowledge and Data Engineering* 28.5 (2016), pp. 1272–1284.

[86] Justine Eustace, Xingyuan Wang, and Yaozu Cui. “Overlapping community detection using neighborhood ratio matrix”. In: *Physica A: Statistical Mechanics and its Applications* 421 (2015), pp. 510–521.

[87] Kun He et al. “Detecting overlapping communities from local spectral subspaces”. In: *2015 IEEE International Conference on Data Mining*. IEEE. 2015, pp. 769–774.

[88] Jin-Xuan Yang and Xiao-Dong Zhang. “Finding overlapping communities using seed set”. In: *Physica A: Statistical Mechanics and its Applications* 467 (2017), pp. 96–106.

[89] Mark EJ Newman and Gesine Reinert. “Estimating the number of communities in a network”. In: *Physical review letters* 117.7 (2016), p. 78301.

[90] Kehui Chen and Jing Lei. “Network cross-validation for determining the number of communities in network data”. In: *Journal of the American Statistical Association* 113.521 (2018), pp. 241–251.

[91] D Franco Saldana, Yi Yu, and Yang Feng. “How many communities are there?” In: *Journal of Computational and Graphical Statistics* 26.1 (2017), pp. 171–181.

[92] Can M Le and Elizaveta Levina. “Estimating the number of communities in networks by spectral methods”. In: *arXiv preprint arXiv:1507.00827* (2015).

[93] Jie Chen and Yousef Saad. “Dense subgraph extraction with application to community detection”. In: *IEEE Transactions on Knowledge and Data Engineering* 24.7 (2012), pp. 1216–1230.

[94] Andrea Lancichinetti et al. “Finding statistically significant communities in networks”. In: *PloS one* 6.4 (2011), e18961.

[95] Zibin Zheng et al. “Finding weighted k-truss communities in large networks”. In: *Information Sciences* 417 (2017), pp. 344–360.

[96] Wanyun Cui et al. “Local search of communities in large graphs”. In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM. 2014, pp. 991–1002.

[97] Mauro Sozio and Aristides Gionis. “The community-search problem and how to plan a successful cocktail party”. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 939–948.

[98] Nicola Barbieri et al. “Efficient and effective community search”. In: *Data mining and knowledge discovery* 29.5 (2015), pp. 1406–1433.

[99] Xin Huang et al. “Approximate closest community search in networks”. In: *arXiv preprint arXiv:1505.05956* (2015).

[100] Lu Qin et al. “Locally densest subgraph discovery”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, pp. 965–974.

[101] Rong-Hua Li et al. “Influential community search in large networks”. In: *Proceedings of the VLDB Endowment* 8.5 (2015), pp. 509–520.

[102] Yubao Wu et al. “Robust local community detection: on free rider effect and its elimination”. In: *Proceedings of the VLDB Endowment* 8.7 (2015), pp. 798–809.

[103] Haoran Wen, EA Leicht, and Raissa M D'Souza. "Improving community detection in networks by targeted node removal". In: *Physical Review E* 83.1 (2011), p. 16114.

[104] Dongxiao He et al. "A model framework for the enhancement of community detection in complex networks". In: *Physica A: Statistical Mechanics and its Applications* 461 (2016), pp. 602–612.

[105] Alireza Khadivi, Ali Ajdari Rad, and Martin Hasler. "Network community-detection enhancement by proper weighting". In: *Physical Review E* 83.4 (2011), p. 46104.

[106] Pasquale De Meo et al. "Enhancing community detection using a network weighting strategy". In: *Information Sciences* 222 (2013), pp. 648–668.

[107] Darong Lai, Hongtao Lu, and Christine Nardini. "Enhanced modularity-based community detection by random walk network preprocessing". In: *Physical Review E* 81.6 (2010), p. 66118.

[108] Jiajun Zhou et al. "Adversarial Enhancement for Community Detection in Complex Networks". In: *arXiv preprint arXiv:1911.01670* (2019).

[109] Pei-Zhen Li et al. "EdMot: An Edge Enhancement Approach for Motif-aware Community Detection". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 479–487.

[110] Shreyansh Bhatt et al. "Knowledge graph enhanced community detection and characterization". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 2019, pp. 51–59.

[111] Zhong-Yuan Zhang, Kai-Di Sun, and Si-Qi Wang. "Enhanced community structure detection in complex networks with partial background information". In: *Scientific reports* 3 (2013), p. 3241.

[112] Liang Yang et al. “A unified semi-supervised community detection framework using latent space graph regularization”. In: *IEEE transactions on cybernetics* 45.11 (2015), pp. 2585–2598.

[113] Wenjun Wang et al. “A Unified Weakly Supervised Framework for Community Detection and Semantic Matching”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2018, pp. 218–230.

[114] Pan Zhang, Cristopher Moore, and Lenka Zdeborová. “Phase transitions in semisupervised clustering of sparse networks”. In: *Physical Review E* 90.5 (2014), p. 52802.

[115] Jianjun Cheng et al. “Active semi-supervised community detection based on must-link and cannot-link constraints”. In: *PloS one* 9.10 (2014).

[116] Eric Eaton and Rachael Mansbach. “A spin-glass model for semi-supervised community detection”. In: *Twenty-Sixth AAAI Conference on Artificial Intelligence*. 2012.

[117] Xiao Liu et al. “Semi-supervised community detection based on non-negative matrix factorization with node popularity”. In: *Information Sciences* 381 (2017), pp. 304–321.

[118] Xiaoke Ma et al. “Semi-supervised clustering algorithm for community structure detection in complex networks”. In: *Physica A: Statistical Mechanics and its Applications* 389.1 (2010), pp. 187–197.

[119] Lei Li et al. “Extremal optimization-based semi-supervised algorithm with conflict pairwise constraints for community detection”. In: *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*. IEEE. 2014, pp. 180–187.

[120] Liang Yang et al. “Active link selection for efficient semi-supervised community detection”. In: *Scientific reports* 5 (2015), p. 9039.

[121] Mark EJ Newman. “Fast algorithm for detecting community structure in networks”. In: *Physical review E* 69.6 (2004), p. 66133.

[122] Mark EJ Newman. “Modularity and community structure in networks”. In: *Proceedings of the national academy of sciences* 103.23 (2006), pp. 8577–8582.

[123] Vincenzo Nicosia et al. “Extending the definition of modularity to directed graphs with overlapping communities”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2009.3 (2009), P03024.

[124] Liang Yang et al. “Modularity Based Community Detection with Deep Learning.” In: *IJCAI*. Vol. 16. 2016, pp. 2252–2258.

[125] Mark EJ Newman. “Equivalence between modularity optimization and maximum likelihood methods for community detection”. In: *Physical Review E* 94.5 (2016), p. 52315.

[126] Peng Gang Sun, Lin Gao, and Yang Yang. “Maximizing modularity intensity for community partition and evolution”. In: *Information Sciences* 236 (2013), pp. 83–92.

[127] Sonia Cafieri, Pierre Hansen, and Leo Liberti. “Locally optimal heuristic for modularity maximization of networks”. In: *Physical Review E* 83.5 (2011), p. 56105.

[128] Jonathan Q Jiang and Lisa J McQuay. “Modularity functions maximization with nonnegative relaxation facilitates community detection in networks”. In: *Physica A: Statistical Mechanics and its Applications* 391.3 (2012), pp. 854–865.

[129] Ju Xiang et al. “Local modularity for community detection in complex networks”. In: *Physica A: Statistical Mechanics and its Applications* 443 (2016), pp. 451–459.

[130] Shuqin Zhang and Hongyu Zhao. “Normalized modularity optimization method for community identification with degree adjustment”. In: *Physical Review E* 88.5 (2013), p. 52802.

[131] James P Bagrow. “Communities and bottlenecks: Trees and treelike networks have high modularity”. In: *Physical Review E* 85.6 (2012), p. 66118.

[132] Zhan Bu et al. “A fast parallel modularity optimization algorithm (FPMQA) for community detection in online social network”. In: *Knowledge-Based Systems* 50 (2013), pp. 246–259.

[133] Ling Chen, Qiang Yu, and Bolun Chen. “Anti-modularity and anti-community detecting in complex networks”. In: *Information Sciences* 275 (2014), pp. 293–313.

[134] Mingming Chen, Konstantin Kuzmin, and Boleslaw K Szymanski. “Community detection via maximization of modularity and its variants”. In: *IEEE Transactions on Computational Social Systems* 1.1 (2014), pp. 46–65.

[135] Mark EJ Newman. “Spectral methods for community detection and graph partitioning”. In: *Physical Review E* 88.4 (2013), p. 42822.

[136] Joan Bruna et al. “Spectral networks and locally connected networks on graphs”. In: *arXiv preprint arXiv:1312.6203* (2013).

[137] Alaa Saade, Florent Krzakala, and Lenka Zdeborová. “Spectral clustering of graphs with the bethe hessian”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 406–414.

[138] Austin R Benson, David F Gleich, and Jure Leskovec. “Tensor spectral clustering for partitioning higher-order network structures”. In: *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM. 2015, pp. 118–126.

[139] Raj Rao Nadakuditi and Mark EJ Newman. “Graph spectra and the detectability of community structure in networks”. In: *Physical review letters* 108.18 (2012), p. 188701.

[140] Karl Rohe, Sourav Chatterjee, Bin Yu, et al. “Spectral clustering and the high-dimensional stochastic blockmodel”. In: *The Annals of Statistics* 39.4 (2011), pp. 1878–1915.

[141] Jing Lei, Alessandro Rinaldo, et al. "Consistency of spectral clustering in stochastic block models". In: *The Annals of Statistics* 43.1 (2015), pp. 215–237.

[142] Kamalika Chaudhuri, Fan Chung, and Alexander Tsiatas. "Spectral clustering of graphs with general degrees in the extended planted partition model". In: *Conference on Learning Theory*. 2012, pp. 35–1.

[143] Antony Joseph, Bin Yu, et al. "Impact of regularization on spectral clustering". In: *The Annals of Statistics* 44.4 (2016), pp. 1765–1791.

[144] Jiashun Jin et al. "Fast community detection by SCORE". In: *The Annals of Statistics* 43.1 (2015), pp. 57–89.

[145] Michael W Mahoney, Lorenzo Orecchia, and Nisheeth K Vishnoi. "A local spectral method for graphs: With applications to improving graph partitions and exploring data graphs locally". In: *Journal of Machine Learning Research* 13.Aug (2012), pp. 2339–2365.

[146] Xiao Zhang and Mark EJ Newman. "Multiway spectral community detection in networks". In: *Physical Review E* 92.5 (2015), p. 52808.

[147] Brian Karrer and Mark EJ Newman. "Stochastic blockmodels and community structure in networks". In: *Physical review E* 83.1 (2011), p. 16107.

[148] Anderson Y Zhang, Harrison H Zhou, et al. "Minimax rates of community detection in stochastic block models". In: *The Annals of Statistics* 44.5 (2016), pp. 2252–2280.

[149] YX Rachel Wang, Peter J Bickel, et al. "Likelihood-based model selection for stochastic block models". In: *The Annals of Statistics* 45.2 (2017), pp. 500–528.

[150] Jing Lei et al. "A goodness-of-fit test for stochastic block models". In: *The Annals of Statistics* 44.1 (2016), pp. 401–424.

[151] Chao Gao et al. “Community detection in degree-corrected block models”. In: *The Annals of Statistics* 46.5 (2018), pp. 2153–2185.

[152] Elchanan Mossel and Jiaming Xu. “Density evolution in the degree-correlated stochastic block model”. In: *Conference on Learning Theory*. 2016, pp. 1319–1356.

[153] Yunpeng Zhao, Elizaveta Levina, Ji Zhu, et al. “Consistency of community detection in networks under degree-corrected stochastic block models”. In: *The Annals of Statistics* 40.4 (2012), pp. 2266–2292.

[154] Alain Celisse, Jean-Jacques Daudin, Laurent Pierre, et al. “Consistency of maximum-likelihood and variational estimators in the stochastic block model”. In: *Electronic Journal of Statistics* 6 (2012), pp. 1847–1899.

[155] Xiaoran Yan et al. “Model selection for degree-corrected block models”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2014.5 (2014), P05007.

[156] Emmanuel Abbe and Colin Sandon. “Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery”. In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE. 2015, pp. 670–688.

[157] Simon Heimlicher, Marc Lelarge, and Laurent Massoulié. “Community detection in the labelled stochastic block model”. In: *arXiv preprint arXiv:1209.2910* (2012).

[158] Se-Young Yun and Alexandre Proutiere. “Accurate community detection in the stochastic block model via spectral algorithms”. In: *arXiv preprint arXiv:1412.7335* (2014).

[159] Se-Young Yun and Alexandre Proutiere. “Optimal cluster recovery in the labeled stochastic block model”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 965–973.

[160] Tai Qin and Karl Rohe. “Regularized spectral clustering under the degree-corrected stochastic blockmodel”. In: *Advances in Neural Information Processing Systems*. 2013, pp. 3120–3128.

[161] Tiago P Peixoto. “Efficient Monte Carlo and greedy heuristic for the inference of stochastic block models”. In: *Physical Review E* 89.1 (2014), p. 12804.

[162] Tiago P Peixoto. “Nonparametric Bayesian inference of the microcanonical stochastic block model”. In: *Physical Review E* 95.1 (2017), p. 12317.

[163] Purnamrita Sarkar, Peter J Bickel, et al. “Role of normalization in spectral clustering for stochastic blockmodels”. In: *The Annals of Statistics* 43.3 (2015), pp. 962–990.

[164] Vince Lyzinski et al. “Perfect clustering for stochastic blockmodel graphs via adjacency spectral embedding”. In: *Electronic journal of statistics* 8.2 (2014), pp. 2905–2922.

[165] Jiaming Xu, Laurent Massoulié, and Marc Lelarge. “Edge label inference in generalized stochastic block models: from spectral theory to impossibility results”. In: *Conference on Learning Theory*. 2014, pp. 903–920.

[166] Elchanan Mossel, Joe Neeman, and Allan Sly. “Belief propagation, robust reconstruction and optimal recovery of block models”. In: *Conference on Learning Theory*. 2014, pp. 356–370.

[167] Tiago P Peixoto. “Entropy of stochastic blockmodel ensembles”. In: *Physical Review E* 85.5 (2012), p. 56122.

[168] Wei-Lin Chiang et al. “Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 257–266.

[169] Chun Wang et al. “Mgae: Marginalized graph autoencoder for graph clustering”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 889–898.

[170] Fei Tian et al. “Learning deep representations for graph clustering”. In: *Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.

[171] Joan Bruna and X Li. “Community detection with graph neural networks”. In: *Stat* 1050 (2017), p. 27.

[172] Xiao Wang et al. “Community Preserving Network Embedding.” In: *AAAI*. 2017, pp. 203–209.

[173] Sandro Cavallari et al. “Learning community embedding with community detection and node embedding on graphs”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM. 2017, pp. 377–386.

[174] Ming Shao et al. “Deep Linear Coding for Fast Graph Clustering.” In: *IJCAI*. 2015, pp. 3798–3804.

[175] Vincent W Zheng et al. “From node embedding to community embedding”. In: *arXiv preprint arXiv:1610.09950* (2016).

[176] Benedek Rozemberczki et al. “Gemsec: Graph embedding with self clustering”. In: *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 2019, pp. 65–72.

[177] Bing-Jie Sun et al. “A Non-negative Symmetric Encoder-Decoder Approach for Community Detection”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM. 2017, pp. 597–606.

[178] Peihao Huang et al. “Deep embedding network for clustering”. In: *2014 22nd International conference on pattern recognition*. IEEE. 2014, pp. 1532–1537.

[179] Da Kuang, Chris Ding, and Haesun Park. "Symmetric nonnegative matrix factorization for graph clustering". In: *Proceedings of the 2012 SIAM international conference on data mining*. SIAM. 2012, pp. 106–117.

[180] Da Kuang, Sangwoon Yun, and Haesun Park. "SymNMF: nonnegative low-rank approximation of a similarity matrix for graph clustering". In: *Journal of Global Optimization* 62.3 (2015), pp. 545–574.

[181] Fanghua Ye, Chuan Chen, and Zibin Zheng. "Deep autoencoder-like nonnegative matrix factorization for community detection". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2018, pp. 1393–1402.

[182] Yu Zhang and Dit-Yan Yeung. "Overlapping community detection via bounded nonnegative matrix tri-factorization". In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2012, pp. 606–614.

[183] Zhong-Yuan Zhang, Yong Wang, and Yong-Yeol Ahn. "Overlapping community detection in complex networks using symmetric binary matrix factorization". In: *Physical Review E* 87.6 (2013), p. 62803.

[184] Wenhui Wu et al. "Nonnegative matrix factorization with mixed hypergraph regularization for community detection". In: *Information Sciences* 435 (2018), pp. 263–281.

[185] Ioannis Psorakis et al. "Overlapping community detection using bayesian nonnegative matrix factorization". In: *Physical Review E* 83.6 (2011), p. 66114.

[186] Yulong Pei1 Nilanjan Chakraborty and Katia Sycara. "Nonnegative matrix tri-factorization with graph regularization for community detection in social networks". In: *AAAI*. AAAI Press. 2015, pp. 2083–2089.

[187] Shudong Huang et al. “Robust graph regularized nonnegative matrix factorization for clustering”. In: *Data Mining and Knowledge Discovery* 32.2 (2018), pp. 483–503.

[188] Yulong Pei, Nilanjan Chakraborty, and Katia Sycara. “Nonnegative matrix tri-factorization with graph regularization for community detection in social networks”. In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.

[189] Fei Wang et al. “Community discovery using nonnegative matrix factorization”. In: *Data Mining and Knowledge Discovery* 22.3 (2011), pp. 493–521.

[190] Zhirong Yang et al. “Clustering by nonnegative matrix factorization using graph random walk”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 1079–1087.

[191] Fanhua Shang, LC Jiao, and Fei Wang. “Graph dual regularization non-negative matrix factorization for co-clustering”. In: *Pattern Recognition* 45.6 (2012), pp. 2237–2250.

[192] Xiaohua Shi et al. “Community detection in social network with pairwisely constrained symmetric non-negative matrix factorization”. In: *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. 2015, pp. 541–546.

[193] Xianchao Tang et al. “Uncovering community structures with initialized bayesian nonnegative matrix factorization”. In: *PloS one* 9.9 (2014).

[194] Martin Rosvall and Carl T Bergstrom. “Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems”. In: *PloS one* 6.4 (2011), e18209.

[195] Martin Rosvall et al. “Memory in network flows and its effects on spreading dynamics and community detection”. In: *Nature communications* 5.1 (2014), pp. 1–13.

[196] Christian Persson et al. “Maps of sparse Markov chains efficiently reveal community structure in network flows with memory”. In: *arXiv preprint arXiv:1606.08328* (2016).

[197] Martin Rosvall and Carl T Bergstrom. “Maps of random walks on complex networks reveal community structure”. In: *Proceedings of the National Academy of Sciences* 105.4 (2008), pp. 1118–1123.

[198] Di Jin et al. “A Markov random walk under constraint for discovering overlapping communities in complex networks”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2011.5 (2011), P05031.

[199] Kyle Kloster and David F Gleich. “Heat kernel based community detection”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 1386–1395.

[200] Vinko Zlatić, Andrea Gabrielli, and Guido Caldarelli. “Topologically biased random walk and community finding in networks”. In: *Physical Review E* 82.6 (2010), p. 66109.

[201] Dongxiao He et al. “A network-specific Markov random field approach to community detection”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[202] Vsevolod Salnikov, Michael T Schaub, and Renaud Lambiotte. “Using higher-order Markov models to reveal flow-based communities in networks”. In: *Scientific reports* 6 (2016), p. 23194.

[203] Lorenzo Orecchia and Zeyuan Allen Zhu. “Flow-based algorithms for local graph clustering”. In: *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2014, pp. 1267–1286.

[204] Jian Liu and Tingzhan Liu. “Detecting community structure in complex networks using simulated annealing with k-means algorithms”. In: *Physica A: Statistical Mechanics and its Applications* 389.11 (2010), pp. 2300–2309.

[205] Hui-Jia Li et al. “Potts model based on a Markov process computation solves the community structure problem effectively”. In: *Physical Review E* 86.1 (2012), p. 16109.

[206] Renaud Lambiotte and Martin Rosvall. “Ranking and clustering of nodes in networks with smart teleportation”. In: *Physical Review E* 85.5 (2012), p. 56107.

[207] Yang Yang et al. “Closed walks for community detection”. In: *Physica A: Statistical Mechanics and its Applications* 397 (2014), pp. 129–143.

[208] Wenjun Wang et al. “Fuzzy overlapping community detection based on local random walk and multidimensional scaling”. In: *Physica A: Statistical Mechanics and its Applications* 392.24 (2013), pp. 6578–6586.

[209] Symeon Papadopoulos et al. “Community detection in social media”. In: *Data Mining and Knowledge Discovery* 24.3 (2012), pp. 515–554.

[210] Pierre Latouche, Etienne Birmelé, Christophe Ambroise, et al. “Overlapping stochastic block models with application to the french political blogosphere”. In: *The Annals of Applied Statistics* 5.1 (2011), pp. 309–336.

[211] Haoran Xie et al. “Community-aware user profile enrichment in folksonomy”. In: *Neural Networks* 58 (2014), pp. 111–121.

[212] Srijan Kumar et al. “An army of me: Sockpuppets in online discussion communities”. In: *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2017, pp. 857–866.

[213] Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. “How community feedback shapes user behavior”. In: *arXiv preprint arXiv:1405.1429* (2014).

[214] Cristian Danescu-Niculescu-Mizil et al. “No country for old members: User life-cycle and linguistic change in online communities”. In: *Proceedings of the 22nd international conference on World Wide Web*. ACM. 2013, pp. 307–318.

[215] Jure Leskovec and Julian J Mcauley. “Learning to discover social circles in ego networks”. In: *Advances in neural information processing systems*. 2012, pp. 539–547.

[216] Sanjay Ram Kairam, Dan J Wang, and Jure Leskovec. “The life and death of online groups: Predicting group growth and longevity”. In: *Proceedings of the fifth ACM international conference on Web search and data mining*. 2012, pp. 673–682.

[217] Ullas Gargi et al. “Large-scale community detection on youtube for topic discovery and exploration”. In: *Fifth International AAAI Conference on Weblogs and Social Media*. 2011.

[218] Mrinmaya Sachan et al. “Using content and interactions for discovering communities in social networks”. In: *Proceedings of the 21st international conference on World Wide Web*. 2012, pp. 331–340.

[219] Yu Wang et al. “Community-based greedy algorithm for mining top-k influential nodes in mobile social networks”. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 1039–1048.

[220] Amanda L Traud et al. “Comparing community structure to characteristics in online collegiate social networks”. In: *SIAM review* 53.3 (2011), pp. 526–543.

[221] Federico Botta and Charo I del Genio. “Analysis of the communities of an urban mobile phone network”. In: *PloS one* 12.3 (2017), e0174198.

[222] Mert Ozer, Nyunsu Kim, and Hasan Davulcu. “Community detection in political Twitter networks using Nonnegative Matrix Factorization methods”. In: *2016*

IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE. 2016, pp. 81–88.

[223] Nagarajan Natarajan, Prithviraj Sen, and Vineet Chaoji. “Community detection in content-sharing social networks”. In: *Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining*. 2013, pp. 82–89.

[224] Zhongying Zhao et al. “Topic oriented community detection through social objects and link analysis in social networks”. In: *Knowledge-Based Systems* 26 (2012), pp. 164–173.

[225] Javier O Garcia et al. “Applications of community detection techniques to brain graphs: Algorithmic considerations and implications for neural function”. In: *Proceedings of the IEEE* 106.5 (2018), pp. 846–867.

[226] Ying Liu, Jason Moser, and Selin Aviyente. “Network community structure detection for directional neural networks inferred from multichannel multisubject EEG data”. In: *IEEE Transactions on Biomedical Engineering* 61.7 (2014), pp. 1919–1930.

[227] Tamás Nepusz, Haiyuan Yu, and Alberto Paccanaro. “Detecting overlapping protein complexes in protein-protein interaction networks”. In: *Nature methods* 9.5 (2012), p. 471.

[228] Anna CF Lewis et al. “The function of communities in protein interaction networks at multiple scales”. In: *BMC systems biology* 4.1 (2010), p. 100.

[229] Danielle S Bassett et al. “Extraction of force-chain network architecture in granular materials using community detection”. In: *Soft Matter* 11.14 (2015), pp. 2731–2744.

[230] Manish Gupta et al. “Evolutionary clustering and analysis of bibliographic networks”. In: *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*. IEEE. 2011, pp. 63–70.

[231] Tanmoy Chakraborty and Abhijnan Chakraborty. “OverCite: Finding overlapping communities in citation network”. In: *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*. IEEE. 2013, pp. 1124–1131.

[232] Andrés E Coca and Liang Zhao. “Musical rhythmic pattern extraction using relevance of communities in networks”. In: *Information Sciences* 329 (2016), pp. 819–848.

[233] Hanyin Fang et al. “Community-based question answering via heterogeneous social network learning”. In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[234] Ruiqi Hu et al. “Co-clustering enterprise social networks”. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2016, pp. 107–114.

[235] Jure Leskovec and Andrej Krevl. “{SNAP Datasets}:{Stanford} Large Network Dataset Collection”. In: (June 2015).

[236] Michael E Smoot et al. “Cytoscape 2.8: new features for data integration and network visualization”. In: *Bioinformatics* 27.3 (2010), pp. 431–432.

[237] Manlio De Domenico, Mason A Porter, and Alex Arenas. “MuxViz: a tool for multilayer analysis and visualization of networks”. In: *Journal of Complex Networks* 3.2 (2015), pp. 159–176.

[238] Aric Hagberg et al. “Networkx. High productivity software for complex networks”. In: *Webová strá nka https://networkx. lanl. gov/wiki* (2013).

[239] Gabor Csardi, Tamas Nepusz, et al. “The igraph software package for complex network research”. In: *InterJournal, Complex Systems* 1695.5 (2006), pp. 1–9.

[240] Wouter De Nooy, Andrej Mrvar, and Vladimir Batagelj. *Exploratory social network analysis with Pajek*. Cambridge University Press, 2018.

[241] Ghi-Hoon Ghim, Namun Cho, and Jeongsu Seo. “NetMiner”. In: *Encyclopedia of Social Network Analysis and Mining*. Springer, 2014, pp. 1025–1037.

[242] Balázs Adamcsek et al. “CFinder: locating cliques and overlapping modules in biological networks”. In: *Bioinformatics* 22.8 (2006), pp. 1021–1023.

[243] Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy, et al. “Gephi: an open source software for exploring and manipulating networks.” In: *Icwsm* 8.2009 (2009), pp. 361–362.

[244] Stephen P Borgatti, Martin G Everett, and Linton C Freeman. “Ucinet”. In: *Encyclopedia of social network analysis and mining*. Springer, 2014, pp. 2261–2267.

[245] Kathleen M Carley. “ORA: A toolkit for dynamic network analysis and visualization”. In: *Encyclopedia of social network analysis and mining* (2014), pp. 1219–1228.

[246] Jaewon Yang and Jure Leskovec. “Defining and evaluating network communities based on ground-truth”. In: *Knowledge and Information Systems* 42.1 (2015), pp. 181–213.

[247] Günce Keziban Orman, Vincent Labatut, and Hocine Cherifi. “Comparative evaluation of community detection algorithms: a topological approach”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2012.8 (2012), P08001.

[248] Darko Hric, Richard K Darst, and Santo Fortunato. “Community detection in networks: Structural communities versus ground truth”. In: *Physical Review E* 90.6 (2014), p. 62805.

[249] Zhao Yang, René Algesheimer, and Claudio J Tessone. “A comparative analysis of community detection algorithms on artificial networks”. In: *Scientific reports* 6 (2016), p. 30750.

[250] Bruno Abrahao et al. “On the separability of structural classes of communities”. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012, pp. 624–632.

[251] Meng Wang et al. “Community detection in social networks: an in-depth benchmarking study with a procedure-oriented framework”. In: *Proceedings of the VLDB Endowment* 8.10 (2015), pp. 998–1009.

[252] Gabriel Murray, Giuseppe Carenini, and Raymond Ng. “Using the omega index for evaluating abstractive community detection”. In: *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*. Association for Computational Linguistics. 2012, pp. 10–18.

[253] Tanmoy Chakraborty et al. “On the permanence of vertices in network communities”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 1396–1405.

[254] Hui-Jia Li, Hao Wang, and Luonan Chen. “Measuring robustness of community structure in complex networks”. In: *EPL (Europhysics Letters)* 108.6 (2015), p. 68009.

[255] Hua-Wei Shen and Xue-Qi Cheng. “Spectral methods for the detection of network community structure: a comparative analysis”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2010.10 (2010), P10020.

[256] Rodrigo Aldecoa and Ignacio Marín. “Surprise maximization reveals the community structure of complex networks”. In: *Scientific reports* 3.1 (2013), pp. 1–9.

[257] Yanqing Hu et al. “Measuring the significance of community structure in complex networks”. In: *Physical Review E* 82.6 (2010), p. 66106.

[258] Rodrigo Aldecoa and Ignacio Marín. “Closed benchmarks for network community structure characterization”. In: *Physical Review E* 85.2 (2012), p. 26109.

[259] Pan Zhang. “Evaluating accuracy of community detection using the relative normalized mutual information”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2015.11 (2015), P11006.

[260] Alexander G Nikolaev, Raihan Razib, and Ashwin Kucheriya. “On efficient use of entropy centrality for social network analysis and community detection”. In: *Social Networks* 40 (2015), pp. 154–162.

[261] Elizabeth A Leicht and Mark EJ Newman. “Community structure in directed networks”. In: *Physical review letters* 100.11 (2008), p. 118703.

[262] Alex Arenas et al. “Size reduction of complex networks preserving modularity”. In: *New Journal of Physics* 9.6 (2007), p. 176.

[263] Leon Danon et al. “Comparing community structure identification”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2005.9 (2005), P09008.

[264] Aaron F McDaid, Derek Greene, and Neil Hurley. “Normalized mutual information to evaluate overlapping community finding algorithms”. In: *arXiv preprint arXiv:1110.2515* (2011).

[265] Steve Gregory. “Fuzzy overlapping communities in networks”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2011.2 (2011), P02017.

[266] Frank Havemann et al. “Identification of overlapping communities and their hierarchy by locally calculating community-changing resolution levels”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2011.1 (2011), P01023.

[267] Aditya Grover and Jure Leskovec. “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 855–864.

[268] Andrea Lancichinetti and Santo Fortunato. “Limits of modularity maximization in community detection”. In: *Physical review E* 84.6 (2011), p. 66122.

[269] Tatsuro Kawamoto and Yoshiyuki Kabashima. "Limitations in the spectral method for graph partitioning: Detectability threshold and localization of eigenvectors". In: *Physical Review E* 91.6 (2015), p. 62803.

[270] Ju Xiang et al. "Multi-resolution modularity methods and their limitations in community detection". In: *The European Physical Journal B* 85.10 (2012), p. 352.

[271] Rodrigo Aldecoa and Ignacio Marín. "Exploring the limits of community detection strategies in complex networks". In: *Scientific reports* 3.1 (2013), pp. 1–11.

[272] Ju Xiang and Ke Hu. "Limitation of multi-resolution methods in community detection". In: *Physica A: Statistical Mechanics and its Applications* 391.20 (2012), pp. 4995–5003.

VITA

Vita may be provided by doctoral students only. The length of the vita is preferably one page. It may include the place of birth and should be written in third person. This vita is similar to the author biography found on book jackets.