ABSTRACT

Title of dissertation: ANALYZING SEMI-LOCAL LINK COHESION TO DETECT COMMUNITIES AND ANOMALIES IN COMPLEX NETWORKS

Catherine Schwartz, Doctor of Philosophy, 2021

Dissertation directed by: Professor Wojciech Czaja
Department of Mathematics

Link cohesion is a new type of metric used to assess how supported an edge is relative to other edges, accounting for nearby alternate paths and associated vertex degrees. A deterministic, scalable, and parallelizable link cohesion metric was shown to be useful in supporting edge scoring and simplifying highly connected networks, making key cohesive subgraphs easier to detect. In this dissertation, the link cohesion metric and a modified version of the metric are analyzed to determine their ability to improve the communities detected in different types of networks when used as a pre-weighting step to traditional algorithms like the Louvain method. Additional observations are made on the utility of analyzing the modified metric to gain insights on whether a network has community structure. The two different link cohesion metrics are also used to create vertex-level features that have the potential for being useful in detecting fake accounts in online social networks. These features are used in conjunction with a new interpretable anomaly detection method which performs well with a small amount of training data, yielding the potential for human-in-the-loop interactions that can allow users to tailor the type of anomalies to prioritize.

ANALYZING SEMI-LOCAL LINK COHESION TO DETECT COMMUNITIES AND ANOMALIES IN COMPLEX NETWORKS

by

Catherine Schwartz

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Professor Wojciech Czaja, Chair
Professor John Benedetto
Professor Radu Balan
Professor Vince Lyzinski
Professor Ramani Duraiswami

© Copyright by
Catherine Schwartz
2021

Dedication

To my sweet little Arthur.

Acknowledgments

There are many people to thank for their support in finishing my thesis while working full-time. Without their advice and encouragement, I would not have been able to complete my doctorate.

First, I thank my advisor, Wojtek, for never giving up on me, for guiding me in the right directions, for always being patient and kind, and for believing I could get it done even when I didn't. Thank you to Alex Cloninger for suggesting that Wojtek be my advisor, and to Chae Clark and Arijit Sehanobish for making classes and grad school life more enjoyable. Thank you to Dr. Duraiswami for mentoring me through my Masters thesis and to Dr. Duraiswami, Dr. Benedetto, Dr. Balan, and Dr. Lyzinski for being on my committee. I'd also like to thank Dr. O'Leary and Dr. Levy for their guidance and help. Many thanks to Alverda McCoy, Jessica Sadler, and Mark Lai for helping me through all the bureaucratic aspects of graduate school, at which I am so terrible.

I owe many thanks to the Johns Hopkins University Applied Physics Laboratory, which provided an ideal environment to complete my degree while working. Thank you to Cetin Savkli for involving me in internal research and development projects that have led to published papers on interesting topics that I could expand into my thesis, and thank you to Amanda Galante and Jonathan Cohen for being collaborators. I am grateful for my supervisors Lauren Kennell, Aaron Katz, Amanda Galante, Cetin Savkli, and David Silberberg who encouraged me to focus on completing my thesis and also to Cathy Hurley, John Piorkowski, Phil Graff, Jeff Lin, and Paul Markakis for their support. JHU/APL has incredible benefits for those looking to work on their PhD, with paid time for employees to use towards complete their thesis, flexibility to go part-time easily, and encouragement to attend conferences. Many ideas in the thesis came from insightful conversations with Carlo Vittorio Cannistraci at the Complex Networks 2019 Conference in Lisbon, which I attended courtesy of JHU/APL.

I would also like to thank my mentors, supervisors, and colleagues from my previous jobs at Sphere of Influence and Lockheed Martin. I am grateful to John Sipple for endless guidance, interesting conversations, and encouragement. I have learned so much through working and talking with you throughout the years. Thank you to Chris Kauffman, Thad Sheer, Teresa Smith, Erik Stein, Scott Pringle, Gail Raper, Mark Levedahl, Cheri Erb, Bob Kennedy, and Don Harmon for being great leaders and for supporting and encouraging me. Thanks to Joel Klein, Zach Price, Scott Love, Kathleen Crumrine, Sanda Klacar, Art Dorsey, Dennis Freeman, Debbie Pederson, Gary Roth, and Dick Walker for making work enjoyable and fun. I've learned so much from all of you.

Without the great experience in the mathematics department of my undergraduate institution, James Madison University, I would have never considered getting my doctorate. Thank you to the great professors there, including Dr. Deb Warne, Dr. Pruett, Dr. Taalman, Dr. Brown, Dr. Rosenhouse, and Dr. Lubert, for making mathematics fun and encouraging me to study it further.

To Callie, Rachael, Kelli, and Stephanie, thank you for always being there for me. I've loved our many adventures and fun times together and cherish our friendships. I am also grateful for my board game friends both in Alexandria and at JHU/APL for making life more fun.

I'm eternally grateful for my mom and dad who have always been there for me and to my in-laws who have been extremely supportive as well. Lastly, I'd like to thank my husband, Jon, for his unwavering support and love, and my son, Arthur, whose arrival made me focus on completing my degree. You are already such an amazing little man, so friendly to people you meet, and so encouraging to me, making me want to be the best person I can be.

Table of Contents

Dedication ii

Acknowledgements iii

1 Introduction 1

2 Graph and Network Science Preliminaries 7
- 2.1 Introduction to Graph Theory . . . 7
- 2.2 The Structure of Real-World Networks . . . 15
- 2.3 Community Detection . . . 29
  - 2.3.1 Clustering Perspective of Community Detection . . . 30
  - 2.3.2 Derived Edge Weights for Modularity Maximization . . . 34
  - 2.3.3 Other Perspectives of Community Detection . . . 36
  - 2.3.4 Community Evaluation . . . 38
- 2.4 Anomaly Detection . . . 40
- 2.5 Network Models . . . 43
  - 2.5.1 Random Graphs . . . 44
  - 2.5.2 Benchmark Graphs with Planted Partitions . . . 49
  - 2.5.3 Popularity×Similarity Graphs . . . 50
  - 2.5.4 Popularity×Similarity Graphs with Community Structure . . . 53

3 Link Cohesion for Networks Influenced by Popularity×Similarity Optimization 56
- 3.1 Introduction . . . 56
- 3.2 Motivation for Studying Popularity×Similarity Graphs (nPSO Graphs) . . . 59
- 3.3 Related Work . . . 66
- 3.4 Inversely Proportional Link Cohesion . . . 67
- 3.5 Maximum Density Core (MDCore) . . . 76
- 3.6 Implementation and Experimental Setup . . . 78
- 3.7 Results . . . 79
  - 3.7.1 Benchmark nPSO Graphs . . . 81
  - 3.7.2 S2 verses S3 verses Combined Weights . . . 82

3.7.3 Louvain Method with Pre-Weighted nPSO Graphs 85
3.7.4 Scalability of Inversely Proportional Link Cohesion 88
3.7.5 Experiments with Other Community Detection Algorithms 92
3.7.6 Graphs with Degree Assortative Community Structure 97
3.7.7 Communities in the Email Network from the Research Institute . . . 100
3.8 Conclusions 104

4 Link Cohesion for Networks with Degree Assortative Community Structure 106
4.1 Introduction 106
4.2 Motivation for Studying LFR Benchmark Graphs 108
4.3 Related Work 112
4.4 Local Community Link Cohesion 113
4.5 Implementation and Experimental Setup 123
4.6 Results 124
4.6.1 Benchmark LFR Graphs 125
4.6.2 Benefits to Using Only S2 and S3 Weights 126
4.6.3 Comparison of the Two Different of Link Cohesion Metrics 130
4.6.4 Louvain Method with Pre-Weighted LFR Graphs 131
4.6.5 Scalability of Local Community Link Cohesion 133
4.6.6 Experiments with Other Community Detection Algorithms 137
4.6.7 Additional Observations about Local Community Link Cohesion . . 146
4.6.8 Communities in the Co-authorship Network of Published Researchers 156
4.7 Conclusions 162

5 Sybil Detection in Online Social Networks Using Link Cohesion 165
5.1 Introduction 165
5.2 Motivation for Using Link Cohesion Metrics to Detect Sybils 168
5.3 Related Work 171
5.4 Local and Semi-Local Structural Attributes for Sybil Detection 172
5.5 Noise-Enhanced Automatic Gaussian Mixture Model (NEAGMM) 178
5.6 Random Subspace Mixture Model using NEAGMMs 182
5.7 Implementation and Experimental Setup 185
5.8 Results 186
5.8.1 Subspace Selection Study 188
5.8.2 Small Percent Training Study on Other Benchmark Datasets 189
5.8.3 Benchmark Graphs with Injected Anomalies 191
5.8.4 Anomaly Detection on Benchmark Graphs with Injected Anomalies 195
5.8.5 Interpretability of Anomaly Detection Results 198
5.8.6 Sybil Detection with Human-in-the-Loop Interaction 206
5.9 Conclusions 210

Bibliography 213

List of Tables

2.1 Community sizes inspired by research from anthropologist Robin Dunbar. . 39

3.1 Graph-level metrics for G_{EU}, G_{LFR}, and G_{nPSO} 60
3.2 Baseline pre-weighting methods used for community detection 80
3.3 Community detection methods used on weighted and unweighted graphs . . 80
3.4 Parameters for nPSO benchmark graphs. 81

4.1 Parameters for LFR benchmark graphs. 126
4.2 Predicted community size distribution in G_{DBLP} from different methods. . . 159

5.1 Attribute sets used for Sybil detection. 177
5.2 Anomaly detection methods used as baselines for comparison to RSMM . . 186
5.3 Summary of benchmark anomaly detection datasets 188
5.4 ROC-AUC on benchmark anomaly detection datasets with 60% training . . 190
5.5 ROC-AUC on benchmark anomaly detection datasets with 5% training . . . 190
5.6 Avg. ROC-AUC on benchmark graphs with injected Sybils w/ 60% training 196
5.7 Avg. precision on benchmark graphs with injected Sybils w/ 60% training . 196
5.8 Avg. maximum percentile of different anomalies in the benchmark graphs . 197
5.9 Avg. ROC-AUC on pre-processed graphs w/ injected Sybils w/ 60% training 207
5.10 Avg. precision on pre-processed graphs w/ injected Sybils w/ 60% training . 208
5.11 Avg. maximum percentile of different anomalies in pre-processed graphs . . 208

List of Figures

1.1 Toy networks representing two different community structure paradigms. . . 3

2.1 The seven bridges of Königsberg 7
2.2 Examples of different types of graphs 9
2.3 Different types of special graphs 12
2.4 Degree distribution and related plots for an example scale-free network . . . 18
2.5 Graphs with and without 'community structure' 24
2.6 Highlighted trusses of a graph 26
2.7 Rich-club coefficient plots of an example network 28
2.8 The configuration model (CM) 47
2.9 The Maslov-Sneppen (MS) method 48
2.10 A small example Popularity×Similarity graph (PSO graph) 52
2.11 A small example non-uniform Popularity×Similarity graph (nPSO graph) . 54

3.1 Degree distribution and related plots for G_{EU}, G_{LFR}, and G_{nPSO} 61
3.2 Rich-club coefficient plots for G_{EU}, G_{LFR}, and G_{nPSO} 62
3.3 Community related plots for G_{EU}, G_{LFR}, and G_{nPSO} 63
3.4 Delineation of all 1-hop, 2-hop, and 3-hop simple paths between two seed vertices. 70
3.5 Four examples of connected seed vertices. 74
3.6 Maximum Density Core(MDCore) pruning algorithm on a random graph. . 77
3.7 InvProLinkCo verses raw and normalized S2 and S3 weights using NMI . . 83
3.8 InvProLinkCo verses raw and normalized S2 and S3 weights using CCE . . 84
3.9 NMI and CCE for nPSO_EU using the Louvain method with pre-weighting 86
3.10 NMI and CCE for nPSO_BHLP1 using the Louvain method with pre-weighting 87
3.11 NMI and CCE for scaled nPSO_EU graphs using the Louvain method . . . 89
3.12 Compute times for nPSO graphs using the Louvain method 90
3.13 NMI and CCE for nPSO graphs using other community detection methods . 93
3.14 NMI for InvPrLinkCo verses unweighted community detection on nPSO_EU 94
3.15 CCE for InvPrLinkCo verses unweighted community detection on nPSO_EU 94
3.16 NMI for InvPrLinkCo verses unweighted community detection on nPSO_BHLP1 95
3.17 CCE for InvPrLinkCo verses unweighted community detection on nPSO_BHLP1 95
3.18 NMI and CCE of the Louvain method with InvPrLinkCo on LFR_SSGC . . 98
3.19 Community structure of LFR_SSGC with $\mu = 0.6$ and its MDCore 99
3.20 Comparison of results from the benchmark graphs to G_{EU} 101

3.21 Community structure of G_{EU} and its MDCore 103

4.1 Examples of a disconnected and a connected $S2_{uv}$ local community 116
4.2 A connected $S3_{uv}$ local community 117
4.3 Two different disconnected $S2_{uv}$ local communities 121
4.6 Comparison of LoCoLinkCo to InvPrLinkCo on toy networks 131
4.7 NMI and CCE for LFR_BHLP using the Louvain method with pre-weighting 133
4.8 NMI and CCE for scaled LFR_BHLP graphs using the Louvain method . . 135
4.9 Compute times for LFR graphs using the Louvain method 136
4.10 NMI and CCE for nPSO graphs using other community detection methods . 138
4.11 Compute time comparison to other community detection methods 139
4.12 NMI for LoCoLinkCo verses unweighted community detection on LFR_BHLP 140
4.13 CCE for LoCoLinkCo verses unweighted community detection on LFR_BHLP 140
4.14 NMI for LoCoLinkCo verses unweighted community detection on LFR_BHLP_S 141
4.15 CCE for LoCoLinkCo verses unweighted community detection on LFR_BHLP_S 141
4.16 NMI for LoCoLinkCo verses unweighted community detection on LFR_MSPM 142
4.17 CCE for LoCoLinkCo verses unweighted community detection on LFR_MSPM 142
4.18 NMI for LoCoLinkCo verses unweighted community detection on LFR_KRH 143
4.19 CCE for LoCoLinkCo verses unweighted community detection on LFR_KRH 143
4.20 NMI for LoCoLinkCo verses unweighted community detection on LFR_SSGC 144
4.21 CCE for LoCoLinkCo verses unweighted community detection on LFR_SSGC 144
4.22 Assortativity and rich-clubness of benchmark graphs 148
4.23 LoCoLinkCo verses geom. mean of an edge's degree for LFR_KHR 150
4.24 LoCoLinkCo verses geom. mean of an edge's degree for other LFR graphs . 152
4.25 LoCoLinkCo verses geom. mean of an edge's degree for nPSO graphs . . . 154
4.26 Example HSBM graph 156
4.27 LoCoLinkCo verses geom. mean of an edge's degree for HSBM graph . . . 157
4.28 Community structure of G_{DBLP} and its MDCore 158
4.29 Comparison of detected communities from different methods for G_{DBLP}. . . 160

5.1 Toy scenarios demonstrating the combination of subspace scores 184
5.2 Example attribute subspaces using link cohesion metrics 193
5.3 Subspaces of an example LFR benchmark graph with injected Sybils 199
5.4 Subspaces of an example nPSO benchmark graph with injected Sybils . . . 200
5.5 Interpretable bar plots showing attributes most anomalous for a state vector 202
5.6 Subspaces of a true normal data point predicted anomalous 204
5.7 Subspaces of a true anomalous data point predicted anomalous 205
5.8 Subspace densities with and without pre-processing 209

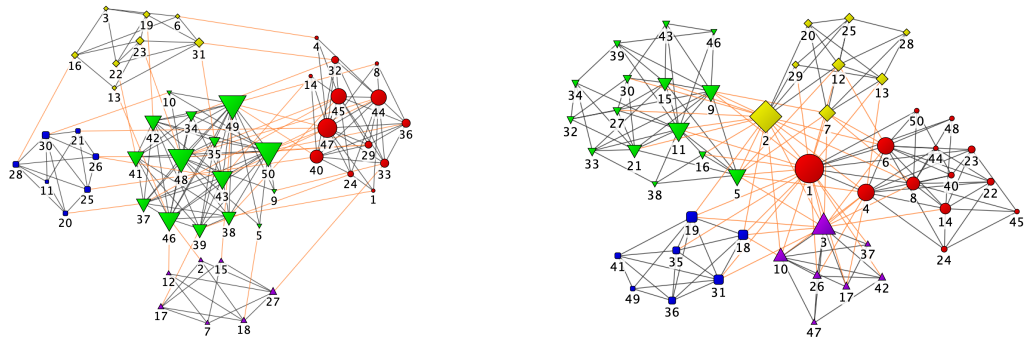
Chapter 1: Introduction

Graphs, which are structures containing the pairwise relationships between objects, have been studied since the 18th century [96]. A graph generated from a real-world system is typically referred to as a network and complex networks are real-world graphs with non-trivial topological structure. Complex networks are everywhere, from people connected by friendships in social networks, to web pages connected by hyperlinks in the world wide web, to neurons connected by synapses in the brain. Since the turn of the 21st century, the ability to collect, store, and analyze exponentially more data than ever before has enabled scientists to discover properties of networks that are common across many different fields of study, leading to a new field of study called network science [19]. A common property that most networks are thought to have is community structure, meaning a network can be partitioned in a way that the partitions, called communities, are more densely connected within than between. Knowing the community structure of a network can provide valuable information: for social networks, communities might represent groups that share a high level of mutual trust or could provide insights on how information or viruses are spread [249]; for the world wide web, communities could provide topical categorizations of web pages [281]; in connectomes, communities may provide insights on how the brain functions [109]. Because of its importance, community detection has been well studied over the past two decades as illustrated by the fact the initial paper on the topic, written by Michelle Girvan and Mark Newman in 2002, has been cited over 15,700 times [111].

In many real-world networks, it has been found that the sizes of communities within a

network can vary widely, often distributed in such a way that there are many more smaller communities and a few larger communities [141]. It is conventionally thought that objects in a network that are highly connected to other objects would be members of larger communities, like in the toy network shown in Figure 1.1a [102, 141, 199]. In 2018, Alessandro Muscoloni and Carlo Vittorio Cannistraci proposed an alternative paradigm for community structure where highly connected objects can be found across all community sizes like in the toy network shown in Figure 1.1b. Both paradigms create realistic community structure patterns observed in the real world. For the latter, imagine members of a large research institute interacting with each other. A network can be created where each member represents an object in the graph and two members are connected in the graph if they have sent an email back and forth to each other at least once in an 18-month time period. If the research institute is divided into departments, it seems likely that the structural patterns in the network would be affected by it. Research institutes are often structured in a hierarchical fashion. For example, at the Johns Hopkins University Applied Physics Laboratory where I am employed, the lab is divided into departments, which are further divided into branches, which are further divided into groups, which in turn are further divided into sections. Individuals within a section are highly likely to be connected to each other and likely to be connected to others in their group. It is less likely that people in unrelated sections of different departments would be connected to one another. At the same time, leaders at each level are likely to be connected with those that fall under their leadership and connected to other leaders at varying levels in the hierarchy. Lastly, it is possible that small departments can play key roles in the organization, requiring some individuals in these departments to reach out to a larger number of people across other departments. Since the sizes of the departments can vary widely, there potentially can be very well-connected members in small departments, similar to the community structure in the new paradigm. Alternatively, consider computer science researchers that collaborate on published journal

articles, conference and workshop proceedings, or monographs. A network of researchers can be created in which a pair are connected if they co-authored at least one paper in the period of several decades. One could argue that the type of communities expected in such a network would better correspond the conventional paradigm of community structure. It seems reasonable to assume that smaller communities would be made up of researchers that frequently collaborate with each other, but less with others. Since they have fewer people to collaborate with, they would not be amongst the highest connected in the network. Alternatively, researchers who collaborate with a lot of other researchers are likely to collaborate with each other and also collaborate with less connected researchers, creating larger communities.



(a) Conventional Community Structure (b) Alternative Community Structure

Figure 1.1: Toy networks representing two different community structure paradigms. The color and shape of the objects in the toy networks correspond to communities. The size of each object is proportional to how connected it is to others. Black lines connect objects in the same community while orange lines connect objects in different communities. (a) A toy network with community structure following the conventional paradigm in which objects that are highly connected to other objects are only found in larger communities (b) A toy network with the community structure following the alternative paradigm where highly connected objects can be found in communities of any size.

While the e-mail network from the research institute and the co-authorship network of published researchers have more intuitive notions of what types of community structure to expect, there are complex networks where it is less clear. For example, online social networks very likely have strong community structure but which paradigm it would fall

under is harder to determine – it may be a mix or neither. As an additional complication, it can be hard for the maintainers of online social networks to prevent people from creating multiple false identities [88]. While it would be difficult for a person not working at the research institute be able to obtain an internal email account or for a 'fake' computer science researcher to be able to collaborate with other researchers in a peer reviewed publication, it can be easy to create fake accounts in online social networks. Nefarious actors can take advantage of this by creating numerous fake accounts, connecting them together and requesting connections with benign users who may be apt to connect to anyone who makes a request [139]. Accounts created in such a way, called Sybils, can be difficult to distinguish from real users, but being able to make the distinction is important – Sybils can be used to gain private user information usually only shared with friends of friends [99], to scam vulnerable individuals out of large sums of money [142, 204], or to trick people into sending images of themselves that can be used for blackmail [142, 190]. In 2012, malicious actors created a fake Facebook account impersonating NATO's most senior commander, Admiral James Stavridis, and were able to deceive other high-ranking officials into accepting friendship requests from the account [149]. Personal information was then harvested, including email addresses, phone numbers, dates of birth, family pictures, and messages revealing beliefs and states of mind. Such information could be used to produce detailed profiles of potential targets, to blackmail individuals, or to create convincing deep fakes of people not typically in the public spotlight [24, 58]. Detecting Sybils is an example of an anomaly detection task. Anomaly detection in networks is a newer research area that has a large number of important applications. Anomalies in a network are objects, relationships, or substructures that deviate significantly from the typical patterns observed in the network and could correspond to fake malicious accounts in online social networks or diseases in the brain. A 2015 survey by Leman Akoglu, Hanghang Tong, and Danai Koutra on graph-based anomaly detection has been cited over 950 times [7].

The work in this dissertation analyzes a new type of metric called link cohesion, a measure used to assess the strengths of the relationships within a network proportional to how supported it is by other relationships within semi-close proximity, for the purpose of improving the communities and anomalies detected by traditional algorithms. This type of measure is particularly beneficial in social networks – if two people are connected, they are more likely to have a stronger connection if they have many friends in common or have many friends that are friends with each other, particularly if those involved are not connected to many others in the network. A deterministic, scalable, and easily parallelizable link cohesion metric was originally proposed by Cetin Savkli, Amanda Galante, Jonathan Cohen and myself in 2019 [239]. The link cohesion metric was used to create informative simplified graphs of highly connected networks for both visualization purposes and to be used to detect cohesive subgraphs that highlight important groups of objects that are amongst the most densely connected in a network.

Chapter 2 provides a preliminary introduction to graph theory and network science, including definitions related to the structure of real-world networks, a review of community detection and anomaly detection in networks, and descriptions of the different network models used throughout the dissertation. Chapter 3 analyzes the benefits of using the link cohesion metric proposed in [239] to improve communities detected in real and synthetic networks that have community structure similar to the alternative paradigm described above. Chapter 4 analyzes the benefits of using a modified link cohesion metric to improve communities detected in real and synthetic networks that have community structures similar to the conventional paradigm described above. The modified metric incorporates equations used for link reliability and link prediction proposed by Carlo Vittorio Cannistraci, Alessandro Muscoloni, and Ilyes Abdelhamid in 2018 [45, 182]. Additional observations are made on the potential of analyzing the modified metric to gain insights on if a network has community structure or not, with the possibility of obtaining clues on the type of com-

munity structure a network may have. Chapter 5 analyzes the benefits of using both of the link cohesion metrics to create vertex-level features that can be used to detect Sybils in online social networks. Chapter 5 additionally makes use of an interpretable anomaly detection method originally proposed by Cetin Savkli and myself earlier this year [238].

It has been shown that when the appropriate link cohesion metric is used to pre-weight a graph with one of the community structure paradigms described above, communities detected by algorithms that maximize modularity are improved in all benchmark graphs. In certain situations, other community detection algorithms pre-weighted with the link cohesion metrics obtain even better performance improvements though in other situations performance could sometimes degrade. To assist in determining which link cohesion metric is expected to provide a benefit when used to pre-weight a network, several exploratory data analysis tools are introduced. These tools are used to better understand real-world networks that exhibit both types of community structure. Additionally, the performance on benchmark graphs that are generated to have similar properties as these real-world networks were found to be informative in determining which algorithms would perform well on the real-world networks and which would not. In addition to being useful to detect communities, the two link cohesion metrics are shown to be beneficial for anomaly detection. Different attribute sets and different anomaly detection algorithms were all used to detect anomalies similar to Sybils in online social networks that are injected into benchmark graphs with each type of community structure. The best performing anomaly detection method was designed to be interpretable both for understanding what attributes contribute most to a data point being consider anomalous and for determining which subspaces a data point is most anomalous in relative to the training data. A user could use the tools for human-in-the-loop interactions that may further improve performance results which is made further possible since the method was shown to perform well with a small percent training data.

Chapter 2: Graph and Network Science Preliminaries

2.1 Introduction to Graph Theory

Graph theory can be traced back to a paper published by Leonhard Euler in 1736 where he proved that there was no way to walk through the city of Königsberg that would cross each of its seven bridges, illustrated in Figure 2.1a, once and only once [96]. He completed the task by converting the problem into a simplified structure shown in Figure 2.1b. Although the term graph was not introduced until 1878 by James Joseph Sylvester [256], the structure in Figure 2.1b is what now is refer to as an graph.

(a) The seven bridges of Königsberg

(b) Graph of the seven bridges

Figure 2.1: Graph theory originated in 1736 with a paper published by Leonhard Euler on the seven bridges of Königsberg. (a) Map of Königsberg with the seven bridges highlighted in green and connected land masses enumerated (b) Graph of the seven bridges.

Definition 2.1.1. A *graph* $G = (V, E)$ is defined by a set of *vertices* $V = \{v_1, v_2, ..., v_N\}$ and a set of pairs of vertices $E = \{(u, v)$, where $u, v \in V\}$. A pair of vertices $e = (u, v)$ is

called an *edge*. A *directed graph* has ordered pairs of vertices. An *undirected graph* has an edge set E such that if $(u,v) \in E, (v,u) \in E$ as well. A *multigraph* is a special type of graph where the set of edges is a multiset, meaning pairs of vertices can have *multiple edges*.

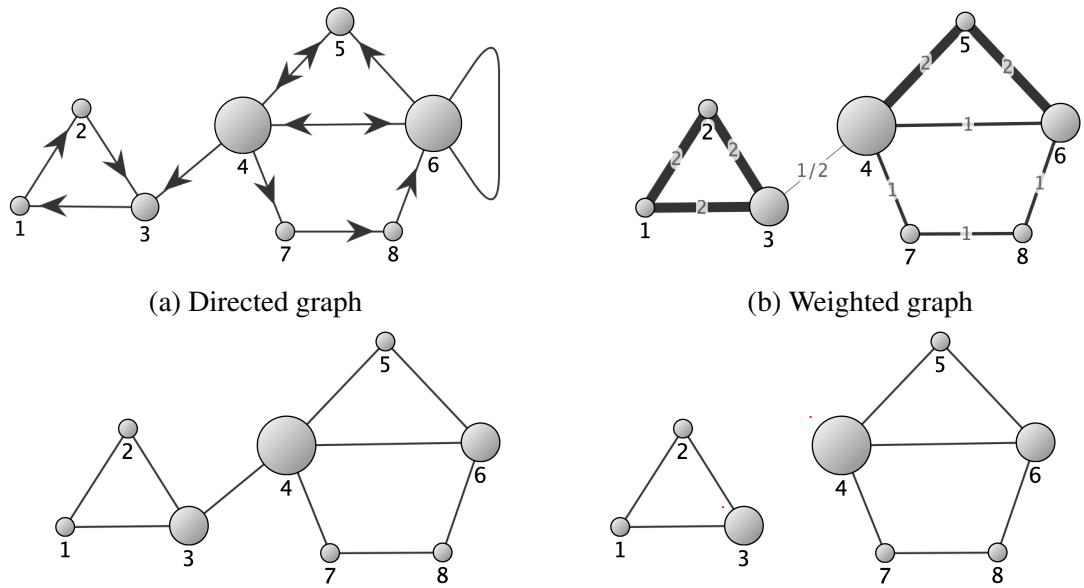
Sylvester introduced the new term to draw connections between structures found in the field of Chemistry and structures studied in Algebra. Graphs are sometimes referred to as *networks* when they come from a real-world example, with the vertices sometimes referred to as *nodes* and the edges sometimes referred to as *links*. While this section will review some basic concepts in graph theory, common structural patterns observed in real-world networks will be discussed in the next section. The vertex and edge terminology will be used throughout the dissertation for both networks and graphs for consistency, with a graph being called a network only if it comes from a real-world system.

As stated in the definition, graphs can be directed or undirected. An example of a directed graph is shown in Figure 2.2a. Euler's graph in Figure 2.1b is an example of an undirected graph. Additionally, graphs can be weighted or unweighted.

Definition 2.1.2. A *weighted graph* $G = (V,E,\omega)$ is defined by a set of vertices $V = \{v_1, v_2, ..., v_N\}$, a set of pairs of vertices $E = \{(u,v)$, where $u,v \in V\}$, and a function $\omega : V \times V \to \mathbb{R}$ with $\omega(V \times V \setminus E) = 0$. If the graph is undirected, then $\omega(u,v) = \omega(v,u)$. If $\omega(E) = 1$, then G is an *unweighted graph*.

An example of a weighted graph is shown in Figure 2.2b. The graphs in Figures 2.1b and 2.2a are both unweighted graphs. Another type of graph is a simple graph.

Definition 2.1.3. A *simple graph* is an unweighted, undirected graph containing no self-loops or multiple edges. A *self-loop* is an edge that connects a vertex to itself. As stated previously, *multiple edges* occurs when two vertices have more than one edge (in the same direction for directed graphs) connecting them.



(a) Directed graph (b) Weighted graph

(c) Simple graph (d) Simple graph with two components

Figure 2.2: Examples of different types of graphs. The size of each vertex is relative to its degree. (a) Directed graph (b) Weighted graph (c) Simple graph (d) Simple graph with two components

Vertex v_6 in the graph shown in Figure 2.2a has a self-loop. There are multiple edges between vertices v_1 and v_4 and between vertices v_3 and v_4 in the graph shown in Figure 2.1b. Figure 2.2c is a *simplified* version of the graphs in Figure 2.2a and Figure 2.2b where directed edges, self-loops, multiple edges, and edge weights are removed to create a simple graph.

A common way to represent a graph is with its adjacency matrix.

Definition 2.1.4. Let $G = (V, E, \omega)$ be a graph with $N = |V|$. The *adjacency matrix* for G is the $N \times N$ matrix A such that

$$A_{uv} = \begin{cases} 1 & \text{if } (u,v) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

If the graph is unweighted, then $A_{uv} = \omega(u,v)$.

The adjacency matrix can be used to compute different metrics and measures of a graph.

Metrics and measures at the graph-level, vertex-level, and edge-level are used to describe the structure of a graph. Often vertex-level measures are additionally summarized with graph-level metrics. Basic graph-level measures include the size of the graph, the number of edges in the graph, and the density of the graph.

Definition 2.1.5. Let $G = (V, E)$ be a graph. The *size of the graph* is defined to be the number of vertices $N = |V|$ and the *number of edges* in the graph is defined to be $L = |E|$. The *density* of the graph is defined as $\delta = L/L_{max}$ where $L_{max} = \binom{N}{2} = N(N-1)/2$ if G is undirected and $L_{max} = 2\binom{N}{2} = N(N-1)$ if G is directed.

If the graph is weighted, then another graph-level measure of importance is the total weight.

Definition 2.1.6. The *total weight* of a weighted graph $G = (V, E, \omega)$ is $W = \frac{1}{2}\sum_{u,v \in V} \omega(u, v)$.

The most common vertex-level metric is the degree.

Definition 2.1.7. The *degree* of a vertex $v \in V$ of a graph $G = (V, E)$ is $k_v = |\{(u, v) : u \in V, (u, v) \in E\}|$. If the graph is directed, then the *degree* or *total degree* is $k_v = k_v^{total} = k_v^- + k_v^+$ where the *in-degree* is $k_v^- = |\{(u, v) : u \in V, (u, v) \in E\}|$ and the *out-degree* is $k_v^+ = |\{(v, u) : u \in V, (v, u) \in E\}|$. If the degree of a vertex is 0, the vertex is called an *isolated vertex*. If $v = v_i$, then k_{v_i} is shortened to k_i.

The degree can be seen as a measure of centrality of a vertex. A *centrality measure* is a metric capturing the importance, power, or influence of a vertex or edge in a given graph, typically used in the context of real-world networks [33]. Terms related to degree are neighbors and neighborhood.

Definition 2.1.8. Let $G = (V, E)$ be a graph with $u, v \in V$. u and v are said to be *connected*, *adjacent*, or *neighbors* if $(u, v) \in E$. The *neighborhood* of a vertex $v \in V$ is defined as

$\mathcal{N}(v) = \{x \in V | (x,v) \in E\}$. $x \in V$ is said to be a *common neighbor* to $u, v \in V$ if $x \in \mathcal{N}(u)$ and $x \in \mathcal{N}(v)$.

If $G = (V,E)$ is a simple graph, then the degree of vertex $v \in V$ is equal to the size of its neighborhood, $k_v = |\mathcal{N}(v)|$, and can be computed by summing up the rows or columns of its corresponding adjacency matrix. The statistics of the degrees of the vertices in a graph can be rolled up into graph-level metrics.

Definition 2.1.9. Let $G = (V,E)$ be a graph with $N = |V|$. The *maximum degree* of the graph is defined to be $k_{max} = \max\{k_v | v \in V\}$, the *minimum degree* is defined to be $k_{min} = \min\{k_v | v \in V\}$, and the *average degree* is defined to be $\langle k \rangle = \frac{1}{N}\sum_{v \in V} k_v$. The *degree sequence* is defined to be a list of N numbers $[k_1, k_2, ..., k_N]$ where k_i is the degree of the ith vertex.

If a graph is weighted, then in addition to the degree of each vertex, the weighted degree can be calculated. This term is also referred to as the strength of the vertex.

Definition 2.1.10. The *weighted degree* or *strength* of a vertex $v \in V$ of a weighted graph $G = (V,E,\omega)$ is $s_v = \sum_{u \in \mathcal{N}(v)} \omega(u,v)$. If the weighted graph is also a directed graph, then the *strength* or *total strength* is $s_v = s_v^{total} = s_v^- + s_v^+$ where the *in-strength* is $s_v^- = \sum_{u \in \mathcal{N}(v)} \omega(u,v)$, and the *out-strength* is $s_v^+ = \sum_{u \in \mathcal{N}(v)} \omega(v,u)$.

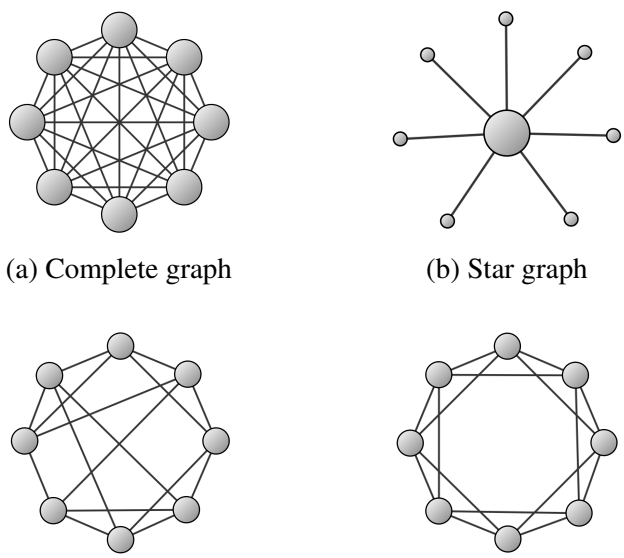
Another related term is the average strength in a weighted graph.

Definition 2.1.11. The *average strength* of a vertex $v \in V$ of a weighted graph $G = (V,E,\omega)$ can be defined as $s_v^{avg} = \frac{1}{k_v}\sum_{u \in \mathcal{N}(v)} \omega(u,v)$.

Special types of graphs can be described in terms of the conditions on the degrees of their vertices.

Definition 2.1.12. Let $G = (V, E)$ be a graph with $N = |V|$. If $k_v = N - 1$ for all $v \in V$, then the graph is a *complete graph*. If $N - 1$ vertices have degree 1 and 1 vertex has degree $N - 1$, then the graph is a *star graph*. If $k_v = k$ for all $v \in V$, then the graph is a *k-regular graph*. A *ring lattice* is a k-regular graph whose vertices $v \in V$ are connected to its $k/2$ closest neighbors on each side when laid out in a circle equi-distanced apart in $\mathbb{R}^2$.

A complete graph is a k-regular graph with $k = N - 1$. Examples of these special types of graphs are shown in Figure 2.3. Another special type of graph is a subgraph.



(a) Complete graph (b) Star graph

(c) 4-regular graph (d) Ring lattice with k=4

Figure 2.3: Different types of special graphs with $N = 8$. The vertex's size is relative to its degree. (a) Complete graph (b) Star graph (c) An example 4-regular graph (d) Ring lattice with k=4.

Definition 2.1.13. A *subgraph* $G_S = (V_S, E_S)$ of a graph $G = (V, E)$ is a graph whose vertex set V_S is a subset of V and whose edge set E_S is a subset of E, that is $V_S \subseteq V$ and $E_S \subseteq E$. If the edge set $E_S = \bigcup_{u,v \in V_S} (u, v) \in E$, then the subgraph is called an *induced subgraph*. The graph G is referred to as a *supergraph* of G_S.

Certain subgraphs have special names based on their structure. Two types of subgraphs with special names are called cliques and connected components.

Definition 2.1.14. A *clique* is a subgraph in which each of its vertices are connected to all others, making it a complete graph. A *maximal subgraph* with respect to a given property is a subgraph that satisfies the property such that no larger subgraph with that property exists in G, which contains them. A clique is generally defined only if it is a maximal subgraph.

Figure 2.2c contains two 3-vertex cliques: one 3-vertex clique is made up of vertices $V_{S_1} = \{v_1, v_2, v_3\}$ and the other 3-vertex clique is made up of vertices $V_{S_2} = \{v_4, v_5, v_6\}$. The study of complete subgraphs goes back to at least 1935 [94] when Paul Erdős and George Szekeres used the special subgraphs in their reformulation of Ramsey theory, although the term clique was not introduced until 1949 when Duncan Luce and Albert Perry used the term to describe a group of people whom all know each other in a social networks [157].

Definition 2.1.15. A subgraph is *connected* if every pair of vertices can be joined by a path. A maximal connected subgraph is called a *connected component* or just *component*. A graph is connected if it only has one connected component.

The number of connected components is an important graph-level topological invariant of a graph. In topological graph theory, it can be interpreted as the zeroth Betti number of a graph. In algebraic graph theory, it equals the multiplicity of 0 as the eigenvalue of the Laplacian matrix of a graph. All of the example graphs up to now have been connected. The graph in Figure 2.2d is not connected, and instead has two components: one component is made up of vertices $V_{S_1} = \{v_1, v_2, v_3\}$ and the other component made up of vertices $V_{S_2} = \{v_4, v_5, v_6, v_7, v_8\}$. There are two components in Figure 2.2d because, for example, there is no path that connects vertex v_1 to vertex v_8.

Definition 2.1.16. A *path* (sometimes referred to as a *walk*) between $u, v \in V$ of a graph $G = (V, E)$ is a sequence of edges in E which join vertex u to vertex v. The *length of a path* is equal to the number of edges in the path. A path of length h from vertex u to vertex v,

sometimes referred to as *h-hop* path, can be described as the ordered list of vertices starting from u and ending with v that are connected by the edges in the path.

Special types of paths of interest are shortest paths, simple paths, trails, and cycles.

Definition 2.1.17. A *shortest path* between two distinct vertices $u, v \in V$ in a graph $G = (V, E)$ is a path connecting vertex u and vertex v with the shortest length. The length of the shortest path between u and v is called the distance between u and v. The length of the longest shortest path between any pair of distinct vertices in a graph is called the *diameter* of the graph. The average distance across all pairs of distinct vertices is called the *characteristic path length*, $\mathcal{L}$.

The diameter of a graph and the characteristic path length of a graph are graph-level metrics. The characteristic path length of the graph in Figure 2.2c is 1.78. The two shortest path between v_1 and v_8 in the graph are $(v_1, v_3, v_4, v_7, v_8)$ and $(v_1, v_3, v_4, v_6, v_8)$ which both have length 4. 4 is the diameter of the graph. Shortest paths do not visit the same vertex twice, making them both simple paths and trails.

Definition 2.1.18. A *simple path* is a path with no repeated vertices. A *trail* is a path with no repeated edges.

Definition 2.1.19. A *cycle* is a non-empty trail which the only repeated vertices are the first and last. A *k-cycle* is a cycle of length k. A *triangle*, $\blacktriangle$, is a 3-cycle. A *rectangle*, $\blacksquare$, is a 4-cycle.

An example of a triangle in Figure 2.2c is (v_1, v_2, v_3) and an example of a rectangle in the same graph is (v_4, v_6, v_7, v_8). Analyzing cyclic properties between connected edges can be used to provide alternative definitions of adjacency.

Definition 2.1.20. Two edges a and b are *triangle adjacent*, denoted $a \blacktriangle b$, if a and b share a triangle. Two edges a and b are *triangle-connected* if $a \blacktriangle b$ or if there exist edges $e_1, e_2, ..., e_n$

such that $a \blacktriangle e_1$, $e_n \blacktriangle b$, and $e_i \blacktriangle e_{i+1}$, for $i = 1, ..., n-1$. Two edges a and b are *rectangle-adjacent* denoted $a \blacksquare b$, if a and b share a rectangle, that is, a 4-cycle. Two edges a and b are *rectangle-connected* if $a \blacksquare b$ or if there exists edges $e_1, e_2, ..., e_n$ such that $a \blacksquare e_1$, $e_n \blacksquare b$, and $e_i \blacksquare e_{i+1}$ for $i = 1, ..., n-1$ [71].

In the Königsberg bridge problem described at the beginning of this section, Euler's goal was to find a trail that visits each edge exactly once. By analyzing the degrees of vertices in graphs, Euler was able to prove that such a path would only be possible in graphs that have only 0 or 2 vertices with odd degree. Since the graph in Figure 2.1b has 4 vertices with odd degree, no such path exists. Graph theory has been used to solve a number of difficult real-world problems by analyzing characteristics of the underlying mathematical structures including the traveling salesman problem [32], the Chinese postman problem [171, 260], and the four-color theorem [12].

2.2 The Structure of Real-World Networks

While graph theory is the study of mathematical structures, network science is an interdisciplinarity field focused on finding and understanding common patterns found in real-world complex networks with non-trivial topological structures across a broad range of domains [45, 46, 139, 182, 208, 212, 254, 261]. Similar to how Euler represented the real-world Königsberg bridge problem as a graph made up of land masses connected by bridges, many complex systems can be represented as graphs with up to billions or more vertices and even more edges – from people connected by friendships in social networks, to web pages connected by hyperlinks in the world wide web, to neurons connected by synapses in our brain [199]. Network science is a relatively new field of study that emerged at the beginning of the 21st century [19]. Prior to the advent of computers and the internet (which itself is an example of a network of computers connected by data connections such as optical fiber

cables or telephone lines [199]), understanding commonalities between diverse networks was near impossible due to the difficulties in mapping out the connections in large networks and sharing that information across disciplines [19]. With modern day tools, storage capacity, and ease of collaboration, many observations have been made that has improved our ability to locate terrorists and criminals in social networks [212], find information on the internet [208], and understand neurological disorders in our brains [254].

This section will review some of the main distinctive patterns found in the structure of real-world complex networks across many domains. These patterns are sometimes studied based on how they would vary if the size of the network grew infinitely large. While definitions in this section are specifically defined in terms of networks, they can also be used for graphs. In the last section of this chapter, network models that scientists have used to better understand the patterns observed in real-world networks will be reviewed.

The first observation relates to the sparsity of networks.

Definition 2.2.1. A network $G = (V,E)$ with $N = |V|$ is *sparse* if its density $\delta \to 0$ as $N \to \infty$. Otherwise, the network is *dense*.

Sparsity regimes can be defined by how the average degree, $\langle k \rangle$, of the network grows as the size of the network grows [159].

Definition 2.2.2. A network $G = (V,E)$ with $N = |V|$ is *extremely sparse* if the average degree, $\langle k \rangle$, remains constant as $N \to \infty$ [37, 199].

In extremely sparse networks, $L = |E|$ increases in proportion to $N = |V|$, which in turn implies that $O(L+N)$ algorithms are equivalent to $O(N)$. Many real-world networks are extremely sparse. For example, in social networks the number of friends a person has is not expected to depend on the population of the whole world. Most networks discussed in [199] are extremely sparse networks or networks where $\langle k \rangle \to \log N$ as $N \to \infty$. Another

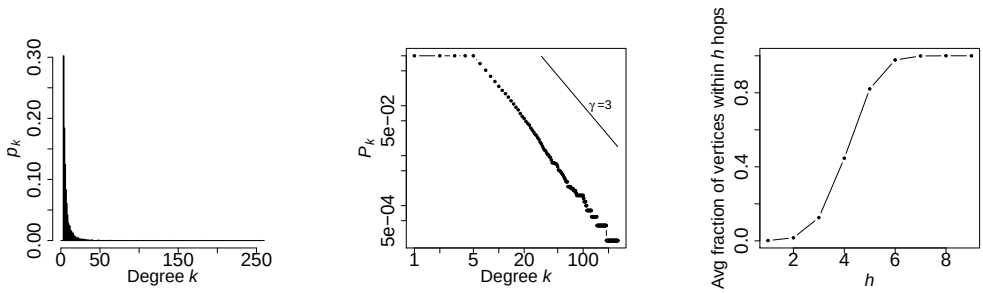
sparsity regime are networks where $\langle k \rangle \to \sqrt{N} \log \log N$ as $N \to \infty$ [159]. Often times, only a static network is available for analysis. In this case, sparsity of the network can be measured based on if $\langle k \rangle \leq \lceil \log N \rceil$ (sparse network), if $\lceil \log N \rceil < \langle k \rangle \leq \lceil \sqrt{N} \log \log N \rceil$ (highly connected network), or if $\langle k \rangle > \lceil \sqrt{N} \log \log N \rceil$ (denser network). Most real-world networks are not dense, though research has shown that certain types of networks densify over time, accumulating new edges faster than new vertices [147].

The next set of observations relate to the degrees of the vertices in a network.

Definition 2.2.3. The *degree distribution* $p_k = p(k) : \mathbb{Z}^{0+} \to \mathbb{R}^{0+}$ of a network $G = (V, E)$ is defined as $p_k = \frac{|\{v \in V | k_v = k\}|}{|V|}$.

For a given k, p_k is equal to the fraction of vertices in the network with degree k, or equivalently the probability that a vertex chosen uniformly at random from the network will have degree k. An example degree distribution is shown in Figure 2.4a. Almost all real-world networks have degree distributions that are *right-skewed* [199]. The degree distribution in Figure 2.4a additionally follows a power law $k^{-\gamma}$ with $\gamma = 3$ starting at $k^*_{min} = 20$. This property can more easily be seen in the *cumulative distribution function (CDF)* $P_k = \sum_{k'=k}^{\infty} p_{k'}$ shown in Figure 2.4b where $P_k \propto k^{-(\gamma-1)}$. Degree distributions that exhibit this type of property are called *scale-free*. Although there is so single agreed upon definition, scale-free networks are often studied in network science. The formal definition from [43] is provided here as a reference.

Definition 2.2.4. A network is a *perfect scale-free network* if $p_k = (\gamma - 1) k_{min}^{\gamma-1} k^{-\gamma}$ for $2 < \gamma < 3$ [199]. A network is a *(weak) scale-free network* if its degree distribution $p_k \propto k^{-\gamma}$ for $k \geq k^*_{min} > 1$ where k^*_{min} must cover at least 50 vertices. A network is a *strong scale-free network* if additionally $2 < \gamma < 3$ [43]. The *power-law cut-off degree* of a scale-free network is $k_{cut} = k_{min} N^{\frac{1}{\gamma-1}}$ which is the expected maximum degree of a perfect scale-free network [19].



(a) Degree distribution (b) Degree CDF (c) Hop plot

Figure 2.4: Scale-free network with $N = 5000$ vertices where $\gamma = 3$ and characteristic path length $\mathcal{L} = 4.6$. (a) Degree distribution of the network (b) Cumulative degree distribution of the network (c) Hop plot of the network

The emphasis on $2 < \gamma < 3$ is motivated by additional properties of networks that occur in this range [19, 199]. While there have been numerous claims that many real-world networks are scale-free, the prevalence may be less frequent than originally thought [43, 64]. Nevertheless, scale-free networks have degree distributions that are strongly right-skewed as seen in many real-world networks and there exists networks models that can provide an explanation for their prevalence [20, 209], as discussed in Section 2.5.

The degree distribution or related degree sequence of a network alone cannot be considered as a comprehensive characterization of a network. Because of this, Mark Newman introduced the concept of assortativity in 2002 [192].

Definition 2.2.5. Let $G = (V, E)$ represent a network with corresponding adjacency matrix A and with $L = |E|$. The *assortativity coefficient* is the Pearson correlation coefficient of degree between pairs of adjacent vertices,

$$r = \frac{\sum_{u,v \in V} (A_{uv} - k_u k_v / 2L) k_u k_v}{\sum_{u,v \in V} (k_u \delta_{uv} - k_u k_v / 2L) k_u k_v}, \tag{2.1}$$

where δ_{uv} is the Kronecker delta. Networks with $r > 0$ are called *assortatively mixed* and networks with $r < 0$ are called *disassortatively mixed*.

While r can theoretically range from $[-1,1]$, often the magnitude of r is not large meaning the correlation is not especially strong. That said, knowing the sign of the coefficient can be useful in understanding behavior on the network [199]. It has also been shown for certain network models the assortativity coefficient tends to 0 as the size of the network grows [206]. In 2003, Mark Newman published that social networks tend to be assortatively mixed whereas most other types of networks tend to be disassortatively mixed [199, 202]. In 2004, Sergei Maslov, Kim Sneppen and Alexei Zaliznyak found that many real-world networks tend to be disassortatively mixed because the number of edges that can fall between high-degree vertex pairs is limited [163]. Although the assortativity coefficient relates specifically to the degrees of the vertices in the network, assortativity can be used to describe other attributes related to the network including non-structural attributes that may be associated with the network, such as age, race or gender in social networks [199]. A survey of research related to the assortativity in networks was completed by Rogier Noldus and Piet Van Mieghem in 2015 [206].

Another observation of real-world networks relates to the characteristic path length of its underlying graph. Often times, the characteristic path length scales logarithmically with the size of the network ($\mathcal{L} \sim \log N$) [167]. A useful plot related to the characteristic path length is called the *hop plot*, introduced by Michalis Faloutsos, Petros Faloutsos and Christos Faloutsos in 1999 [97], which depicts the average fraction of vertices within h hops. The hop plot corresponding to the degree distribution in Figure 2.4a is shown in Figure 2.4c. Even as a network grows large, often the number of hops to reach any other vertex is relatively small in real-world networks.

The next set of observations relate to the number of triangles found in the network, sometimes referred to as *transitivity*, which can be measured by common vertex-level and graph-level metrics. The vertex-level metric called the local clustering coefficient will first be introduced.

Definition 2.2.6. The *local clustering coefficient* of a vertex v in a network $G = (V, E)$ is

$$cc_v = \frac{2|\{(u,x) : u,x \in \mathcal{N}(v), (u,x) \in E\}|}{k_v(k_v - 1)}. \tag{2.2}$$

cc_v is the count of all distinct pairs of vertices that are neighbors of vertex v and neighbors of each other (which create triangles) divided by the total number of pairs of neighbors of vertex v. It is a measure of how close a vertex's neighborhood is to being a clique and was introduced in 1998 by Duncan Watts and Steven Strogatz [273]. If $cc_v = 1$, then the 1-hop neighborhood of v is a complete graph making the subgraph a clique. If $cc_v = 0$, then the 1-hop neighborhood of v is a star graph. Two related graph-level metrics are the average local clustering coefficient and the global clustering coefficient.

Definition 2.2.7. The *average local clustering coefficient* of a network $G = (V, E)$ with $N = |V|$ is defined as

$$\overline{cc} = \frac{1}{N} \sum_{v \in V} cc_v. \tag{2.3}$$

Definition 2.2.8. The *global clustering coefficient* of a network $G = (V, E)$ is defined as

$$gcc = \frac{3 \times |\blacktriangle|}{|\text{connected triples of vertices}|}, \tag{2.4}$$

where a $\blacktriangle$ is a 3-cycle and a *connected triple* is a vertex connected directly to an unordered pair of others.

The factor of three in the numerator of the global clustering coefficient equation arises because each $\blacktriangle$ gets counted three times when connected triples are counted in the network. *gcc* represents the probability in a the network that two of a vertex's neighbors would be neighbors of each other. A form of the global clustering coefficient was originally proposed by Duncan Luce and Albert Perry in 1949 [157]. Although the two metrics measure a similar value, they can be orders of magnitude different in some cases. A pattern observed in

real-world networks is that lower degree vertices tend to have higher cc_v values on average [97, 199, 266]. Additionally, real-world networks tend to have significantly higher clustering than random graphs generated with the same graph-level metrics. A network that has high clustering compared to random graph, yet a similar characteristic path length is considered to be a *small-world network* [126, 191, 273].

High clustering values can be an indication of another important observation of real-world networks: networks tend to have stronger clustering within specified groups of vertices than between those groups of vertices, though it is typically stated that these groups are just more densely connected within than between. While the induced subgraphs created using such groups of vertices, often referred to as *communities*, can be overlapping or non-overlapping, the focus of this dissertation is on non-overlapping communities. There is no single commonly agreed upon definition for communities. A general definition is provided and will be compared to other definitions that have been proposed.

Definition 2.2.9. A undirected network $G = (V, E, \omega)$ is said to have *community structure* if its vertices can be partitioned into M mutually exclusive groups of vertices V_{C_i} for $i = 1, .., M$ such that the induced subgraph of each group creates *communities*, $C_i = G_{C_i} = (V_{C_i}, E_{C_i}, \omega_{C_i})$ for $i = 1, .., M$, which satisfies that

$$\forall v \in C_i, \quad \frac{\sum_{u \in V_{C_i}} \omega(u,v)}{|V_{C_i}| - 1} > \max \left[\frac{\sum_{u \in V_{C_j}} \omega(u,v)}{|V_{C_j}|}, \; j = 1, 2, .., M \text{ and } j \neq i \right]. \tag{2.5}$$

A community is a subgraph such that the average strength of each vertex with the other members of its group exceeds the average strength of the vertex with the vertices of any other group. Often times, networks are unweighted and then the average strength with respect to a group can be interpreted as the average number of edges with other members the group. While this definition does not take clustering (the presence of triangles) into

account, it is possible to derive edge weights that do. This is discussed in Chapters 3 and 4.

Before continuing the discussion of communities in networks, some community-level and vertex-level measures related to communities will be defined.

Definition 2.2.10. Given a community $C = G_C = (V_C, E_C)$ of a network $G = (V, E)$, the *size* of the community is equal to $N_C = |V_C|$ and the number of internal edges in the community is equal to $L_C = |E_C|$. Let $V_{C'} = V \setminus V_C$ be the set of vertices not part of community C. An edge $(u, v) \in E$ is an *internal edge* if $u, v \in V_C$ and it is an *external edge* if $u \in V_C$ and $v \in V_{C'}$ or if $u \in V_{C'}$ and $v \in V_C$. The *internal degree* of the of a vertex v with respect to community C is defined as

$$k_v^{int}(C) = |(u, v) : u, v \in V_C, (u, v) \in E|. \tag{2.6}$$

If C is the true community of v then it can be abbreviated as k_v^{int}. The *external degree* of the vertex v with respect to community C is defined as

$$k_v^{ext}(C) = |(u, v) : u \in V_C \text{ and } v \in V_{C'}, (u, v) \in E|. \tag{2.7}$$

If C is the true community of v then it can be abbreviated as k_v^{ext}. For each vertex v, the *mixing rate* is equal to

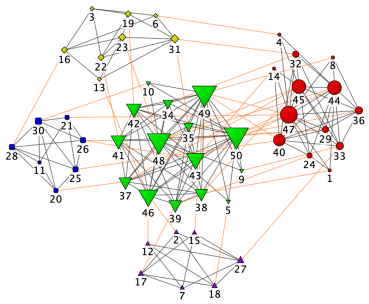
$$\mu_v = \frac{k_v^{ext}}{(k_v^{ext} + k_v^{int})}. \tag{2.8}$$

Additional community-level and graph-level metrics will be introduced in the next section. A graph with community structure is shown in Figure 2.5a where there are 5 communities, each represented in a different color and shape, with the internal edges in black and the external edges in orange. Definition 2.2.9 is similar to the weak definition of community used by Santo Fortunato and Darko Hric in their 2016 User Guide to Community Detection in Networks, except he assumes the weights are probabilities. Often, the graphs used for community detection are unweighted and one does not have access to the weight

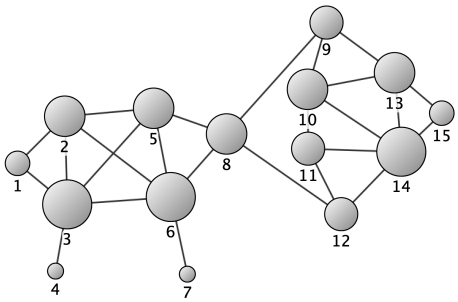
of each edge. While it is possible to use the definition for unweighted graphs, there are cases in which it would seem a graph should have community structure but does not based on the definition. For example, the unweighted graph in Figure 2.5b does not have community structure based on Definition 2.2.9. While it might seem as though the graph could be split into two communities, no matter which community v_8 is placed into the criteria would not be met. There are also cases in which an unweighted graph generated from a random model can be said to have community structure based on Definition 2.2.9, such as the one in Figure 2.5c.

Fortunato uses the Definition 2.2.9 as an improvement on definitions proposed by Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi in 2004 [222] and Yanqing Hu, Hongbin Chen, Peng Zhang, Menghui Li, Zengru Di and Ying Fan in 2008 [125]. The previous definitions used sums instead of averages and therefore did not handle community structures with potentially large differences in community sizes, which is often seen in real-world networks [141]. For example, the graph in Figure 2.5d would not be considered to have community structure by either of the previous definitions mentioned.

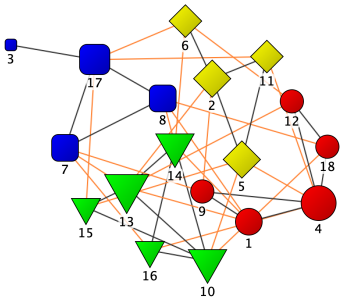
While the proposed definitions of community have been shown to have some flaws, they can be useful in understanding what the goal of a community is. Many other definitions have been defined in addition to these. For example, in 2006, Mark Newman proposed that a community could be defined as an indivisible subgraph [197]. In 2012, Comandur Seshadhri, Tamara Kolda, and Ali Pinar propose that a community is a module with large internal clustering coefficient, using this statement to prove that any such community must contain a dense Erdős -Rényi subgraph (see Subsection 2.5.1 for details about Erdős -Rényi graphs) [248]. In 2018, Alessandro Muscoloni and Carlo Vittorio Cannistraci propose that communities can be defined by angularly separated regions in a hyperbolic space where the radius represents vertex centrality and the angular displacement between two vertices



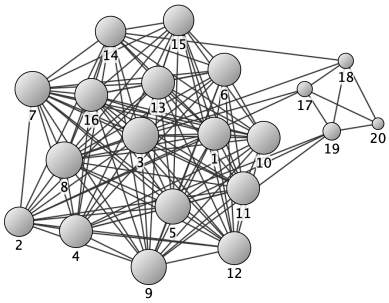
(a) Graph w/ ‘community structure’



(b) Graph w/o ‘community structure’ 1



(c) Random graph w/ ‘community structure’



(d) Graph w/o ‘community structure’ 2

Figure 2.5: Graphs with and without community structure. Graphs with community structure based on Definition 2.2.9 show each community represented in a different color and shape. Internal edges are shown in black and external edges are shown in orange. (a) Unweighted graph with community structure based on all definitions (b) Unweighted graph without community structure based on Definition 2.2.9 (c) Unweighted random graph with community structure based on Definition 2.2.9 and based on the definition proposed by Yanqing Hu et al. in 2008 [125] (d) Unweighted graph without community structure based on definitions proposed by Filippo Radicchi et al. in 2004 [222] and Yanqing Hu et al. in 2008 [125].

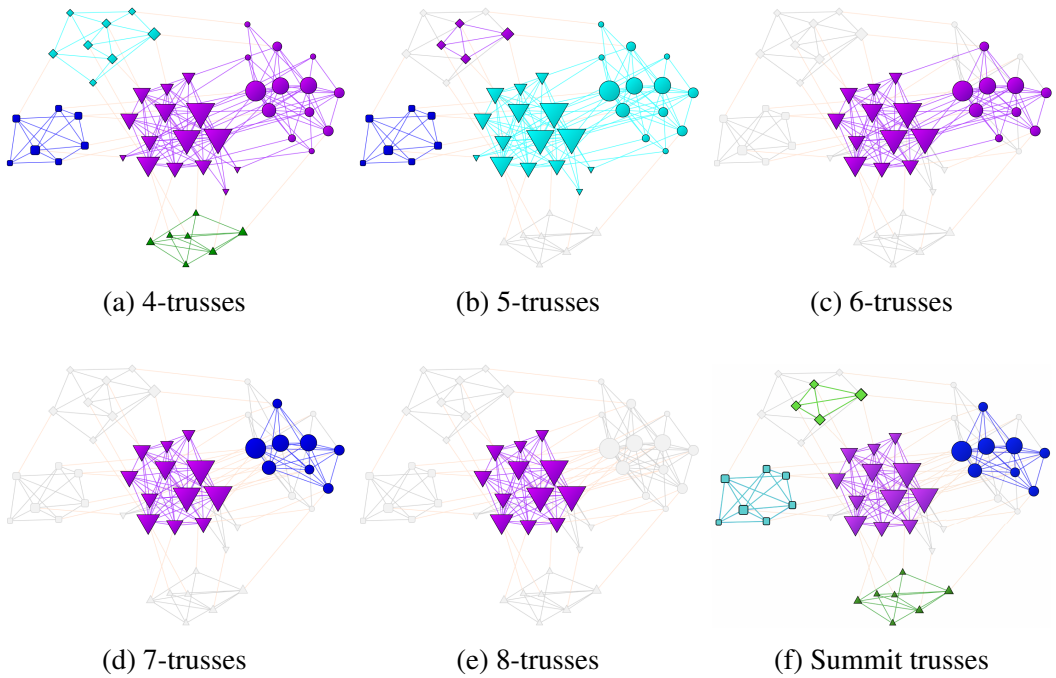
resembles their topological proximity. Since angular displacement represents topological proximity, angularly separated regions would contain groups of spatially close vertices that are more densely connected between each other than with the rest of the network (See Subsections 2.5.3 and 2.5.4 for more context) [185]. While there have been numerous other proposed definitions, one commonality is that communities should represent partitions of the network where each partition is more densely connected internally than externally. It has been observed that many real-world networks exhibit community structure, and the communities can provide a deeper understanding of the overall network. For social networks, communities might represent groups that share a high level of mutual trust or could

provide insights on how information or viruses are spread; for the world wide web, communities could provide topical categorizations of web pages; in connectomes, communities may provide insights on how the brain functions [199]. Partitioning the network into communities based on the structure of the network is a task in itself called *community detection*. Community detection will be discussed in the next section.

Certain types of networks, especially social networks, often additionally have a large number of interesting *cohesive subgraphs*, community-like objects focused only on how densely connected the subgraph are. Cliques are the most cohesive subgraph with a density equal to 1, but there are often a number of cohesive subgraphs that occur more frequently than expected yet are not complete graphs. n-cliques [9, 156], n-clans [173], n-clubs [173], and k-plexes [247] are relaxations of a clique that can be valuable to study, but are difficult to enumerate on large networks. Introduced by Stephen Seidman in 1983, the *k-core* is a maximal (single-component) subgraph in which each member is adjacent to at least k other members [246]. While k-cores are easier to enumerate, they are not necessarily cohesive. In 2008, Jonathan Cohen introduced a new type of cohesive subgraph with a goal of defining a structure that had benefits of the more cohesive subgraphs like n-cliques, n-clans, n-clubs, and k-plexes, but were easier to enumerate like the less cohesive k-cores [69].

Definition 2.2.11. A *k-truss* is a non-trivial, one-component subgraph such that each edge is reinforced by at least $k-2$ pairs of edges making a triangle with that edge. A *summit truss* is a maximal truss whose edges are not in a truss of higher support.

The trusses of Figure 2.5a are highlighted in Figure 2.6. In 2019, Cohen presented a weighted extension to trusses and defined a *strong k-truss* to be a k-truss with the additional constraint that each pair of its edges are triangle-connected. In the same paper, he also introduced *k-trapeze* which have square requirements instead of the triangle requirements of the k-truss [71].



(a) 4-trusses (b) 5-trusses (c) 6-trusses

(d) 7-trusses (e) 8-trusses (f) Summit trusses

Figure 2.6: Highlighted trusses in the graph from Figure 2.5a.
(a) 4-trusses (b) 5-trusses (c) 6-trusses (d) 7-trusses (e) 8-trusses (f) Summit trusses

The cohesive subgraphs introduced above are all independent of the type of vertices involved, as long as they satisfy the properties of the subgraph structures. Certain cohesive subgraphs are of interest by paying special attention to the type of vertices involved, which can play a large role in the network as a whole. In particular, a wide range of networks from different domains have been shown to have its highest degree vertices form a more cohesive subgraph than would be expected in a random graph with the same degree distribution. First, a set of graph-level metrics are defined that measures how densely connected the highest degree vertices are and then a graph-level metric is defined for determining if they are more densely connected than would be expected in a random graph with the same degree distribution.

Definition 2.2.12. Let $G_{>k} = (V_{>k}, E_{>k})$ with $N_{>k} = |V_{>k}|$ be the induced subgraph of a graph $G = (V, E)$ such that $\forall v \in V_{>k}$, $k_v > k$. The *rich-club coefficient* of a network is

defined as

$$\phi(k) = \frac{2|E_{>k}|}{N_{>k}(N_{>k}-1)}. \tag{2.9}$$

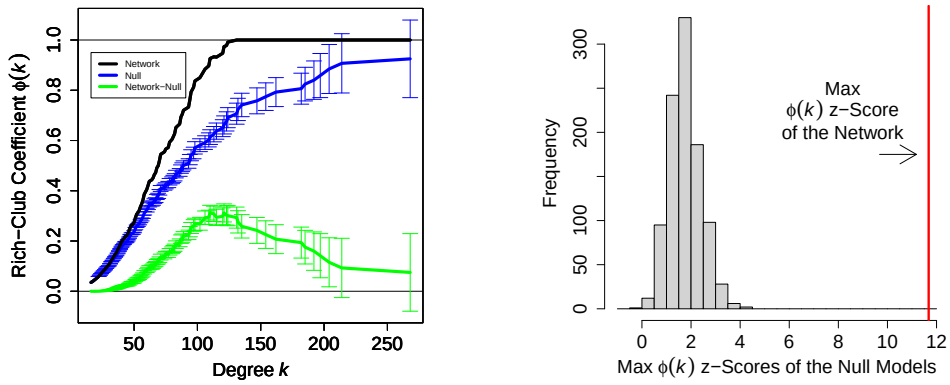
The rich-club coefficient is a metric first proposed by Shi Zhou and Raúl J. Mondragón in 2004 [289]. Because of its dependence on k, the overall network will have a set of rich-club coefficients that describe it. Some networks with right-skewed degree distributions, such as power-law distributions, create densely connected subgraphs with its highest degree vertices by random chance, as can be confirmed by a high expected number of edges in a random degree preserving multigraph (see Definition 2.5.1). In order to understand if the highest degree vertices are connected more than random, a set of null models are generated and compared against to create a single scalar graph-level metric independent of k.

Definition 2.2.13. Let $G = (V,E)$ be a network with corresponding rich-club coefficients $\phi(k)$ for all degrees k in G. Let $\phi_i^{MS}(k)$ for $i = 1,...,1000$ be the rich-club coefficients of null models, created by randomly rewiring G using the MS method (see Subsection 2.5.1). Let $\phi_{\text{mean}}^{MS}(k) = \frac{1}{1000}\sum_i \phi_i^{MS}(k)$ be the average value for each k of the null models and $\phi_{\text{sd}}^{MS}(k) = \sqrt{\frac{\sum(\phi_i^{MS}(k)-\phi_{\text{mean}}^{MS}(k))^2}{1000}}$ be the standard deviation for each k of the null models. Additionally, let $\phi^z(k) = \frac{\phi(k)-\phi_{\text{mean}}^{MS}(k)}{\phi_{\text{sd}}^{MS}(k)}$ be the normalized z-scores of $\phi(k)$ and $\phi_i^{z,MS}(k) = \frac{\phi_i^{MS}(k)-\phi_{\text{mean}}^{MS}(k)}{\phi_{\text{sd}}^{MS}(k)}$ be the normalized z-scores of each $\phi_i^{MS}(k)$. The network and each of the null models will have a peak z-score, represented as $\phi_{\text{max}}^z = \max[\phi^z(k)]$ and $\phi_{\text{max},i}^{z,MS} = \max[\phi_i^{z,MS}(k)]$, respectively. The *rich-clubness (p-value)* of the network is defined as

$$\mathcal{R} = \frac{|\phi_{\text{max},i}^{z,MS} > \phi_{\text{max}}^z|}{1000}, \tag{2.10}$$

and a network is considered to be a *rich-club network* if $\mathcal{R} < .05$ at a 5% confidence level.

In 2017, Alessandro Muscoloni and Carlo Vittorio Cannistraci proposed the notion of a rich-club p-value to create a single scalar value representing the rich-clubness of a network



(a) $\phi(k)$, $\phi^{MS}_{\text{mean}}(k)$, and $\phi(k)$- $\phi^{MS}_{\text{mean}}(k)$ vs k (b) Histogram of $\phi^{z,MS}_{\text{max},i}$ with ϕ^{z}_{max} depicted

Figure 2.7: Example rich-club network with corresponding rich-club coefficients of the network and null models (1000 MS graphs) depicted. (a) $\phi(k)$ of the network in black with the average of the null models, $\phi^{MS}_{\text{mean}}(k)$, shown in blue, errors bars represent the standard deviation, $\phi^{MS}_{\text{sd}}(k)$, for each k. For illustrative purposes, $\phi(k)$-$\phi^{MS}_{\text{mean}}(k)$ is shown in green with corresponding standard deviation represented with error bars. (b) Histogram of the 1000 $\phi^{z,MS}_{\text{max},i}$ of the null models with ϕ^{z}_{max} shown as a vertical red line. $\mathcal{R}$ is equal to the number of $\phi^{z,MS}_{\text{max},i}$ values that are to the right of the red line divided by 1000, in this case 0. Therefore, the network is very likely a rich-club network.

[183]. Definition 2.2.13 is a refinement of the one proposed by Muscoloni and Cannistraci, utilizing z-scores to account for higher variations sometimes observed for higher degrees in the null models. Additionally, a different null model is used than the one suggested in [183]. Figure 2.7a depicts $\phi(k)$ of an example network in black with the average of the null models, $\phi^{MS}_{\text{mean}}(k)$, shown in blue with errors bars representing the standard deviation, $\phi^{MS}_{\text{sd}}(k)$, for each k. For illustrative purposes, $\phi(k)$-$\phi^{MS}_{\text{mean}}(k)$ is shown in green with corresponding standard deviation represented with error bars. Figure 2.7b is a histogram of the 1000 $\phi^{z,MS}_{\text{max},i}$ of the null models with ϕ^{z}_{max} shown as a vertical red line. $\mathcal{R}$ is equal to the number of $\phi^{z,MS}_{\text{max},i}$ values that are to the right of the red line divided by 1000, in this case 0. Therefore, the network is very likely a rich-club network. Rich-clubness can be viewed as a more specific notion of assortativity, where only the connectivity of vertices beyond a certain degree are of concern. Rich-club networks have been observed in social networks

[265], biological networks [73, 261], and technological networks [289].

2.3 Community Detection

In the previous section, it was stated that communities can play an important role in real-world networks across different domains. But communities need to be detected, which is not a straight-forward task. Part of the reason is that, as stated previously, there is no single agreed upon definition for community. Additionally, different research areas view community detection from different perspectives leading to some confusion. Part of the data science process is understanding the motivation for using an analytic in order to determine which method is best to select. In 2017, Michael Schaub, Jean-Charles Delvenne, Martin Rosvall, and Renaud Lambiotte published a focused review of the different motivations that underpin community detection to provide applied network scientist a problem-driven classification of different methods. In it, they state that community detection can be broken down into four broad perspectives which can be useful in selecting the appropriate method: (1) community detection as minimization of some form of constraint violation; (2) community detection framed as a discretized analogue of data clustering, in which densely knit groups of vertices are to be found; (3) community detection aiming to identify structurally equivalent vertices in a network, leading to notions such as stochastic block models; and (4) community detection looking for simplified descriptions of the dynamical flows occurring on the network [241]. The types of communities introduced in the previous section most closely align with the data clustering perspective. Community detection approached from this perspective will be introduced first and subsequently compared to approaches from other perspectives. Community evaluation metrics will also be discussed.

2.3.1 Clustering Perspective of Community Detection

Some of the early proposed methods to find partitions of densely knit groups of vertices first provided a definition of community and then used the definition in an algorithm searching for and optimal partition. For example, in 2004, Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi proposed a divisive hierarchical clustering algorithm that incrementally removed weakest weighted edges if doing so would not create components that did not meet their community definition [222]. The weakest edges were determined by computing an edge-level metric called the modified edge-clustering coefficient similar to the local clustering coefficient discussed in the previous section. The resulting algorithm had a runtime of $O(N^2)$. In 2008, Yanqing Hu, Hongbin Chen Peng Zhang, Menghui Li Zengru Di, and Ying Fan proposed a heuristic algorithm that assigns vertices to communities and iteratively reassigns them to better communities until a steady state is reached. The final solution ensures the resulting communities satisfied their new definition of community [125] with a runtime of $O(N^2)$. The algorithm developed by Hu et al. is similar in nature to an algorithm proposed by Usha Nandini Raghavan, Réka Albert, and Soundar Kumara in 2007 called *label propagation (Label Prop)* [223]. Label propagation starts with every vertex in its own community, and then iteratively reassigns them to the community "label" occurring with the highest frequency among its neighbors, with ties broken uniformly at random. The steady state solution of label propagation would additionally meet the definition of community proposed by Hu et al. [125] and in practice has near linear runtime, $O(N)$.

The most popular approach used for finding densely connected communities is maximizing a metric called modularity. Modularity is a term introduced by Mark Newman and Michelle Girvan in 2004 [201].

Definition 2.3.1. Let $G = (V, E, \omega)$ be an undirected graph with M communities such that

each of its vertices $v \in V$ is assigned to a community $c_v \in \{C_i\}_{1 \leq i \leq M}$. *Modularity* is defined as

$$Q_\omega = \frac{1}{2W} \sum_{u,v \in V} \left[\omega(u,v) - \frac{s_u s_v}{2W} \right] \delta(c_u, c_v), \tag{2.11}$$

where δ is the Kronecker delta. For an unweighted graph, modularity is equal to

$$Q = \frac{1}{2L} \sum_{u,v \in V} \left[A_{uv} - \frac{k_u k_v}{2L} \right] \delta(c_u, c_v). \tag{2.12}$$

For an unweighted graph the left hand term is equal to $\frac{1}{2L} \sum_{uv} A_{uv} \delta(c_u, c_v)$ measuring the fraction of edges that fall within the communities [63]. Since this term is equal to 1 in the trivial case where all vertices belong to a single community, the expected value of the same quantity in the case of a random degree-preserving multigraph is subtracted from it to obtain a useful metric (see Definition 2.5.1, but note that it has been assumed that when L is sufficiently large the '-1' in the denominator can be ignored). Modularity can range from [-1,1]. In practice, $Q \geq .3$ of an unweighted graph means that a network has significant community structure [63, 201].

Modularity was originally introduced as a way to quantify the strength of community structure at each split of a network for a divisive hierarchical clustering algorithm, specifically one that removed edges with the highest re-calculated edge betweenness metric at each step [201]. This method was shown to produce meaningful communities but was computationally expensive – $O(L^2N)$ or $O(N^3)$ for extremely sparse networks – making it difficult to run on large networks [222]. Mark Newman proposed another algorithm in 2004 that instead optimized the modularity term itself. While finding the true optimal modularity of a network is an NP-hard problem, Newman took a "greedy" approach using agglomerative hierarchical clustering [195] creating an algorithm that would run in $O((L+N)N)$ or $O(N^2)$ for extremely sparse networks. In the same year, Aaron Clauset, Mark New-

man, and Cristopher Moore proposed an algorithm, termed the *Fast Greedy* algorithm, that solves the same greedy optimization as [195] but exploits shortcuts and uses sophisticated data structures to improve runtime to $O(N \log^2 N)$ [63].

Arguably the most popular modularity maximization algorithm, proposed by Vincent Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre in 2008, is called the *Louvain method* which has been shown to run in linear time, $O(N)$, for extremely sparse networks [34]. The Louvain algorithm is a heuristic algorithm divided into two phases which are repeated iteratively. Initially, all vertices $v \in V$ in an assumed weighted network $G = (V, E, \omega)$ are assigned a different community. Then, for each vertex v, the neighbors u of v are considered and evaluated based on the gain in modularity that would take place by removing v from its community and placing it in the community of u. If v is a vertex in an isolated community that is moved into community C, then the gain in modularity can be computed as

$$\triangle Q_\omega = \left[\frac{\Sigma_{\text{in}} + 2s_{v,\text{in}}}{2W} - \left(\frac{\Sigma_{\text{tot}} + s_v}{2W}\right)^2\right] - \left[\frac{\Sigma_{\text{in}}}{2W} - \left(\frac{\Sigma_{\text{tot}}}{2W}\right)^2 - \left(\frac{s_v}{2W}\right)^2\right], \tag{2.13}$$

where $\sum_{in}$ is the sum of the weights of the edges inside C, $\sum_{tot}$ is the sum of the weights of the edges connected to the vertices in C, s_v is strength of vertex v, $s_{v,\text{in}}$ is the sum of the weights of the edges from v to vertices in C and W is the sum of the weights of all the edges in the network. A similar expression is used in order to evaluate the change of modularity when v is removed from its community. The vertex v is placed in the community for which the gain is maximized (with a tie breaking rule), but only if the gain is positive. The process is repeated sequentially for all vertices until no further improvement can be achieved, completing the first phase. The second phase of the algorithm consists in building a new graph whose vertices are now the communities found during the first phase. Edges between vertices of the same community create self-loops in the new graph and the weight of the edges

between the new vertices is equal to the sum of the weights of edges between the vertices in the corresponding communities. Phase 1 and Phase 2 can be iteratively repeated, with the final result being the communities from the iteration that yields the highest modularity.

Despite its popularity, there are number of drawbacks to using modularity maximization algorithms, including the Louvain algorithm. The first is that modularity has a well-known *resolution limit* in which communities with less than $\sqrt{W\xi/2}$ total edge weight may fail to be resolved, where ξ is the maximum weight of an intercommunity edge. Santo Fortunato and Marc Barthelemy first proved in 2007 that this occurs for unweighted networks (where $W = L$ and $\xi = 1$) [101] and Jonathan Berry, Bruce Hendrickson, Randall LaViolette, and Cynthia Phillips extended the result to weighted network in 2011 [30]. Modularity is also extremely sensitive to even individual connections, which is problematic since many real graphs, particularly in biology and in the social sciences, are reconstructed through error prone experiments and surveys. If two small subgraphs happen to be connected by a few false edges, modularity will put them in the same cluster, inferring a relationship between entities that in reality may have nothing to do with each other [100]. Relatedly, Marek Ciglan, Michal Laclavík, and Kjetil Nørvåg demonstrated in 2013 that sometimes the detected communities may be made up a single or a few very large communities that contain a majority of the vertices in the network [60]. This problem was further investigated by Paul Wagenseller, Feng Wang, and Weili Wu in 2018 [269]. Additionally, it has been shown that it is particularly hard to detect communities in sparse networks by using modularity optimization since the expected maximum modularity for a random graph increases when the average degree decreases [100]. There can also be a large number of high modularity partitions close to the maximum that are not structurally similar [114]. This issue, called *extreme degeneracy*, is quite problematic since finding the true maximum is NP-hard and heuristics are used that are often non-deterministic meaning they could provide very different partitions for the exact same graph given a reordering of the vertices. But even if an

algorithm was able to maximize modularity properly, modularity has non-local behavior in which small modifications to graph can cause vertices to have very different community assignments [39]. Additionally, purely random graphs can generate high modularity scores [230]. Modularity also favors equal sized partitions, which was proven by Huiyi Hu, Thomas Laurent, and Mason Porter, and Andrea Bertozzi in 2013 [124]. A number of techniques have been proposed to account for these shortcomings [16, 55–57, 100]. One approach is to weight the edges in a network in a way that incorporates structural information and use the weighted network in the modularization maximization algorithm.

2.3.2 Derived Edge Weights for Modularity Maximization

In 2011, Jonathan Berry, Bruce Hendrickson, Randall LaViolette, and Cynthia Phillips demonstrated that edge weights could be derived that would allow for smaller clusters to be resolved compared to its corresponding unweighted graph. Specifically, they demonstrated that in a weighted graph, communities with less than $\sqrt{W\xi/2}$ total edge weight may fail to be resolved, where ξ is the maximum weight of an intercommunity edge. Therefore, if edge weights could be derived such that ξ is small enough, then there would no longer be a limit on the size of a community that could be detected [30]. In the same year, Alireza Khadivi, Ali Ajdari Rad, and Martin Hasler also demonstrated that appropriately chosen edge weights such that the total sum of the edge weights leaving each community was much smaller than the total edge weight inside the corresponding communities would also mitigate problems with extreme degeneracy [131].

A number of edge weights, which can be considered edge-level metrics or measures, have been used to improve community detection results when used as a pre-weighting step. The metrics can be categorized as local, semi-local, and global.

Definition 2.3.2. A *local* metric depends only on a vertex or edge and its direct neigh-

bors. A *semi-local* metric depends only on a vertex or edge and its neighbors within a small pre-defined neighborhood size greater than 1. A *global* metric requires knowledge of potentially of all of the vertices or edges in a graph to be computed.

Edge betweenness is a global metric that was first used in 2002 by Michelle Girvan and Mark Newman in a divisive hierarchical clustering algorithm [111]. Edge betweenness measures the fraction of shortest paths that pass through the edge and is a generalization of the vertex-level betweenness centrality metric introduced by Linton Freeman in 1977 [105]. In 2004, Santo Fortunato, Vito Latora, and Massimo Marchiori proposed using another global metric, information centrality, as an improvement over edge betweenness in highly mixed networks [103]. Both edge betweenness and information centrality rely on computing the shortest paths across all vertices in the network, which is computationally expensive. In the same year, Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi proposed using a local metric that could be generalized to be a semi-local metric, called the modified edge clustering coefficient which was less computationally expensive than the methods previously proposed [222]. While the modified edge clustering coefficient accounted for short cycles, it did not perform as well in sparse networks where few short cycles exist. To mitigate this issue, I. Vragović and Enrique Louis propose using a semi-local loop coefficient which would take loops of any length into account in a trade-off with computational complexity [267]. In 2011, Jonathan Berry, Bruce Hendrickson, Randall LaViolette, and Cynthia Phillips proposed using a semi-local measure called neighborhood coherence that take cycles of length 3 and 4 into account in the single metric [30]. In the same year, Alireza Khadivi, Ali Ajdari Rad, and Martin Hasler proposed balancing the global edge betweenness metric with a local metric called the common neighbor ratio [131]. In 2012, Pasquale De Meo, Emilio Rerrara, Giacomo Riumara, and Angela Ricciardello developed the κ-path semi-local edge metric which uses

fixed length random walks to efficiently compute the importance of edges [83]. In 2013, Marek Ciglan, Michal Laclavík, and Kjetil Nørvåg proposed using a local metric to account for degree assortativity [60]. In 2015, Haiyan Zhang, Chenxi Zhou, Xun Liang, Xi Zhao, and Yaping Li proposed a semi-local metric that truncates the SimRank algorithm, creating a natural balance between local and global information about a network [286]. In 2017, Alessandro Muscoloni, Josephine Maria Thomas, Sara Ciucci, Ginestra Bianconi, and Carlo Vittorio Cannistraci propose a more involved set of measures derived by finding a coalescent embedding into a hyperbolic circle using machine learning techniques. As an initial step, Muscoloni et al. pre-weight the vertices in the network with a local repulsion attraction metric and the global edge betweenness metric for comparison [189]. Also in 2017, Heman Shakeri, Pietro Poggi-Corradini, Nathan Albin, and Caterina Scoglio investigated using a global analysis called loop modulus which incorporates the probability distribution on all cycles in the graph [250]. In 2020, Behnaz Moradi-Jamei, Heman Shakeri, Pietro Poggi-Corradini, Michael Higgins proposed a method called renewal non-backtracking random walks (RNBRW) which create global edge weights representing the probability of an edge participating in a cycle of any length [176].

2.3.3 Other Perspectives of Community Detection

As mentioned in the introduction to this section, there are other notions of community detection than the clustering perspective that was discussed in Subsection 2.3.1.

Graph partitioning is closely related to community detection, but instead of having a goal of maximizing the internal density of each community, its focus is on minimizing some form of constraint violation. Unlike graph clustering, graph partitioning typically requires a target number of partitions to be specified with an objective, such as minimizing the external degree of the partitions while creating partitions that are as balanced as

possible. A common approach to graph partitioning is to use eigenvectors, first proposed by Brian Kernighan and Shenin Lin 1970 [129]. In 2013, Huiyi Hu, Thomas Laurent, and Mason Porter, and Andrea Bertozzi reformulate modularity optimization into minimizing an energy functional that consists of a total variation term and an ℓ_2 balance term [124].

Stochastic block models or *SBMs* can be used to detect structurally equivalent vertices in a network: vertices are similar if they share the same kind of connection patterns to other vertices. While graph clustering is typically used to find assortative structure in networks, SBMs are able to find more general connectivity patterns than typical graph clustering algorithms, such as core-periphery structure and disassortative structure [102]. SBMs work best with Poisson or Bernoulli degree distributions, which is dissimilar to many real-world networks. In 2011, Brian Karrer and Mark Newman introduced *degree-correlated block models* to preserve the degrees of vertices [127]. In 2016, Vince Lyzinski, Minh Tang, Avanti Athreya, Youngser Park, and Carey Priebe, investigated community detection techniques with *hierarchical stochastic block models (hsbm)* [159]. A planted partition model is a very specific case of the general stochastic block model, in that the intra-community edge probabilities are all equal to the same value p_{in} and the inter-community edge probabilities are all equal to the same value p_{out}. In 2016, Mark Newman proved that optimizing modularity is equivalent to maximizing the likelihood that the planted partition model reproduces the network [198].

The last perspective of community detection focuses on the processes that take place on a network and is closely related to the clustering perspective. Unlike the other three perspectives that focus on the structure of the network, this perspective views communities as dynamical building blocks of the network. In 2005, Pascal Pons and Matthieu Latapy proposed a method falling under this perspective that uses random walks, called *Walktrap* [218]. In 2007, Martin Rosvall and Carl Bergstrom proposed a method called *Infomap* that maps the information flow of the network [232]. While the focus of such algorithms is not

on the structure of the network, densely knit groups within the network would cause, for example, a random walker to be trapped for a significant amount of time [241].

2.3.4 Community Evaluation

Community detection algorithms can be evaluated using both generated benchmark graphs or real-world networks. Often times, ground truth communities of real-world networks are not available to compare performance against. Even if they are, sometimes they are not reliable or not expected to correspond to the structural topology. In these cases, a number of measures are used to investigate if the communities strike a balance between two separate properties: cohesion and separation. A community is considered to have high *cohesion* if it has a relatively high number of internal edges. A community is considered to have high *separation* if it is connected to other communities with relatively few edges.

Definition 2.3.3. Let C be a community of a graph $G = (V, E)$. The *internal edge density* is the ratio between the number of internal edges of C and the number of all possible internal edges,

$$\delta_C^{int} = \frac{k_C^{int}}{N_c(N_c - 1)} \tag{2.14}$$

and the *conductance* of the community is defined as

$$\Phi_C = \frac{k_C^{ext}}{k_C^{int} + k_C^{ext}}. \tag{2.15}$$

Internal edge density is a community-level feature often used to measure the cohesiveness of a community, with higher values desired. Conductance is a popular community-level metric that takes both the cohesion of the community and the separation of the community from the rest of the network into account, with lower values desired. A more complete list of definitions can be found in Santo Fortunato and Hric's 2016 User Guide

to Community Detection in Networks [102]. In 2018, Paul Wagenseller, Feng Wang, and Weili Wu argue that the size of the communities detected by an algorithm matters as a metric for evaluation, since some algorithms can produce large communities that make up a majority of the vertices in the network [269]. They suggest breaking down how many communities fall into the sizes shown in Table 2.1. These sizes correspond to community sizes researched by anthropologist Robin Dunbar [90].

Table 2.1: Community sizes inspired by research from anthropologist Robin Dunbar.

Size	Closeness
4-5	Support clique
6-15	Sympathy group
16-50	Close friend
51-150	Casual friend
151-500	Acquaintance
501-1500	Just a face

In the case that ground truth communities exist or generated benchmark graphs are used, most network scientists evaluate detected communities using normalized mutual information which is a metric introduced by Ana Fred and Anil Jain in 2003 [10] and tailored for community detection in 2005 by Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas [82]. Ground truth communities are considered the true communities, although if they come from a real-world network, it should be kept in mind that they might not align with the structural topology.

Definition 2.3.4. *Normalized mutual information (NMI)* is a metric used to determine how much information the true communities, denoted by $\mathcal{C}$, and detected communities, denoted by $\mathcal{D}$, have in common:

$$\mathrm{NMI}(\mathcal{C},\mathcal{D}) = \frac{2I(\mathcal{C};\mathcal{D})}{H(\mathcal{C})+H(\mathcal{D})}, \tag{2.16}$$

where $H(\mathcal{C})$ is the entropy of $\mathcal{C}$ and $I(\mathcal{C};\mathcal{D})$ is the mutual information between $\mathcal{C}$ and $\mathcal{D}$.

Note that $\text{NMI}(\mathcal{C}, \mathcal{D}) = 1$ if and only if the true communities are identified perfectly, otherwise the value is less than 1. If the detected communities and true communities are independent, the metric has expected value of 0. More information about NMI can be obtained in [200].

Another metric measures how far off the number of communities detected by the algorithm is from the true number of communities, which is a useful network next to NMI.

Definition 2.3.5. Let $\mathcal{C}$ denote the true communities and $\mathcal{D}$ denote the detected communities. Then, *community count error (CCE)* is defined as

$$\text{CCE}(\mathcal{C}, \mathcal{D}) = \frac{|\mathcal{D}|}{|\mathcal{C}|}, \tag{2.17}$$

where $|\mathcal{D}|$ represents the number of communities in $\mathcal{D}$.

Note that $\text{CCE}(\mathcal{C}, \mathcal{D}) = 1$ if and only if the number of detected communities equals the number of true communities. CCE is less than 1 when fewer communities are detected than there are true communities. If CCE is greater than 1, more communities are detected than there are true communities. The community count error is used in a comparative analysis of community detection algorithms completed in 2016 by Zhao Yang, René Algesheimer, and Claudio Tessone [283].

2.4 Anomaly Detection

Community detection algorithms require that every vertex be assigned to a single community, even if some vertices are not good fits for any community. This has the possibility of corrupting the communities detected by certain algorithms. As noted earlier, modularity maximization algorithms can be sensitive to even individual connections [100] and the real-world is full of spurious connections and even spurious entities. Xiaowei Xu,

Nurcan Yuruk, Zhidan Feng, and Thomas Schweiger introduce a novel community detection algorithm called SCAN in 2007 that additionally detects outlier vertices that are not strongly connected to any community and hub vertices which bridge communities [278]. The algorithm utilized a local edge-level structural similarity metric which considered two vertices to be similar if the number of common neighbors shared between their vertices normalized by the geometric mean of their degree was relatively large. Specifically, the algorithm required a threshold on the similarity score to be considered a strong connection and a number of strong connections to be considered a core member of a community. If the vertex did not have enough strong connections to meet the definition of a core member of a community, it was considered either an outlier or a hub depending on if their edges connected to one or more core communities. The downside to this approach is that the two parameters need to be specified by the user. In 2010, Heli Sun, Jianbin Huang, Jiawei Han, Hongbo Deng, Peixiang Zhao, and Boqin Feng introduced a modified version of the algorithm called gSkeletonClu that automatically determines the optimal threshold on the similarity scores [255].

Determining outlier vertices and hub vertices compared to core-community vertices is one approach to detecting anomalies in a graph. In their 2015 survey on graph-based anomaly detection, Leman Akoglu, Hanghang Tong, and Danaiwould Koutra categorize this type of anomaly detection algorithm as a community-based anomaly detection method in static graphs [7]. While Akoglu et al. also discuss anomaly detection on dynamic graphs, the focus this dissertation is anomaly detection on static graphs that do not change over time. Akoglu et al. provide a definition for anomaly detection on static graphs.

Definition 2.4.1. Given a static graph, the goal of *anomaly detection* is to find the vertices and/or edges and/or substructures that are "few and different" or deviate significantly from the patterns observed in the graph.

They categorize anomaly detection algorithms on static graphs as structure-based methods and community-based methods. In 2012, Qi Ding, Natallia Katenka, Paul Barford, Eric Kolaczyk, and Mark Crovella introduce a structure-based method motivated by communities that determines vertices with the lowest local clustering coefficients to be anomalous in terms of intrusion detection [85]. They relate vertices with the lowest clustering coefficient (and highest betweenness) to be cut-vertices and deem them "antisocial" which they hypothesized would be an important indicator of intrusion. This approach worked better than determining antisocial behavior based on communities detected using modularity due to the issues with modularity discussed in Section 2.3. In 2015, Naeimeh Laleh, Barbara Carminati, and Elena Ferrari propose a risk measure based on topological attributes in online social networks to detect Sybils [139]. The topological attributes used by Laleh et. al are similar in concept to the measures used by Ding et al. in that they give an indication of community. Specifically, triangle counts and ratios of degree to triangle counts are used which are more likely to be found within communities.

Another example of a structure-based method is ODDBALL which was proposed by Leman Akoglu, Mary McGlohon, and Christos Faloutsos in 2010 [6]. Akoglu et al. derive a number of attributes for each vertex based on structural characteristics of the 1-hop neighborhood around the vertex called an egonet and observe expected patterns of how those attributes should relate to each other. Specifically, they focus on relationships between pairs of egonet attributes and define anomalies a graph to be vertices that have distinctly different patterns in their paired attributes. While they were able to observe expected behavior in specific pairs of attributes, the authors note that another approach would be to feed generated attributes into traditional outlier detection methods for clouds of points using all attributes. The techniques used to detect anomalies on point cloud datasets can be categorized as Classification Based, Clustering Based, Nearest Neighbor Based, Spectral, Information Theoretic, and Statistical [52]. Akoglu et al. noted the importance of analyz-

ing attributes in pairs and the importance of interpretability, which should be considered when choosing which traditional outlier detection method to use. Luca Becchetti, Carlos Castillo, Debora Donato, Stefano Leonardi, and Ricardo Baeza-Yates proposed a method in 2006 which extracted attributes from a graph and used them in traditional point cloud data algorithms to detect web spam [23]. In 2018, Dieudonne Mulamba, Indrajit Ray, and Indrakshi Ray create a Sybil detection method based on the classification of users based on structural attributes extracted from the graph [177]. Mulamba et al. use traditional classification methods such as AdaBoost, KNN, and Random Forest to classify users as malicious or benign.

Another type of structure-based methods is proximity based, in which the graph structure is used to quantify closeness of vertices in the graph to identify associations. Such approaches are useful to find vertices with close connections to known anomalous vertices. Lastly, another type of community-based anomaly detection is anomalous subgraph detection [251]. Navraj Singh, Benjamin Miller, Nadya Bliss, and Patrick Wolfe define anomalous subgraph detection as a binary hypothesis test and propose using sparse principal component analysis as a solution.

2.5 Network Models

Network models can be used to better understand the significance of the different structural patterns in real-world networks, to provide possible explanations of why certain structural patterns are common and how they are affected by changing certain parameters, or to help test algorithms on their ability to achieve an objective on graphs similar to real-world networks. In previous sections, different random graph models have been used as null models to understand structural patterns in real-world networks, like determining if a network has community structure using modularity or determining if a network is a small-world

network or a rich-club network. Another class of network models are *growing models* that explore the mechanisms by which networks are created. Unlike the models previously mentioned, these models may provide an explanation as to why certain structural patterns are common in the real-world. The most well-known example of this type of network model is called the *Barabási-Albert (BA) model* introduced in 1999 by Albert-László Barabási and Réka Albert [20] at the start of the field of network science (a related model was first studied by Derek Price in the 1970s [221]). With the BA model, graphs are generated using a *preferential attachment* mechanism that provides a possible explanation for why scale-free networks are prevalent in real-world settings. The structure of such graphs are for the most part a result of a succession of random processes in which the most popular vertices are preferred by later vertices added in the model. A contrasting mechanism of network formation is called *structural optimization*. One such model will be introduced that balances the popularity dimension from preferential attachment with a similarity dimension. Lastly, network models can be used to determine the effectiveness of algorithms build for specified tasks. For example, graphs built with different community structure paradigms can be used to determine how well algorithms are able to detect their communities with different parameters settings and also how well they are able to scale to larger and larger graphs.

2.5.1 Random Graphs

A *random graph* is a network model which is generated in a random way aside from having selected properties fixed. The simplest random graph sets only the number of vertices N and the number of edges L. Specifically, L distinct pairs of vertices are chosen uniformly at random from all possible pairs of vertices and connected with an edge. Graphs generated from this type of model are typically referred to as *Erdős -Rényi (ER) graphs* with parameters N and L after Pál Erdős and Alfréd Rényi who published a series of papers

on their properties from 1959-1968 [93].

A related model was introduced by Edgar Gilbert in 1959 in which the number of vertices N is still set, but instead of fixing the number of edges L, each pair of vertices is connected with equal probability p [110]. These models, typically still referred to as ER graphs but with parameters N and p, are easier to study probabilistically, with the ability to derive expected properties of the model. For example, the expect characteristic path length is $\mathcal{L}_{ER} = \frac{\log N}{\log \langle k \rangle}$ and the expected global clustering coefficient is $gcc_{ER} = \frac{\langle k \rangle}{N}$. Comparing a network's characteristic path length and clustering coefficient to the values expected from ER graphs are used to determine if it is a *small-world network*, as mentioned in Section 2.2 [126, 191, 273]. The average number of edges, the average degree, the expected degree distribution, the average number of components, and the average diameter, in addition to other properties can all be calculated probabilistically.

One of the main properties of real-world networks that is not reflected in ER graphs relates to their degree distribution. Instead of having a distribution that are close to a power-law distribution, the degree distributions of ER graphs have Poisson distributions particularly as the sizes of the graphs increase. The degree distribution is an important property of a network, so being able to understand how the degree distribution affects other properties is of interest.

The *configuration model (CM)*, termed by Béla Bollobás in 1980 [36], is a type of random graph network model in which the degree sequence is fixed. Therefore, if the degree sequence creates a power-law distribution, the ensemble of graphs generated using the configuration model will also have power-law distributions by construction, though with a caveat. Given a network, the configuration model generates a graph by cutting each edge in half as shown in Figure 2.8a and then rejoining each stub with equal probability. Alternatively, a degree sequence can be provided and vertices with stubs equivalent to the degree sequence will be created. The procedure of randomly rejoining stubs with equal

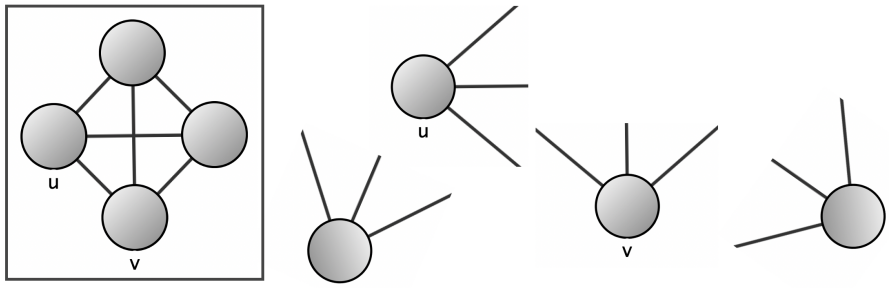
probability allows for the resulting graphs to have self-loops and multiple edges, and therefore multigraphs instead of simple graphs, particularly for small dense graphs as shown in Figure 2.8b. As the size of the graph grows, self-loops and multiple edges are less likely to be prevalent, though as will be discussed they will still often occur. Graphs generated using the CM are called *random degree-preserving multigraphs*. The signature property of the configuration model is the expected number of edges between two specified vertices u and v in a resulting multigraph which is often used as a reference.

Definition 2.5.1. Let $G = (V, E)$ be a multigraph generated using the configuration model. The expected number of edges between vertices $u, v \in V$ is equal to $k_u k_v / (2L - 1)$.

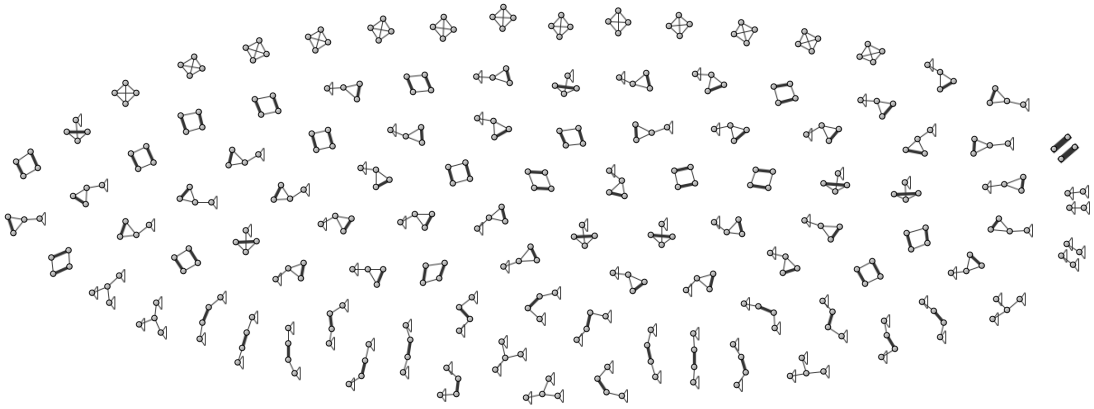
Let $u, v \in V$ be vertices in a multigraph generated using a configuration model $G = (V, E)$. In its generation, let e_{v_1} be one of the stubs of vertex v. e_{v_1} has equal probability of connecting to any of the remaining stubs in the graph. There are $2L - 1$ remaining stubs in the graph, with k_u of them being connected to vertex u. Therefore, the probability that e_{v_1} is connected to one of the stubs connected to vertex u is $k_u / (2L - 1)$. Since there are k_v stubs around vertex v, each contributes equally to the expected number of edges between the two vertices in Definition 2.5.1. As the size of a graph grows, the number of self-loops and multiple edges is expected to decrease, but it is dependent on the properties of the degree sequence.

The expected number of edges between two vertices in the configuration model is most famously used in the modularity metric, though it is assumed that when L is sufficiently large the '-1' in the denominator can be ignored (see Definition 2.3.1) [63, 201]. Other properties that can be calculated probabilistically using a configuration model are the expected clustering coefficient and expected assortativity coefficient [193, 202].

Many real-world networks are scale-free, with many low degree vertices and a few vertices with very high degrees. In these cases, the configuration model tends to create



(a) Stubs of a complete graph with $N = 4$



(b) 100 random graphs built using the configuration model of a complete graph with $N = 4$

Figure 2.8: The configuration model (CM) can be used to generate randomized degree preserving versions of a network, but resulting graph is a multigraph that may have self-loops and/or multiple edges particularly for small and dense graphs. (a) Stubs of a complete graph with $N = 4$ (b) 100 random multigraphs built using the configuration model of a complete graph with $N = 4$

multiple edges amongst the highest degree vertices even if the network is large and sparse [170]. For example, in 2004, Sergei Maslov, Kim Sneppen, and Alexei Zaliznyak showed that for a network with 12,772 edges built from the internet, the two highest degree vertices with $k_v = 1,458$ and $k_u = 750$ would have an expected 43.5 edges between them [163]. Unfortunately, creating a random graph without self-loops or multiple edges via the stubs approach is difficult. Rejecting edges that would create self-loops or multiple edges could lead to many edges not being able to be connected. Therefore, a different approach was proposed by Sergei Maslov and Kim Sneppen in 2002 [162].

Given a network, rewired versions of the network can be created by randomly switching

pairs of edges, so long as no multiple edges or self-loops are created. Specifically, in a given iteration, suppose the edges (u_1, v_1) and (u_2, v_2) are selected, as shown in Figure 2.9. The edges (u_1, v_1) and (u_2, v_2) would be removed and replaced with edges (u_1, v_2) and (u_2, v_1) so long as it would not create multiple edges or self-loops. This is repeated a certain number of times, typically a multiple of the number of edges in the graph, for example $100L$ times. Note that all graph generated using this approach and starting with a complete graph will yield the same complete graph. If starting from a degree sequence and not a network, a random graph is generated using the degree sequence for initialization. While it is not possible to derive expected values of different properties for such a model, it is often useful to generate a large sample of such graphs to understand characteristics of random graphs with a specified degree sequence. For example, this technique is used to determine if a network is a *rich-club network* (see Definition 2.2.13). It is also possible to make additional constraints on the types of edges to select to ensure additional properties exist in the resulting graphs [141, 163, 183, 290]. Graphs using the *Maslov-Sneppen (MS) method* are often called *rewired networks* or *MS graphs*.

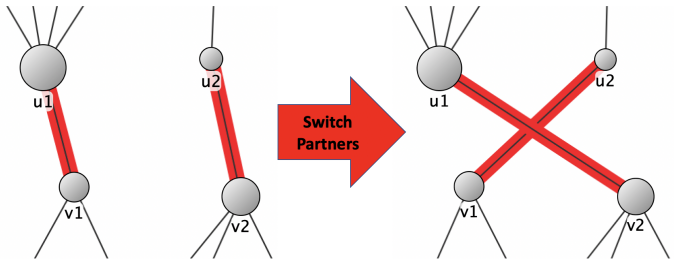


Figure 2.9: In one iteration of generating a Maslov-Sneppen (MS) rewired network, two edges are selected at random. If switching the partners among the two edges would not create self-loops or multiple edges, the partners are switched.

2.5.2 Benchmark Graphs with Planted Partitions

A commonly agreed upon notion is that random graphs do not have community structure [19]. To understand community structure or build graphs to be used to evaluate the performance of community detection algorithms, community structure needs to be part of the network model. One approach is to plant partitions representing communities in a graph based on a set of parameters, which will be discussed in this section.

The first benchmark for community detection was proposed by Michelle Girvan and Mark Newman in 2002 [111], referred to as *GN graphs*. GN graphs consist of 128 vertices, each with an average degree of 16, divided into four communities of 32 vertices each. The amount of mixing between communities can be varied by modifying a parameter corresponding to the conductance of the communities (see Definition 2.3.3) or mixing rate of the vertices (see Definition 2.2.10). All vertices have equal degree and each community is the same size, which is not a good proxy for real networks with community structure.

Proposed by Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi in 2008, *LFR graphs*, generalize different aspects of the GN graphs to allow for different numbers of vertices with degree distributions and community size distributions that can be right-skewed as observed in many real-world networks [141]. To generate LFR graphs, users can specify the number of vertices N, the average degree $\langle k \rangle$, the maximum degree k_{max}, a degree distribution scaling parameter γ, a community size distribution scaling parameter β, the minimum and maximum community sizes, denoted as c_{min} and c_{max} respectively, and the mixing parameter μ in order to perform controlled evaluations of algorithms. The degree distribution and community size distribution are both assumed to follow power laws, with exponents γ and β respectively. The mixing parameter μ is an average of the mixing rate for all the vertices in the graph (see Definition 2.2.10), but as the size of the graph increases, the standard deviation becomes small and most vertices have a mixing rate nearly equal to

the set μ. Since the μ is the same for all vertices, this value is also equal to the conductance of each generated community (see Definition 2.3.3). Typically, μ is varied to study performance under several mixing conditions. LFR networks are planted partition models [102, 198], and it has additionally been shown that LFR networks are special versions of degree-correlated stochastic block models [127].

An LFR graph is generated by first creating a degree sequence with the degree distribution scaling parameter γ that has an average degree $\langle k \rangle$ and maximum degree k_{max}. Then the configuration model (see Subsection 2.5.1) is used to create a random graph with such a degree sequence. Community sizes are then determined using the community size distribution scaling parameter β, c_{min}, and c_{max}. Vertices are initially homeless, then are assigned to communities iteratively so long as the community size exceeds the internal degree of the vertex. Once all vertices are assigned to a community, the network is rewired (See the Maslov-Sneppen method in Subsection 2.5.1) such that the degrees of all vertices stay the same, but the internal and external degrees are modified to match μ as best as possible.

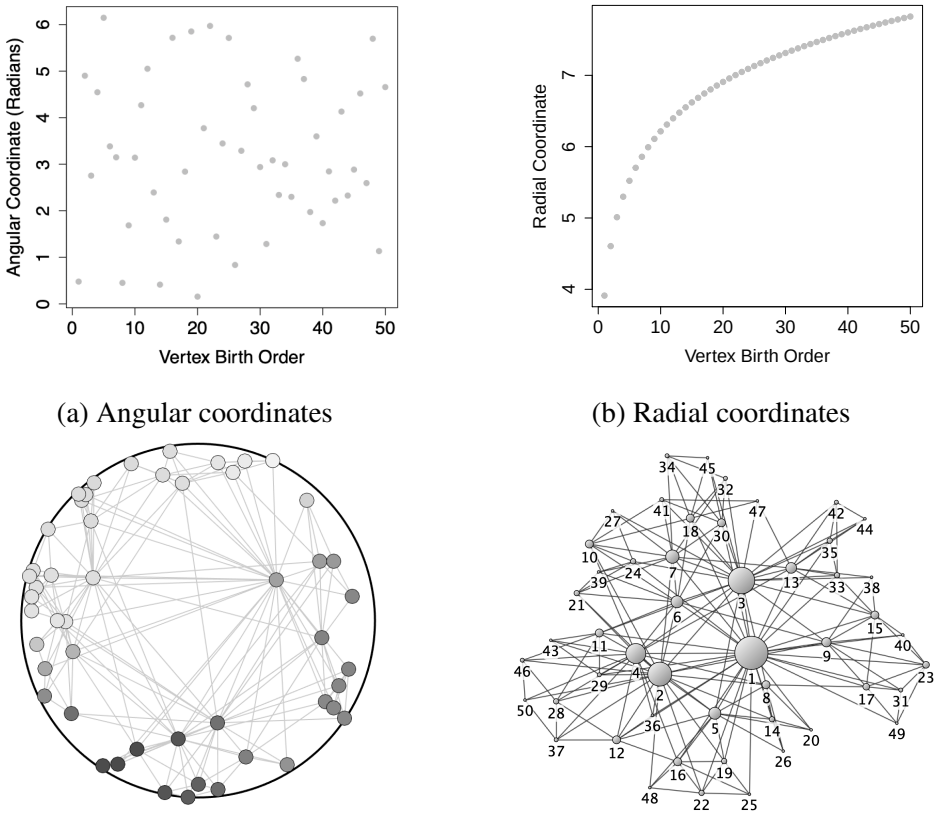
2.5.3 Popularity×Similarity Graphs

The random graphs in Section 2.5.1 and graphs generated with the BA model [20] do not explain the high level of clustering found in many real-world networks. In 2012, Fragkiskos Papadopoulos, Maksim Kitsak, Ángeles Serrano, Marián Boguná, and Dmitri Krioukov proposed a structural optimization growing network model that provides a possible explanation. Their model balances two competing values: popularity and similarity [209]. The concept of popularity is the main force behind the preferential attachment mechanism – newer vertices added to a model prefer to connect to more popular vertices with higher degree than less popular vertices with lower degree. Papadopoulos et al. introduce another concept called similarity, which can be modeled using a measure like cosine simi-

larity, to determine the angular distance of vertices randomly placed on a circle. Laying out vertices in a circle is similar in concept to how a ring lattice graph is created [273]. Instead of the vertices being laid out equi-distance apart, new vertices are added one by one at an angular coordinate chosen uniformly at random, as shown in Figure 2.10a. Popularity can be modeled by mapping birth time to a radial coordinate r_t such that $r_t = \ln Nt$ where N is the number of vertices in the final generated graph, making the oldest vertices more popular than newer vertices as shown in Figure 2.10b. The result projects the vertices into a plane instead of a circle, as shown in Figure 2.10c. In this scenario, optimizing between popularity and similarity is simply analogous to finding the nearest neighbors using hyperbolic distances where popularity is the radial coordinate and similarity is the angular coordinate. Graphs generated using the *Popularity Similarity Optimization (PSO) model* are referred to as *Popularity*$\times$*Similarity graphs* or *PSO graphs*.

Papadopoulos et al. show that graphs generated using their model still provide an explanation for scale-free networks, with different γ values able to be modelled by simulating popularity fading in which older vertices lose popularity over time. Their networks are also able to explain high levels of clustering, since similar vertices have many triadic closures amongst their close neighbors. Varying levels of clustering can be introduced with a temperature parameter $T = [0,1)$, in which high temperatures allow for vertices to be randomly connected to further away vertices. The two other important parameters of the model are the number of vertices in the network, N, and a parameter equal to half the average degree of a vertex, $m = \langle k \rangle / 2$. An example Popularity$\times$Similarity graph built using $T = 0.1$ is shown in Figure 2.10d.

Instead of projecting to a circle, it would be possible to project to the unit sphere to allow for more complex similarity patterns between vertices. Distances on a unit sphere have been studied for various applications [28, 92] and were investigated for the inverse problem of finding angular coordinates of real-world networks [186, 189]. Preliminary investigates



(a) Angular coordinates

(b) Radial coordinates

(c) Angular and radial coordinates in $\mathbb{R}^2$

(d) Standard graph layout

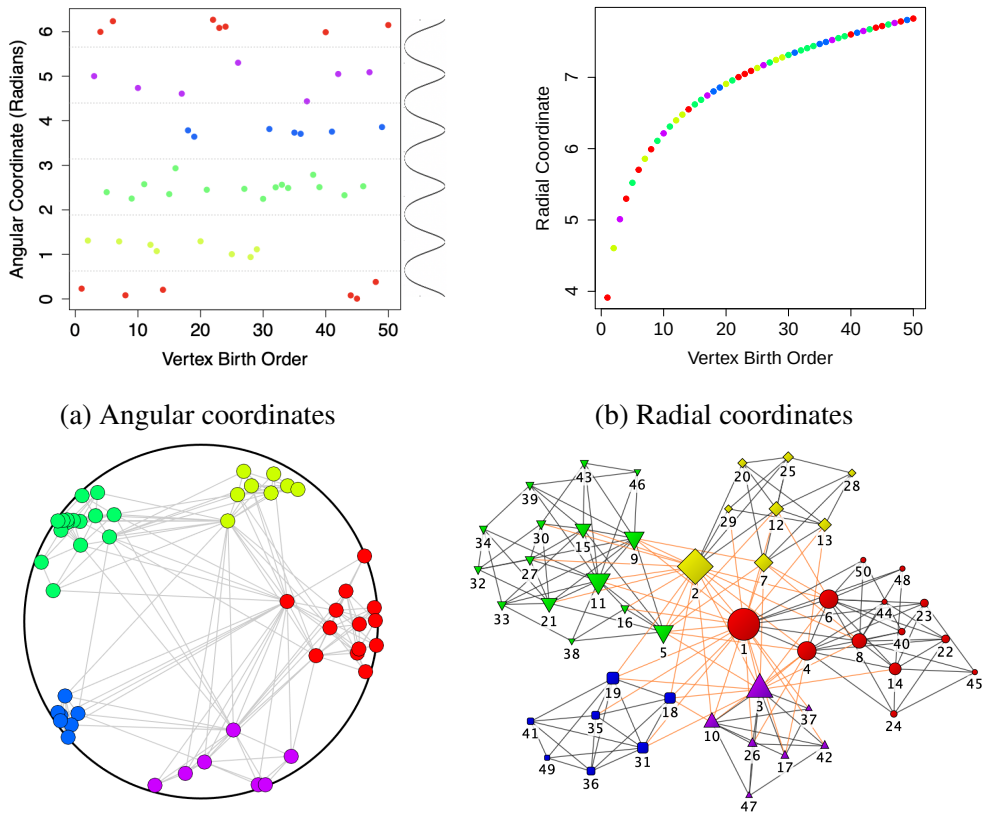
Figure 2.10: A small example Popularity$\times$Similarity graph, (a) angular coordinates vs vertex birth order (b) radial coordinates vs vertex birth order (c) angular and radial coordinates in $\mathbb{R}^2$ (d) Standard graph layout.

of the inverse problem showed that projecting onto a unit sphere did not provide benefits over projecting onto a unit circle for many real-world networks and generated graphs, although this is an unfair assessment for the generated graphs since the Popularity$\times$Similarity graphs are generated using a 2D geometrical model. An example real-world network where there is an improvement in 3D over 2D can be found in [186]. It would be interesting to investigate the inverse problem for a Popularity$\times$Similarity graphs that are generated using a 3D geometrical model.

2.5.4 Popularity×Similarity Graphs with Community Structure

The vertices in Figure 2.10c are placed uniformly at random around a circle and any perceived gaps are due to randomness. Therefore, except for random fluctations, the are not angular regions containing groups of spatially close vertices that are more densely connected between each other than with the rest of the graph. This means there is no specified community structure in Popularity×Similarity graphs that could be used for community detection or link prediction, where link prediction is a task of predicting the existence of an edge between two vertices in a network. To mitigate this, in 2018, Alessandro Muscoloni and Carlo Vittorio Cannistraci introduced a modification to the PSO model that samples angular coordinates in a non-uniform way [185]. Specifically, the only modification between the method proposed by Muscoloni and Cannistraci compared to the method proposed by Papadopoulos et al. is how the angular coordinates are determined.

In the model proposed by Muscoloni and Cannistraci, the user is able to specify the means in radians, standard deviations in radians, and mixing proportions of their desired communities. This means that the user is able to specify the number of communities and relative expected sizes of those communities. The parameters are used to generate a mixture of Gaussians which can be randomly sampled to generate the angular coordinates. The angular coordinates generated in Figure 2.11a were based on 5 equidistant means around the circle with standard deviations equal to $\frac{1}{6} * \frac{2\pi}{5}$ (where * represents scalar multiplcation) and mixing proportions all equal to $\frac{1}{5}$. The radial coordinates in Figure 2.11b and method for creating edges in Figure 2.11c are the same as in the method proposed by Papadopoulos et al., but the resulting graph can be said to have community structure as seen in Figure 2.11d. The internal edges in Figure 2.11d are in black while the external edges are in orange. Otherwise, the colors in Figure 2.11 indicate which of the 5 communities a vertex is assigned to. Graphs generated using the *non-uniform Popularity Similarity Opti-*



(a) Angular coordinates

(b) Radial coordinates

(c) Angular and radial coordinates in $\mathbb{R}^2$

(d) Standard graph layout

Figure 2.11: A small example non-uniform Popularity×Similarity graph with 5 communities equally spaced around a circle with equal proportions. Color (and shape in (d)) indicates the community of the vertex (a) angular coordinates vs vertex birth order (b) radial coordinates vs vertex birth order (c) angular and radial coordinates in $\mathbb{R}^2$ (d) Standard graph layout with internal edges in black and external edges in orange.

mization (nPSO) model can be referred to as *Popularity×Similarity graphs with community structure*, *non-uniform Popularity×Similarity graphs*, or *nPSO graphs*.

It should be noted that researchers have shown that graphs generated using the original PSO model with uniform sampling has shown to create graphs with inherent community structure. Specifically, earlier this year, Bianka Kovács and Gergely Palla demonstrated that graphs generated using the PSO model contain communities for a wide range of the parameters by analyzing the modularity of communities detected on the resulting graphs

using traditional methods like the Louvain method [136]. The random sampling can create natural gaps causing angular separations that create community structure. The nPSO graphs are still needed to test community detection, for ground truth purposes. It should be noted that if the specified community sizes in the non-uniform Popularity$\times$Similarity graphs are large relative to the size of the graph, there may be natural inherent community structure inside the defined communities due to random fluctuations. This could cause community detection algorithms to detect more communities than specified by the model.

Chapter 3: Semi-Local Link Cohesion Metrics for Networks Influenced by Popularity×Similarity Optimization

3.1 Introduction

Network science became a field of study at the beginning of the 21st when modern day tools, storage capacity, and ease of collaboration allowed researchers to study common patterns found in real-world complex networks across different domains [19]. One common pattern that many networks seem to have is community structure, meaning a network can be partitioned in a way that the partitions, called communities, are more densely connected within than between [111]. Knowing the community structure of a network can provide valuable insights: for social networks, communities might represent groups that share a high level of mutual trust or could provide insights on how information or viruses are spread; for the world wide web, communities could provide topical categorizations of web pages; in connectomes, communities may provide insights on how the brain functions [199]. That said, the communities in a network have to be detected which can be a challenging task [100]. Most real-world networks do not have reliable meta-data that corresponds to the underlying structure of the network [123, 215], therefore generated benchmark graphs where the community structure is defined are important for testing the performance of community detection algorithms [102].

The earliest generated benchmark graphs for community detection in network science were proposed by Michelle Girvan and Mark Newman in 2002 [111], called *GN graphs*.

GN graphs consist of 128 vertices with equal degrees broken into four equal sized communities (see Subsection 2.5.2 for more details about GN graphs). In 2008, Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi proposed a new benchmark that generalizes different aspects of the GN graphs to allow for different numbers of vertices with degree distributions and community size distributions that can be right-skewed as observed in many real-world networks [141]. Both the degree distributions and community size distributions in these *LFR graphs* are created artificially and many other patterns found in real-world networks such as higher than random clustering, small-worldness, scale-freeness, and rich-clubness are only observed in LFR graphs due the planted structure (see Subsection 2.5.2 for more details about LFR graphs). That said, methods for community detection are often compared based on their performance on LFR graphs [30, 102, 131, 176, 283, 286]. In 2018, a new type of benchmark for community detection was proposed by Alessandro Muscoloni and Carlo Vittorio Cannistraci where many of the observed patterns of real-world networks come naturally based on how the graphs are generated [185]. These benchmark graphs are build off of a growing network model proposed by Fragkiskos Papadopoulos, Maksim Kitsak, Ángeles Serrano, Marián Boguná, and Dmitri Krioukov in 2012 that optimizes connections of new edges between competing notions of attractiveness: popularity and similarity [209]. In the *Popularity×Similarity Optimization (PSO) model*, vertices are projected such that a radial coordinate corresponds to the popularity of a vertex and an angular coordinate corresponds to the similarity between vertices. As newer and increasingly less popular vertices are added to the growing graph uniformly at random across angular coordinates, hyperbolic distances are used to determine connections to older vertices (see Subsection 2.5.3 for more details about the PSO model). The benchmark graphs proposed by Muscoloni and Cannistraci, called *nPSO graphs*, add non-uniform distributions for angular coordinates to be drawn from yielding a new notion of community structure (see Subsection 2.5.4 for more details about nPSO graphs).

nPSO graphs seem like they should have community structure similar to what is found in real-world networks based on how they are generated, but even somewhat trivial graphs built using the benchmark have communities that do not fit the conventional paradigm. For example, the partition generated using the ground truth communities of the unweighted graph in Figure 1.1b (also shown in Figure 2.11d) would not be considered proper community structure based on the criteria in Definition 2.2.9. In this chapter, we provide an example of why it is important to study nPSO graphs based on their ability to create patterns observed real-world networks that LFR graphs are not able to create. We additional show that a recently proposed link cohesion metric [239] is able to improve the communities detected by traditional algorithms in nPSO graphs when used as a pre-weighting step and demonstrate that the deterministic link cohesion metric computation is scalable and easily parallelizable.

The link cohesion metric was originally introduced by Cetin Savkli, Amanda Galante, Jonathan Cohen and myself in 2019 to create informative simplified graphs of highly connected networks for both visualization purposes and to be used to detect cohesive subgraphs, specifically trusses (see Definition 2.2.11), that highlight important groups of vertices that are amongst the most densely connected in a network [239]. The idea for the metric and for creating simplified graphs was proposed by Savkli, who also wrote the initial implementation in Java. I created the experimental platform for evaluation on LFR graphs and comparison to a baseline algorithm and conducted the experiments. Galante wrote the original paper and helped choose which parameters to use for evaluation. Cohen provided insights used to determine the final form of the metric found in the original paper and suggested its name. At the conference where the paper was presented, Cannistraci suggested I use nPSO graphs as a benchmark in addition to the LFR graphs, leading to this additional research. Wojciech Czaja suggested that the link cohesion metric and related terms should be formalized with mathematical definitions. For this research, I reintroduce

the link cohesion metric with more formal definitions and reimplement the metric using a different approach, enabling it to scale to much larger graphs. The new implementation is written in R, which allows for easier evaluation with various traditional community detection algorithms.

3.2 Motivation for Studying Popularity×Similarity Graphs (nPSO Graphs)

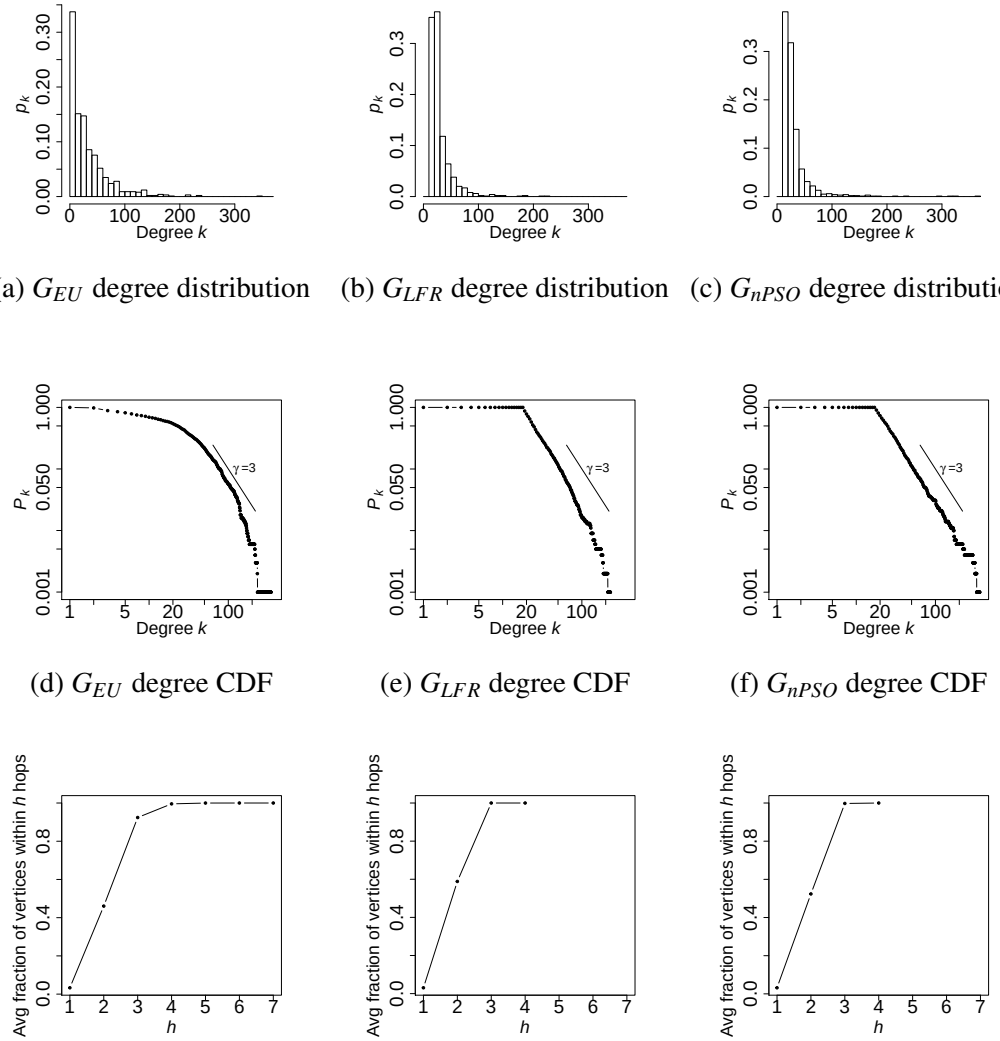
SNAP datasets (the Stanford Large Network Dataset Collection) [148] contains a large number of networks, but only one network has non-overlapping ground truth communities, specifically, the *email-Eu-core* network has 1,005 vertices, 25,571 edges, and 42 non-overlapping communities of varying sizes. The network was generated using email data from a large European research institution where only internal emails were considered, whereas the original dataset also contained emails between members of the institution and people outside the institution. Each vertex in the email-Eu-core network represents a member of the research institution and an edge exists if the two members sent an email back and forth to each other over an 18 month period of time. The ground truth communities correspond to the department an individual belongs to. The original purpose of the dataset was not for community detection, but to study how networks grow over time [147]. The network was first used with its communities denoted in a paper by the original author a decade later to study local graph clustering [284]. The nature of the dataset makes it a good candidate for community analysis because it seems plausible that the department structure of the research institution would lead to groups in the network that are more densely connected within than between. While the original network is directed with self-loops, for community detection purposes a simple graph version (see Definition 2.1.3), G_{EU}, is made resulting in 1,005 vertices and 16,064 edges. Additional graph-level metrics of G_{EU} can be found in Table 3.1. The degree distribution of G_{EU} and related plots can be found in

Figure 3.1. The rich-club coefficient plots of G_{EU} with a comparison to the null models can be found in Figure 3.2. Figure 3.3 contains plots related to the community structure of G_{EU}. See Chapter 2 for related definitions. Note that distances for G_{EU} are based on the largest connected component (see Definition 2.1.15) which contains all but 19 isolated vertices (see Definition 2.1.7).

Table 3.1: Graph-level metrics for the simplified email-Eu-core network, G_{EU}, and for the best LFR graph, G_{LFR}, and best nPSO graph, G_{nPSO}, each chosen out of an ensemble of 1,000 generated using parameters similar to G_{EU}. See Chapter 2 for related definitions.

Graph-level Metrics	G_{EU}	G_{LFR}	G_{nPSO}
Size of Graph	1005	1000	1000
Number of Edges in the Graph	16,064	15,273	15,864
Density of the Graph	0.064	0.061	0.064
Average Degree	32	31	32
Minimum Degree	0	17	16
Maximum Degree	345	228	361
Assortativity	-0.026	-0.002	-0.045
Characteristic Path Length	2.584	2.379	2.445
Average Local Clustering Coefficient	0.450	0.176	0.439
Global Clustering Coefficient	0.267	0.134	0.224
Rich-Clubness (p-value)	0.000	0.000	0.000
Number of Communities	42	41	42
Minimum Community Size	1	11	5
Max Community Size	109	109	95

There are a few reasons one might want to find benchmark graphs similar to G_{EU}. First, ground truth communities in real-world networks are based off of meta-data that are only a proxy for the underlying structure of the network [123, 215]. Finding a benchmark that creates similar structural patterns provides evidence that the ground truth has some merit. Second, once a benchmark similar to a real-world network is found, understanding the performance of different community detection approaches on the benchmark graphs can be used to determine how well the approaches might work on the real-world network. Lastly, developing algorithms that work well on a single network is not necessarily helpful – by



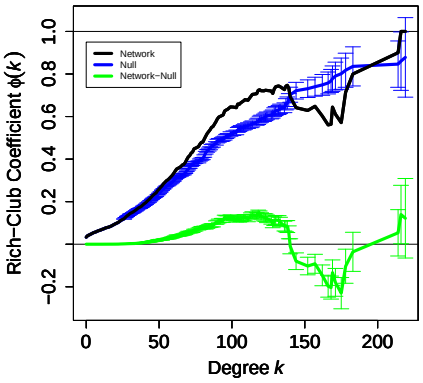
(a) G_{EU} degree distribution (b) G_{LFR} degree distribution (c) G_{nPSO} degree distribution

(d) G_{EU} degree CDF (e) G_{LFR} degree CDF (f) G_{nPSO} degree CDF

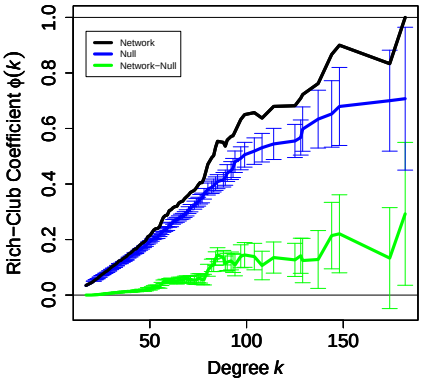
(g) G_{EU} hop plot (h) G_{LFR} hop plot (i) G_{nPSO} hop plot

Figure 3.1: Degree distribution and related plots (see Figure 2.4 for more details) for the simplified email-Eu-core network, G_{EU}, and for the best LFR graph, G_{LFR}, and best nPSO graph, G_{nPSO}, each chosen out of an ensemble of 1,000 generated using parameters similar to G_{EU}. First row contains the degree distributions, second row contains the degree CDFs, third row contains the hop plots. First column are plots corresponding to G_{EU}, second column are plots corresponding to G_{LFR} , third column are plots corresponding to G_{nPSO}. The hop plots of G_{LFR} and G_{nPSO} stop at $h = 4$ because the fraction of vertices within 4-hops reaches 1.0 in both cases.

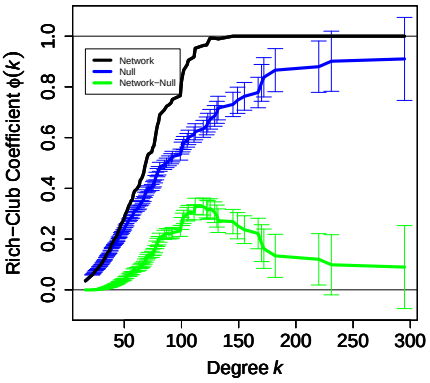
generalizing to a set of benchmark graphs, new approaches can be more broadly understood on when they should be used.



(a) G_{EU} rich-club coefficients



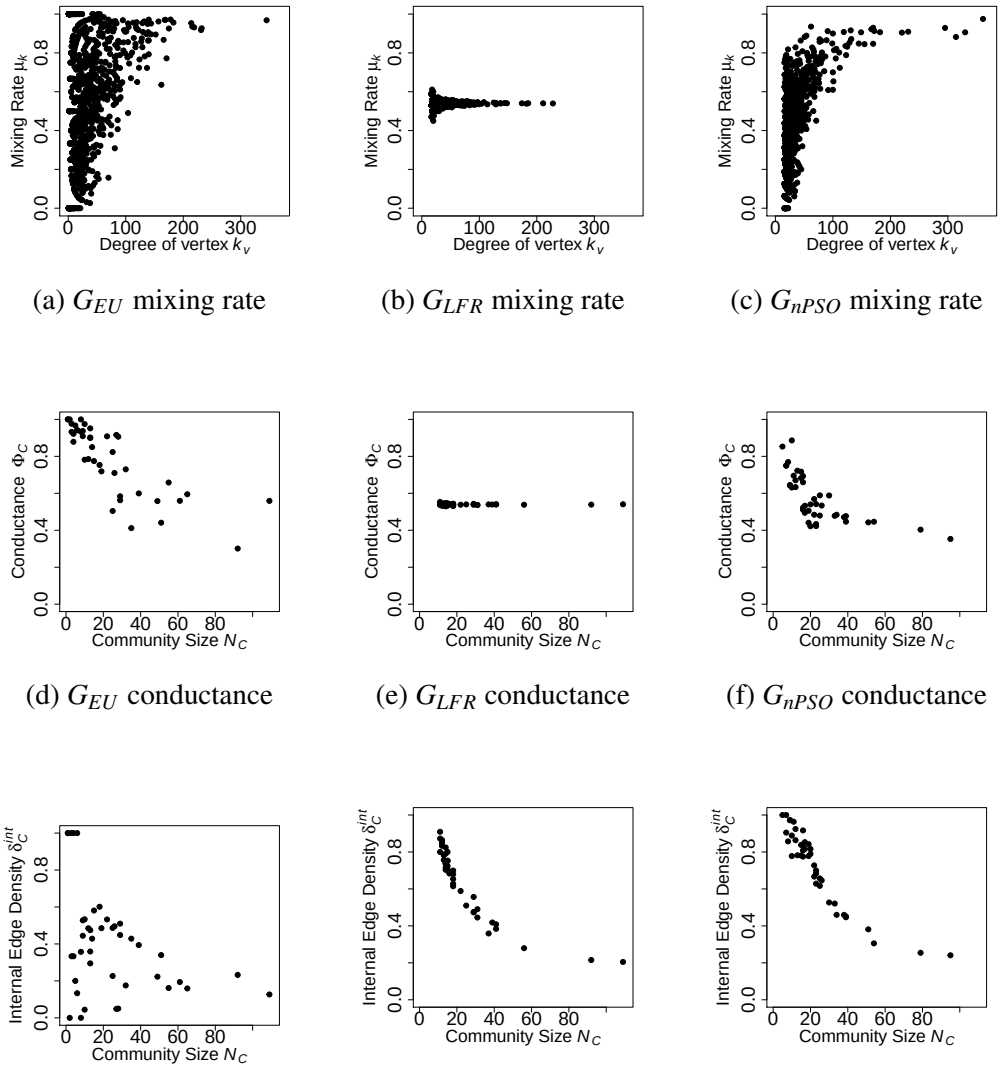
(b) G_{LFR} rich-club coefficients



(c) G_{nPSO} rich-club coefficients

Figure 3.2: Rich-club coefficient plots (see Definition refdef:rcpval and Figure 2.7 for more details) for the simplified email-Eu-core network, G_{EU}, and for the best LFR graph, G_{LFR}, and best nPSO graph, G_{nPSO}, each chosen out of an ensemble of 1,000 generated using parameters similar to G_{EU}. (a) Rich-club coefficients for G_{EU} (b) Rich-club coefficients for G_{LFR} (c) Rich-club coefficients for G_{nPSO}

The goal of creating benchmark graphs similar to G_{EU} is not to find graphs that match G_{EU} exactly. The fact that G_{EU} has 1,005 vertices does not need to be matched exactly, since 19 of those vertices are isolated vertices that would not be reflected in a benchmark anyhow. Additionally, 3 of the 42 communities in G_{EU} have less than 3 vertices which is specifically avoided in the benchmark graphs.



(a) G_{EU} mixing rate (b) G_{LFR} mixing rate (c) G_{nPSO} mixing rate

(d) G_{EU} conductance (e) G_{LFR} conductance (f) G_{nPSO} conductance

(g) G_{EU} internal edge density (h) G_{LFR} internal edge density (i) G_{nPSO} internal edge density

Figure 3.3: Community related plots for the simplified email-Eu-core network, G_{EU}, and for the best LFR graph, G_{LFR}, and best nPSO graph, G_{nPSO}, each chosen out of an ensemble of 1,000 generated using parameters similar to G_{EU}. First row contains the mixing rates (see Definition 2.2.10) for each vertex, second row contains the conductance (see Definition 2.3.3) of each community, third row contains the internal edge density (see Definition 2.3.3) of each community. First column are plots corresponding to G_{EU}, second column are plots corresponding to G_{LFR} , third column are plots corresponding to G_{nPSO}.

An attempt is first made to create LFR graphs similar to G_{EU} with an emphasis placed on getting the most similar maximum degree vertices with the correct number of communi-

ties. To generate LFR graphs, users can specify a number of parameters controlling the size of the network, the degree distribution, and the community size distribution. To obtain LFR graphs similar to G_{EU}, called LFR_EU, the number of vertices was set to $N = 1,000$, the average degree was set to $\langle k \rangle = 32$, the maximum degree was set to $k_{max} = 345$, the degree distribution scaling parameter was set to $\gamma = 2$, the community size distribution scaling parameter was set to $\beta = 1.5$, the minimum and maximum number of communities were set to $c_{min} = 3$ and $c_{max} = 109$ respectively, and the mixing parameter was set to $\mu = .55$. With LFR graphs, differing numbers of communities are made for each realization. Because a high degree vertex cannot be placed in a community unless its internal degree is less than the community size, it is difficult for this particular set of parameters to create the smaller community sizes that would be needed to get up to 42 communities. $\gamma = 2$ is used, yielding an instance that contained 41 communities whereas using $\gamma = 3$ only ever produced a much smaller number of communities. Out of an ensemble of 1,000 LFR graphs with the stated parameters used, the graph with the greatest number of communities and highest degree is considered the best LFR graph, G_{LFR}, for the sole purpose of creating plots and metrics for comparison to G_{EU}.

Unlike LFR graphs, nPSO graphs allow the user to set the number of communities but the maximum degree of a graph is determined by the growing model and is dependent on parameters m and γ. To obtain nPSO graphs similar to G_{EU}, called nPSO_EU, the number of vertices was set to $N = 1,000$, the parameter equal to half the average degree of a vertex is set to $m = 16$, and 42 equidistant means around the circle are set with standard deviations equal to $\frac{1}{6} * \frac{2\pi}{42}$ (where * represents scalar multiplication). The mixing proportions are used to create unequal sized communities, specifically, the proportions are set using a power law with exponent $\beta = 2.6$. An additional condition is made that all communities in a generated nPSO graph must have at least 3 vertices. A temperature of $T = 0.3$ is used to create similar clustering and the degree distribution scaling parameter is set to $\gamma = 3$ for the

nPSO graphs so that maximum degree in each benchmark graph is similar to G_{EU} whereas setting $\gamma = 2$ creates vertices with much larger maximum degrees. Out of an ensemble of 1,000 nPSO graphs with the stated parameters used, the graph with the lowest maximum degree is considered the best nPSO graph, G_{nPSO}, for the sole purpose of creating plots and metrics for comparison to G_{EU}.

Structural details about G_{LFR} and G_{nPSO} can be found in Table 3.1 and Figures 3.1, 3.2, and 3.3. While a number of graph-level and vertex-level metrics similar to G_{EU} can be incorporated into either type of benchmark graphs, a major difference is seen in the characteristics of the communities. The mixing rate (see Definition 2.2.10) for each vertex and the conductance (see Definition 2.3.3) of each community in G_{LFR} are quite uniform, whereas in the real-world network these values can vary quite significantly. Low degree vertices are often connected to high degree vertices outside of their community in G_{EU}, leading to the wide ranges of mixing rates for these vertices and high conductance values for smaller communities. This is expected in networks with hierarchical structure, discussed by Erzsébet Ravasz and Albert Barabási in 2003 [228] and further analyzed by Sara Nadiv Soffer and Alexei Vázquez in 2005 [253].

Claim 3.2.1. (CS) nPSO benchmark graphs are able to generate graphs with hierarchical community structure (see Figures 3.3a, 3.3d, 3.3c, and 3.3f) whereas LFR benchmark graphs cannot (see Figure 3.3b and 3.3e), leading to community structure much more similar to certain real-world networks. For example, the benchmark nPSO_EU described above is a relatively good approximation of G_{EU}, a simplified version of the email-Eu-core network from SNAP datasets [148].

nPSO graphs do not have community structure similar to the conventional paradigm. For example, the partition generated using the ground truth communities of the unweighted graph in Figure 2.11d would not be considered proper community structure based on the

criteria in Definition 2.2.9. Modularity (see Definition 2.3.1) is a measure often used to assess a partitioning of a network into communities where, for an unweighted network, the fraction of edges that fall within the communities is compared to what would be expected in a randomized degree-preserving multigraph (see Definition 2.5.1). Small communities with high conductance (see Definition 2.3.3) have a large fraction of their edges falling outside their communities – if a network is unweighted, these communities may negatively contribute to the modularity score. Modularity maximization is one of the most common techniques in density centric community detection, used by the popular Louvain method (see Subsection 2.3.1).

Claim 3.2.2. (CS) Without proper weighting, community detection algorithms that maximize modularity are at risk of not being able to resolve smaller communities which can have high conductance in nPSO graphs (see Figure 3.3d) and similarly networks influenced by Popularity$\times$Similarity optimization (see Figure 3.3e).

In this chapter, the benefits of pre-weighting networks influenced by Popularity$\times$Similarity optimization with the link cohesion metric proposed in [239] in order to mitigate these problems are analyzed.

3.3 Related Work

While there have been a large number of studies on the performance of community detection algorithms on LFR graphs [30, 102, 131, 176, 283, 286], less research has been conducted on studying the performance of community detection algorithms on nPSO graphs. This is in large part due to the fact that nPSO graphs were proposed only a few years ago, in 2018. In the paper that introduced this new benchmark, an initial glimpse at the detectability of its ground truth communities was conducted by analyzing the NMI of the detected communities using the Louvain method for a variety of parameter settings. A

follow-up study was published the same year by Alessandro Muscoloni and Carlo Vittorio Cannistraci where the performance of the Louvain method was compared to Infomap, Walktrap, and Label Prop in a variety of parameter settings [184] (see Subsections 2.3.1 and 2.3.3 for more details about these methods). Additionally, the performance of these algorithms on the unweighted nPSO graphs are compared to the nPSO graphs weighted using the inverse hyperbolic distances described in [189]. The process of finding inverse hyperbolic distances involves a number of steps, the first among them is to pre-weight the network with either local or global structural metrics. The focus of this chapter is to analyze local, semi-local, and global structural metrics (see Definition 2.3.2) that can improve the communities detected by traditional algorithms. The results in this chapter may lead to a better pre-weighting approach that can be used in the initial step of finding inverse hyperbolic distances as well. The inverse hyperbolic distances are used to weight real-world networks with performance improvement demonstrated for the Louvain method, Infomap, Label Prop and Walktrap in [189]. Some research into new types of community detection algorithms for nPSO graphs has also been investigated, such as greedy routing techniques [47, 98, 187], Markov clustering [91], and self-adapting switching [271].

A wide range of local, semi-local, and global edge metrics (see Definition 2.3.2) have been proposed to improve the communities detected by traditional algorithms, mostly tested on LFR graphs. These are discussed in Subsection 2.3.2.

3.4 Inversely Proportional Link Cohesion

A new type of metric was introduced in [239] called link cohesion.

Definition 3.4.1. *Link cohesion* is a type of metric used to assess how supported an edge is relative to other edges, accounting for nearby alternate paths and associated vertex degrees.

One benefit of link cohesion metrics is that they are semi-local (see Definition 2.3.2)

by design, whereas many other metrics are either local or global. Prior to formalizing the precise definition of the link cohesion metric introduced in [239], definitions related to alternate paths will be introduced. Note that all graphs are assumed to be simple graphs (see Definition 2.1.3).

Definition 3.4.2. Let $G = (V, E)$ be a graph with $u, v \in V$ and let $h \in \mathbb{Z}^+$. The set containing all $n_h \in \mathbb{Z}^{0+}$ simple paths of length h, also called h-hop simple paths, between u and v is defined as

$$\mathrm{S}h_{uv} = \{(u, z_1^1, ..., z_{h-1}^1, v), (u, z_1^2, ..., z_{h-1}^2, v), ...(u, z_1^{n_h}, ..., z_{h-1}^{n_h}, v)\}, \tag{3.1}$$

where u and v are called the *seed vertices* and z_j^i for $i = 1, .., n_h$ and $j = 1, .., h-1$ are the *intermediate vertices* in the simple paths of length h between the seed vertices.

Simple paths were defined in Definition 2.1.18. Additionally, recall that connected vertices are defined in Definition 2.1.8.

Definition 3.4.3. Let $G = (V, E)$ be a graph with $u, v \in V$. Simple paths of length 2 or more between connected seed vertices u and v are considered *alternate paths* of the edge (u, v). If the length of a simple path is equal to 2 or 3 it is considered a *nearby alternate path*.

Although Definition 3.4.2 is generalized for simple paths of any length, 1-hop, 2-hop, and 3-hop simple paths will be analyzed closer.

Definition 3.4.4. Let $G = (V, E)$ be a graph with $u, v \in V$. There is $n_1 = 1$ 1-hop simple path between any connected vertices u and v. Therefore, the set containing all 1-hop simple paths between connected vertices u and v can be defined as

$$\mathrm{S}1_{uv} = \{(u, v)\}. \tag{3.2}$$

Note that there are no intermediate vertices in $\mathrm{S1}_{uv}$.

Definition 3.4.5. Let $G = (V, E)$ be a graph with $u, v \in V$. Given there are $n_2 \in \mathbb{Z}^{0+}$ 2-hop simple paths between vertices u and v and given $l_i = z_1^i$, the set containing all 2-hop simple paths between u and v is defined as

$$\mathrm{S2}_{uv} = \{(u, l_1, v), (u, l_2, v), ..., (u, l_{n_2}, v)\}. \tag{3.3}$$

Note that if $n_2 = 0$, then $\mathrm{S2}_{uv} = \{\}$.

Definition 3.4.6. Let $G = (V, E)$ be a graph with $u, v \in V$. Given there are $n_3 \in \mathbb{Z}^{0+}$ 3-hop simple paths between vertices u and v and given $s_i = z_1^i$, $t_i = z_2^i$, the set containing all 3-hop simple paths between u and v is defined as

$$\mathrm{S3}_{uv} = \{(u, s_1, t_2, v), (u, s_2, t_2, v), ..., (u, s_{n_3}, t_{n_3}, v)\}. \tag{3.4}$$

Note that if $n_3 = 0$, then $\mathrm{S3}_{uv} = \{\}$.

Figure 3.4 provides an example of connected seed vertices u and v with all of the 1-hop, 2-hop, and 3-hop simple paths connecting u and v delineated. The 2-hop and 3-hop simple paths are considered nearby alternate paths of the edge (u, v). Note that 2-hop simple paths between connected seed vertices create triangles and 3-hop simple paths between connected seed vertices create rectangles (see Definition 2.1.19).

One way to measure the cohesiveness of an edge (u, v) is to weight it inversely proportional to number of edges that the vertices u and v are expected to have in a random degree-preserving multigraph (see Definition 2.5.1). This well-known value, $k_u k_v/(2L-1)$ where k_u is equal to the degree of vertex u and L is the number of edges in the graph, is used in a number of equations including modularity (see Definition 2.3.1 and note that the term is approximated as $k_u k_v/2L$).

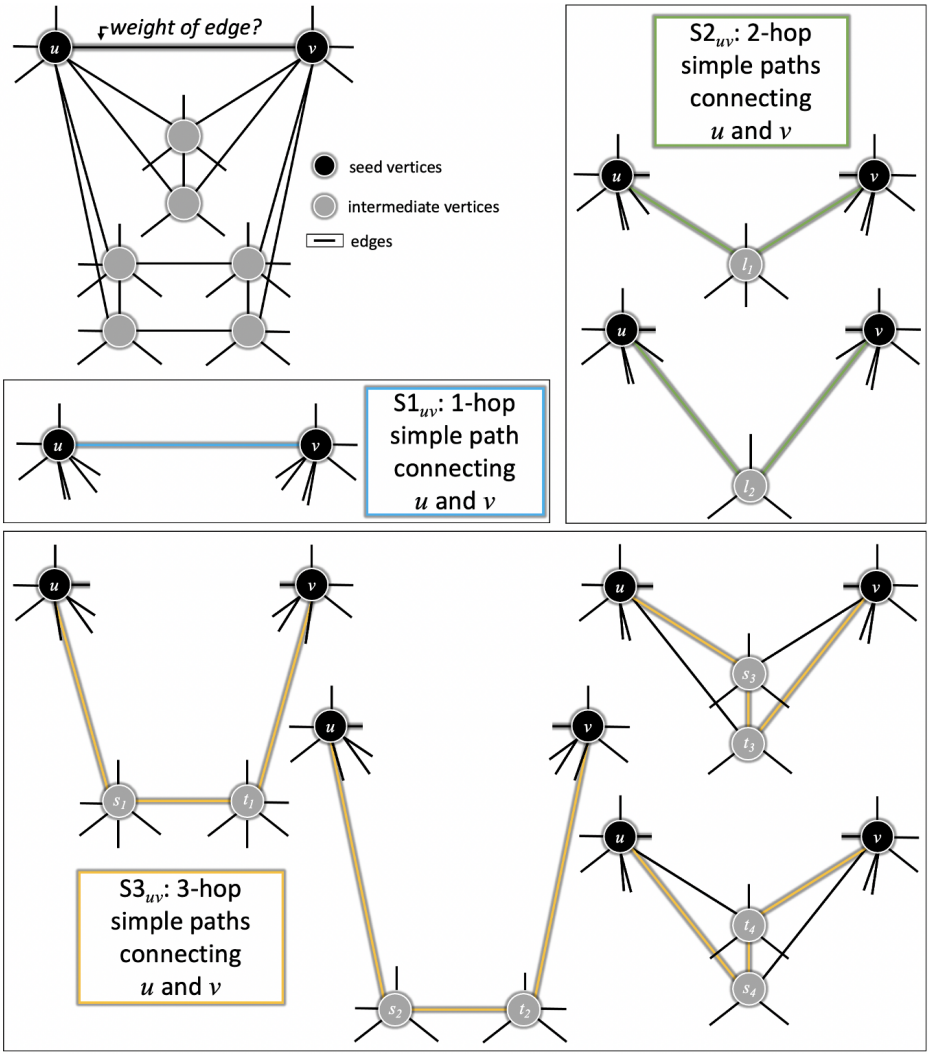


Figure 3.4: Delineation of all 1-hop, 2-hop, and 3-hop simple paths between two seed vertices.

Claim 3.4.7. Edges that have a higher expected number of edges in a random degree-preserving multigraph are less likely to be an indication of cohesiveness [239]. This can be extended to measure the cohesiveness of alternate 2-hop, 3-hop, and other higher order simple paths by weighting each edge of a simple path independently in the same way and multiplying the values together.

While technically the edges are not independent, accounting for the dependency could cause the inverse estimated number of edges to be undefined which is not desirable. Raw

scores for an edge created using these concepts are not meaningful unless they are assessed relative to the scores of other edges in the graph, therefore, the weights are normalized to the interval [0,1). An additional benefit of normalizing the weights between [0,1) is that weights computed taking different simple path lengths into account can be combined into a single metric.

Definition 3.4.8. Let $G = (V, E)$ be a graph with $L = |E|$, let $h \in \mathbb{Z}^+$, and let $\mathrm{S}h_{uv}$ be the set of all h-hop simple paths between vertices $u, v \in V$. The raw Inversely Proportional Link Cohesion weights involving h-hop simple paths, which can be referred to as the *raw* S*h* *Inversely Proportional Link Cohesion weights*, are defined as

$$\mathrm{rInvPrLinkCo_S}h(u,v) = \begin{cases} \frac{1}{k_u * k_v} \sum\limits_{z_1,\ldots,z_{h-1} \in \mathrm{S}h_{uv}} \frac{1}{k_u * k_v * (k_{z_1} * \ldots * k_{z_{h-1}})^2} & \text{if } u, v \in V, \\ 0 & \text{otherwise,} \end{cases} \tag{3.5}$$

where $z_1, \ldots, z_{h-1}$ are the intermediate vertices of $\mathrm{S}h_{uv}$ and * represents scalar multiplication. Since there are no intermediate vertices in $\mathrm{S1}_{uv}$, the default sum is defined equal to 1. If the set of simple paths $\mathrm{S}h_{uv}$ is empty, the default sum is defined to equal 0. The average raw Inversely Proportional Link Cohesion weight involving h-hop simple paths, which can be referred to as the *average raw* S*h* *Inversely Proportional Link Cohesion weights*, is defined as

$$\mu_{\mathrm{rInvPrLinkCo_S}h} = \frac{1}{2L} \sum_{u,v \in V} \mathrm{rInvPrLinkCo_S}h(u,v). \tag{3.6}$$

The normalized Inversely Proportional Link Cohesion weights involving h-hop simple paths, which can be referred to as the S*h* *Inversely Proportional Link Cohesion weights*, are defined as

$$\mathrm{InvPrLinkCo_S}h(u,v) = \frac{\mathrm{rInvPrLinkCo_S}h(u,v)}{\mu_{\mathrm{rInvPrLinkCo_S}h} + \mathrm{rInvPrLinkCo_S}h(u,v)}. \tag{3.7}$$

Inversely Proportional Link Cohesion is defined as

$$\text{InvPrLinkCo}(u,v) = \frac{1}{2}\text{InvPrLinkCo_S2}(u,v) + \frac{1}{2}\text{InvPrLinkCo_S3}(u,v) + \varepsilon, \tag{3.8}$$

where ε is equal to machine epsilon.

The final form of Inversely Proportional Link Cohesion is different than what was proposed in [239] in that it only takes into account 2-hop simple paths with

$$\text{rInvPrLinkCo_S2}(u,v) = \frac{1}{(k_u k_v)^2} \sum_{l \in S2_{uv}} \frac{1}{k_1^2} \tag{3.9}$$

for connected seed vertices and 3-hop simple paths with

$$\text{rInvPrLinkCo_S3}(u,v) = \frac{1}{(k_u k_v)^2} \sum_{s,t \in S3_{uv}} \frac{1}{(k_s k_t)^2} \tag{3.10}$$

for connected seed vertices, leaving out 1-hop simple paths between connected seed vertices. After further analysis, it seems the current proposed definition has some further benefits over the former. Since 1-hop simple paths have no intermediate vertices, connected vertices u and v would have raw S1 Inversely Proportional Link Cohesion weights equal to

$$\text{rInvPrLinkCo_S1}(u,v) = \frac{1}{k_u k_v}. \tag{3.11}$$

This term is already taken into account when calculating Inversely Proportional Link Cohesion for simple paths of higher order. A benefit of not including the 1-hop simple paths is that vertices not connected by any 2-hop or 3-hop alternate paths will have a score equal to ε. While this may not be ideal in some networks, for social networks it can provide valuable insights as a person that is not connected to any friends or friends of friends of another person is not likely to be members of the same community even if the two people are con-

nected. Additional benefits discussed in Subsection 4.6.2 are also relevant here. Subsection 4.6.8 contains an analysis of a network that has some vertices that are only connected to other vertices without nearby alternate paths. The benefit of taking both 2-hop simple paths and 3-hop simple paths into account is analyzed in Subsection 3.7.2 and Subsection 4.6.7. Higher order simple paths are not taken into account because the computational complexity increases and the benefit added is not expected to be much. For social networks, higher order simple paths get increasingly less meaningful as the characteristic path length (see Definition 2.1.17) is often relatively small. Larger characteristic path lengths may indicate that using higher order cycles would be beneficial, observing the corresponding hop plot may provide better insights (see Figure 2.4c as an example).

Claim 3.4.9. Because it takes both the edge connected seed vertices into account along with edges in nearby alternate paths (see Equations 3.5 and 3.8), Inversely Proportional Link Cohesion is one way of measuring the *strength* of an edge based on the cycles it is part of.

A number of researchers have found triangles, rectangles, and cycles of higher order occur more frequently within communities than between [30, 176, 222, 250, 267]. That said, triangles and rectangles commonly occur amongst the highest degree vertices in nPSO graphs and networks influenced by Popularity$\times$Similarity optimization, but these vertices are often spread across all the communities which conflicts with the previous statement. The prevalence of triangles and rectangles amongst the highest degree vertices in nPSO graphs are due to the popularity optimization causing rich-clubness (see Definition 2.2.13) [185]. That does not imply that analyzing cycles is not beneficial in nPSO graphs – relatively more cycles will occur amongst higher degree vertices that are part of the same community and cycles that occur amongst lower degree vertices typically only occur within communities due to the similarity optimization. Inversely Proportional Link Cohesion

would give higher weight to the cycles involving low degree vertices and less weight to cycles involving high degree vertices as would be desired. Vertices that happen to be connected but are not involved in any cycles would have a weight equal to ε to mark it as an edge but not give preference to it for community detection purposes. Other weighting techniques that take cycles into account do not take the degrees of all the vertices involved in the cycles into account the way that Inversely Proportional Link Cohesion does.

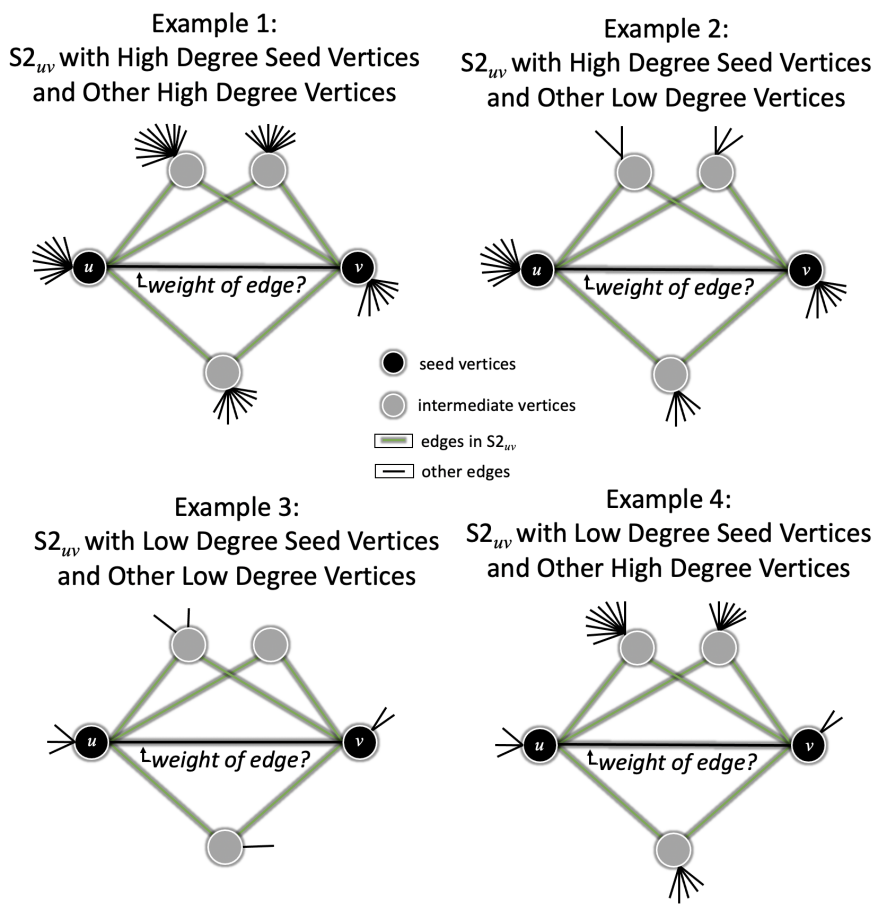


Figure 3.5: Four examples of connected seed vertices with 2-hop alternate paths highlighted. Inversely Proportional Link Cohesion would give the largest weight to the edge connecting the seed vertices in Example 3 and the lowest weight to the edge connecting the seed vertices in Example 1. The edges connecting seed vertices in Examples 2 and 4 would have weights in between.

Consider the four examples in Figure 3.5. All seed vertices are connected by an edge and connected by 3 additional 2-hop simple paths. In Example 1, the seed vertices are

both high degree vertices connected by other high degree vertices, which would obtain a relatively small Inversely Proportional Link Cohesion weight. While it is possible that these seed vertices are part of the same community, they may just be connected because of their popularity if they are influenced by Popularity$\times$Similarity optimization. In Example 2, the seed vertices have the same degrees as the seed vertices in Example 1, but are connected by other lower degree vertices. Relative to Example 1, it would seem more likely that the seed vertices in Example 2 are part of the same community. In Example 3, the seed vertices are both low degree vertices connected by other low degree vertices. While it is possible that these seed vertices are part of different communities, it seems more likely that they are part of the same community due to their similarity if they are influenced by Popularity$\times$Similarity optimization. In Example 4, the seed vertices have the same degrees as the seed vertices in Example 3, but are connected by other high degree vertices. It is less likely that the seed vertices in Example 4 are part of the same community compared to the seed vertices in Example 3, though it is still possible.

Claim 3.4.10. (CS) Inversely Proportional Link Cohesion weights provide relatively larger scores to edges that connect low degree vertices more likely to be connected in an nPSO graph due to their similarity and relatively smaller scores to edges that connect high degree vertices more likely to be connected in an nPSO graph due to their popularity as expected, assuming they are from the same graph.

All of the comparisons are relative – if the average degree in the graph that these examples came from is 60, all vertices in these examples would have degrees on the lower end compared to the rest of the graph, meaning all may be indicative that the seed vertices are part of the same community. Because of this, normalization is important for more meaningful scores.

3.5 Maximum Density Core (MDCore)

When link cohesion metrics and the now named Inversely Proportional Link Cohesion metric were introduced, the link cohesion metric was used to create a simplified graph called MDCore.

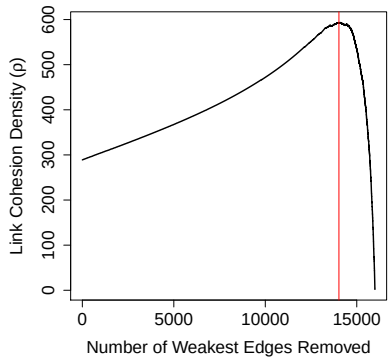
Definition 3.5.1. Let $G = (V, E, \text{LinkCo})$ be a graph weighted by a link cohesion metric, abbreviated as LinkCo. Let $N_{>0}$ represent the number of vertices with degree greater than 0 and $\overline{LinkCo}$ represent the average link cohesion score for the remaining edges. *Link cohesion density* is defined as

$$\rho = N_{>0} * \overline{LinkCo}, \tag{3.12}$$

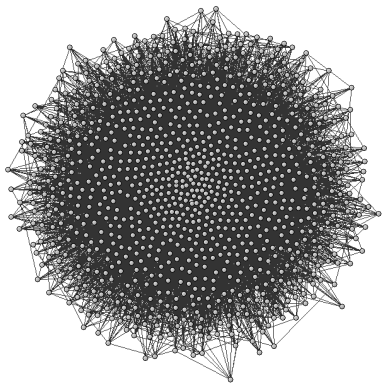
where * represents scalar multiplication. The Maximum Density Core *MDCore* is a subgraph of G obtained by removing edges in G starting with the weakest LinkCo weight, computing the ρ after each edge is removed, until all edges are removed. The vertices in G and edges that are included when the link cohesion density, ρ, is maximized is called the MDCore of G.

The link cohesion metric does not need to be recalculated during the pruning process since they are informed by the original graph. An example of MDCore pruning algorithm on a random graph is shown in Figure 3.6. Figure 3.6a shows the link cohesion density as a function of the number of weakest edges removed, with the maximum density marked with a red vertical line. Figure 3.6b is the random graph used in Figure 3.6a and Figure 3.6c is the resulting MDCore of the random graph.

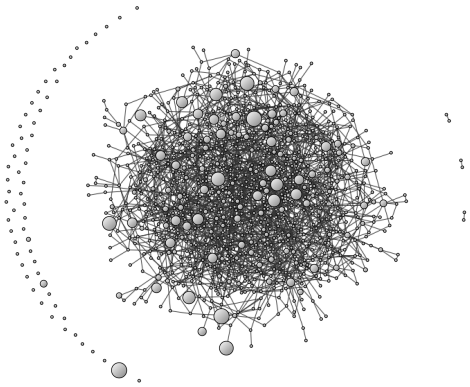
The simplified MDCore of the random graph is much easier to visualize than the random graph itself, with only 12% of the vertices that are in its supergraph. There is not any obvious community structure in the random graph, unlike in the other examples where MDCore is used in this dissertation that show clear community structure, specifically in



(a) Maximum Density Core(MDCore) pruning algorithm on a random graph



(b) The random graph: MS graph (see Subsection 2.5.1) of the graph in Figure 3.19a



(c) MDCore of the random graph with vertices sized by the degrees in the supergraph

Figure 3.6: Maximum Density Core(MDCore) pruning algorithm on a random graph. (a) Link cohesion density ρ as a function of the number of weakest edges removed to demonstrate the maximum density pruning algorithm. (b) The random graph used in the MDCore pruning algorithm example (c) The resulting MDCore with vertices sized by the degrees in the supergraph.

Figures 3.19c, 3.21c, and 4.28c.

Claim 3.5.2. (CS) The MDCore of a graph can be used as a tool to determine if the graph has community structure (see Figures 3.19c, 3.21c, and 4.28c) or not (see Figure 3.6c).

Determining that there is not obvious community structure was not possible in the unsimplified graph, as Figure 3.6b and Figure 3.19a look very similar without the pruning. The benefit of MDCore was originally shown based on the ability to find meaningful cohesive subgraphs, specifically trusses (see Definition 2.2.11), in the simplified graph which

was not possible in highly conneced graphs otherwise [238].

Although it is out of the scope of the dissertation, it has been observed that creating an MDCore with InvPrLinkCo weights on nPSO graphs often separates communities into different connected components due to the consistent patterns generated by the nPSO model. Similarly, it has been observed that creating an MDCore with LoCoLinkCo weights on LFR graphs often separates communities into different connected components due to the consistent patterns generated by the planted partition model. On real-world nerworks, MDCore has been observed to look more like Figures 3.19c, 3.21c, and 4.28c. MDCore works best on large networks or networks that are highly connected, which, as will be seen in Subsection 3.7.5 and Subsection 4.6.6, it is also when the link cohesion metrics provide the most benefit when used as a pre-weighting step for community detection.

3.6 Implementation and Experimental Setup

To calculate the deterministic Inversely Proportional Link Cohesion weights for a particular graph, a simple approach was used as a proof of concept, though a number of techniques could be used to create much faster implementations. To compute Inversely Proportional Link Cohesion for a particular edge, an approach similar to depth first search is used to loop through the neighbors of the seed vertex with fewer neighbors to find simple paths between connected seed vertices. When a path is found, the degrees are used to add to the metric. The computational complexity of computing all edge weights is $O(N)$ for extremely sparse networks (which are often assumed in network science [199], see Definition 2.2.2) with a constant term equal to $\langle k \rangle^2$ where $\langle k \rangle$ represents the average degree of the network, defined to be constant as the size of extremely sparse networks grow. Note that if only S2 weights are calculated, the constant term is equal to $\langle k \rangle$ as implemented, and more specifically incorporating paths of length h would have a constant term equal to $\langle k \rangle^{h-1}$. The

algorithm is also easily parallelizable, as each edge can be computed independently with semi-local information. The original implementation used matrix multiplies with special attention paid to ensure no circular loops between pairs of vertices were included. While fast and efficient for smaller sized networks, the approach does not scale to larger networks with tens of thousands or more vertices.

The Inversely Proportional Link Cohesion computation was written in R to leverage the community detection algorithms written in the igraph package [75]. The nPSO benchmark graphs are written in Matlab [185], so the matlabr package is utilized to run blocks of Matlab code [181]. The LFR benchmark graphs are written in c++ and are called by invoking a system command [141]. The future.apply package is used for parallelization [29]. All baseline weighting metrics are also written in R, parallelizing when possible. RNBRW is a different type of parallelization requiring the doParallel [116] and foreach [117] packages.

All experiments were run on MacBook Pro laptops, either on my old laptop with a 2.9 GHz Quad-Core Intel Core i7 Processor and 16 GB of memory, a loaner laptop when my old laptop had problems which had a 2.9 GHz Dual-Core Intel Core i7 Processor and 16 GB of RAM, or my new laptop with a 2.4 GHz 8-Core Intel Core i9 Processor with 64 GB of RAM.

3.7 Results

Inversely Proportional Link Cohesion (InvPrLinkCo) introduced in this chapter (see Definition 3.4.8) will be compared against several baseline pre-weighting methods, listed in Table 3.2. Local Community Link Cohesion (LoCoLinkCo) introduced in the next chapter (see Definition 4.4.7) will also be used for comparison purposes. Some cycle-based methods are more computationally expensive even when parallelized, so a comparison to cycle-based baseline methods will be completed separately from the other methods with

a more narrow focus. The baseline pre-weighting methods were initially introduced in Subsection 2.3.2.

Table 3.2: Baseline pre-weighting methods used for community detection with the type designating if it is a local, semi-local, or global metric (see Definition 2.3.2). Also included is a short description and a reference to the paper initially introducing the method. A '°' symbol in the type column highlights methods that are cycle-based. Methods were originally introduced in Subsection 2.3.2.

Name	Type	Description	Initital Paper
CN Ratio	Local	Percentage of common neighbor vertices	[130]
RA2	Local	A repulsion attraction rule	[189]
Modified ECC	Local°	Modified edge clustering coefficient	[222]
NeighborCo	Semi-Local°	Neighborhood coherence	[30]
SimRank3	Semi-Local	Truncated SimRank after 3 iterations	[286]
RNBRW	Global°	Renewal non-backtracking random walks	[176]
Khadivi et al.	Global	'Inverse EB' multiplied by 'CN Ratio'	[131]
Inverse EB	Global	Normalized inverse edge betweenness	[111, 131]

The main community detection algorithm used will the Louvain method, but other community detection methods are used to understand the benefits of pre-weighting with InvPrLinkCo for a wider range of algorithms. All of the community detection methods can be found in Table 3.3, which were originally introduced in Section 2.3.

Table 3.3: Community detection methods used on weighted and unweighted graphs. Methods were originally introduced in Section 2.3.

Name	Description	Initital Paper
Louvain method	Heuristic method that maximizes modularity	[34]
Fast Greedy	Agglomerative hierarchical clustering that maximizes modularity	[63]
Label Prop	Heuristic method that propagates labels until a steady state is reached	[223]
Infomap	Method that minimizes the description length of a random walk	[232]
Walktrap	Method assuming that short random walks stay inside a community	[218]
HSBM	Hierarchical Stochastic Block Models	[159]

Because ground truth is avaliable for all of the benchmark graphs and G_{EU}, the results are presented as normalized mutual information, abbreviated as NMI (see Definition 2.3.4), and community count error, abbreviated as CCE (see Definition 2.3.5). Results

are summarized from 30 random repetitions. When comparing against non-cycle-based methods, mean values are plotted with 95% confidence intervals, assuming normal distributions. When comparing against cycle-based methods, boxplots are shown. The baseline pre-weighting methods, community detection methods, and evaluations metrics will also be used in the next chapter.

3.7.1 Benchmark nPSO Graphs

In addition to the parameters for nPSO graphs introduced in Section 3.2, which will be referred to as nPSO_EU, the benchmark graphs used in [184] will be used to motivate using a combined S2 and S3 weight. Specifically, nPSO graphs with all combinations of $N = [100, 500, 100]$, $m = [10, 12, 14]$, and $T = [0.1, 0.3, 0.5]$ are used with $\gamma = 0.3$, where angular coordinates are sampled according to a Gaussian mixture distribution with equal proportions and 8 components. Additionally, a benchmark was created to be more similar to the LFR benchmark graphs introduced in [30], which is referred to as nPSO_BHLP1. The degree distribution and corresponding plots for a graph generated using these parameters with $T = 0.3$ can be found in Figure 2.4. nPSO_BHLP1 graphs are much sparser than nPSO_EU and nPSO_BHLP1 graphs have positive assortativity unlike most nPSO graphs [185]. Additionally, benchmark graphs are generated that scale nPSO_EU to larger sizes. The parameters for all the benchmark nPSO graphs are summarized in Table 3.4.

Table 3.4: Parameters for nPSO benchmark graphs.

Parameter	Baselines [184]	nPSO_EU	nPSO_EU_S1	nPSO_EU_S2	nPSO_BHLP1
N	[100,500,100]	1000	[1000,..,10000]	[1000,..,10000]	5000
m	[10,12,14]	16	16	16	4
T	[0.1,0.3,0.5]	[0.1,0.3,0.5]	0.3	0.3	[0.1,0.3,0.5]
γ	3	3	3	3	3
β	NA	2.6	2.6	2.6	2
Number of Communities	8	42	N*.042	42	150

3.7.2 S2 verses S3 verses Combined Weights

The first experiment uses the benchmark graphs introduced in [184] and compares the raw Inversely Proportional Link Cohesion weights (rInvPrLinkCo_S2 and rInvPrLinkCo_S3 from Equation 3.5) and the normalized Inversely Proportional Link Cohesion weights (InvPrLinkCo_S2 and InvPrLinkCo_S3 from Equation 3.7) to InvPrLinkCo (see Equation 3.8), which combines InvPrLinkCo_S2 and InvPrLinkCo_S3 equally into a single value. The mean NMI in Figure 3.7 and the mean CCE in Figure 3.8 are obtained by comparing the ground truth nPSO communities to the detected communities from the Louvain method when the above mentioned methods are used to pre-weight the graphs. The results of the detected communities by the Louvain method on the unweighted graphs are also shown for comparison.

As expected from Claim 3.2.2, the Louvain method consistently undercounts the true number of communities. While the S2 scores (both raw and normalized) yield higher mean NMI values when $N = 100$, the normalized S3 score yield higher mean NMI values when $N = 1000$ (see Figure 3.7). Comparing the mean CCE in Figure 3.8 leads to better insights on why. With lower $N = 100$ and the same m, a parameter equal to half the average degree, the communities are being merged together by the algorithm whereas with higher $N = 1000$ and the same m, using the edge weights breaks ground truth communities apart. Since communities in nPSO graphs are defined based on angular separation, when the community sizes are large relative to the number of vertices in the graph, natural angular separation occurs randomly as discussed in Subsection 2.5.4 [136]. While the separation may not be as large as the pre-defined breaks, it makes sense that certain weighting schemes may highlight the separations. As shown in [185], the characteristic path lengths (see Definition 2.1.17) are smaller for nPSO graphs where $N = 100$ compared to nPSO graphs where $N = 1000$ that have the same m. Specifically, when the characteristic path length is 2 or

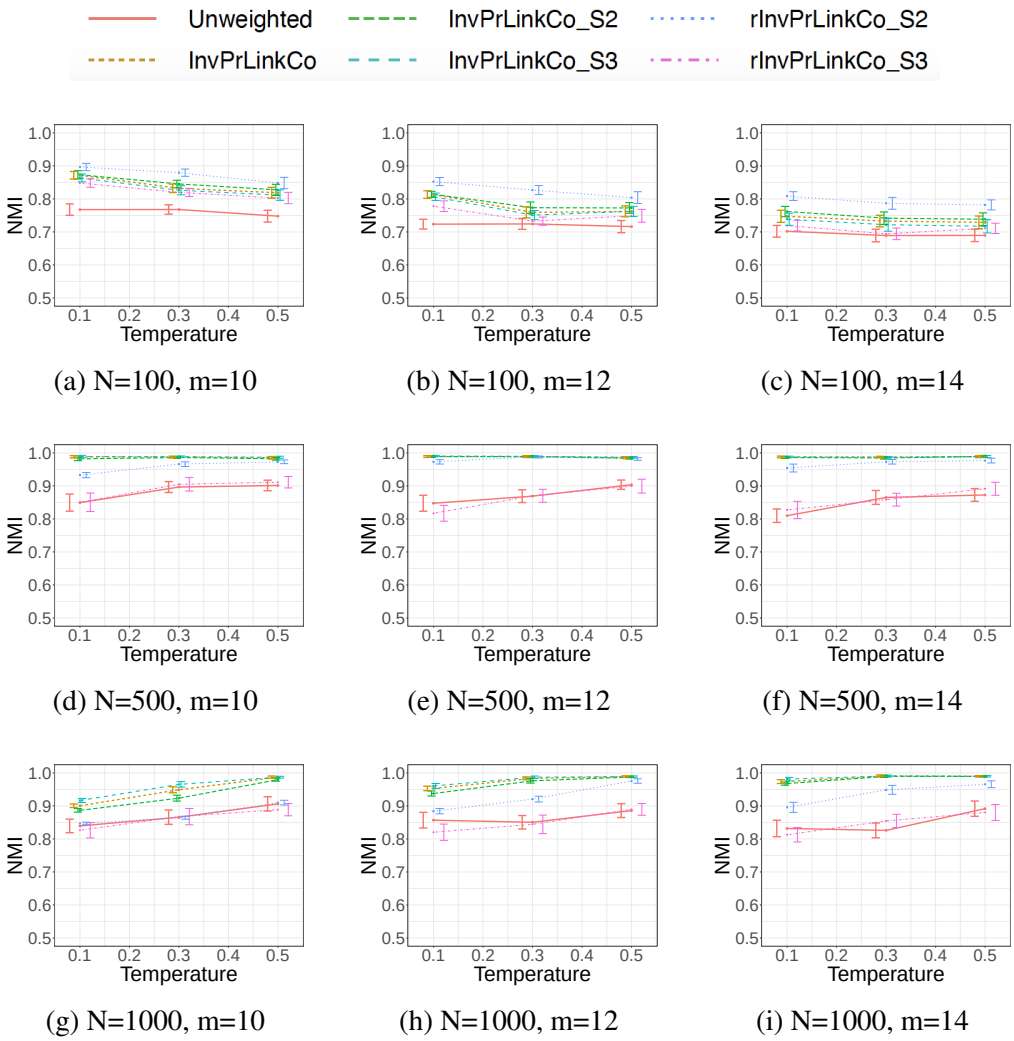


Figure 3.7: Community detection on nPSO graphs with 8 communities. The nPSO graphs have parameters $N = [100, 500, 1000]$, $m = [10, 12, 14]$, $T = [0.1, 0.3, 0.5]$, $\gamma = 3$ and angular coordinates sampled according to a Gaussian mixture distribution with equal proportions and 8 components. The combined Inversely Proportional Link Cohesion metric is compared to normalized and raw S2 and S3 Inversely Proportional Link Cohesion weights. The unweighted graph is also used as a reference. The plots report for each parameter combination the mean NMI with 95% confidence intervals from 30 random repetitions.

less, using just S2 may be beneficial. As a default, using the combined S2 and S3 weights yield the best overall performance with the InvPrLinkCo_S2 weights also yielding good results overall. It is assumed that the general behavior is similar for all link cohesion metrics.

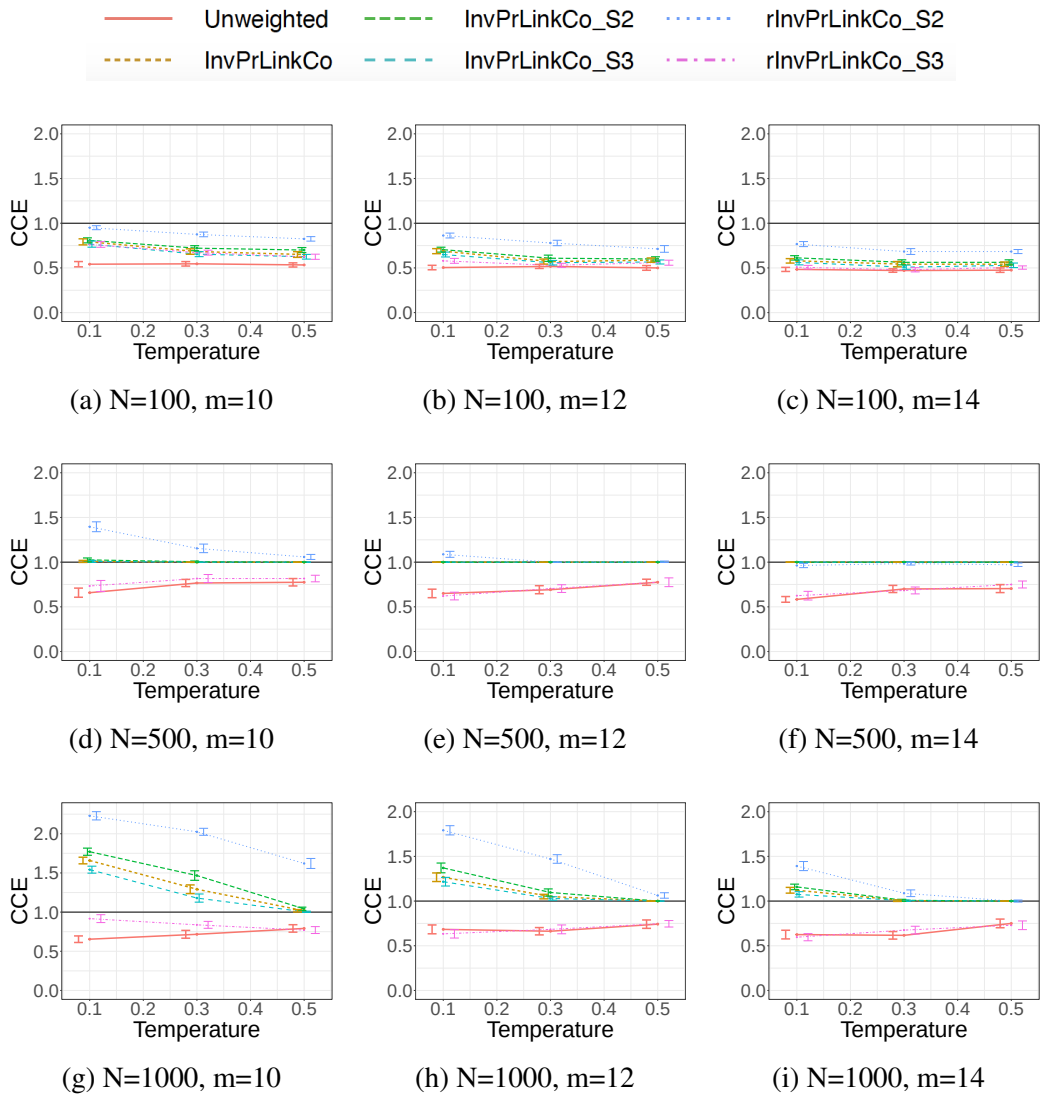


Figure 3.8: Community detection on nPSO graphs with 8 communities. The nPSO graphs have parameters $N = [100, 500, 1000]$, $m = [10, 12, 14]$, $T = [0.1, 0.3, 0.5]$, $\gamma = 3$ and angular coordinates sampled according to a Gaussian mixture distribution with equal proportions and 8 components. The combined Inversely Proportional Link Cohesion metric is compared to normalized and raw S2 and S3 Inversely Proportional Link Cohesion weights. The unweighted graph is also used as a reference. The plots report for each parameter combination the mean CCE with 95% confidence intervals from 30 random repetitions.

Claim 3.7.1. (CS) A link cohesion metric that combines scores from nearby alternate paths of length 2, which have been found to be better at resolving small communities (see Figures 3.8a, 3.8b and 3.8c), and scores from nearby alternate paths of length 3, which have been

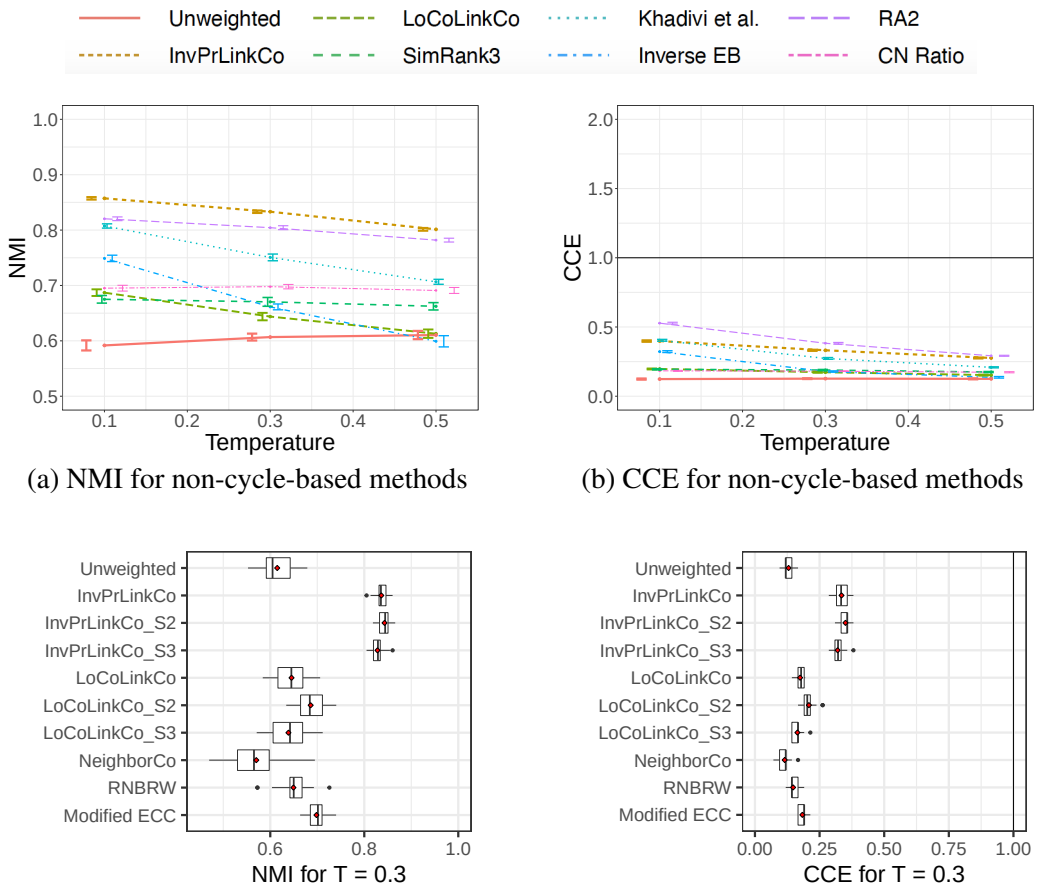
found to be better at keeping larger communities intact (see Figures 3.8g, 3.8h and 3.8i), was shown to provide improved performance when used to pre-weight the Louvain method across benchmark graphs with just small communities and benchmark graphs with just large communities (see Figure 3.7).

The benefit of using only S2 and S3 weights (as opposed to S1 weights) are shown in Subsection 4.6.2 and additional benefits of using both S2 and S3 for another link cohesion metric is shown in Subsection 4.6.7.

3.7.3 Louvain Method with Pre-Weighted nPSO Graphs

Most real-world networks do not have equal community sizes [141], which were assumed in the benchmark graphs of the previous section. In this section, community detection results from the Louvain method will be presented on benchmark graphs created for the purpose of reflecting particular real-world networks. Specifically, nPSO_EU was built to be similar to G_{EU} from Section 3.2 and nPSO_BHLP1 was modelled after parameters for LFR graphs introduced in [30] where the average degree was chosen based on its similarity to the average degree of the world wide web. In each case, InvPrLinkCo is compared against the baseline pre-weighting methods shown in Table 3.2 and LoCoLinkCo which is introduced in the next chapter (see Definition 4.4.7). Because some of the cycle-based methods are more computationally expensive, experiments are run on a single temperature, $T = 0.3$, instead of varying the value, as is done with the non-cycle-based methods. The InvPrLinkCo_S2, InvPrLinkCo_S3, along with LoCoLinkCo_S2 and LoCoLinkCo_S3 from the next chapter are also used as a comparison in the cycle-based methods. The community detection results for nPSO_EU are in Figure 3.9 and the community detection results for nPSO_BHLP1 are in Figure 3.10.

Although the two different benchmark graphs are quite different from each other in



(a) NMI for non-cycle-based methods

(b) CCE for non-cycle-based methods

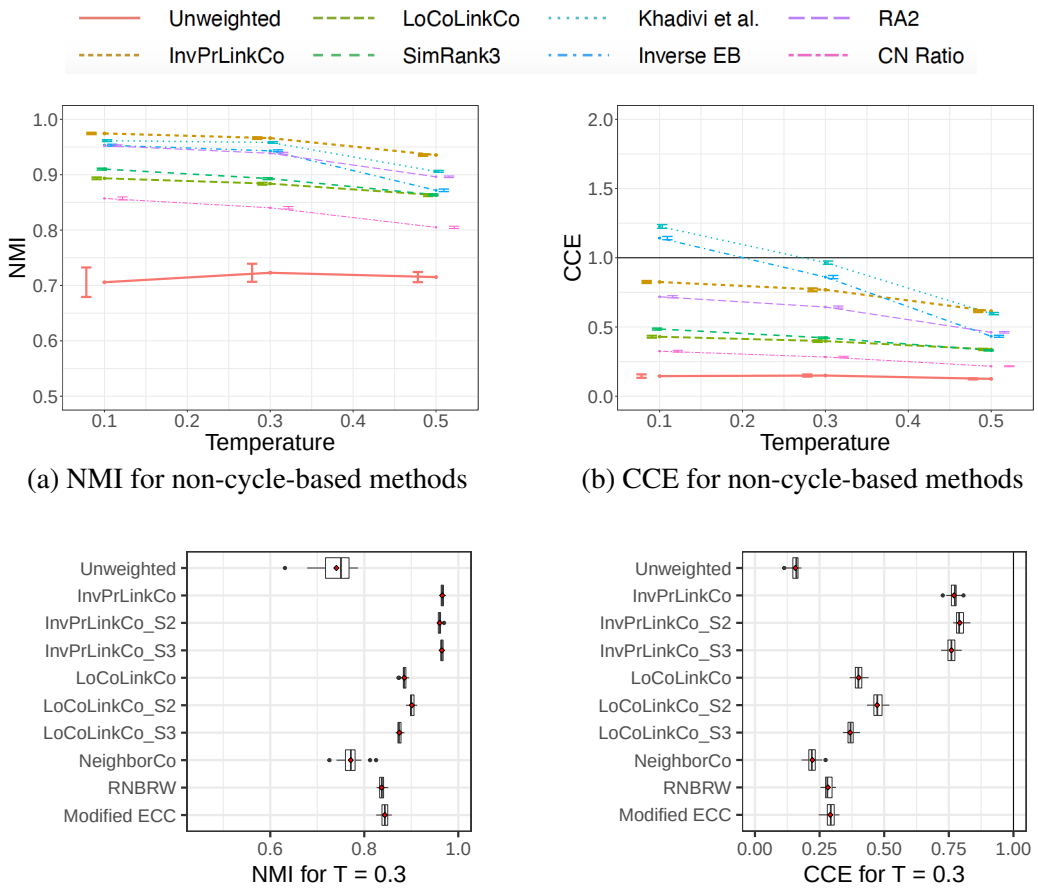
(c) NMI for cycle-based methods

(d) CCE for cycle-based methods

Figure 3.9: Community detection results for nPSO_EU using the Louvain method with different pre-weighting methods. The unweighted graph is also used as a reference. The plots in (a) and (b) report the mean NMI and mean CCE for non-cycle-based methods with 95% confidence intervals from 30 random repetitions. The plots (c) and (d) report for T=0.3 the boxplots of NMI and CCE for cycle-based methods over 30 random repetitions.

terms of their average degree, their characteristic path length (see Table 3.1 and Figure 2.4), and their assortativity (nPSO_EU graphs are disassortatively mixed while nPSO_BHLP1 graphs are assortatively mixed), InvPrLinkCo performs better than the other baseline pre-weighting methods.

Claim 3.7.2. (CS) InvPrLinkCo can improve the communities detected by the Louvain method in a wide range of real-world inspired parameter settings for nPSO graphs with unequal community sizes (see Figures 3.7, 3.8, 3.9, and 3.10) and improved the communi-



(a) NMI for non-cycle-based methods

(b) CCE for non-cycle-based methods

(c) NMI for cycle-based methods

(d) CCE for non-cycle-based methods

Figure 3.10: Community detection results for nPSO_BHLP1 using the Louvain method with different pre-weighting methods. The unweighted graph is also used as a reference. The plots in (a) and (b) report the mean NMI and mean CCE for non-cycle-based methods with 95% confidence intervals from 30 random repetitions. The plots (c) and (d) report for T=0.3 the boxplots of NMI and CCE for cycle-based methods over 30 random repetitions.

ties detected by the Louvain method more than other baseline pre-weighting methods (see Figures 3.9 and 3.10).

Interestingly, using the InvPrLinkCo_S2 weights by themselves also works well on these more realistic benchmarks likely because there tends to be a larger number of smaller communities and a few larger communities [141]. As will be seen in the next section, the InvPrLinkCo_S2 weights can be commuted faster than the combined weights, so it certain situations it may be beneficial to use just the S2 weights instead of both.

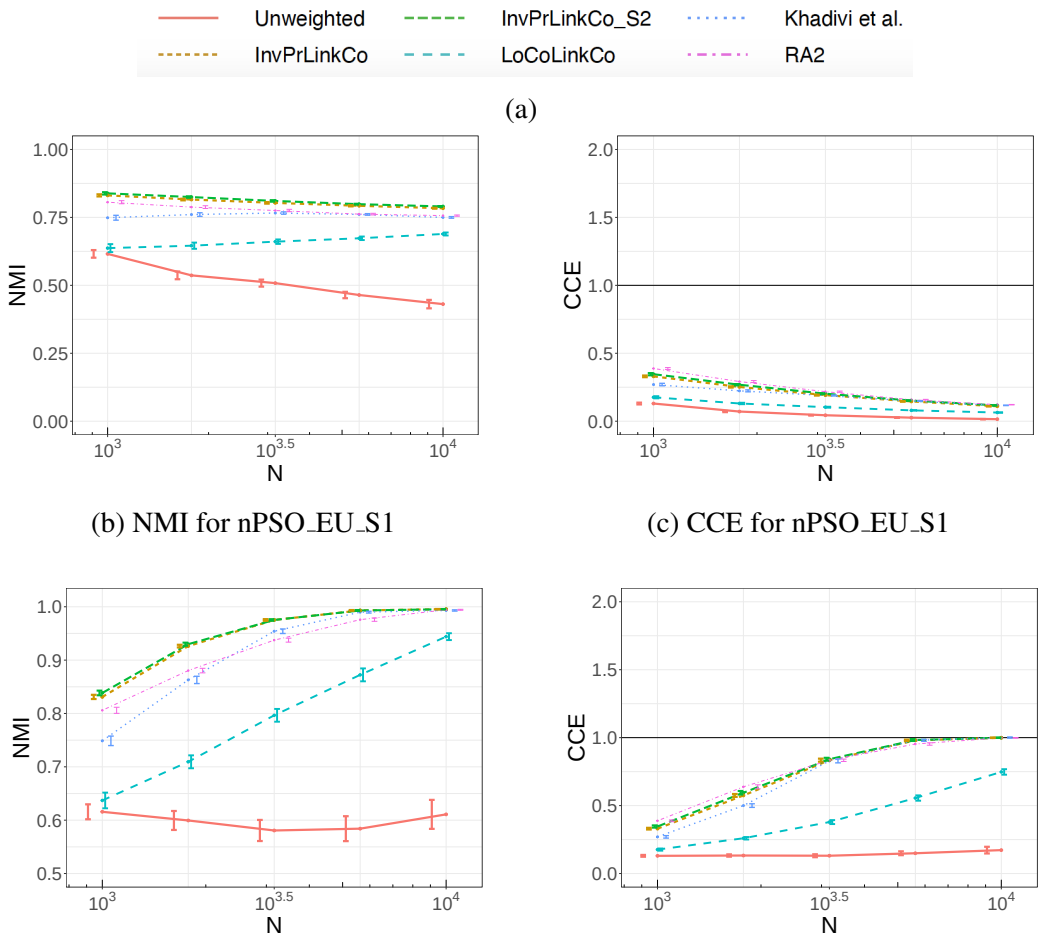
Nearly all of the methods undercount the number of true communities, particularly in the denser benchmark. This is as expected from Claim 3.2.2. The next best performing baseline pre-weighting methods are RA2, the local repulsion attraction metric [189], and the measure proposed by Khadivi et al. that combines the global edge betweenness metric and local metric common neighbor ratio metric [130, 131].

Claim 3.7.3. (CS) Taking the degrees of the nearby alternate paths into account creates cycle-based weights that are better than the other cycle-based baselines (see Figures 3.9c and 3.10c) which cannot differentiate between triangles and rectangles among high degree vertex often from different communities that create a rich-club due to their popularity and triangles and rectangles among low degree vertices tied together due to their similarity (see Figures 3.9d and 3.10d).

It should also be noted that while LoCoLinkCo, the link cohesion metric introduced in the next chapter, usually provides a benefit over an unweighted approach, it does not perform as well as some of the other pre-weighting methods.

3.7.4 Scalability of Inversely Proportional Link Cohesion

The scalability experiments were conducted on nPSO graphs similar to the nPSO_EU graphs. The nPSO model fails to generate graphs with much more than 10,000 vertices, therefore graphs are scaled from 1,000 to 10,000 using logarithmic spacing. Two different versions of scaling the parameters are analyzed: nPSO_EU_S1 which scales number of communities proportional to the size of the graph and nPSO_EU_S2 which keeps the number of communities the same as nPSO_EU. Both scaling approaches use a constant $T = 0.3$. The community detection results from the Louvain method are shown in Figure 3.11 with the corresponding compute times shown in Figure 3.12. Figure 3.12 also contains compute times for the cycle-based methods for nPSO_EU and nPSO_BHLP1 with $T = 0.3$.



(a)

(b) NMI for nPSO_EU_S1

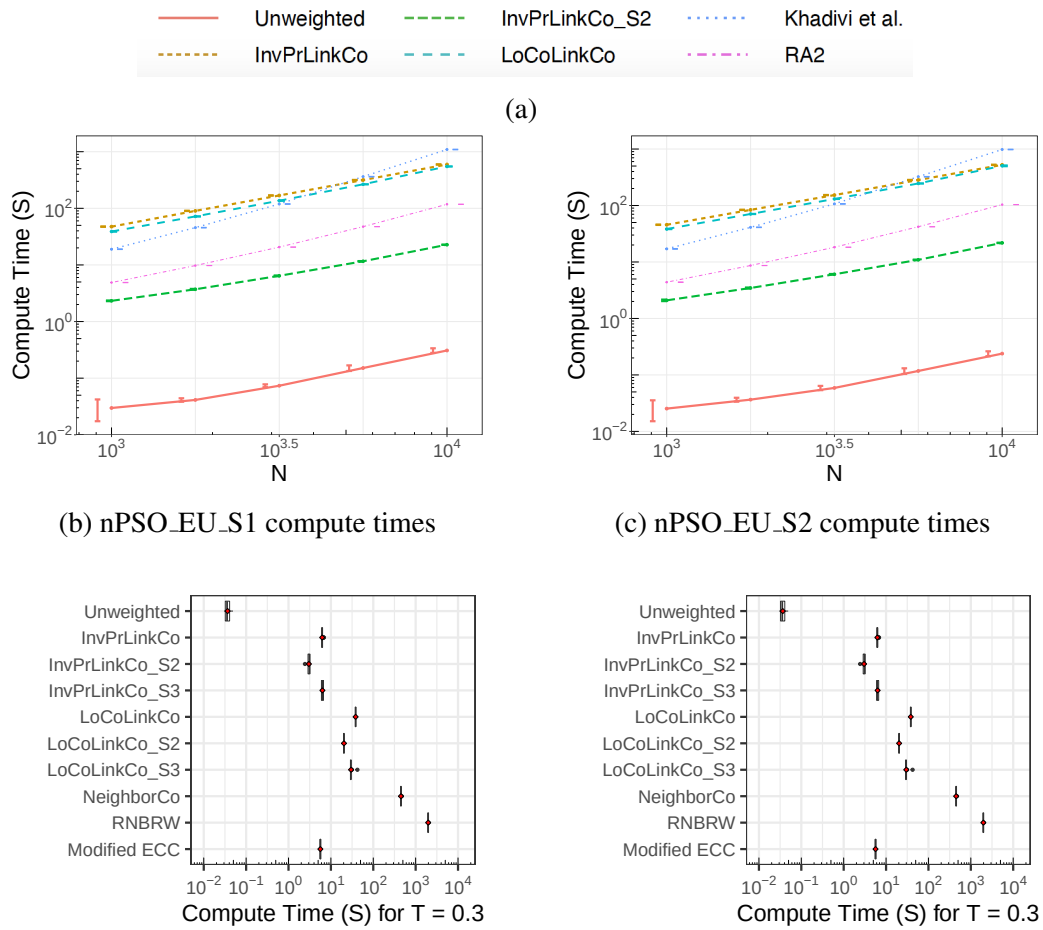
(c) CCE for nPSO_EU_S1

(d) NMI for nPSO_EU_S2

(e) CCE for nPSO_EU_S2

Figure 3.11: Community detection results for scaled nPSO_EU graphs using the Louvain method with the selected pre-weighting methods. The unweighted graph is also used as a reference. The plots in (a) and (b) report the mean NMI and CCE for nPSO_EU_S1 with 95% confidence intervals from 30 random repetitions. The plots in (c) and (d) report the mean NMI and CCE for nPSO_EU_S2 with 95% confidence intervals from 30 random repetitions.

The performance of the algorithms degrade for the unweighted Louvain method (with a confidence interval that the true difference in means is not equal to 0 being (-0.2061,-0.1635) using a Welch Two Sample t-test), the Louvain method weighted with InvPrLinkCo (with a confidence interval that the true difference in means is not equal to 0 being (-0.0517, 0.0429) using a Welch Two Sample t-test), the Louvain method weighted with InvPrLinkCo_S2 (with a confidence interval that the true difference in means is not equal



(a)

(b) nPSO_EU_S1 compute times

(c) nPSO_EU_S2 compute times

(d) nPSO_EU compute times with T=0.3 using cycle-based pre-weighting methods

(e) nPSO_BHLP1 compute times with T=0.3 using cycle-based pre-weighting methods

Figure 3.12: Compute times for nPSO graphs using the Louvain method. The plots in (a) and (b) contain the mean compute times with 95% confidence intervals from 30 random repetitions for nPSO_EU_S1 and nPSO_EU_S2 using the Louvain method with the selected pre-weighting methods. The plots in (c) and (d) contain the boxplots of compute times from 30 random repetitions for the cycle-based methods on nPSO_EU graphs and nPSO_BHLP1 graphs with T=0.3.

to 0 being (-0.0530, -0.0434) using a Welch Two Sample t-test), and the Louvain method weighted with RA2 (with a confidence interval that the true difference in means is not equal to 0 being (-0.0557, -0.0430) using a Welch Two Sample t-test) as the size of the graphs increase when the community sizes are proportional to the size of the graphs (nPSO_EU_S1). The nPSO_EU graphs have very small communities that are hard to resolve, and keeping

the community sizes proportional to the size of the graphs enables the very small community sizes to persist. The resolution limit states that communities smaller than $\sqrt{W\xi/2}$ total edge weight may fail to be resolved, where ξ is the maximum weight of an intercommunity edge [30, 101]. Therefore, it makes sense that in general as the size of the graphs grow the performance degrades. Interestingly, the communities detected using the LoCoLinkCo as the pre-weighting method improve (with a confidence interval that the true difference in means is not equal to 0 being (0.0358, 0.0682) using a Welch Two Sample t-test). Based on how the LoCoLinkCo weights are designed, this makes sense because although there are a constant number of communities, they become more spread out as the size of the graphs grow as there is a constant average degree. The performance of the Louvain method weighted with the method proposed by Khadivi et al. did not significantly change (with a confidence interval that the true difference in means is not equal to 0 being (0.0085, 0.0109) using a Welch Two Sample t-test).

When the number of communities remain constant as the size of the graphs grow (nPSO_EU_S2), the performance increases for all of the tested pre-weighting methods. Interestingly, the performance of the Louvain method on the unweighted graphs does not improve. As the size of the graphs increase, the sizes of the communities grow since there are a constant number of communities. Therefore, the resolution limit becomes less of a problem [101]. With proper weighting, the ξ is small enough such that the Louvain method is able to resolve the smaller communities of the larger graphs [30] but in an unweighted network, $\xi = 1$ and the smaller communities are still unable to be resolved. The pre-weighting method proposed by Khadivi et al. improves faster than RA2 for larger graphs, although at $N = 10,000$ both algorithms obtain near perfect community detection similar to InvPrLinkCo. LoCoLinkCo is the only pre-weighting method tested that does not obtain near perfect detected communities at $N = 10,000$ due to merging communities, which is somewhat expected based on how LoCoLinkCo is designed compared to how

Popularity×Similarity graphs are formed.

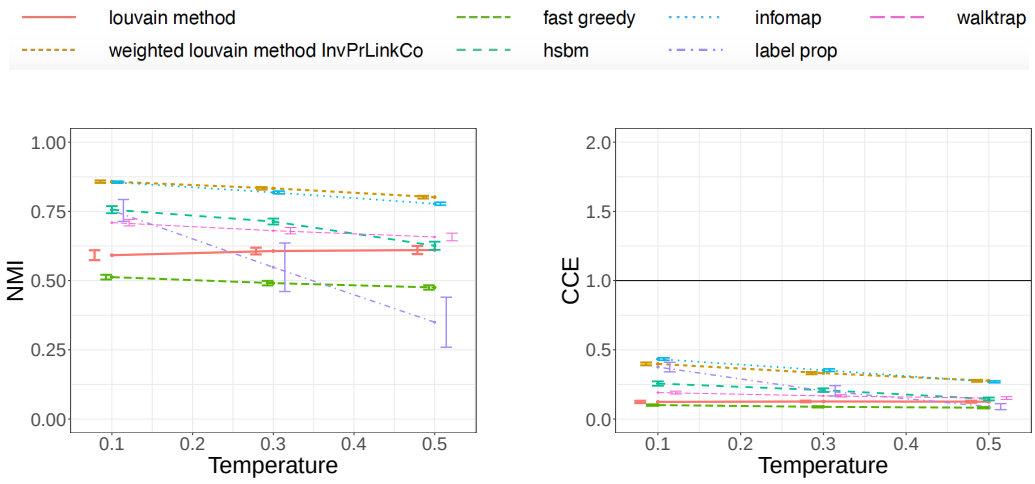
InvPrLinkCo_S2 performs about the same as InvPrLinkCo and is significantly faster, as shown in Figure 3.12.

Claim 3.7.4. (CS) Although the factors are different, both InvPrLinkCo and InvPrLinkCo_S2 scale similarly to the Louvain method on unweighted graphs (see Figures 3.12b and 3.12c).

3.7.5 Experiments with Other Community Detection Algorithms

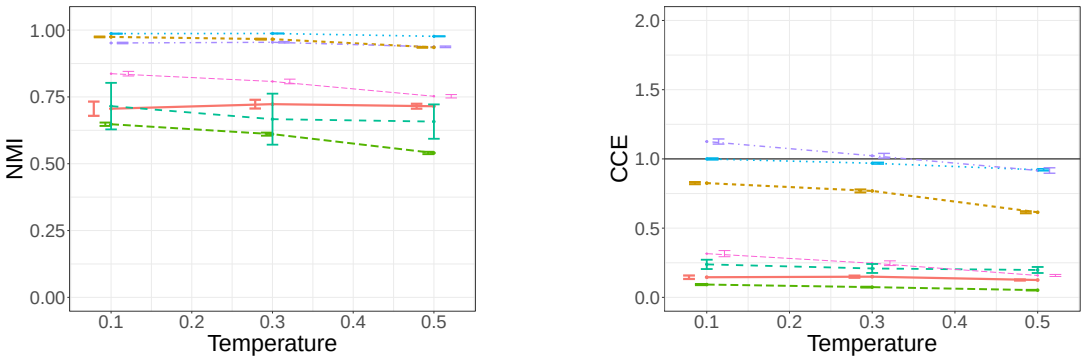
While pre-weighting the Louvain method with InvPrLinkCo provides better results than other pre-weighting methods and the unweighted Louvain method, there are other community detection options that may perform better on the benchmark nPSO graphs. First, the InvPrLinkCo weighted and unweighted Louvain method is compared against the other community detection methods listed in Table 3.3 on nPSO_EU and nPSO_BHLP1 with results shown in Figure 3.13.

While the Louvain method pre-weighted with InvPrLinkCo obtains higher mean NMI than Infomap on nPSO_EU with statistically significance for $T = 0.3$ with a 95% confidence interval from a paired t-test equal to (0.0078,0.0221) and for $T = 0.5$ with a 95% confidence interval from a paired t-test equal to (0.0168,0.0307), it is not true for $T = 0.1$ which has a 95% confidence interval from a paired t-test equal to (-0.0030, 0.0066). Infomap obtains higher mean NMI for nPSO_BHLP1 for all temperatures and Label Prop obtains higher mean NMI than the Louvain method pre-weighted with InvPrLinkCo on nPSO_BHLP1 with statistically significance for $T = 0.1$ with a 95% confidence interval from a paired t-test equal to (0.0120,0.0255) and for $T = 0.3$ with a 95% confidence interval from a paired t-test equal to (0.0085,0.0150), it is not true for $T = 0.5$ which has a 95% confidence interval from a paired t-test equal to (-0.0046, 0.0008). The InvPrLinkCo are not specific to the Louvain method and have the potential to improve the communities



(a) NMI for nPSO_EU Cluster Comparison

(b) CCE for nPSO_EU Cluster Comparison



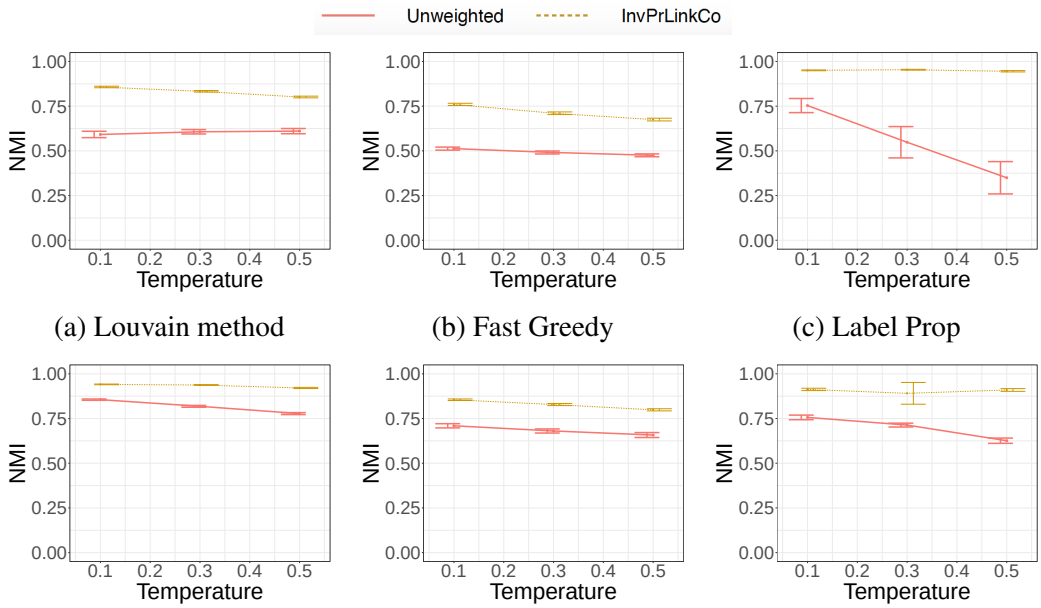
(c) NMI for nPSO_BHLP1 Cluster Comparison

(d) CCE for nPSO_BHLP1 Cluster Comparison

Figure 3.13: Community detection results for nPSO graphs using the Louvain method plus InvPrLinkCo compared to other unweighted community detection methods described in Section 2.3. The plots in (a) and (b) report the mean NMI and CCE for nPSO_EU with 95% confidence intervals from 30 random repetitions. The plots in (c) and (d) report the mean NMI and CCE for nPSO_BHLP1 with 95% confidence intervals from 30 random repetitions.

detected by other methods. Therefore, Figures 3.14, 3.15, 3.16, and 3.17 show results from experiments testing the potential performance improvements from using InvPrLinkCo to pre-weight the other community detection methods. All experiments use 30 random repetitions and the 95% confidence intervals are reflected in the plots.

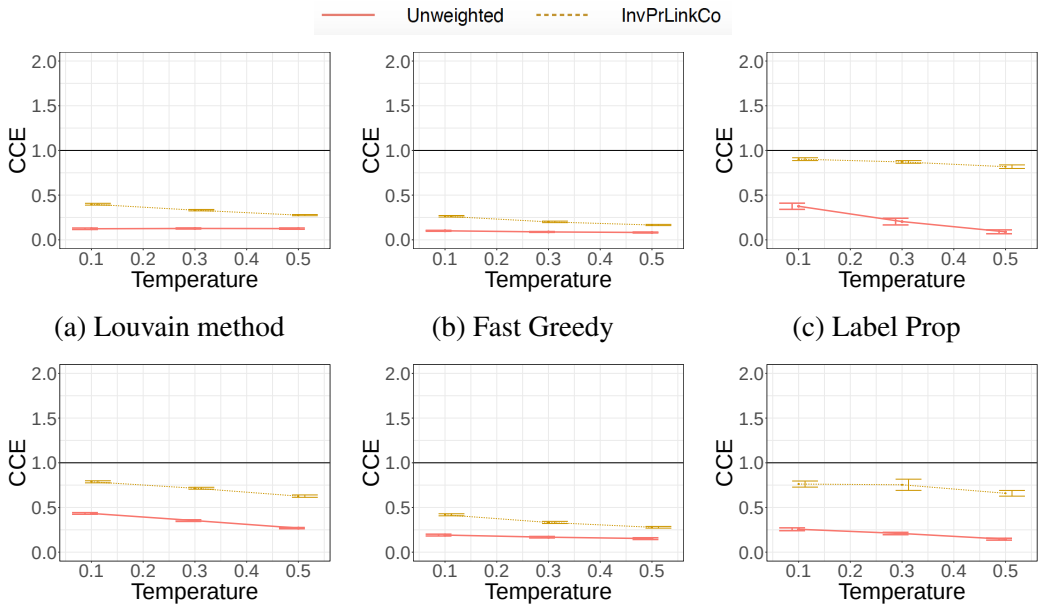
Experiments using InvPrLinkCo to pre-weight a variety of community detection algorithms lead to a number of observations.



(a) Louvain method (b) Fast Greedy (c) Label Prop

(d) Infomap (e) Walktrap (f) HSBM

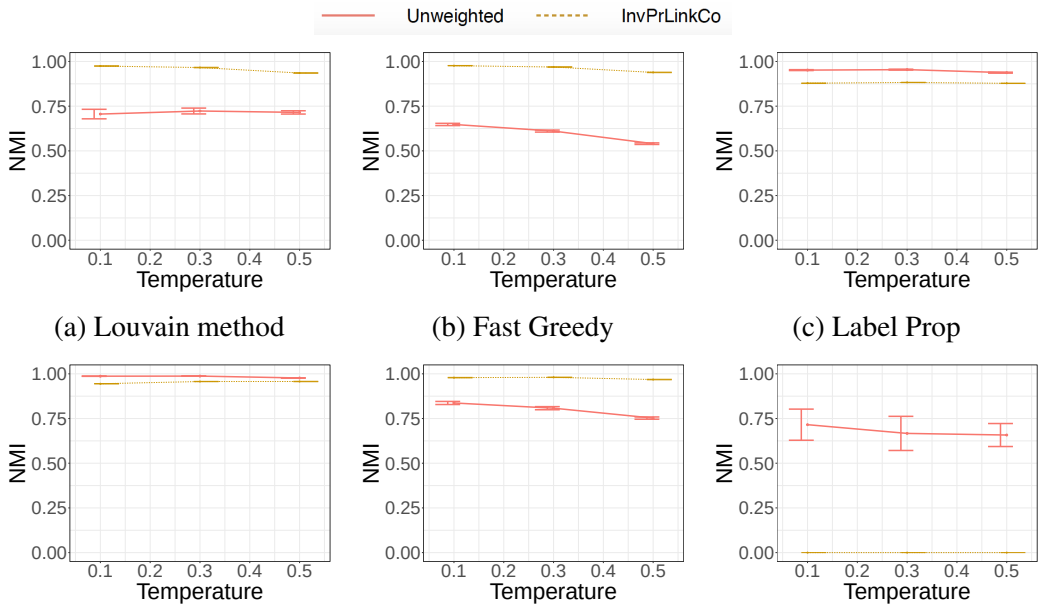
Figure 3.14: NMI for InvPrLinkCo verses unweighted community detection on nPSO_EU.



(a) Louvain method (b) Fast Greedy (c) Label Prop

(d) Infomap (e) Walktrap (f) HSBM

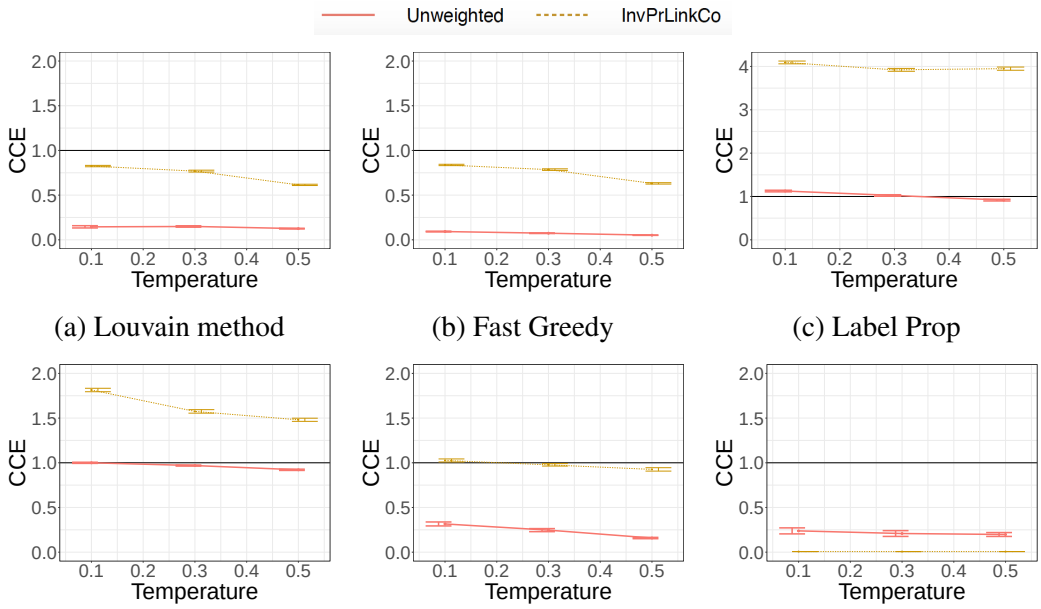
Figure 3.15: CCE for InvPrLinkCo verses unweighted community detection on nPSO_EU.



(a) Louvain method (b) Fast Greedy (c) Label Prop

(d) Infomap (e) Walktrap (f) HSBM

Figure 3.16: NMI for InvPrLinkCo verses unweighted community detection on nPSO_BHLP1.



(a) Louvain method (b) Fast Greedy (c) Label Prop

(d) Infomap (e) Walktrap (f) HSBM

Figure 3.17: CCE for InvPrLinkCo verses unweighted community detection on nPSO_BHLP1.

Claim 3.7.5. (CS) Pre-weighting the highly connected nPSO_EU graphs with InvPrLinkCo provide higher mean NMI values (see Figure 3.14) and more accurate predictions on the number of communities (see Figure 3.15) for all community detection methods tested (e.g. the Louvain method, the Fast Greedy algorithm, Label Prop, Infomap, Walktrap, and HSBM).

Label Prop pre-weighted with InvPrLinkCo obtains higher mean NMI values for all other methods except HSBM pre-weighted with InvPrLinkCo for $T = 0.3$. That said, Label Prop is able to most closely predict the correct number of communities even compared to HSBM. The fact that InvPrLinkCo is able to improve the performance of all community detection methods implies that all methods had some sort of resolution limit, as the improvement was due to the ability to resolve more communities.

For the sparser nPSO_BHLP1 graphs, not all community detection methods improved when pre-weighted with InvPrLinkCo, although the Louvain method, Fast Greedy, and Walktrap do improve significantly. Pre-weighting modularity-based methods such as the Louvain method and the Fast Greedy algorithm properly is expected to resolve some issues with modularity in general, which will be discussed in Subsection 4.6.2, and specifically for nPSO graph which was stated in Claim 3.2.2 and the performance results on the benchmark graphs reflect this.

Claim 3.7.6. (CS) InvPrLinkCo has been shown to consistently improve the communities detected by algorithms that maximize modularity (e.g. the Louvain method and Fast Greedy algorithm) on all of the nPSO benchmark graphs tested, which include graphs with only small communities (see (a)-(c) in FIgures 3.7 and 3.8), graphs with only large communities (see (g)-(i) in FIgures 3.7 and 3.8)), and both highly connected (see Figures 3.14a, 3.14b, 3.15a, 3.15b, and 3.11) and sparse graphs (see Figures 3.16a, 3.16b, 3.17a, 3.17b) with unequal community sizes.

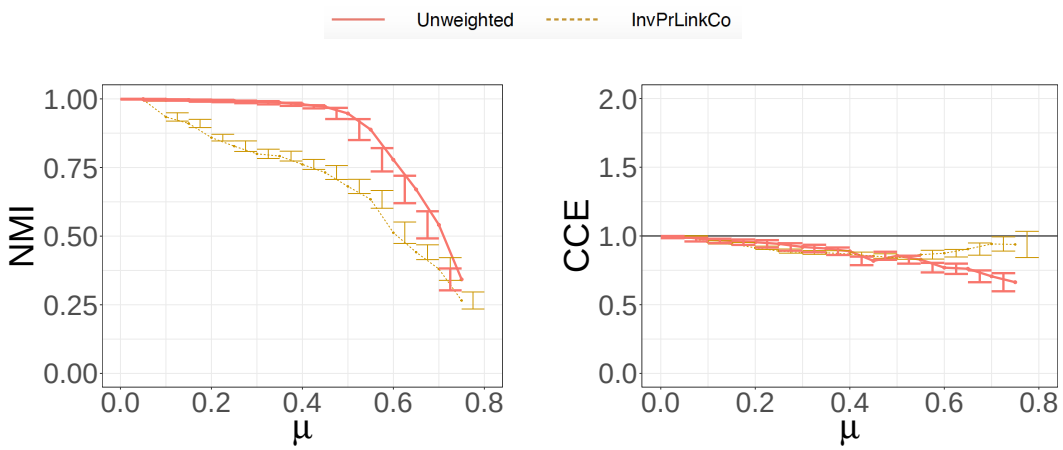
HSBM threw an exception when pre-weighted with InvPrLinkCo. Infomap and Label Prop had obtained high mean NMI on the unweighted graphs with the ability to accurately determine the correct number of communities whereas pre-weighting with InvPrLinkCo causes these methods to break true communities apart.

Claim 3.7.7. (CS) Certain non-modularity based community detection algorithms obtain degraded performance when InvPrLinkCo is used to pre-weight sparse networks, with a hypothesis for why this occurs being that both Infomap and Label Prop are able to utilize cycles of higher order into their solution on unweighted graphs, therefore focusing on only shorter cycles when the characteristic path length is relatively high causes a degradation in the performance.

3.7.6 Graphs with Degree Assortative Community Structure

Based on how the InvPrLinkCo weights are derived and how Popularity$\times$Similarity graphs are formed, it is expected that pre-weighting nPSO graphs and networks influenced by Popularity$\times$Similarity optimization would help improve the communities detected by traditional community detection algorithms when they are highly connected, as seen in the previous section. Highly connected graphs of this type yield low characteristic path lengths and many short cycles that particularly involve high degree vertices due to their popularity even if they are not part of the same community [228, 253]. An alternative paradigm for community structure is degree assortative community structure, where higher degree vertices are only found in the larger communities, as seen in Figure 1.1a. Figure 3.18 shows the community detection results of the unweighted Louvain method compared to the Louvain method pre-weighted with InvPrLinkCo on a highly connected benchmark with degree assortative community structure.

In the largest communities of a graph with degree assortative community structure, a



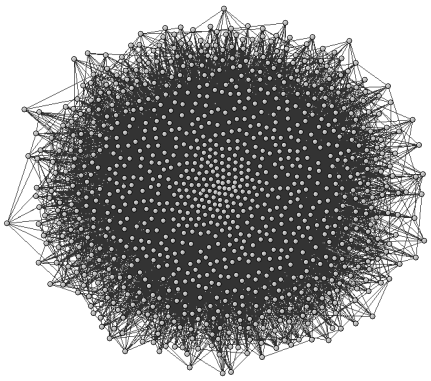
(a) NMI of the Louvain method on LFR_SSGC (b) NMI of the Louvain method on LFR_SSGC

Figure 3.18: NMI and CCE of the Louvain method with InvPrLinkCo verses an unweighted Louvain method on highly connected LFR benchmark LFR_SSGC (see Table 4.1).

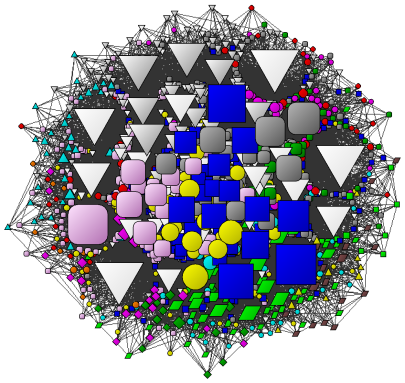
disproportionate number of high degree vertices would be members meaning connections with these vertices are as meaningful as connections to lower degree vertices, particularly if there are many triangles and rectangles. InvPrLinkCo would take these connections into account, but heavily discount them. This causes the largest communities to have many internal edges to have low weight, which when used as pre-weighting to community detection methods like the Louvain method can degrade performance as seen in Figure 3.18. Subsection 4.6.3 illustrates the InvPrLinkCo weights on a toy network with degree assortative community structure, which makes the degradation in NMI understandable. While it can be seen that the mean NMI degrades with InvPrLinkCo in Figure 3.18a, the number of communities does not change significantly as shown in Figure 3.18b.

Visualizing the MDCore of the graph (see Section 3.5) can provide insights into the particular vertices that have the lowest corresponding edge weights, making them more likely to be the vertices placed into the wrong communities. Specifically, Figure 3.19 is an example highly connected graph with degree assortative community structure built using LFR graphs, specifically, the LFR_SSGC benchmark (see Table 4.1) whose community re-

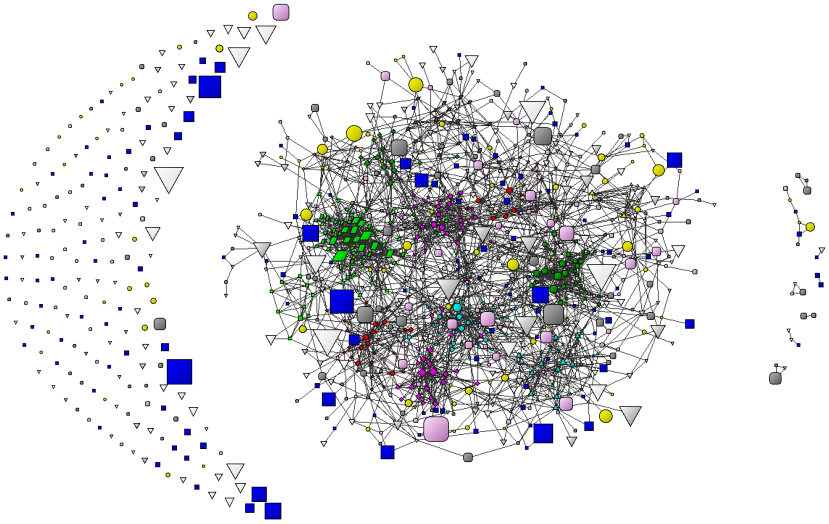
sults were shown in Figure 3.18. Figure 3.19c illustrates the MDCore of the graph with a large number of the isolated vertices coming from the largest communities, as can be determined by looking at the community of the highest degree vertices (the size of the vertices correspond to the degree of the supergraph). Vertices part of the largest communities that are part of the largest connected component are not tightly bunched together, as are many of the smaller communities made up of only low degree vertices.



(a) LFR_SSGC graph with $\mu = 0.6$ visualized using Pointillist



(b) Community Structure of LFR_SSGC graph with $\mu = 0.6$ visualized using Pointillist



(c) MDCore of LFR_SSGC graph with $\mu = 0.6$ visualized using Pointillist

Figure 3.19: Community structure of the highly connected LFR benchmark LFR_SSGC (see Table 4.1) with $\mu = 0.6$ and its MDCore. (a) LFR_SSGC with $\mu = 0.6$ graph (b) Community Structure of LFR_SSGC with $\mu = 0.6$ (c) MDCore of the LFR_SSGC with $\mu = 0.6$ with InvPrLinkCo [239].

Claim 3.7.8. (CS) Using InvPrLinkCo weights on highly connected graphs with degree assortative community structure may cause algorithms that maximize modularity to place vertices that are part of the largest ground truth communities into the wrong predicted communities (see Figures 3.18b and 3.19c), leading to a degredation in NMI (see Figure 3.18a).

Because InvPrLinkCo provides large benefits to nPSO graphs with hierarchical community structure but not LFR graphs with degree assortative community structure, it is important to make efforts to understand the type of community structure to expect from a network before using the pre-weighting method and to understand that communities with a disproportionate number or higher degree vertices may fail to be resolved when InvPrLinkCo is used to pre-weight a community detection method.

3.7.7 Communities in the Email Network from the Research Institute

Now that it is understood how pre-weighting affects the performance of community detection methods on benchmark graphs, the performance will be measured on the real world network, G_{EU}, discussed in Section 3.2. The nPSO_EU benchmark was created to be similar to G_{EU} when $T = 0.3$ (see Claim 3.2.1). The performance of all the community detection methods with and without pre-weighting using InvPrLinkCo and all the other baseline pre-weighting methods with the Louvain method was measured on their ability to predict the ground truth communities, corresponding to departments a person is a member of, in G_{EU}. The NMI values of each approach was compared to the mean NMI of the approach obtained on *nPSO_EU* when $T = 0.3$, with results shown in Figure 3.20. Three methods are left off of the plot, specifically the weighted and unweighted HSBM which threw an exception probably due to the isolated vertices in the graph and the unweighted Label Prop which obtained an NMI of 0.03 due to placing all of the vertices in the largest

connected component into a single community.

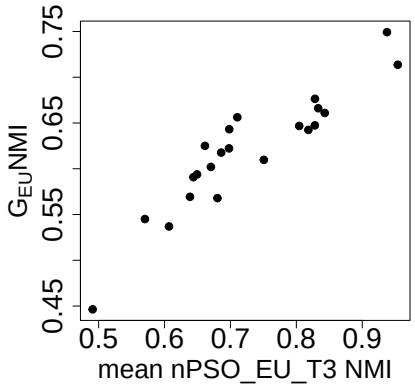


Figure 3.20: Comparison of the mean NMI from the benchmark graphs to the NMI from G_{EU} for all weighted and unweighted approaches tested on the benchmark graphs.

Claim 3.7.9. (CS) While the NMI values on the real-world network G_{EU} (a simplified version of the email-Eu-core network from SNAP datasets [148]) are consistently lower than the mean NMI on the benchmark graphs which were created to be similar to G_{EU}, the linear correlation between the two values is 0.92 (see Figure 3.20) across all of the community detection methods that provided reasonable results.

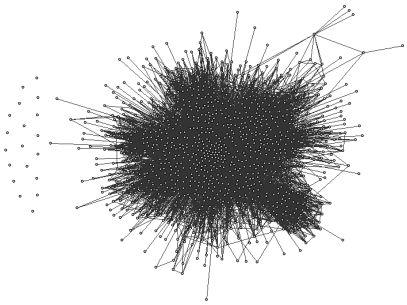
The two top performing approaches switch ordering, with Infomap pre-weighted with InvPrLinkCo obtaining an NMI of 0.75 and Label Prop pre-weighted with InvPrLinkCo obtaining an NMI of 0.71. Walktrap pre-weighted with InvPrLinkCo obtains the third highest NMI of 0.68, the Louvain method pre-weighted with InvPrLinkCo obtains the fourth highest NMI of 0.67, and the Louvain method pre-weighted with InvPrLinkCo_S2 obtains the fifth highest NMI of 0.66.

Since there is ground truth to G_{EU}, further analysis can be conducted to understand the consistent degradation in performance on the real-world network compared to the benchmark graphs. Figure 3.21 contains visualizations of G_{EU} and its ground truth communities.

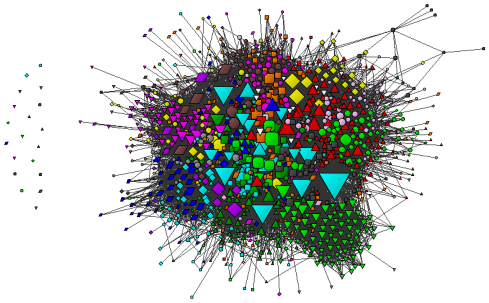
Figure 3.21c illustrates the MDCore of G_{EU} (see Section 3.5) and it can be seen that many of the highest degree vertices are part of the same community, which is broken apart in the MDCore. Interestingly, it seems the community is not as densely connected as the highest degree vertices typically are in nPSO graphs, a shown in Figure 3.2a. The induced subgraph generated using the vertices of this community along with the vertices of three other communities are shown in Figure 3.21d.

The fact that many of the highest connected vertices are all part of the same community is different than nPSO graphs. Specifically, 10 of the top 23 highest degree vertices are part of the ground truth community corresponding to department 36 including all of top 6 highest degree vertices. This is also different than degree assortative graphs, as department 36 is the 18th ranked department in terms of size, only a fifth of the size of the largest department. By randomly selecting three other departments of smaller and larger sizes to department 36, it can be seen that they all strongly interact with department 36. Departments 19 and 21 interact a fair bit with one another, whereas department 3 does not interact with departments 19 and 21 aside from a single connection. It is additionally interesting to note that 13 of the vertices in department 21 do not interact at all with other members in their department which is also dissimilar to what would be expected in an nPSO graph. It would be hard for a community detection algorithm to place the 13 vertices into the correct department when there are no connections to other vertices in the department. This analysis was easy to do due to the interpretable visualization of MDCore.

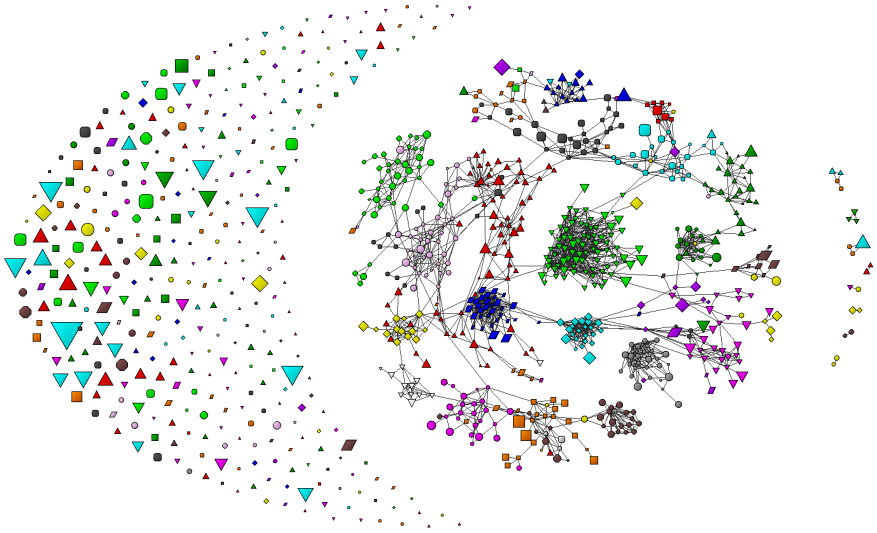
Claim 3.7.10. (CS) As demonstrated in this subsection and Subsection 3.7.6, the MDCore (see Definition 3.5.1) of a graph can be used to easily understand what errors a community detection algorithm would likely make when using a link cohesion metric (see Figures 3.19c and 3.21c and related commentary).



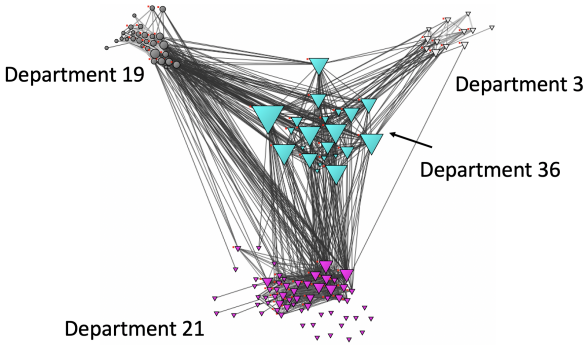
(a) G_{EU} graph visualized using Pointillist



(b) Community Structure of G_{EU} graph visualized using Pointillist



(c) MDCore of G_{EU} visualized using Pointillist



(d) Selected communities in G_{EU} graph where higher edge weights are highlighted in white

Figure 3.21: Community structure of G_{EU} and its MDCore. (a) G_{EU} graph (b) Community Structure of G_{EU} (c) MDCore of G_{EU} with InvPrLinkCo [239]. (d) Selected communities in G_{EU} graph where higher edge weights are highlighted in white.

3.8 Conclusions

In this chapter, we make a case for studying nPSO graphs (see Claim 3.2.1), but without proper weighting algorithms that maximize modularity would have trouble resolving smaller communities (see Claim 3.2.2). A new link cohesion metric (see Definition 3.4.1) called Inversely Proportional Link Cohesion (see Definition 3.4.8) is built on the proposition that edges that have a higher expected number of edges in a degree-preserving random graph are less likely to be an indication of cohesiveness (see Claim 3.4.7). The metric is a cycle-based method (see Claim 3.7.3) that is particularly suited for nPSO graphs (see Claim 3.4.10). Although the factors are different, Inversely Proportional Link Cohesion is observed to scale similarly to the Louvain method (see Claim 3.7.4) as expected (see Section 3.6) with a benefit that it is easily parallelizable since each edge cam be computed independently.

We provide motivation for combining nearby alternate paths of length 2 and 3 into a single metric (see Claim 3.7.1) which is shown to outperform other baseline pre-weighting methods (see Claim 3.7.2) including other cycle-based methods (see Claim 3.4.9), in diverse nPSO graphs which were chosen to be similar to real-world networks. Additionally, Inversely Proportional Link Cohesion is shown to consistently improve all community detection methods tested (e.g. the Louvain method, the Fast Greedy algorithm, Label Prop, Infomap, Walktrap, and HSBM) on a highly connected nPSO benchmark (see Claim 3.7.5) and to consistently improve the communities detected by algorithm that maximize modularity (e.g. the Louvain method and the Fast Greedy algorithm) on a highly connected and sparse benchmark graphs (see Claim 3.7.6). Not all methods yielded higher NMIs on the sparse nPSO benchmark, with a hypothesis being that Infomap and Label Prop are able to make use of higher order cycles (see Claim 3.7.7). It was also shown that some effort should be taken to understand the expected community structure of a real-world network, as using

Inversely Proportional Link Cohesion to pre-weight highly connected graphs with degree assortative community structure can lead to degraded communities (see Claim 3.7.8). A preliminary analysis on how to determine the community structure on a network without ground truth is explored in Subsection 4.6.7 of the next chapter.

The performance of all of the community detection methods with and without weighting on G_{EU}, the simplified version of the real-world email-Eu-core network from SNAP datasets [148], was compared against the mean performance on the benchmark graph created to be similar to G_{EU} (see Claim 3.2.1) and they were found to have a strong linear correlation (see Claim 3.7.9). The Maximum Density Core (MDCore – see Definition 3.5.1) was shown to highlight the community structure in graphs with the ability to draw some insights from visualizing it (see Claim 3.7.10) while not falsely creating community structure when used on random graphs (see Claim 3.5.2).

Because Inversely Proportional Link Cohesion was shown to degrade certain graphs with degree assortative community structure (Claim 3.7.8)), the next chapter introduces a new link cohesion metric (see Definition 4.4.7) that is design to perform better on networks with such structure. While this chapter and the next are focused on the benefits of using link cohesion metrics to improve the communities detected by traditional community detection methods, link cohesion metrics may also be used for various anomaly detection purposes (see Section 2.4). For example, the isolated vertices in the MDCore of a graph can be seen as one type of anomaly, similar to the hub vertices and outlier vertices detected in [278]. Link cohesion metrics can also be used in proximity-based anomaly detection that are useful to find vertices with close connections to known anomalous vertices [7]. In Chapter 5, the link cohesion metrics proposed in this chapter and the next will be used to create vertex-level features (see Definitions 5.4.7 and 5.4.8) that can be used to detect fake accounts in online social networks.

Chapter 4: Semi-Local Link Cohesion Metrics for Networks with Degree Assortative Community Structure

4.1 Introduction

In Chapter 3, a case was made for studying nPSO graphs as a benchmark for community detection by demonstrating that they could create structural patterns observed in real-world networks that LFR graphs were unable to create. Does that make studying LFR graphs obsolete? In this chapter, we advocate for why it is still important to study LFR graphs based on their ability to create structural patterns observed in real-world networks that nPSO graphs are not able to create. As demonstrated in Subsection 3.7.6, community detection techniques shown to work well for one paradigm of community structure do not necessarily work well for a different paradigm of community structure. The fact that LFR graphs can generate patterns observed in real-world networks that the nPSO graphs cannot and vise-versa provides some evidence that researchers and practitioners should take time to understand the type of community structure they expect from a network they are studying before determining what methods to use for community detection.

The link cohesion metric proposed in Chapter 3, called Inversely Proportional Link Cohesion (see Definition 3.4.8), heavily degraded the communities detected by traditional algorithms on highly connected LFR graphs with degree assortative community structure (see Subsection 3.7.6). Because most highly connected vertices in nPSO graphs are joined together based on their popularity, not because they are part of the same community, a

metric that heavily discounts connections among the highest degree vertices makes sense. In highly connected LFR graphs with degree assortative community structure, the vertices with the highest degrees are often part of the same community. Heavily discounting the connections involving the highest degree vertices in such a graph may cause some vertices belonging to the largest communities to have larger weights connecting them to vertices outside their community. If Inversely Proportional Link Cohesion is used to pre-weight the network, such weights could cause these vertices to be placed in the wrong community when traditional algorithms are used.

That said, determining edge weights by taking the degrees of the vertices along simple paths connecting seed vertices into account has shown to be beneficial. Traditional community detection algorithms still have difficulty on certain LFR graphs due to issues such as the resolution limit [30, 100, 101], extreme degeneracy [100, 114, 131], non-locality [39], sensitivity to noise [100], and the tendency to create a giant community consisting of a majority of the vertices [60, 269] that may be benefitted by a related approach. In this chapter, we define a new link cohesion metric that incorporates equations used for link reliability and link prediction proposed by Carlo Vittorio Cannistraci, Alessandro Muscoloni, and Ilyes Abdelhamid in 2018 [45, 182]. These equations take the degrees of vertices along simple paths into account, but rewards connections among these vertices instead of penalizing them as is done in Inversely Proportional Link Cohesion [239]. In this chapter, we demonstrate how, when used to pre-weight networks, the new link cohesion metric is able to improve the communities detected by traditional algorithms in LFR graphs, which in many cases have degree assortative community structure, and demonstrate that the computation of the new deterministic metric is scalable and parallelizable. We additionally make observations of the new metric which may provide insights on determining what type of community structure a network may have, or if it has community structure at all.

Using the equations from link reliability and link prediction applications in a link cohe-

sion metric was proposed by Cannistraci at the conference where the original link cohesion paper was presented [239]. Cannistraci also made Matlab code available to compute the equations. Wojciech Czaja suggested the new metric and related terms should be formalized with mathematical definitions. For this research, I introduce the equations used for link reliability and link prediction from a different perspective and implement the new metric in R, which allows for easier evaluation with various traditional community detection algorithms.

4.2 Motivation for Studying LFR Benchmark Graphs

While nPSO graphs (see Subsection 2.5.4) have been show to be able to create realistic community structure found in real-world networks (see Section 3.2), they are unable to create degree assortative community structure where higher degree vertices are only found in the larger communities, as seen in Figure 1.1a. Specifically, the degree of a vertex is encoded in a radial dimension independent of the angular dimension representing similarity, as illustrated in Figure 2.11. Since communities are determined by angular separation, it is not possible to create communities that do not include higher degree vertices without additional modifications to the model.

Is degree assortative community structure a good paradigm for real-world networks? A number of researchers have indicated it is. Specifically, in their 2016 User Guide to Community Detection in Networks, Santo Fortunato and Darko Hric suggested that degrees can be used as proxy for group membership in certain types of networks [102]. In his second edition textbook on Networks published in 2018, Mark Newman suggested that smaller average values of the local clustering coefficients for higher degree vertices compared to lower degree vertices may be due to smaller groups being constrained to having lower degree because they have relatively few fellow group members to connect to [199]. He

also suggested that social networks are assortatively mixed for the same reasons. Consider computer science researchers that collaborate on published journal articles, conference and workshop proceedings, or monographs. A network of researchers can be created in which a pair are connected if they co-authored at least one paper in the period of several decades. It seems reasonable to assume that smaller communities would be made up of researchers that frequently collaborate with each other, but less with others. Since they have fewer people to collaborate with, they would not be amongst the highest connected in the network. Alternatively, researchers who collaborate with a lot of other researchers are likely to collaborate with each other and also collaborate with less connected researchers, creating larger communities. These are the exact characteristics of degree assortative community structure.

One of the networks in SNAP datasets (the Stanford Large Network Dataset Collection) [148] is the DBLP collaboration network which corresponds to the largest connected component (see Definition 2.1.15) of the network of computer science researchers described above, created using the DBLP bibliography [79]. The DBLP network, which will be referred to as G_{DBLP}, has $N = 317,080$ vertices, $1,049,866$ edges, and $13,477$ ground truth overlapping communities where authors who published to a certain journal or conference form a community. It should be noted that the paper first introducing the DBLP network explicitly states that its ground truth communities are very different than 'communities' that identify tightly-connected clusters within a given graph [15]. Although the ground truth communities are repurposed for density-based community detection [280], they are not used in this paper. It is possible that separating the original network to only the largest connect component could make a some of the ground truth communities meaningless. That said, it should still be a reasonable assumption that the density-based communities would have degree assortative community structure in its largest connected component.

Although they do not necessarily create degree assortative community structure, many parameter settings for LFR graphs (see Subsection 2.5.2) do create communities of this

nature. Because a vertex cannot be placed in communities whose sizes are smaller than its corresponding internal degree, the resulting community structure in LFR graphs are degree assortative in many cases. Specifically, when the sizes of the communities are small relative to the largest degrees in the graph, the most highly connected vertices will only be able to be placed in the larger communities. If the sizes of all communities are larger than the degree of the most highly connected vertex in the graph, then the resulting community structure will not necessarily be degree assortative (see Subsection 2.5.2 for more details about LFR graphs).

The average degree of the G_{DBLP} is 6.6 with a minimum degree of 1 and a maximum degree of 343. While it is good practice to scale up to larger graph sizes using a benchmark, starting with graphs of smaller sizes are usually easier to study. Based solely on the minimum degree, maximum degree, and the size of the G_{DBLP}, a maximum degree for a similar graph with a smaller number of vertices can be determined by assuming a power-law distribution and determining the matching power-law cut-off degree (See Definition 2.2.4). In this case, using $\gamma = 3.17$ would create a power-law cut-off degree of 342.8 for the G_{DBLP} and would correspond to a maximum degree of 50.7 for a graph with 5,000 vertices. In extremely sparse networks (see Definition 2.2.2), which most social networks are assumed to be, the average degree remains constant as the size of the network grows, so a scaled down G_{DBLP} graph would have the same average degree. In 2011, Jonathan Berry, Bruce Hendrickson, Randall LaViolette, and Cynthia Phillips created a benchmark using LFR graphs, called LFR_BHLP, where the number of vertices was set to $N = 5,000$, the average degree was set to $\langle k \rangle = 8$, the maximum degree was set to $k_{max} = 50$, the degree distribution scaling parameter was set to $\gamma = 2$, the community size distribution scaling parameter $\beta = 1.5$, the minimum and maximum number of communities were set to $c_{min} = 3$ and $c_{max} = 105$ respectively, and the mixing parameter was varied using values, $\mu \in [.1, .2, .3, .4, .5]$. Since the parameters are similar to the properties of the scaled down G_{DBLP} graph and the com-

munity structure of the benchmark is degree assortative, as the community sizes can be smaller than the degrees of the vertices, this benchmark is used for related experiments.

Claim 4.2.1. (CS) LFR benchmark graphs are able to generate graphs with degree assortative community structure whereas nPSO benchmark graphs cannot, leading to community structure asummed to be more similar to certain real-world networks such as G_{DBLP}, the DBLP collaboration network from SNAP datasets [148]. Based on similar graph properties, the LFR_BHLP benchmark is deemed to be a good approximation of G_{DBLP}.

Degree assortative community structure and the graphs generated by the LFR benchmark have community structure similar to the conventional paradigm. Even for somewhat higher mixing rates, the average strength of a vertex is expected to be larger with members in its own community compared to average strength of the vertex with the vertices of any other community which aligns with Definition 2.2.9. That said, the settings where LFR graphs create degree assortative community structure are also ones where modularity-based algorithms have particular difficulty. Specifically, modularity-based algorithms have a well-known resolution limit where communities smaller than $\sqrt{L/2}$ may fail to be resolved, where L is the number of edges in the network [101]. Additionally, it has been shown that it is particularly hard to detect communities in sparse networks by using modularity optimization since the expected maximum modularity for a random graph increases when the average degree decreases [100]. The parameter settings for LFR graphs similar to the DBLP network have an average degree $\langle k \rangle = 8 < \log(5000) = 8.5$ even for the smallest graphs analyzed, making it an extremely sparse network (See Definition 2.2.2). Additionally, the fewer the edges, the more likely that random connections unrelated to community structure can cause smaller communities to be merged together leading to a tendency to create a giant component consisting of a majority of the vertices [60, 100, 269]. Small changes in the network may also cause inconsistencies in local regions of a network [39] and it is

possible that there are a large number of high modularity partitions close to the maximum that are not structurally similar, a problem called extreme degeneracy [100, 114, 131]. While a number of these issues are only seen in modularity optimization algorithms, many community detection methods have resolution limit issues and difficulty with very sparse networks [100]. In this chapter, the benefits of pre-weighting networks that have degree assortative community structure with a new link cohesion metric in order to mitigate these problems are analyzed.

Claim 4.2.2. Without proper weighting, community detection algorithms that maximize modularity are at risk of not being able to resolve smaller communities in LFR graphs and similar networks with degree assortative community structure as it is the precise setting where there is a well-known resolution limit [101]. Additionally, if the graphs are extremely sparse, as G_{DBLP} and LFR_BHLP are, additional problems arise [100] including issues that happen at a local scale [39] where adding semi-local information may be beneficial.

In this chapter, the benefits of pre-weighting networks that have degree assortative community structure with a link cohesion metric that incorporates equations from link reliability and link prediction [45, 182] in order to mitigate these problems are analyzed.

4.3 Related Work

A number of techniques have been proposed to account for these shortcomings, for example, [16, 55–57, 100]. One approach is to weight the edges in a network in a way that incorporates structural information and use the weighted network in the modularization maximization algorithm. A wide range of local, semi-local, and global edge metrics (see Definition 2.3.2) have been proposed to improve the communities detected by traditional algorithms. These are discussed in Subsection 2.3.2.

4.4 Local Community Link Cohesion

As stated in Definition 3.4.1, *link cohesion* is a type of metric used to assess how supported an edge is relative to other edges, accounting for nearby alternate paths (see Definition 3.4.3) and associated vertex degrees. One benefit of link cohesion metrics is that they are semi-local (see Definition 2.3.2) by design, whereas many other metrics are either local or global. In Chapter 3, a link cohesion metric called Inversely Proportional Link Cohesion was defined (See Definition 3.4.8) that weighted each edge in an alternate path inversely proportional to their expected number of edge in a random degree-preserving multigraph (see Definition 2.5.1). This gives less weight to alternate paths made up of high degree vertices even if those high degree vertices are part of many alternate paths between two seed vertices. In such a case, the intermediate vertices in the alternate paths may have many connections amongst themselves that could be used as an indication that the seed vertices are part of the same community, particularly in networks with degree assortative community structure. Prior to defining the new link cohesion metric that takes such a notion into account, which will be referred to as Local Community Link Cohesion, a few terms are defined that can be used to identify the connections between the intermediate vertices in the alternate paths. Note that all graphs are assumed to be simple graphs (see Definition 2.1.3).

Definition 4.4.1. Let $G = (V, E)$ be a graph, let $h \in \mathbb{Z}^+$, and let $\mathrm{S}h_{uv}$ be the set of all h-hop simple paths between vertices $u, v \in V$. The *set of* $\mathrm{S}h_{uv}$ *common neighbors* is defined as

$$\mathrm{CN_S}h_{uv} = \{z_j^i\}_{\substack{1 \le i \le n_h \\ 1 \le j \le h-1}}, \tag{4.1}$$

where z_j^i for $i = 1, .., n_h$ and $j = 1, .., h-1$ are the intermediate vertices of $\mathrm{S}h_{uv}$. If $x \in \mathrm{CN_S}h_{uv}$, then x is an Sh *common neighbor* of u and v.

Sh_{uv} is first defined in Definition 3.4.2. Note that a common neighbor from Definition 2.1.8 is equivalent to an S2$_{uv}$ common neighbor.

Definition 4.4.2. Let $G = (V, E)$ and let CN_Sh_{uv} be the set of Sh_{uv} common neighbors of $u, v \in V$ where $h \in \mathbb{Z}^+$. The Sh_{uv} *local community* of u and v is defined as

$$\text{LoCo_S}h_{uv} = G_{\text{CN_S}h_{uv}} = (V_{\text{CN_S}h_{uv}}, E_{\text{CN_S}h_{uv}}), \tag{4.2}$$

where $V_{\text{CN_S}h_{uv}} = \text{CN_S}h_{uv}$ and

$$E_{\text{CN_S}h_{uv}} = \bigcup_{x,y \in V_{\text{CN_S}h_{uv}}} (x, y) \in E. \tag{4.3}$$

The Sh_{uv} local community, LoCo_Sh_{uv}, is the induced subgraph with the vertex set being equal to CN_Sh_{uv}. It is important to note that the Sh_{uv} local community is defined in terms of Sh_{uv} common neighbors. Although more vertices may be common neighbors for different path lengths h, only Sh_{uv} common neighbors and connections between those vertices are considered for the Sh_{uv} local community. See Supplemental Figure 2 in [182] for an illustration of the distinction. An Sh_{uv} local community is *connected* if it is has only one connected component and it is *disconnected* if all its vertices are isolated vertices.

Definition 4.4.3. Let $G = (V, E)$ be a graph, let CN_Sh_{uv} be the set of Sh_{uv} common neighbors of $u, v \in V$ where $h \in \mathbb{Z}^+$, and let LoCo_Sh_{uv} be the Sh_{uv} local community of u and v. The *other vertices* in V not including u, v, or the Sh_{uv} common neighbors are defined as

$$\text{O_S}h_{uv} = V \setminus (\text{CN_S}h_{uv} \cup \{u, v\}). \tag{4.4}$$

An edge $(x, y) \in E$ is an *internal local community edge* relative to LoCo_Sh_{uv} if $x, y \in$ CN_Sh_{uv} and it is an *external local community edge* relative to LoCo_Sh_{uv} if $x \in$ CN_Sh_{uv}

and $y \in \text{O_S}h_{uv}$ or $x \in \text{O_S}h_{uv}$ and $y \in \text{CN_S}h_{uv}$. An edge is a *non-local community edge* relative to $\text{LoCo_S}h_{uv}$ if it is not an internal local community edge or an external local community edge relative to $\text{LoCo_S}h_{uv}$. The *internal local community degree* of a vertex x relative to $\text{LoCo_S}h_{uv}$ is defined as

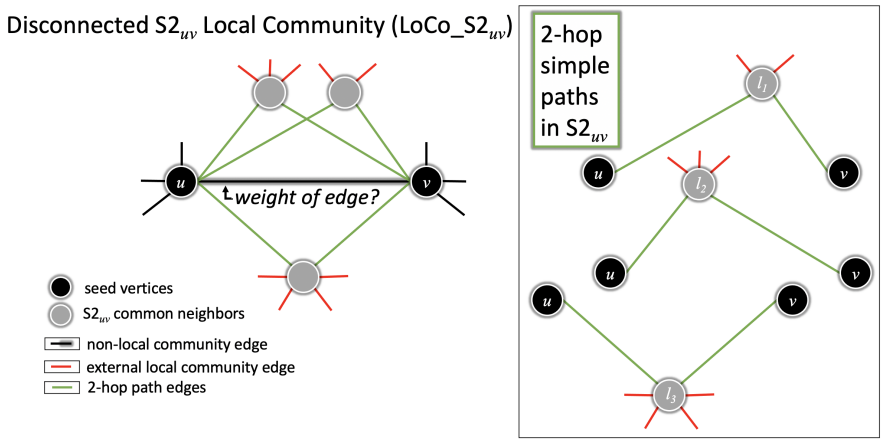
$$k_x^{int} = |(x,y) : x,y \in \text{CN_S}h_{uv}, (x,y) \in E| \tag{4.5}$$

and the *external local community degree* of the vertex x relative to $\text{LoCo_S}h_{uv}$ is defined as

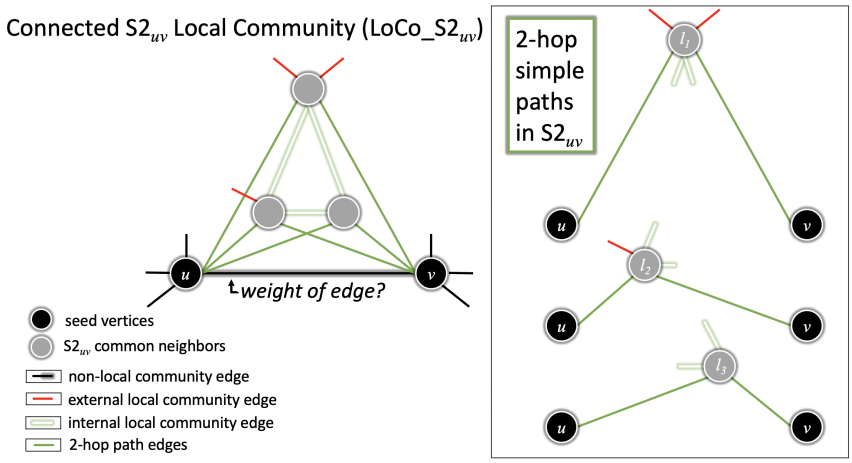
$$k_x^{ext} = |(x,y) : x \in \text{CN_S}h_{uv} \text{ and } y \in \text{O_S}h_{uv}, (x,y) \in E|. \tag{4.6}$$

Figure 4.1a is an example of a disconnected S2_{uv} local community, Figure 4.1b is an example of a connected S2_{uv} local community, and Figure 4.2 is an example of a connected S3_{uv} local community. The connected (see Definition 2.1.8) seed vertices in Figure 4.1a and 4.1b have the same degree, the same number of 2-hop simple paths (see Definition 2.1.18) connecting them, and the vertices in the 2-hop simple paths have the same degrees in both examples. This means that the edges connecting the seed vertices would have the same weight computed using Inversely Proportional Link Cohesion. The two examples can be differentiated if the cross interactions among the S2_{uv} local communities are analyzed. In Figure 4.1a, none of the vertices in the S2_{uv} local community are connected to each other so all the edges of the S2_{uv} common neighbors are external to the S2_{uv} local community (with the exception of the edges connecting them to u and v which are considered non-local) whereas in Figure 4.1b, all of the vertices in the S2_{uv} local community are connected to each other yielding several internal S2_{uv} local community edges. All else being equal, it would seem more desirable to have a higher cohesion score for the edge connecting the seed vertices in Figure 4.1b compared to the edge connecting the seed vertices in Figure

4.1a.



(a) A disconnected S2$_{uv}$ local community



(b) A connected S2$_{uv}$ local community

Figure 4.1: Examples of S2$_{uv}$ local communities with all 2-hop simple paths between seed vertices delineated. (a) a disconnected example (b) a connected example.

Similar to how communities from a partitioning of a network are evaluated (see Definition 2.3.3 and Definition 2.3.1), it is desirable for local communities to both be cohesive with more internal local community edges and to be separated from the rest of the network with less external local community edges. One way to create a measure that takes these into account would be to place the internal local community degrees of the vertices in the local community into the numerator of an equation and to place the external local commu-

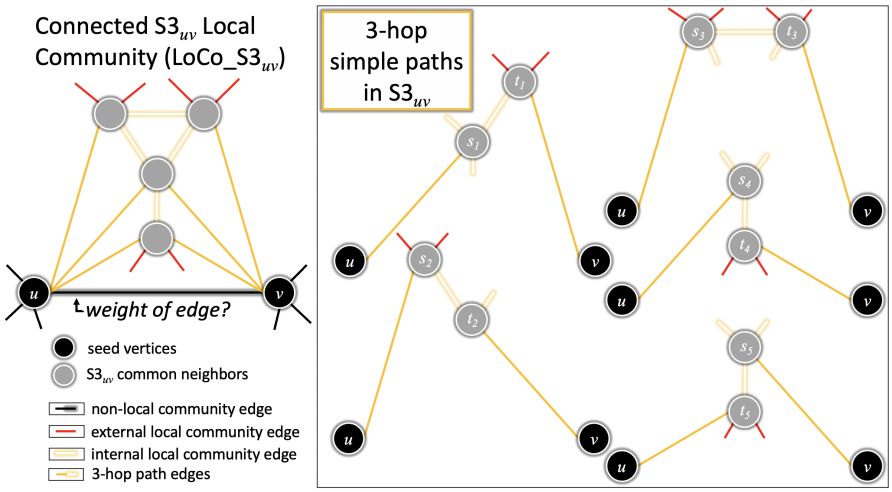


Figure 4.2: A connected S3$_{uv}$ local community with all 3-hop simple paths between seed vertices delineated. Some of the local community edges are internal and some are external.

nity degrees of the vertices in the local community into the denominator of an equation. It is possible that either internal local community degree or the external local community degree of a vertex could equal 0 in some scenarios. For example, the internal local community degree of all the S2$_{uv}$ common neighbors in Figure 4.1a is 0 and the external local community degree of l_3 in Figure 4.1b is 0. Because of this, using these values in certain equations could cause a term to be undefined if in the denominator or cause a term to equal 0 if in the numerator, removing valuable additional information. Therefore, modified versions of the terms are defined.

Definition 4.4.4. Let $G = (V, E)$ be a graph and let LoCo_Sh_{uv} be the Sh_{uv} local community of $u, v \in V$ where $h \in \mathbb{Z}^+$. The *modified internal local community degree* of a vertex x relative to LoCo_Sh is defined as

$$\widetilde{k_x^{int}} = k_x^{int} + 1 \tag{4.7}$$

and the *modified external local community degree* of the vertex x relative to LoCo_Sh_{uv} is

defined as

$$\widetilde{k_x^{ext}} = k_x^{ext} + 1. \tag{4.8}$$

This simple modification is similar in concept to treating the edges along the simple paths as independent in the calculation of Inversely Proportional Link Cohesion. There are a number of ways one could combine the modified internal local community degrees in the numerator of an equation and the modified external local community degrees in the denominator of an equation. The equations for link prediction take the geometric mean of each, so that if one common neighbor along a simple path has low degree it is strongly reflected in the score for the whole simple path. Geometric means are also used in [238] for similar reasons.

Definition 4.4.5. Let $G = (V, E)$ be a graph, let $h \in \mathbb{Z}^+$, and let $\mathrm{S}h_{uv}$ be the set of all simple paths of length h between $u, v \in V$. The *Cannistraci-Hebb formulas* for simple paths of length h are defined as

$$\mathrm{CH2_S}h(u,v) = \sum_{z_1,\ldots,z_{h-1} \in \mathrm{S}h_{uv}} \frac{\left(\widetilde{k_{z_1}^{int}} * \ldots * \widetilde{k_{z_{h-1}}^{int}}\right)^{\frac{1}{h-1}}}{\left(\widetilde{k_{z_1}^{ext}} * \ldots * \widetilde{k_{z_{h-1}}^{ext}}\right)^{\frac{1}{h-1}}}, \tag{4.9}$$

where * represents scalar multiplication.

The 2 in the abbreviated name for the Cannistraci-Hebb formulas is to remain somewhat consistent with the naming in [182] and these formulas were a revision to a previously proposed formula [182, 188]. Note that for paths of length 2, the Cannistraci-Hebb formula is equal to

$$\mathrm{CH2_S2}(u,v) = \sum_{l \in S2_{uv}} \frac{\widetilde{k_l^{int}}}{\widetilde{k_l^{ext}}} \tag{4.10}$$

and for paths of length 3, the Cannistraci-Hebb formula is equal to

$$\text{CH2_S3}(u,v) = \sum_{s,t \in S3_{uv}} \frac{\sqrt{\widetilde{k_s^{int}} * \widetilde{k_t^{int}}}}{\sqrt{\widetilde{k_s^{ext}} * \widetilde{k_t^{ext}}}}. \tag{4.11}$$

Claim 4.4.6. (CS) The Cannistraci-Hebb formulas in Definition 4.4.5 would provide relatively higher scores to edges that have a local community with lower conductance (see Definition 2.3.3) and high modularity (see Definition 2.3.1), which corresponds to the conventional paradigm of community structure (see Definition 2.2.9) that LFR graphs generate (see Section 2.5.2).

While link cohesion metrics are specifically to measure the *strength* of existing edges in a network, the Cannistraci-Hebb formulas are used for link prediction and are therefore intended to be used when an edge is not present in a network. Carlo Vittorio Cannistraci, Gregorio Alanis-Lobato, and Timothy Ravasi originally motivated similar equations for link prediction by analyzing the statement made by Donald Olding Hebb in 1949: neurons that fire together wire together [121]. One interpretation of the statement is that new connectivity between non-interacting neurons emerge and form due to how the neurons are embedded and interact as a cohort. Considering only the network topology, this interpretation can be formalized as topological link prediction in complex networks [46]. Imagining that the seed vertices are not already connected by an edge, Figures 4.1a, 4.1b, and 4.2 can illustrate the link prediction problem and were originally used to do so in [188]. Similar formulas to those in Definition 4.4.5 are also used for link reliability in [45], which would be used for existing edges in a network. In both link reliability and link prediction, the degrees of the seed vertices are not necessarily important and are therefore not part of the equations.

For a link cohesion metric, taking the seed vertices into account is important as highly

connected vertices are more likely to be connected at random than lower degree vertices. In Figure 4.3, two example disconnected $S2_{uv}$ local communities are shown. Each example is based on connected seed vertices, each connected by 3 additional 2-hop simple paths. The intermediate vertices in the two examples have the same degrees, so if the seed vertices are not taken into account their scores would be equal. To motivate why this is not ideal, consider the DBLP network discussed in Secion 4.2. Assume there are two prominent computer science researchers that collaborate with the greatest number of other researchers but generally study very different topics. It is possible that the two have found a common topic of interest to publish on together. It is also possible that they have a few common collaborators, either because those researchers switched topics from graduate school to post graduate research or for other reasons. The fact that they have so many collaborators makes a few common collaborators less notable and should be reflected with a smaller Local Community Link Cohesion score. Alternatively, consider two researchers that have hardly collaborated with any other researchers. If they published a paper together and have of few common collaborators, it would be a better indication that they are part of the same community compared to same situation with the prominent researchers. Therefore, a term is added in the Local Community Link Cohesion metric that weights the connected seed vertices inversely proportional to their expected number of edges in a random degree-preserving multigraph (see Definition 2.5.1) to create the relative scoring as desired. Raw scores for an edge created using these concepts are not meaningful unless they are assessed relative to the scores of other edges in the graph, therefore, the weights are normalized to the interval [0,1). An additional benefit of normalizing the weights between [0,1) is that weights computed taking different simple path lengths into account can be combined into a single metric.

Definition 4.4.7. Let $G = (V, E)$ be a graph with $L = |E|$, let $h \in \mathbb{Z}^+$, and let Sh_{uv} be the set

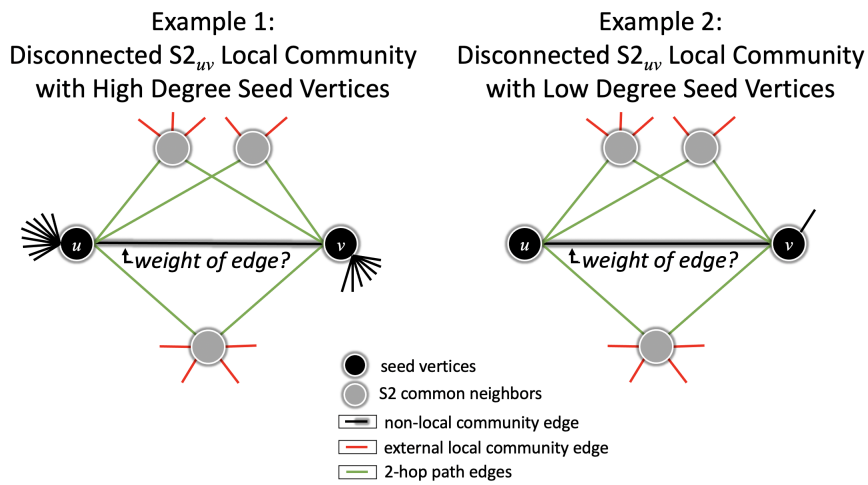


Figure 4.3: Two example disconnected S2$_{uv}$ local communities. Example 1 has high degree connected seed vertices and Example 2 has low degree connected seed vertices, each connected by 3 additional 2-hop simple paths. The intermediate vertices in Example 1 and Example 2 have the same degrees. Local Community Link Cohesion is defined such that the edge connecting the seed vertices in Example 2 has a higher weight than the edge connected the seed vertices in Example 1.

of all simple paths of length h between $u, v \in V$. The *raw Local Community Link Cohesion weights* involving paths of length h are defined as

$$\text{rLoCoLinkCo_S}h(u,v) = \begin{cases} \frac{1}{k_u * k_v} \sum\limits_{z_1,\ldots,z_{h-1} \in \text{S}h_{uv}} \frac{(\widetilde{k^{int}_{z_1}} * \ldots * \widetilde{k^{int}_{z_{h-1}}})^{\frac{1}{h-1}}}{(\widetilde{k^{ext}_{z_1}} * \ldots * \widetilde{k^{ext}_{z_{h-1}}})^{\frac{1}{h-1}}} & \text{if } u, v \in V, \\ 0 & \text{otherwise,} \end{cases} \tag{4.12}$$

where $z_1, \ldots, z_{h-1}$ are the intermediate vertices of Sh_{uv} and * represents scalar multiplication. Since there are no intermediate vertices in S1$_{uv}$, the default sum is defined equal to 1. If the set of simple paths Sh_{uv} is empty, the default sum is defined to equal 0. The *average raw Local Community Link Cohesion weight* involving paths of length h is defined as

$$\mu_{\text{rLoCoLinkCo_S}h} = \frac{1}{2L} \sum_{u,v \in V} \text{rLoCoLinkCo_S}h(u,v). \tag{4.13}$$

The *normalized Local Community Link Cohesion weights* involving paths of length h,

which can be referred to as the S*h Local Community Link Cohesion weights*, are defined as

$$\text{LoCoLinkCo_S}h(u,v) = \frac{\text{rLoCoLinkCo_S}h(u,v)}{\mu_{\text{rLoCoLinkCo_S}h} + \text{rLoCoLinkCo_S}h(u,v)}. \tag{4.14}$$

Local Community Link Cohesion is defined as

$$\text{LoCoLinkCo}(u,v) = \frac{1}{2}\text{LoCoLinkCo_S2}(u,v) + \frac{1}{2}\text{LoCoLinkCo_S3}(u,v) + \varepsilon, \tag{4.15}$$

where ε is equal to machine epsilon.

Claim 4.4.8. (CS) Because it takes both the edge connected seed vertices into account along with edges in nearby alternate paths (see Equations 4.13 and 4.15), Local Community Link Cohesion is one way of measuring the *strength* of an edge based on the cycles it is part of.

Aside from how the intermediate vertices along the simple paths are weighted, Local Community Link Cohesion follows the same design choices as Inversely Proportional Link Cohesion. Specifically, only 2-hop simple paths with

$$\text{rLoCoLinkCo_S2}(u,v) = \frac{1}{k_u k_v} \sum_{l \in S2_{uv}} \frac{\widetilde{k_l^{int}}}{\widetilde{k_l^{ext}}} \tag{4.16}$$

between connected seed vertices and 3-hop simple paths with

$$\text{rLoCoLinkCo_S3}(u,v) = \frac{1}{k_u k_v} \sum_{s,t \in S3_{uv}} \frac{\sqrt{\widetilde{k_s^{int}} * \widetilde{k_t^{int}}}}{\sqrt{\widetilde{k_s^{ext}} * \widetilde{k_t^{ext}}}} \tag{4.17}$$

between connected seed vertices are taken into account. Therefore, vertices not connected by any 2-hop or 3-hop alternate paths will have a score equal to ε. While this may not be ideal in some networks, for social networks it can provide valuable insights as a person

that is not connected to any friends or friends of friends of another person is not likely to be members of the same community even if the two people are connected. Additional benefits discussed in Subsection 4.6.2. Subsection 4.6.8 will show that a number of the researchers in the DBLP network have connections to other researchers but without any nearby alternate paths. The benefit of taking both 2-hop simple paths and 3-hop simple paths into account is analyzed in Subsection 3.7.2 and Subsection 4.6.7. Higher order simple paths are not taken into account because the computational complexity increases and the benefit added is not expected to be much. As stated previously, for social networks, higher order simple paths get increasingly less meaningful as the characteristic path length (see Definition 2.1.17) is often quite small. Larger characteristic path lengths may indicate that using higher order cycles would be beneficial, observing the corresponding hop plot may provide better insights (see Figure 2.4c as an example). It also may be beneficial to use cycles of higher order in networks where all the communities are larger than the highest degree vertex. Note that this statement is made independent of sparsity, but community sizes in sparser networks would need to be relatively smaller than the community sizes in denser networks in order for the previous statement to hold. This is discussed in Subsection 4.6.7.

4.5 Implementation and Experimental Setup

To calculate the deterministic Local Community Link Cohesion weights for a particular graph, an approach similar to the one used to compute the link reliability and link prediction scores was taken [1]. The computational complexity of computing all edge weights is $O(\langle|\text{CN_S3}_{uv}|\rangle N)$ for extremely sparse networks, where $\langle|\text{CN_S3}_{uv}|\rangle$ is the average number of S3 common neighbors between two seed vertices. Note that if only S2 weights are

[1] https://github.com/biomedical-cybernetics/Cannistraci_Hebb_network_automata

calculated, the term is equal to $\langle|\text{CN_S2}_{uv}|\rangle$ as implemented, and more specifically incorporating paths of length h would have a constant term equal to $\langle|\text{CN_S}h_{uv}|\rangle$. The term is inside the big O notation, because it depends on how the sizes of the communities scale. The algorithm is easily parallelizable, as each edge can be computed independently with semi-local information.

The Local Community Link Cohesion computation was written in R to leverage the community detection algorithms written in the igraph package [75]. The LFR benchmark graphs are written in c++ and are called by invoking a system command [141]. The future.apply package is used for parallelization [29]. All baseline weighting metrics are also written in R, parallelizing when possible. RNBRW is a different type of parallelization requiring the doParallel [116] and foreach [117] packages.

Most experiments were run on MacBook Pro laptops, either on my old laptop with a 2.9 GHz Quad-Core Intel Core i7 Processor and 16 GB of memory, a loaner laptop when my old laptop had problems with a 2.9 GHz Dual-Core Intel Core i7 Processor and 16 GB of RAM, or my new laptop with a 2.4 GHz 8-Core Intel Core i9 Processor with 64 GB of RAM. Some of the larger run are done on a cluster using slurm on compute nodes with 128 AMD ROME Cores and 2 TB of RAM.

4.6 Results

Local Community Link Cohesion (LoCoLinkCo) introduced in this chapter (see Definition 4.4.7) will be compared against several baseline pre-weighting methods, listed in Table 3.2. Inversely Proportional Link Cohesion (InvPrLinkCo) introduced in the previous chapter (see Definition 3.4.8) will also be used for comparison purposes. Some cycle-based methods are more computationally expensive even when parallelized, so a comparison to cycle-based baseline methods will be completed separately from the other methods with

a more narrow focus. The baseline pre-weighting methods were initially introduced in Subsection 2.3.2.

The main community detection algorithm used will the Louvain method, but other community detection methods are used to understand the benefits of pre-weighting with InvPrLinkCo for a wider range of algorithms. All of the community detection methods can be found in Table 3.3, which were originally introduced in Section 2.3.

Because ground truth is available for the benchmark graphs, the results for the benchmark graphs are presented as normalized mutual information, abbreviated as NMI (see Definition 2.3.4), and community count error, abbreviated as CCE (see Definition 2.3.5). Results are summarized from 30 random repetitions except for the benchmark that scales to 500,000 vertices and using community detection methods other than the Louvain method where only 10 random repetitions were used. Note that the experiments on the benchmark that scales to 500,000 using only the Louvain method were done with 30 random repetitions. When comparing against non-cycle-based methods, mean values are plotted with 95% confidence intervals, assuming normal distributions. When comparing against cycle-based methods, boxplots are shown.

Because reliable ground truth is not available for G_{DBLP}, the communities are evaluated based on the number of communities detection in various size ranges which were chosen based on research conducted by Robin Dunbar (see Table 2.1), as was done in [269]. Additionally, conductance (see Definition 2.3.3) and internal edge density (see Definition 2.3.3) are used as well.

4.6.1 Benchmark LFR Graphs

In addition to the parameters for LFR graphs introduced in Section 4.2, which will be referred to as LFR_BHLP, the benchmark graphs proposed by other authors will also be

analyzed [131, 176, 239]. Additionally, LFR_BHLP will be scaled up to graphs that contain 500,000 vertices. The parameters for all the benchmark LFR graphs are summarized in Table 4.1.

Table 4.1: Parameters for LFR benchmark graphs.

Parameter	LFR_BHLP [30]	LFR_BHLP_S	LFR_MSPM [176]	LFR_KRH [131]	LFR_SSGC [239]
N	5000	[5000,..,500000]	500	1000	1000
$\langle k \rangle$	8	8	7	20	30
k_{max}	50	$N^{1/2.17}$	12	50	300
γ	2	2	2	2	2
β	1.5	1.5	1	1	1.5
c_{min}	10	10	30	10	9
c_{max}	105	$N^{1/1.83}$	70	50	300
μ	[0.1,..,0.5]	0.3	[0.1,..,0.5]	[.05,..,0.8]	[.05,..,0.75]

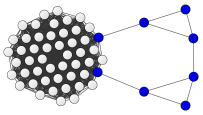
Claim 4.6.1. (CS) All of the benchmark graphs (e.g. LFR_BHLP, LFR_BHLP_S, LFR_KRH, LFR_SSGC) except for LFR_MSPM have degree assortative community structure, since the community sizes can be smaller than the highest degree vertices in the graphs. For the LFR_MSPM benchmark, the highest degree vertices have degree 12 but the smallest community sizes are 30, therefore the highest degree vertices can appear in any of the communities making the community structure not degree assortative.

4.6.2 Benefits to Using Only S2 and S3 Weights

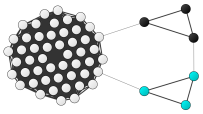
Modularity (see Definition 2.3.1)is the measure optimized in the Louvain method and the Fast Greedy algorithm. The equation for modularity reveals the competing trade-off of optimization algorithms that work by maximizing modularity: the first term is maximal when there are many edges contained within the communities whereas the second subtracted term is minimized when a graph is split into many small communities with low total degrees. Because of this, algorithms that optimize modularity can often form large communities by merging communities that are smaller than a certain threshold which de-

pends on the total number of edges in the network and on the degree of inter-connectivity between the communities [140]. In [30], it was shown that weighted modularity algorithms may fail to resolve communities with less than $\sqrt{W\xi/2}$ total edge weight, where W is the total edge weight in the network and ξ is the maximum weight of an intercommunity edge. When the graph is unweighted, $W = |E|$ and $\xi = 1$. In the extreme case, algorithms that optimize modularity are unable to separate cliques (see Definition 2.1.14) connected only by a single edge into disjoint communities. Adding weights to a network that lowers the weight of the edges between communities can therefore help modularity maximization algorithms resolve smaller communities. Figures 4.4a, 4.4c, and 4.4e illustrate the resolution limit problem with 3 example unweighted graphs used in [154] and Figures 4.4b, 4.4d, and 4.4f illustrate how add LoCoLinkCo can help mitigate the resolution limit. Here, K$\hat{n}$ denotes a complete subgraph, or clique, with $\hat{n}$ vertices.

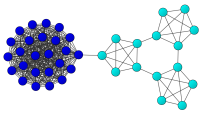
In the example of Figure 4.4a which contains a clique with 50 vertices and two cliques with 3 vertices all connected by a single edge, the Louvain method merges the two small cliques K3-K3 into one community and pulls the connected vertices of K50 into the small community. By adding the LoCoLinkCo as edge weights in 4.4b, the Louvain method is able to separate the cliques correctly. This is because the edges connecting the cliques are not supported by any 2- or 3- hop alternative paths and therefore $\xi = \varepsilon$ where ε is equal to machine epsilon, and very small communities are able to be detected. The example of Figure 4.4c contains a clique with 25 vertices and three cliques with 5 vertices. The Louvain method merges the three smaller communities K5-K5-K5 into one community in the unweighted graph, but is able to resolve all of the cliques with the LoCoLinkCo weights. In Figure 4.4e, a ring of cliques with 5 vertices each are tied together by single edges. The optimal modularity is obtained by merging K5 communities together in pairs as seen with the communities generated by the Louvain method. Adding the LoCoLinkCo edge weights helps the Louvain method separate each clique into it own community, as shown in Figure



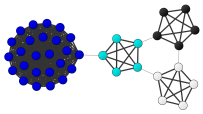
(a) K50-K3-K3
without edge weights



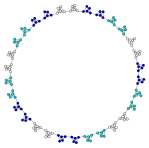
(b) K50-K3-K3
with LoCoLinkCo edge weights



(c) K25-K5-K5-K5
without edge weights



(d) K25-K5-K5-K5
with LoCoLinkCo edge weights



(e) 30K5 clique circle
without edge weights

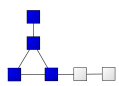
(f) 30K5 clique circle
with LoCoLinkCo edge weights

Figure 4.4: Resolution limit in modularity maximization algorithms mitigated with LoCoLinkCo edge weights. Communities shown in different colors are those detected by Louvain method. Similar benefits would be obtained using InvPrLinkCo.

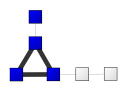
4.4f. InvPrLinkCo would also yield similar benefits as it also only uses S2 and S3 weights. Note that the weighting proposed in [239] also used S1 weights, which would not provide as much benefit due to the larger intercommunity weights it would assign.

Although modularity seems as though it would be a local quality measure since each community contributes separately to the value, it tends to exhibit counterintuitive non-local behavior [39]. Specifically, adding a single vertex to a small graph can completely change the optimal partitioning of the graph as shown in Figures 4.5a and 4.5c. Modularity optimization algorithms also have trouble with satellite vertices as shown in Figure 4.5e. The optimal clustering of a clique is to keep all its vertices together in the same community, but adding leaves to a clique will separate the vertices of the clique into different communities. Lastly, figures 4.5e and 4.5g show the scaling behavior of modularity. By simply doubling

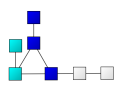
the graph in Figure 4.5e, the detected communities are completely altered in 4.5g. Figures 4.5b, 4.5d, 4.5f, and 4.5h illustrate how adding LoCoLinkCo edge weights mitigates the counterintutive behavior by encoding semi-local community information into the edges of the graphs. InvPrLinkCo would also yield similar benefits. Fixing these non-local behaviors are particularly crucial in sparser graphs, such as G_{DBLP}, where satellite vertices are expected.



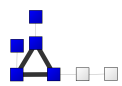
(a) Non-local behavior 1
without edge weights



(b) Non-local behavior 1
with LoCoLinkCo edge weights



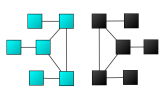
(c) Non-local behavior 2
without edge weights



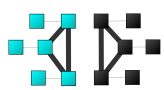
(d) Non-local behavior 2
with LoCoLinkCo edge weights

(e) K3 with leaves
without edge weights

(f) K3 with leaves
with LoCoLinkCo edge weights



(g) Scaling behavior
without edge weights



(h) Scaling behavior
with LoCoLinkCo edge weights

Figure 4.5: Counterintuitive behavior of modularity mitigated with LoCoLinkCo edge weights. Communities shown in different colors are those detected by the Louvain method. Adding a single vertex to (a) as shown in (c) completely changes the detected communities. A clique with satellite vertices are split apart in (e), but if the graph is doubled a different behavior is seen. Pre-weighting with LoCoLinkCo makes the communities detected by the Louvain method exhibit more desirable behavior. Similar benefits would be obtained using InvPrLinkCo.

Claim 4.6.2. (CS) Using the combined S2 and S3 weights that does not include S1 weights or weights from higher order cycles should in theory be beneficial to both highly connected

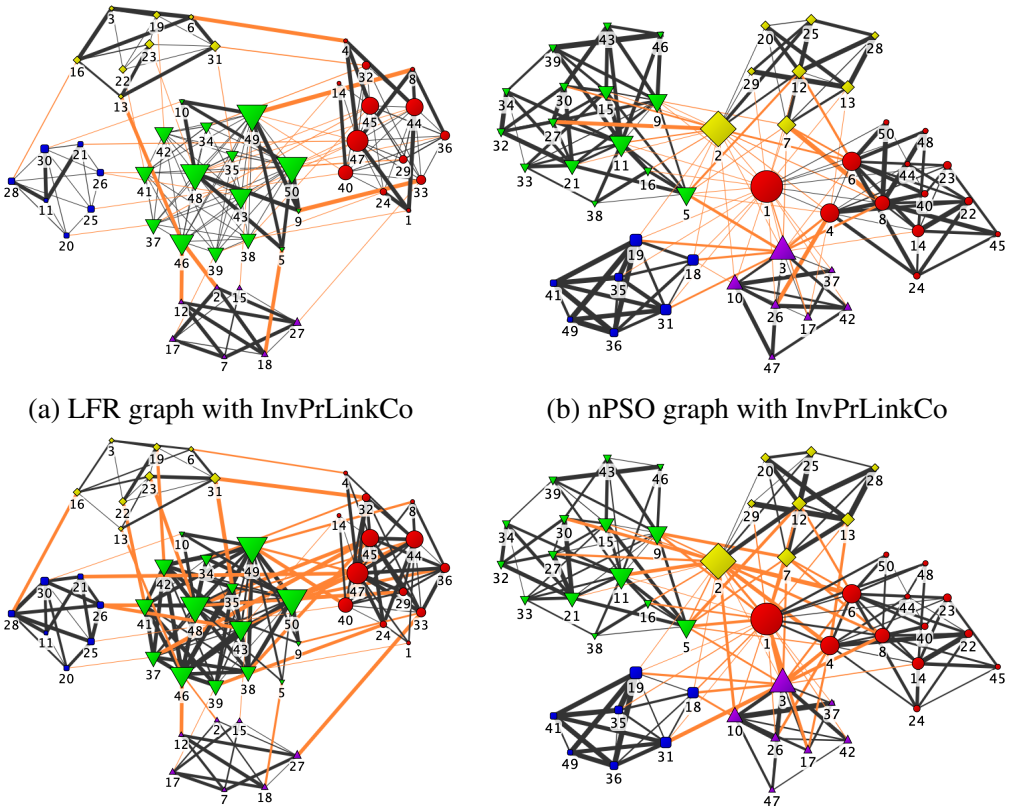
graphs affected by the resolution limit (see Figure 4.4) and sparse graphs affected by counterintuitive non-local behavior of modularity (see Figure 4.5) when used to pre-weight algorithms that maximize modularity.

4.6.3 Comparison of the Two Different of Link Cohesion Metrics

To better understand the similarity and differences between the link cohesion metric proposed in this chapter to the link cohesion metric proposed in the previous chapter, a quick analysis is conducted on the toy networks from Chapter 1 (see Figure 1.1). Figure 4.6 illustrates the LoCoLinkCo edge weights proposed in this chapter and the InvPrLinkCo edge weights proposed in the previous chapter by varying the line thickness of the edges.

The InvPrLinkCo weights are clearly higher inside communities in the nPSO graph shown in Figure 4.6b. While InvPrLinkCo weights tend to be higher inside some communities in the LFR graph shown in Figure 4.6a, there are some several high edge weights between communities and many of the edge weights within the largest community are very small as expected. The LoCoLinkCo weights are high within communities in the LFR graph as shown in Figure 4.6c, and although there are high weights between communities, the highest edge weight of a vertex tends to be within its true community. This is not necessarily the case for LoCoLinkCo weights in the nPSO graph shown in Figure 4.6d, particularly for the higher degree vertices.

The Louvain method correctly finds the ground truth communities in the unweighted LFR graph and the weighted graphs with InvPrLinkCo and LoCoLinkCo. The Louvain method places the two highest degree vertices in the unweighted nPSO graph in the wrong communities and merges the two smallest communities into one. With the LoCoLinkCo weights, the Louvain method is able to separate the two smaller communities, but still places the two highest degree vertices in the wrong communities. With the InvPrLinkCo



(a) LFR graph with InvPrLinkCo

(b) nPSO graph with InvPrLinkCo

(c) LFR graph with LoCoLinkCo

(d) nPSO graph with LoCoLinkCo

Figure 4.6: Comparison of Local Community Link Cohesion (LoCoLinkCo) to Inversely Proportional Link Cohesion (InvPrLinkCo) on the example LFR graph from Figure 1.1a and the example non-uniform Popularity×Similarity (nPSO) graph from Figure 1.1b. The thickness of each edge corresponds to its weight. (a) LFR graph with InvPrLinkCo edge weights (b) nPSO graph with InvPrLinkCo edge weights (c) LFR graph with LoCoLinkCo edge weights (d) nPSO graph with LoCoLinkCo edge weights.

weights, the Louvain method only places the highest degree vertex in the wrong community.

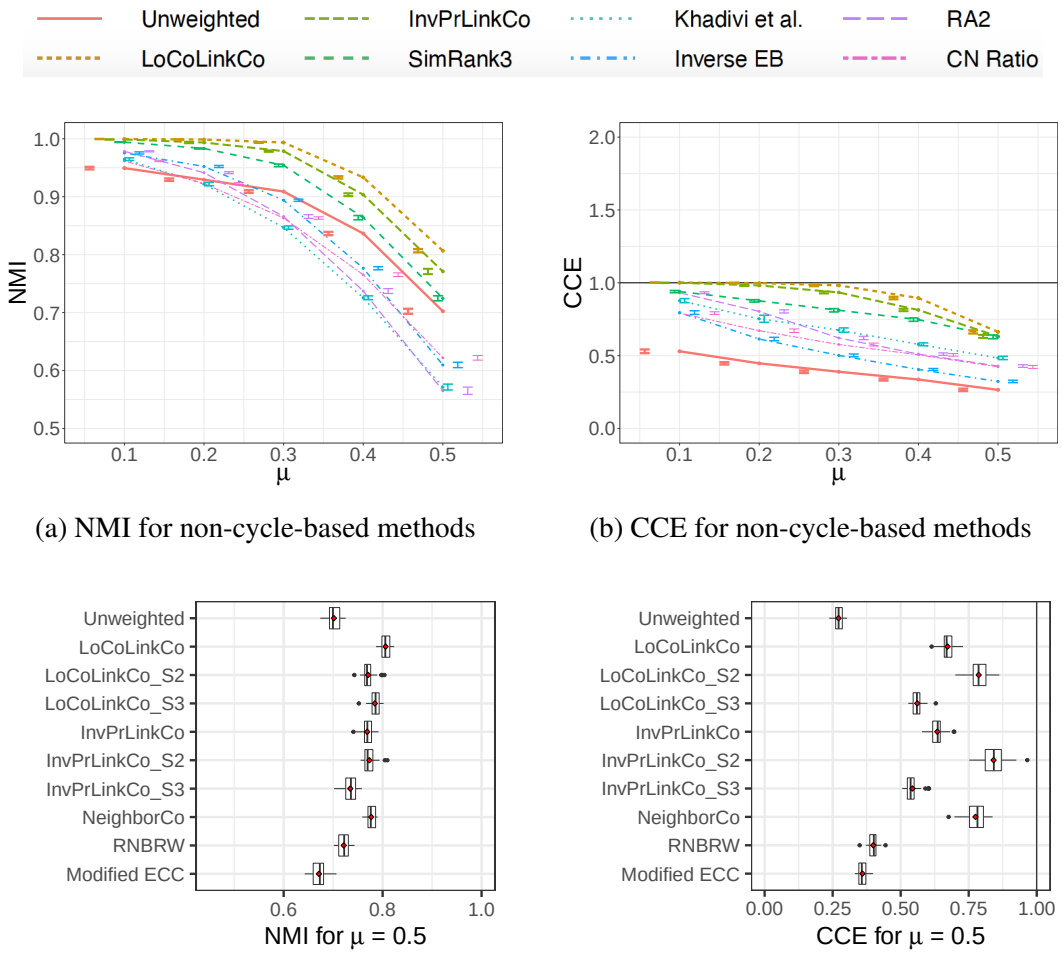
4.6.4 Louvain Method with Pre-Weighted LFR Graphs

The main benchmark used in this chapter is LFR_BHLP as is motivated in Section 4.2, which will be used to compare LoCoLinkCo to other pre-weighting methods shown in Table 3.2 along with InvPrLinkCo from the previous chapter with the Louvain method.

Additional experiments with the Louvain method on the other benchmark graphs will be shown in Subsection 4.6.6 for side by side comparison to other community detection algorithms. Because some of the cycle-based methods are more computationally expensive, experiments are run on a single mixing parameter $\mu = 0.5$, instead of varying the value, as is done with the non-cycle-based methods. The normalized S2 and S3 Local Community Link Cohesion weights (LoCoLinkCo_S2 and LoCoLinkCo_S3) along with normalized S2 and S3 Inversely Proportional Link Cohesion weights (InvPrLinkCo_S2 and InvPrLinkCo_S3) from the previous chapter are also used as a comparison in the cycle-based methods. The community detection results for are shown in Figure 4.7.

LoCoLinkCo significantly outperforms non-cyclic-based methods, with higher NMI and closer estimates to the true number of communities. It should be noted that unlike in Subsection 3.7.6, InvPrLinkCo provides significant benefit when used as a pre-weighting step to the Louvain method compared to the unweighted Louvain method for this benchmark. This is true for many of the LFR benchmark graphs, which are not as highly connected as the benchmark graph used in 3.7.6 (specifically LFR_SSGC from Table 4.1) and do not have high degree vertices as large as in LFR_SSGC. The next best performing baseline pre-weighting method is the cyclic-based NeighborCo which was originally introduced in the same paper that the parameters for this benchmark graph were introduced [30]. That said, LoCoLinkCo still performs significantly better, as indicated by the paired t-test on the NMI which yields a confidence interval of (0.0273,0.0321).

Claim 4.6.3. (CS) LoCoLinkCo can improve the communities detected by the Louvain method in a difficult real-world inspired parameter setting for LFR graphs with both resolution limit issues and sparsity issues (see Claim 4.2.2) more than other baseline pre-weighting methods (see Figures 4.7).



(a) NMI for non-cycle-based methods

(b) CCE for non-cycle-based methods

(c) NMI for cycle-based methods

(d) CCE for non-cycle-based methods

Figure 4.7: Community detection results for LFR_BHLP using the Louvain method with different pre-weighting methods. The unweighted graph is also used as a reference. The plots in (a) and (b) report the mean NMI and CCE for non-cycle-based methods with 95% confidence intervals from 30 random repetitions. The plots (c) and (d) report for T=0.3 the boxplots of NMI and CCE for cycle-based methods over 30 random repetitions.

4.6.5 Scalability of Local Community Link Cohesion

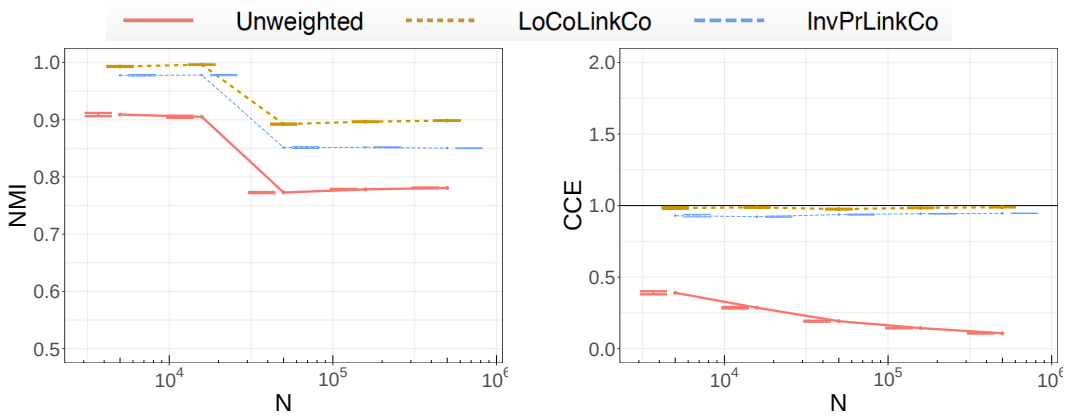
The scalability tests were conducted on LFR graphs similar to the LFR_BHLP graphs. Graphs are scaled from 5,000 to 500,000 using logarithmic spacing. By assuming a power-law distribution and determining the matching power-law cut-off degree (See Definition 2.2.4), a $\gamma = 3.17$ creates the maximum degree of the LFR_BHLP. Therefore, $N^{1/2.17}$ is

used as the maximum degree as N increases. A similar approach is used to determine the maximum community size. It is assumed that the community sizes follow a power-law distribution, so $\beta = 2.83$ is used to match the benchmark. Note that in both cases, the minimum value is assumed to be 1, which is not actually true. The community detection results from the Louvain method are shown in Figure 4.8 with the corresponding compute times shown in Figure 4.9. Results are shown in Figures 4.8a, 4.8b, and 4.9a are experiments run on a laptop with 4 logical cores and results shown in Figures 4.8c, 4.8d, and 4.9b are experiments run on a single large machine in a cluster. Results shown in Figures 4.9c and 4.9d are run on a cluster using Slurm so that the number of cores could be specified.

It is interesting to note that the NMI of all methods take a large dip starting at $N = 31,622$. This is due to the LFR benchmark switch the minimum degree from 3 to 2 to balance the increasing maximum degree with the constant average degree, where a number of the vertices with degree 2 are not supported by any 2-hop or 3-hop alternate paths, as can be determined by counting the number of vertices that have an average strength (see Definition 2.1.11) equal to machine epsilon. This will be discussed more in Subsection 4.6.8. As the size of the graphs grow, all methods are able to determine the correct number of communities. This means the dip in mean NMI causes vertices to be placed into incorrect communities compared to the community detection method splitting or merging communities. This makes sense, since the dip is due to the vertices with edges that are not supported by any 2-hop or 3-hop alternate paths. Instead of creating new communities from these vertices, they are misplaced due to equal strengthed edges.

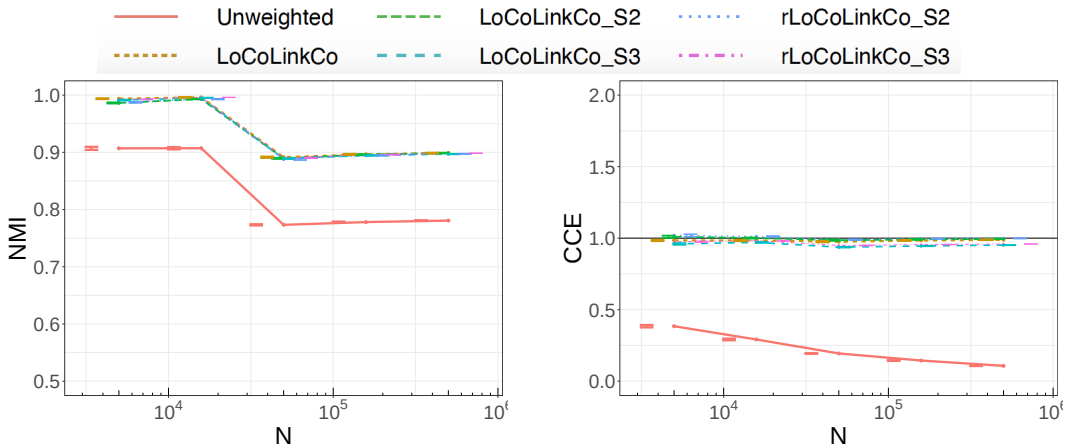
In Figures 4.9a and 4.9b, a change in the scaling of the compute times is noticed at the same point where the performance of the mean NMI degrades. This may be due to an increase in the number of common neighbors for some vertices with the scaled parameters as the highest degree vertices increases slower than the size of the largest communities.

As shown in Figures 4.9c and 4.9d, all baseline cycle-based method are parallelized, but



(a) NMI for LFR_Berry scaled with proportional number of communities

(b) CCE for LFR_Berry scaled with proportional number of communities

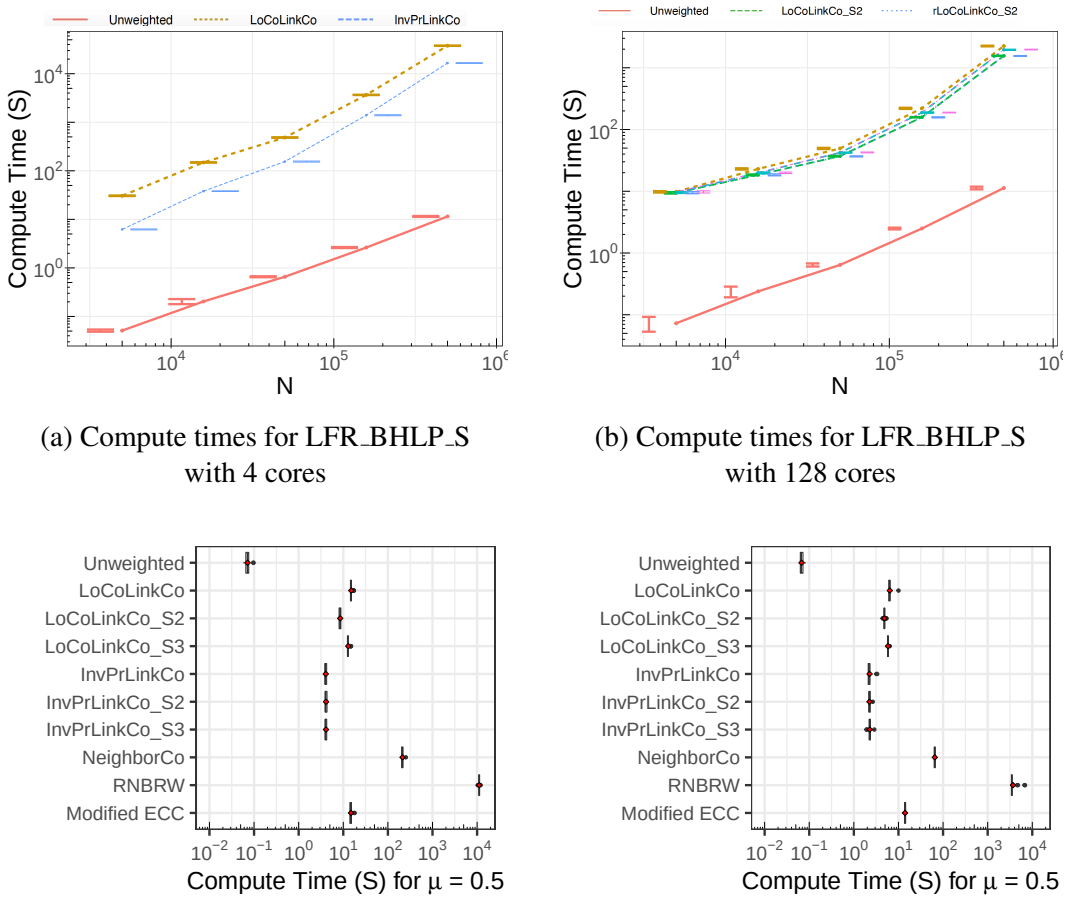


(c) NMI for LFR_Berry scaled with proportional number of communities

(d) CCE for LFR_Berry scaled with proportional number of communities

Figure 4.8: Community detection results for LFR_BHLP_S using the Louvain method with different pre-weighting methods. The unweighted graph is also used as a reference. The plots in (a) and (b) report the mean NMI and CCE for LoCoLinkCo and InvPrLinkCo with 95% confidence intervals from 30 random repetitions run on a laptop with 4 cores. The plots in (a) and (b) report the mean NMI and CCE for LoCoLinkCo with raw and normalized S2 and S3 LoCoLinkCo weights with 95% confidence intervals from 30 random repetitions run on a cluster with 128 cores.

some of the methods are much more computationally expensive than LoCoLinkCo. Specifically, NeighborCo and RNBRW are quite computationally expensive as implemented. NeighborCo is implemented in a similar way to InvPrLinkCo, except the method relies on several iterations to determine edge weights. RNBRW is implemented similar to the



(a) Compute times for LFR_BHLP_S with 4 cores

(b) Compute times for LFR_BHLP_S with 128 cores

(c) Compute times for LFR_BHLP using cycle-based pre-weighting methods with $\mu = 0.5$ and 4 cores

(d) Compute times for LFR_BHLP using cycle-based pre-weighting methods with $\mu = 0.5$ and 16 cores

Figure 4.9: Compute times for LFR graphs using the Louvain method. The plots in (a) and (b) contain the mean compute times with 95% confidence intervals from 30 random repetitions for LFR_BHLP_S using the Louvain method with the selected pre-weighting methods, (a) with 4 cores on a laptop and (b) with 128 cores on a cluster. The plots in (c) and (d) contain the boxplots of compute times from 30 random repetitions for the cycle-based methods on LFR_BHLP graphs with μ=0.5, (c) with 4 cores using Slurm on a cluster and (d) with 16 cores using Slurm on a cluster.

Python implementation shown in [175].

4.6.6 Experiments with Other Community Detection Algorithms

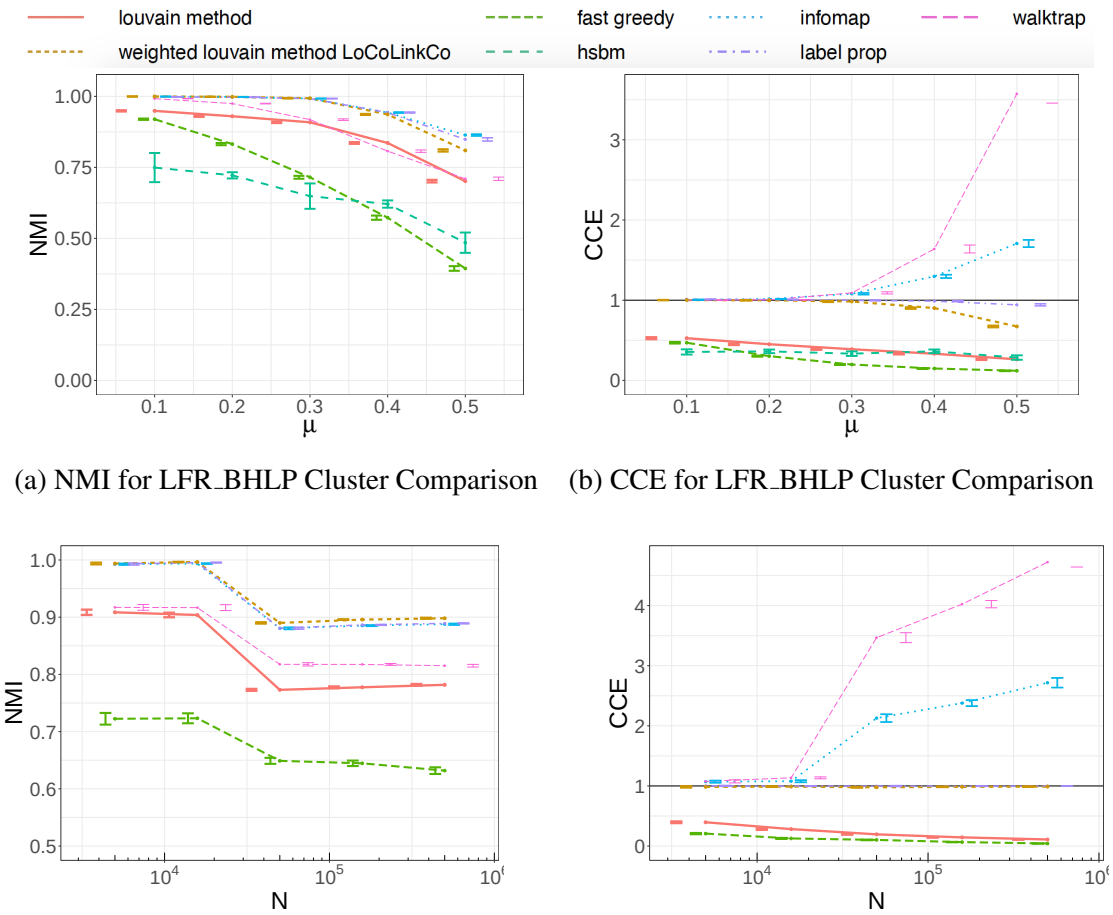
While pre-weighting the Louvain method with LoCoLinkCo provides better results than other pre-weighting methods and the unweighted Louvain method, there are other community detection options that may perform better on the benchmark LFR graphs. First, the LoCoLinkCo weighted and unweighted Louvain method is compared against the other community detection methods listed in Table 3.3 on LFR_BHLP and LFR_BHLP_S with results shown in Figure 3.13. The compute times for the other community detection algorithms, as implemented in the igraph library, are shown in Figure 4.11.

Both Infomap and Label Prop obtain high mean NMI, with Label Prop able to more accurately predict the correct number of communities and Infomap overshooting the number of communities. All of the baseline community detection algorithms also has the dip in performance at $N = 31,622$. Note that only 10 random repetitions were run for LFR_BHLP_S and that HSBM was not run on LFR_BHLP_S.

The LoCoLinkCo are not specific to the Louvain method and have the potential to improve the communities detected by other methods. Therefore, Figures 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.18, 4.19, 4.20, and 4.21 show results from experiments testing the potential performance improvements from using LoCoLinkCo to pre-weight the other community detection methods. All experiments use 30 random repetitions and the 95% confidence intervals are reflected in the plots.

Experiments using InvPrLinkCo to pre-weight a variety of community detection algoirithms lead to a number of observations.

Claim 4.6.4. (CS) LoCoLinkCo has been shown to consistently improve the communities detected by algorithms that maximize modularity (e.g. the Louvain method and Fast Greedy algorithm) on all of the tested LFR benchmark graphs with degree assortative community structure, which include graphs with degree assortative community structure(see



(a) NMI for LFR_BHLP Cluster Comparison

(b) CCE for LFR_BHLP Cluster Comparison

(c) NMI for LFR_BHLP_S Cluster Comparison

(d) CCE for LFR_BHLP_S Cluster Comparison

Figure 4.10: Community detection results for LFR graphs using the Louvain method plus InvPrLinkCo compared to other unweighted community detection methods described in Section 2.3. The plots in (a) and (b) report the mean NMI and CCE for LFR_BHLP with 95% confidence intervals from 30 random repetitions. The plots in (c) and (d) report the mean NMI and CCE for LFR_BHLP_S with 95% confidence intervals from 10 random repetitions.

Figures 4.12a, 4.12b, 4.14a, 4.14b, 4.18a, 4.18b, 4.20a, and 4.20b), graphs without degree assortative community structure (see FIgures 4.16a and 4.16b). Only in the benchmark graph that does not have degree assortative community structure (LFR_MSPM) does pre-weighting with LoCoLinkCo cause the modularity maximization algorithms to predict a number of communities further away from the true number of communities (see Figures 4.13a, 4.13b, 4.15a, 4.15b, 4.19a, 4.19b, 4.21a, 4.17a, and 4.17b).

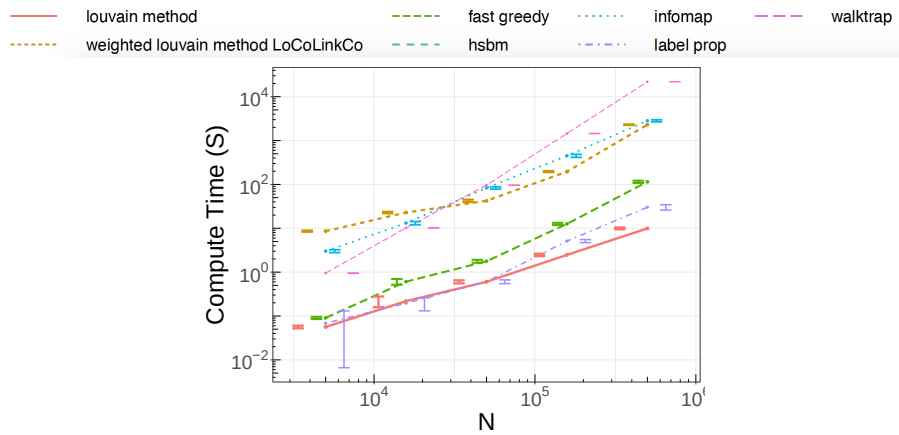
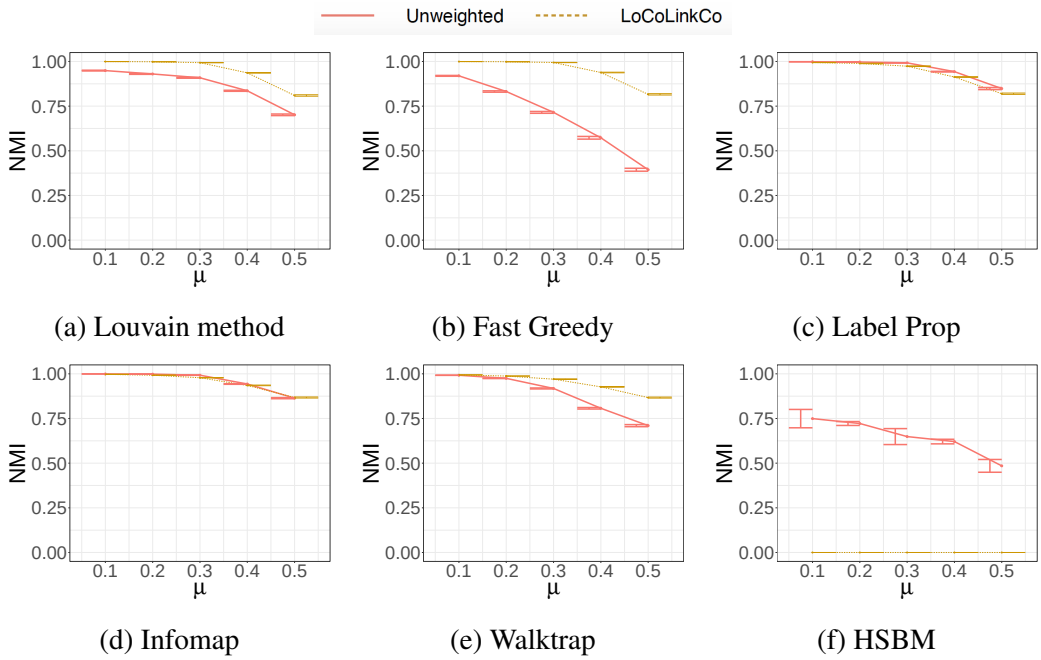


Figure 4.11: Compute time comparison to other community detection methods using LFR_BHLP_S. The plot contains the mean compute times with 95% confidence intervals from 10 random repetitions.

As stated, while the improvement in the LFR benchmark graphs without degree assortative community structure is statistically significant for the Louvain method pre-weighted with LoCoLInkCo compared to the unweighted Louvain method (with confidence intervals that the true difference in means is not equal to 0 being (0.0030, 0.0082) for $\mu = 0.1$, (0.0113, 0.0261) for $\mu = 0.2$, (0.0423, 0.0870) for $\mu = 0.3$, (0.0678, 0.1105) for $\mu = 0.4$, and (0.0569, 0.0796) for $\mu = 0.5$ using a Welch Two Sample t-test) – see Figure 4.16a) and more obviously the Fast greedy algorithm (see Figure 4.16b), the weights cause the algorithm to split communities apart as shown in Figures 4.17a 4.17b. This is due to the large community sizes relative to the degree of the largest vertices in the graphs – lower cycles are not very common even within a community.

Claim 4.6.5. (CS) Pre-weighting the highly connected LFR_SSGC graphs with LoCoLinkCo provide higher mean NMI values (see Figure 4.20) and more accurate predictions for the number of communities (see 4.21) for all community detection methods except HSBM (see Figures 4.20f and 4.21f).

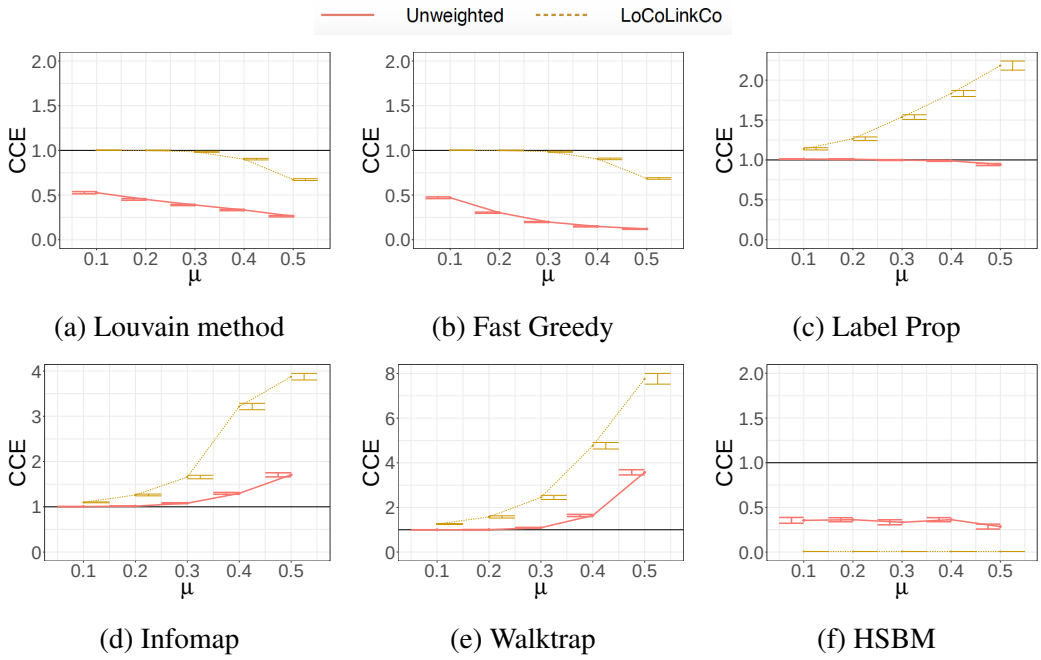
It should be noted that HSBM determines the number of communities by using the



(a) Louvain method (b) Fast Greedy (c) Label Prop

(d) Infomap (e) Walktrap (f) HSBM

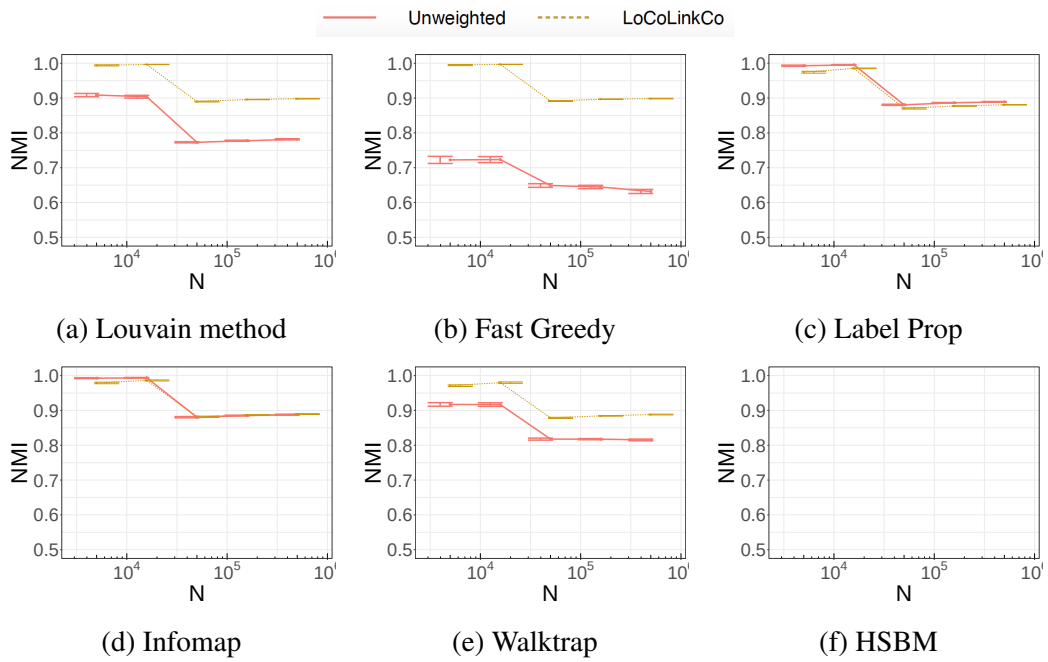
Figure 4.12: NMI for LoCoLinkCo verses unweighted community detection on LFR_BHLP.



(a) Louvain method (b) Fast Greedy (c) Label Prop

(d) Infomap (e) Walktrap (f) HSBM

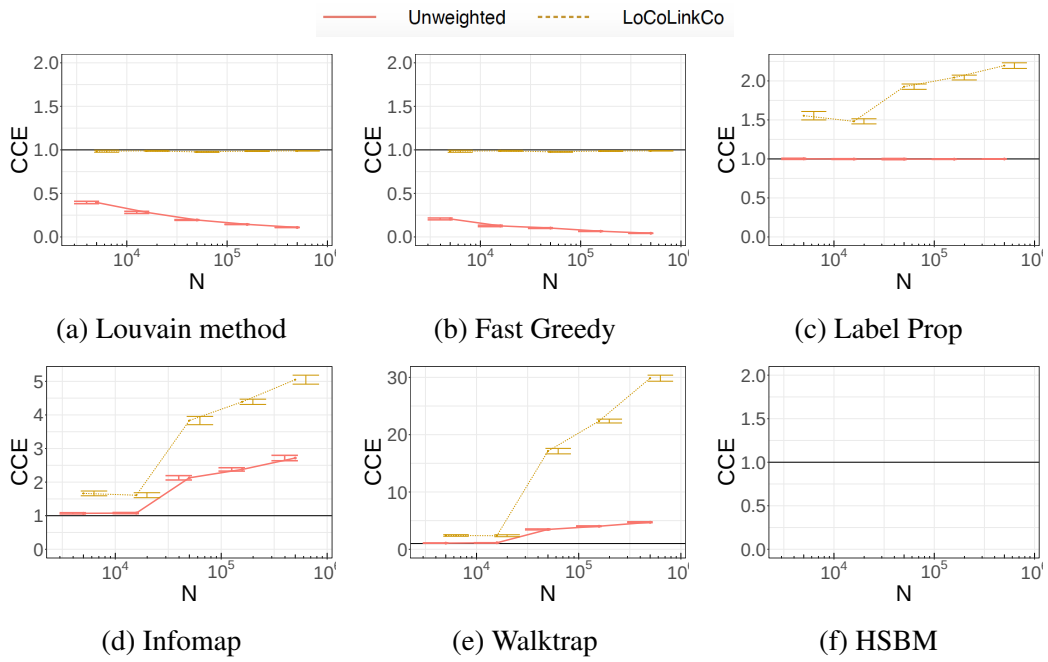
Figure 4.13: CCE for LoCoLinkCo verses unweighted community detection on LFR_BHLP.



(a) Louvain method (b) Fast Greedy (c) Label Prop

(d) Infomap (e) Walktrap (f) HSBM

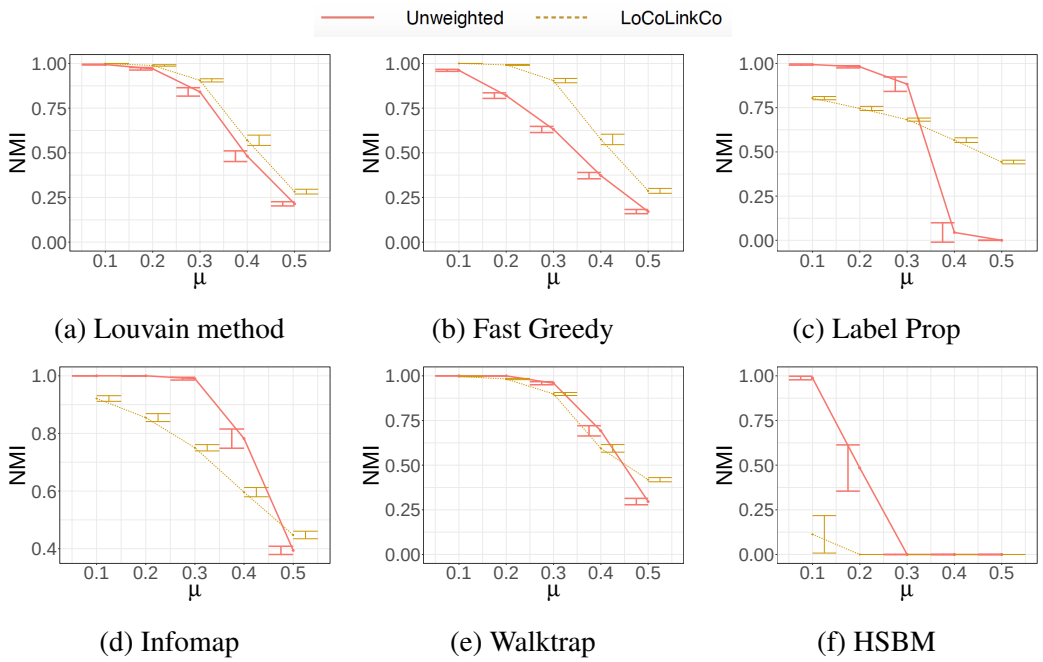
Figure 4.14: NMI for LoCoLinkCo verses unweighted community detection on LFR_BHLP_S.



(a) Louvain method (b) Fast Greedy (c) Label Prop

(d) Infomap (e) Walktrap (f) HSBM

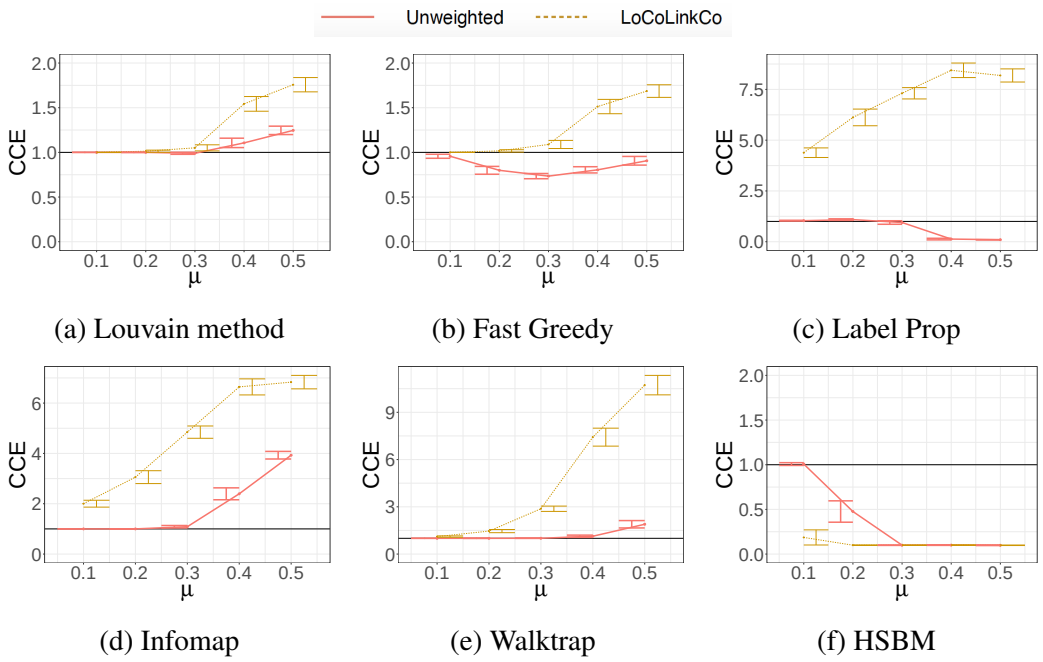
Figure 4.15: CCE for LoCoLinkCo verses unweighted community detection on LFR_BHLP_S.



(a) Louvain method (b) Fast Greedy (c) Label Prop

(d) Infomap (e) Walktrap (f) HSBM

Figure 4.16: NMI for LoCoLinkCo verses unweighted community detection on LFR_MSPM.



(a) Louvain method (b) Fast Greedy (c) Label Prop

(d) Infomap (e) Walktrap (f) HSBM

Figure 4.17: CCE for LoCoLinkCo verses unweighted community detection on LFR_MSPM.

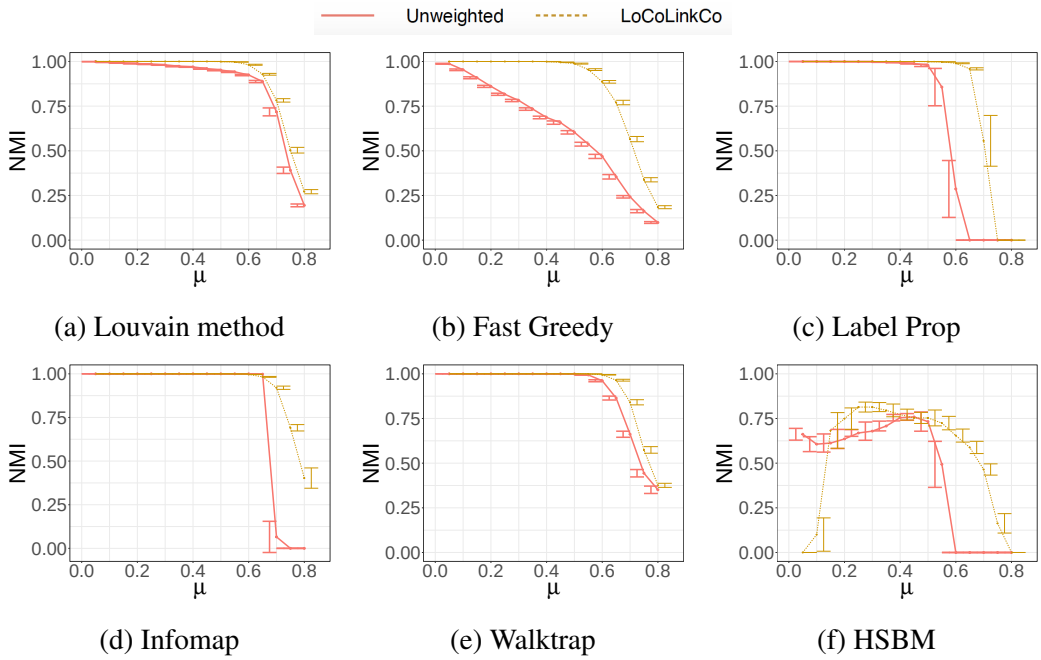


Figure 4.18: NMI for LoCoLinkCo verses unweighted community detection on LFR_KRH.

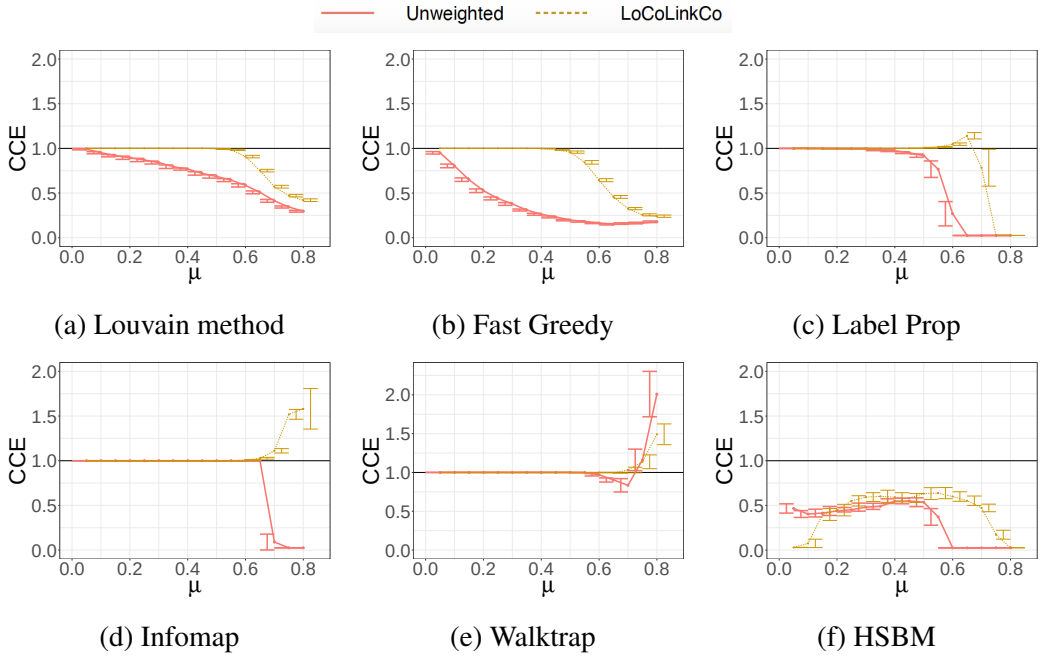
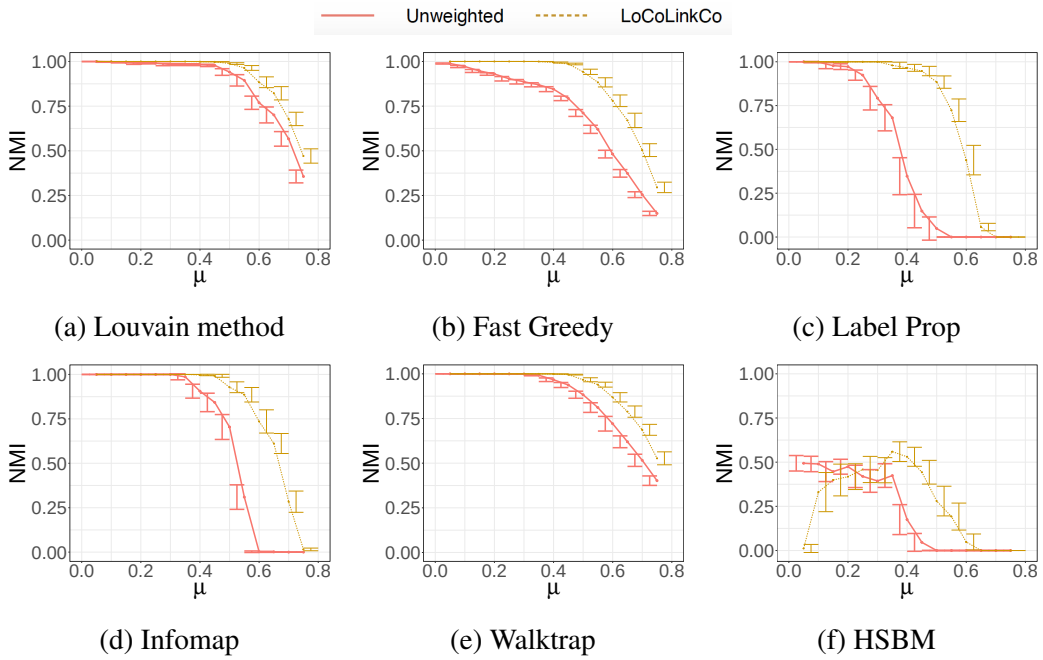


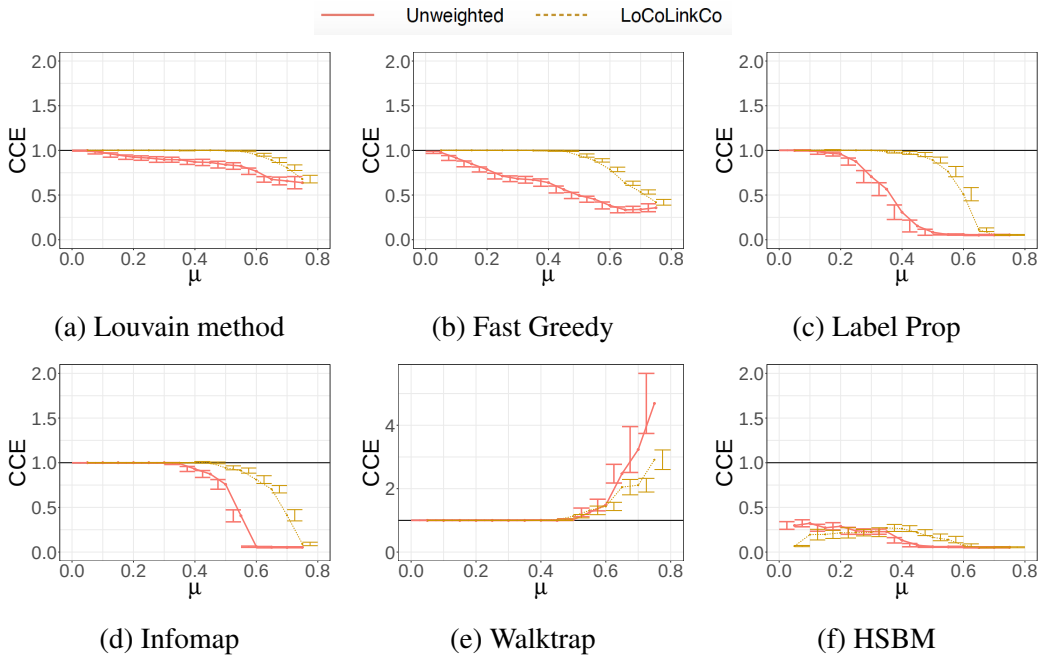
Figure 4.19: CCE for LoCoLinkCo verses unweighted community detection on LFR_KRH.



(a) Louvain method (b) Fast Greedy (c) Label Prop

(d) Infomap (e) Walktrap (f) HSBM

Figure 4.20: NMI for LoCoLinkCo verses unweighted community detection on LFR_SSGC.



(a) Louvain method (b) Fast Greedy (c) Label Prop

(d) Infomap (e) Walktrap (f) HSBM

Figure 4.21: CCE for LoCoLinkCo verses unweighted community detection on LFR_SSGC.

standard "elbow-finding" methodologies [159], and it is possible that tuning could increase the number of communities it detects.

With the LFR_KRH benchmark graphs shown in Figures 4.18 and 4.19, which are also relatively highly connected, most methods under most of the mixing conditions show an improvement in mean NMI and mean CCE when pre-weighted with LoCoLinkCo. The exception again is HSBM and when $\mu = 0.65$ for Infomap. Sometimes adding the LoCoLinkCo weighting causes the methods to over predict the number of communities, as is seen with Infomap and Label Prop for higher mixing parameters, although the added weights helps the Walktrap algorithm detect fewer communities yielding closer values to ground truth.

Sparser graphs, as are the LFR_BHLP (Figures 4.12 and 4.13), LFR_BHLP_S (Figures 4.14 and 4.15), and LFR_MSPM (Figures 4.16 and 4.17) benchmarks, the benefits are less clear for algorithm not based on maximizing modularity. This is similar to the observations made in Chapter 3 (see Subsection 3.7.5). In many of these cases, pre-weighting with LoCoLinkCo causes the community detection algorithms to break true communities apart. As stated previously, a hypothesis for why this occurs is that Infomap, Label Prop, and Walktrap (for graphs not influenced by Popularity$\times$Similarity optimization) are able to utilize cycles of higher order into their solution on unweighted graphs, therefore focusing on only shorter cycles when the characteristic path length is relatively high causes a degradation in the performance.

Claim 4.6.6. (CS) Similar to Claim 3.7.7, certain non-modularity based community detection algorithms (e.g. Infomap, Label Prop and sometimes Walktrap) obtain degraded NMI when LoCoLinkCo is used to pre-weight sparse networks (see Figures 4.12, 4.14, and 4.16), but unline in the nPSO graphs (see Figures 3.17c and 3.17d), Infomap and Walktrap all seem to have a tendency to over predict the true number of clusters when unweighted

(see Figures 4.13, 4.15, and 4.17) which is exacerbated when weighted with LoCoLinkCo.

The degradation is less significant when the ground truth communities are small in size, leading to resolution limits that may be benefitted by the weights. When the ground truth communities are large, as is the case with the LFR_MSPM benchmark, pre-weighting with LoCoLinkCo yields significantly degraded detected communities.

4.6.7 Additional Observations about Local Community Link Cohesion

While in many cases, pre-weighting graphs with LoCoLinkCo edge weights yield detected communities more similar to ground truth for many community detection algorithms, having a way to determine when pre-weighting would likely degrade performance is important. It is also important to be able to determine if a graph has no underlying community structure, as obtaining predicted communities in a random graph is not helpful. Link cohesion weights are deterministic, unlike most community detection algorithms. Therefore, making determinations on whether or not a graph has community structure or if LoCoLinkCo is expected to improve performance based on the distribution in link cohesion weights has some appeal. An additional concern is determining if the underlying community structure is more similar to the type of community structure obtained from Popularity$\times$Similarity optimization or degree assortative community structure, as using the right link cohesion metric has been shown to make a difference.

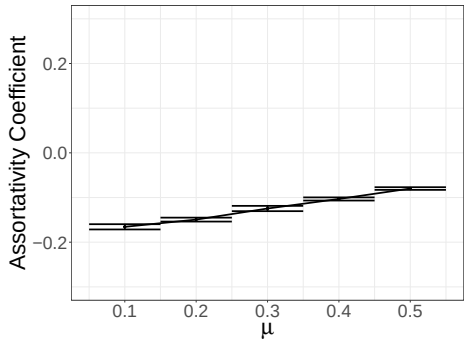
While certain graph-level attributes such as the size of a graph and the average local clustering coefficeint (see Definition 2.2.7) or global clustering coefficient (see Definition 2.2.8) can be obtained in either nPSO graphs or LFR graphs, it is of interest to know if other graph-level attributes could be used to classify which type of community structure a graph likely has. In general, LFR graphs tend to have smaller maximum degree vertices, otherwise the largest communities would be very large. That said, there are many graphs

that have maximum degrees in a range that could yield either type of community structure. In 2005, Manuel Middendorf, Etay Ziv, and Chris H. Wiggins investigated using the frequency of certain subgraphs to determine network type [169]. In 2011, James A. Henderson and Peter A. Robinson analyzed attributes related to the clustering coefficient (see Definition 2.2.8) and the characteristic path length (see Definition 2.1.17) on its effects on network structure [122]. Before analyzing local community link cohesion, which contains semi-local information related to clustering and characteristic path length which encoding details about the presence of triangle (3-cycle) and rectangle (4-cycles) subgraphs while additionally taking degrees of the vertices in the subgraphs into account, the assortativity coefficient (see Definition 2.2.5) and the rich-clubness (p-value) (see Definition 2.2.13) are studied as candidates for making the distinction on the type of community structure. Figure 4.22 shows the values obtained from these measures for the LFR_BHLP, LFR_MSPM, and LFR_KRH benchmark graphs. Not shown are LFR_SSGC which consistently yield negative assortativity coefficient values with rich-clubness (p-value) close to 0 for small μ, nPSO_EU which consistently yield negative assortativity coefficient values with rich-clubness (p-value) always equal to 0, and nPSO_BHLP1 which consistently yield positive assortativity coefficient values with rich-clubness (p-value) always equal to 0.

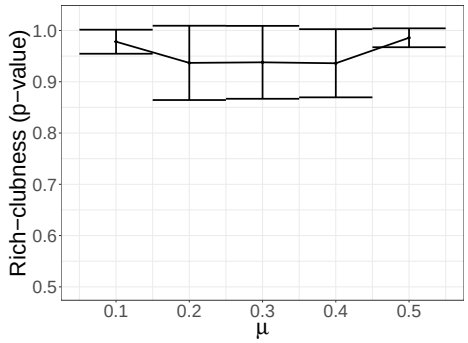
While nPSO graphs are almost always rich-club networks, some examples in [185] show some parameter settings that create graphs that are not rich-club networks (a different null model is used for determining if a graph is a rich-club network, with difference scores instead of z-scores [183]).

Claim 4.6.7. (CS) Assortatively mixed rich-club networks, disassortatively mixed rich-club networks, and assortatively mixed and disassortatively mixed graphs that are not rich-club networks can be created using both nPSO graphs and LFR graphs.

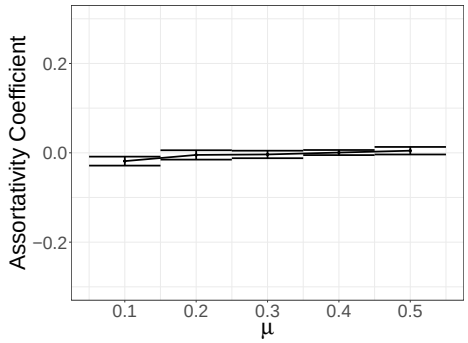
One interesting observation of certain LFR graphs are that they can sometimes have



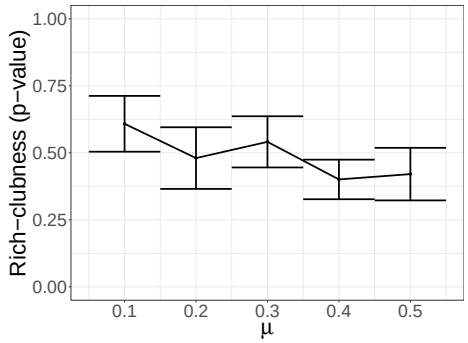
(a) Assortativity Coefficient for LFR_BHLP



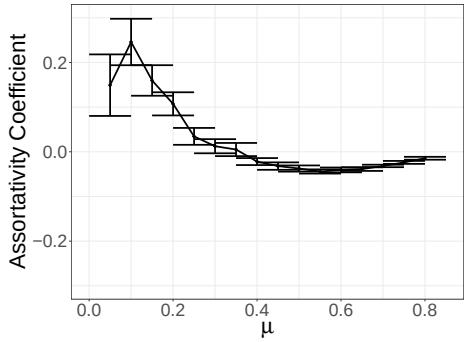
(b) Rich-clubness (p-value) for LFR_BHLP



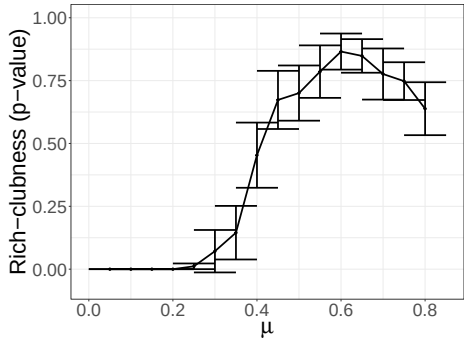
(c) Assortativity Coefficient for LFR_MSPM



(d) Rich-clubness (p-value) for LFR_MSPM



(e) Assortativity Coefficient for LFR_KRH



(f) Rich-clubness (p-value) for LFR_KRH

Figure 4.22: Assortativity and rich-clubness of benchmark graphs. The plots in (a) and (b) report the mean assortativity coefficient and rich-clubness (p-value) for LFR_BHLP with 95% confidence intervals from 30 random repetitions. (c) and (d) show similar plots to (a) and (b) for LFR_MSPM. (e) and (f) show similar plots to (a) and (b) for LFR_KRH.

anti rich-club network behavior – rich-clubness p-values that are consistently high as seen in Figure 4.22b. This is probably due to the highest degree vertices being placed in different

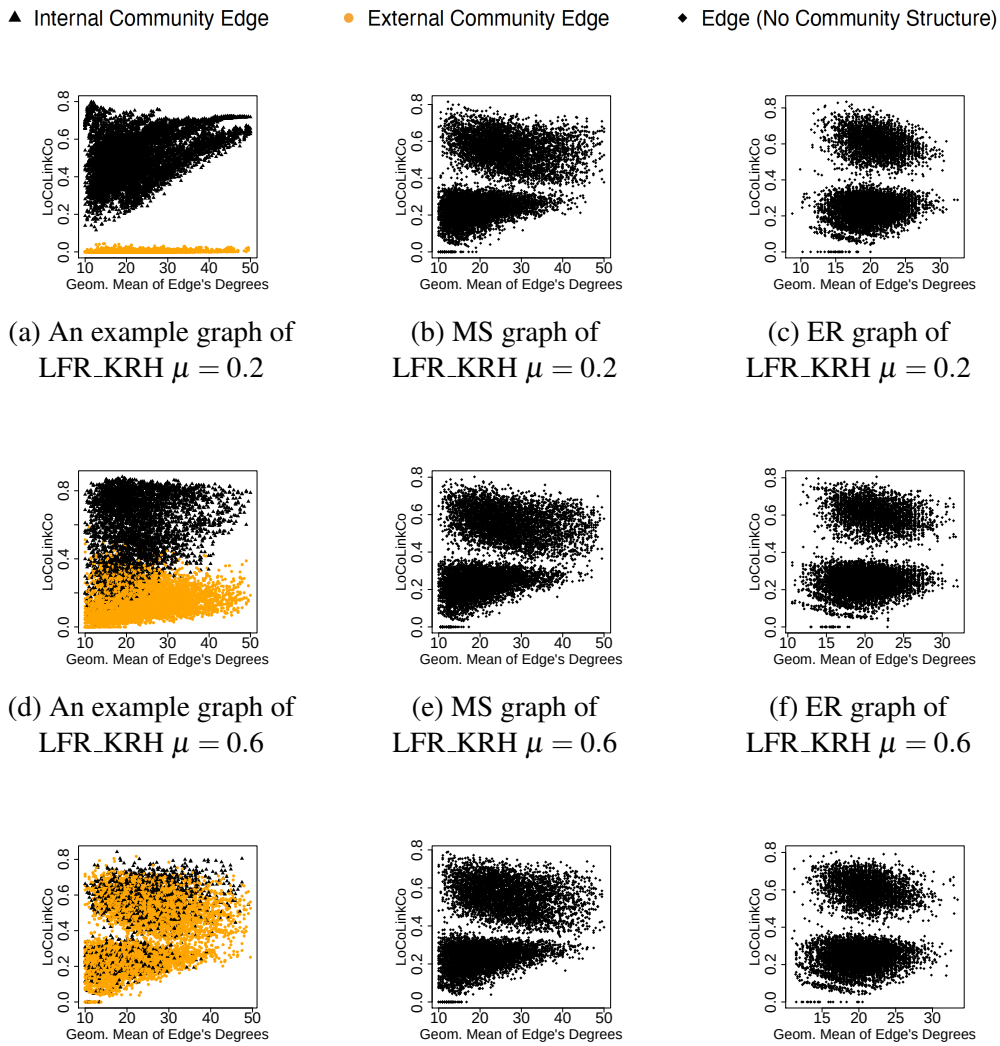
communities. With a majority of the connections being made within the communities, they are less likely to be connected to each other than a fully random graph.

Claim 4.6.8. (CS) An anti rich-club network with rich-clubness p-values that are consistently high, as seen in Figure 4.22b, is characteristic of an LFR graph that is not likely to be observed in nPSO graphs.

That said, this may be a characteristic of the LFR benchmark as such a network has not been observed in the real-world networks analyzed though the highest degree vertices in Figure 3.2a seem to be anti rich-club between degrees $k = 150$ to $k = 200$.

A preliminary analysis is conducted on analyzing the LoCoLinkCo weights to determine if a graph has community structure. Specifically, LoCoLinkCo weights are plotted verses geometric mean of vertex degrees of the edges in a graph. The first set of plots shown in Figure 4.23 are using the LFR_KRH benchmark that has very different assortativity coefficients and rich clubness p-values as μ is varied. When the mixing between communities is very low e.g. $\mu = 0.2$ in Figure 4.23a, there is a separation between the internal community edges and the external community edges. Also shown in the top row are similar plots for a rewired version of the graph in Figure 4.23b (an MS graph, see Figure 2.9) and an ER graph in Figure 4.23c (See Subsection 2.5.1) generated with the same number of vertices and edges. Both random graphs have a distinctive pattern due to the LoCoLinkCo_S2 weight and LoCoLinkCo_S3 weights being uncorrelated.

As the mixing rate is increased to $\mu = 0.6$ in Figure 4.23d, the edges with low geometric mean of their vertex degrees obtain a wider range of weights with internal community edges taking on values that external community edges also have. The edges connecting the highest degree vertices tend to have larger LoCoLinkCo edge weights if they are part of the same community and lower LoCoLinkCo edge weights if they are part of different communities. The corresponding random graphs have the same distinctive pattern as the



(a) An example graph of LFR_KRH $\mu = 0.2$

(b) MS graph of LFR_KRH $\mu = 0.2$

(c) ER graph of LFR_KRH $\mu = 0.2$

(d) An example graph of LFR_KRH $\mu = 0.6$

(e) MS graph of LFR_KRH $\mu = 0.6$

(f) ER graph of LFR_KRH $\mu = 0.6$

(g) An example graph of LFR_KRH $\mu = 0.8$

(h) MS graph of LFR_KRH $\mu = 0.8$

(i) ER graph of LFR_KRH $\mu = 0.8$

Figure 4.23: LoCoLinkCo verses geom. mean of an edge's degree for LFR_KHR. The first row is based on an example graph with $mu = 0.2$, the second row is based on an example graph with $mu = 0.6$, and the third row is based on an example graph with $\mu = 0.8$. In each case, the LFR graph is compared to rewired version, aka an MS graph, and an ER graph with the same size and average degree.

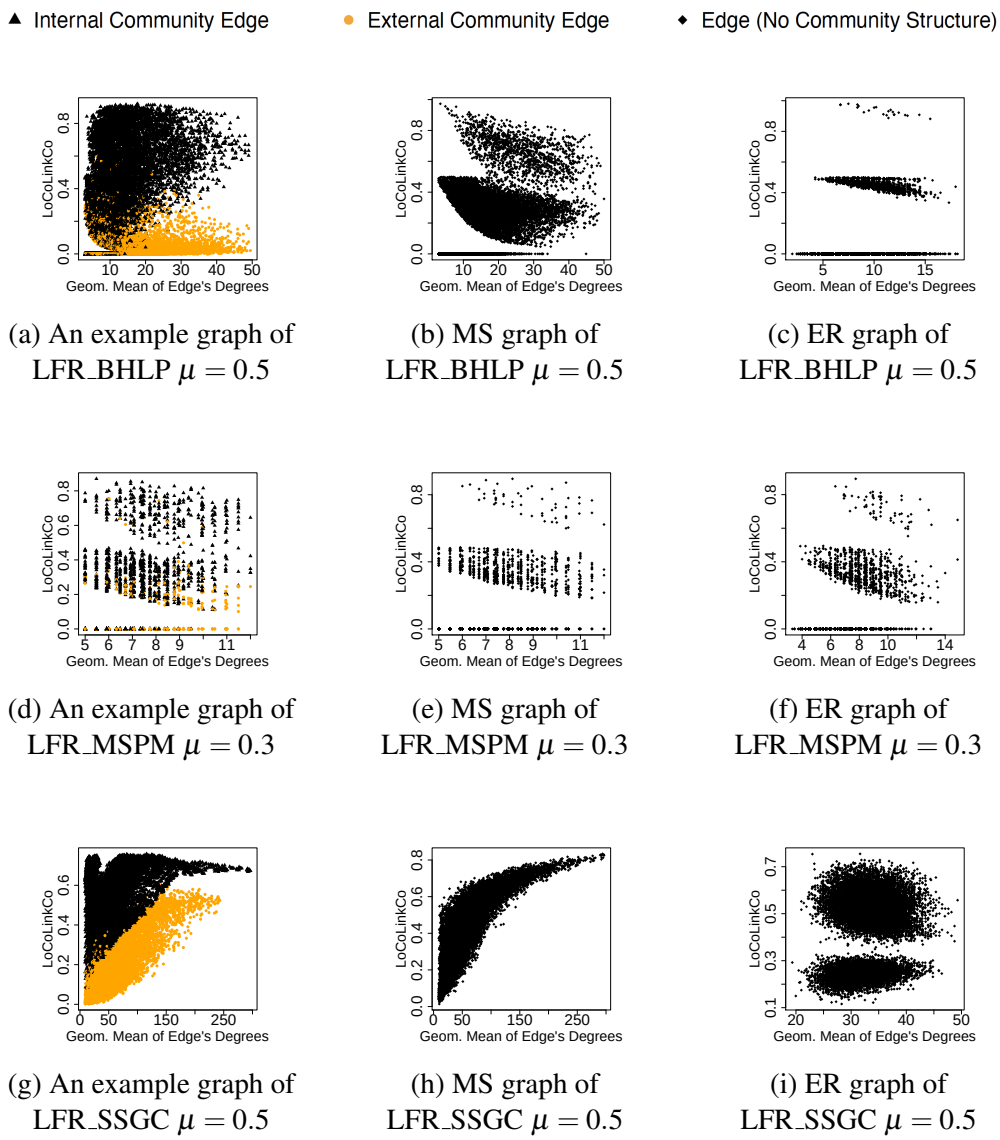
other random graphs. When $\mu = 0.8$ in Figure 4.23g, the LFR_KRH graphs seem to become mostly random as well, obtaining very similar patterns to the randomized versions of the graph, particuarly the rewired version.

Unlike the assortativity coefficient, which is a graph-level metric, visualizing LoCoL-

inkCo edge weights verses the geometric mean of the vertex degrees of the edges in a graph gives insights into how the highest degree vertices are mixed. While the rich-clubness p-value also provides a value that can lead to insights on how the highest degree vertices are mixed, the single value is not able to reflect the difference between highly connected vertices within a community that tend to be connected with large LoCoLinkCo weights and highly connected vertices in different communities that tend to be connected with low LoCoLinkCo weights.

Figure 4.24 contains similar examples for the other LFR benchmkark graphs. Both the example LFR_BHLP graph shown in Figure 4.24a and the example LFR_SSGC graph shown in Figure 4.24g have similar separation among edges that connect the highest degree vertices. The example LFR_MSPM graph shown in Figure 4.24d looks similar to a random graph. This is because it is the LFR benchmark without degree assortative community structure. Since the graph is sparse and the community sizes are much larger than the highest degree vertices in the graphs, communities likely contain more higher order cycles than triangles and rectangles. Note that the density of the LFR_MSPM graphs are higher than the densities of the LFR_BHLP since the LFR_BHLP graphs have an order of magnitude more vertices. The pattern differences are due to the sizes of the communities relative to the degrees of the vertices. The rewired graph of LFR_SSGC shown in Figure 4.24h has a very different pattern than the rewired graphs of the other examples. This is because the LFR_SSGC graph has highly connected vertices with a maximum degree of 300 compared to the other benchmark graphs that have much lower maximum degrees. Therefore, the highest degree vertices are all likely to be connected to each other in the MS graphs simply because they have many more edges compared to other vertices.

The next set of plots shown in Figure 4.25 are of the nPSO benchmark graphs from Chapter 3. Unlike the distinctive separation seen in the edge weights connecting the highest degree vertices in the benchmark graphs with degree assortative community structure,



(a) An example graph of LFR_BHLP $\mu = 0.5$

(b) MS graph of LFR_BHLP $\mu = 0.5$

(c) ER graph of LFR_BHLP $\mu = 0.5$

(d) An example graph of LFR_MSPM $\mu = 0.3$

(e) MS graph of LFR_MSPM $\mu = 0.3$

(f) ER graph of LFR_MSPM $\mu = 0.3$

(g) An example graph of LFR_SSGC $\mu = 0.5$

(h) MS graph of LFR_SSGC $\mu = 0.5$

(i) ER graph of LFR_SSGC $\mu = 0.5$

Figure 4.24: LoCoLinkCo verses geom. mean of an edge's degree for other LFR graphs. The first row is based on an LFR_BHLP graph with $mu = 0.5$, the second row is based on an LFR_MSP graph with $mu = 0.3$, and the third row is based on an LFR_SSGC graph with $\mu = 0.8$. In each case, the LFR graph is compared to rewired version, aka an MS graph, and an ER graph with the same size and average degree.

the highest degree vertices in the nPSO graphs all tend to have similar values. This makes sense, as the most popular vertices are typically from various communities but connected to each other due to their popularity. In highly connected graphs, such as the example nPSO_EU graph and the real-world G_{EU} graph, the edges connecting the highest degree

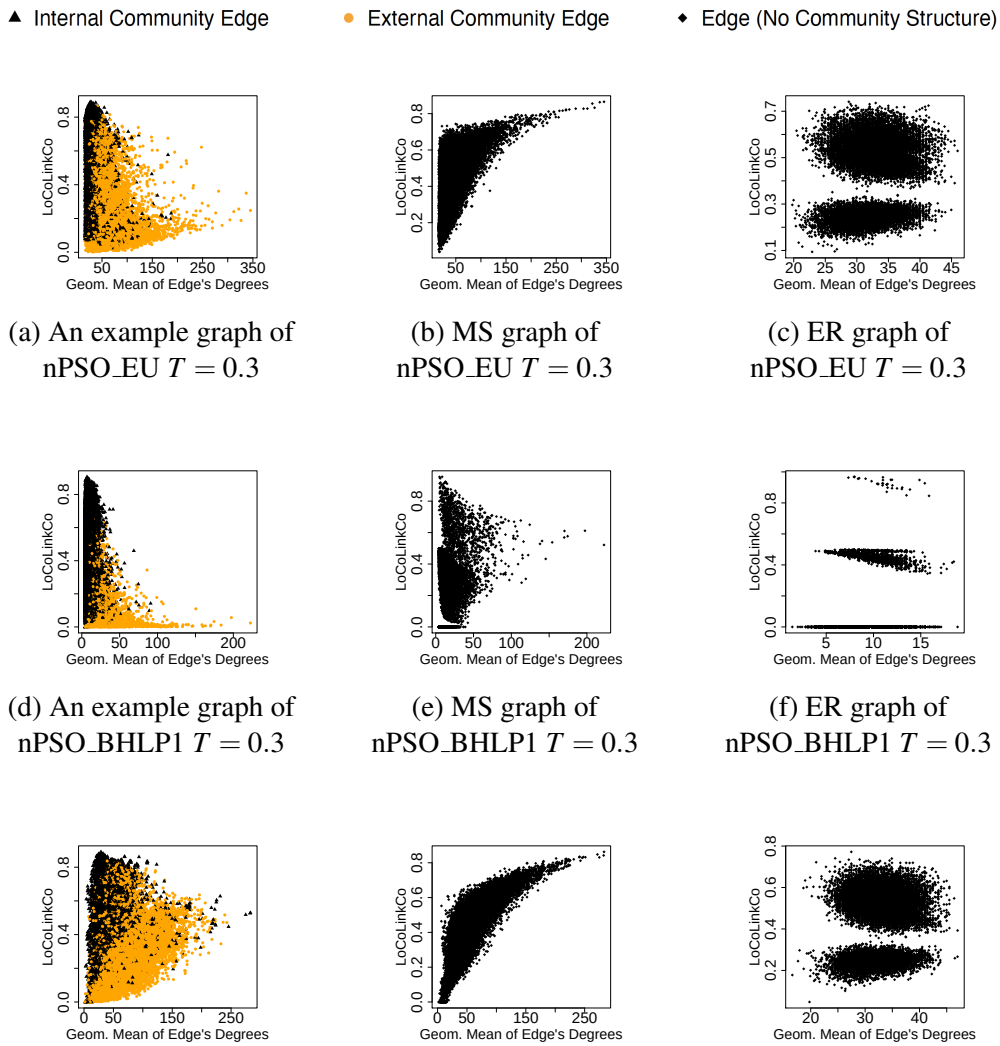
vertices have mid-range LoCoLinkCo edge weights. In sparser graphs, such as the example nPSO_BHLP1 graph, the edges connecting the highest degree vertices have very low LoCoLinkCo edge weights since there is larger separation in the graph overall. When this is the case, there are no edges that connect high degree vertices with high LoCoLinkCo edge weights, as was seen in Figures 4.23 and 4.24.

The random rewired networks (MS graphs) of the highly connected graphs influenced by Popularity×Similarity optimization follow a pattern similar to the rewired network of LFR_SSGC shown in Figure 4.24h for the same reasons. The highest degree vertices in the MS graphs have many more edges than the other vertices and with so many connections, they are very likely to be connected. The rewired graph of nPSO_BHLP1 shows a very different pattern which makes it look like the highly connected graphs influenced by Popularity×Similarity optimization. The main difference is that there are much fewer vertices that have higher LoCoLinkCo edge weights by chance, compared to the graphs with true community structure.

Based on these observations, a number of preliminary claims are made that need to be further analyzed.

Claim 4.6.9. (CS) Graphs with degree assortative community structure and more than one large community may have two distinct sets of weights for edges connecting the highest degree vertices (see Figures 4.23a, 4.23d, 4.24a, and 4.24g) if the mixing rate of the vertices (see Definition 2.2.10) is not too high (see Figure 4.23g) while graphs influenced by Popularity×Similarity optimization are only expected to have one set that has mid-range weights (see Figures 4.25a and 4.25g) to low weights (see Figure 4.25d),, but not high weights due to the high penalty of cycles made up of just high degree vertices (see Claim 3.4.10).

Claim 4.6.10. (CS) Erdős -Rényi (ER) graphs have a distinct disjoint pattern in graphs



(a) An example graph of nPSO_EU $T = 0.3$

(b) MS graph of nPSO_EU $T = 0.3$

(c) ER graph of nPSO_EU $T = 0.3$

(d) An example graph of nPSO_BHLP1 $T = 0.3$

(e) MS graph of nPSO_BHLP1 $T = 0.3$

(f) ER graph of nPSO_BHLP1 $T = 0.3$

(g) The simplified graph of the email-Eu-core [148], G_{EU}

(h) MS graph of G_{EU}

(i) ER graph of G_{EU}

Figure 4.25: LoCoLinkCo verses geom. mean of an edge's degree for nPSO graphs. The first row is based on an nPSO_EU graph with $T = 0.3$, the second row is based on an nPSO_BHLP1 graph with $T = 0.3$, and the third row is the G_{EU} graph from Section 3.2. In each case, the LFR graph is compared to rewired version, aka an MS graph, and an ER graph with the same size and average degree.

plotting the LoCoLinkCo weight of an edge verses the geometric mean of the vertices degrees of the edge with two clusters away from a LoCoLinkCo weight of 0 (see Figures 4.23c, 4.23f, 4.23i, 4.24c, 4.24f, 4.24i, 4.25c, 4.25f, and 4.25i) due to the LoCoLinkCo_S2 and LoCoLinkCo_S3 weights being uncorrelated.

Claim 4.6.11. (CS) Graphs with a maximum degree on the same order of magnitude as an expected maximum degree of an ER graph may yield MS graphs similar to ER graphs (see Figures 4.23b, 4.23e, 4.23h, 4.24b, and 4.24e), whereas graphs with a maximum degree an order of magnitude larger than the expected maximum degree of an ER graph may yield MS graphs that are strongly correlated (see Figures 4.24h, 4.25b, and 4.25h) if the graphs are not sparse (see Figure 4.25e).

Claim 4.6.12. (CS) Graphs not influenced by Popularity$\times$Similarity optimization that have large communities relative to the size of the maximum degree in the graph may have similar patterns to ER graphs (see Figure 4.24d). This may be indicative that higher order cycles are needed to detect community structure.

The last analysis of the LoCoLinkCo edge weights to determine community structure is done on a Hierarchical Stochastic Block Model (HSBM) introduced by Vince Lyzinski, Minh Tang, Avanti Athreya, Youngser Park, and Carey Priebe in 2016 [159]. Such a graph has block community structure with different motif patterns within the communities, as shown in Figure 4.26. The Louvain method pre-weighted with LoCoLinkCo weights are able to correctly identify the 8 communities. The LoCoLinkCo weights for the graph are shown in Figure 4.27. In Figure 4.27, the LoCoLinkCo weights for internal edges of each of the predicted communities, which align exactly with the ground truth communities, are shown separately as well.

First, there is clear separation between the edges found within communities and edges between communities, as shown in FIgure 4.27a. Additionally, different motifs show different patterns in the LoCoLinkCo edge weights of the internal community edges shown in (b)-(h) of Figure 4.27. Unlike in [159] where the exact block structure can be recovered, only relative patterns in the probability of being connected is reflected in each community. Specifically, it is hard to discern the B1 and B3 which have similar connection patterns but

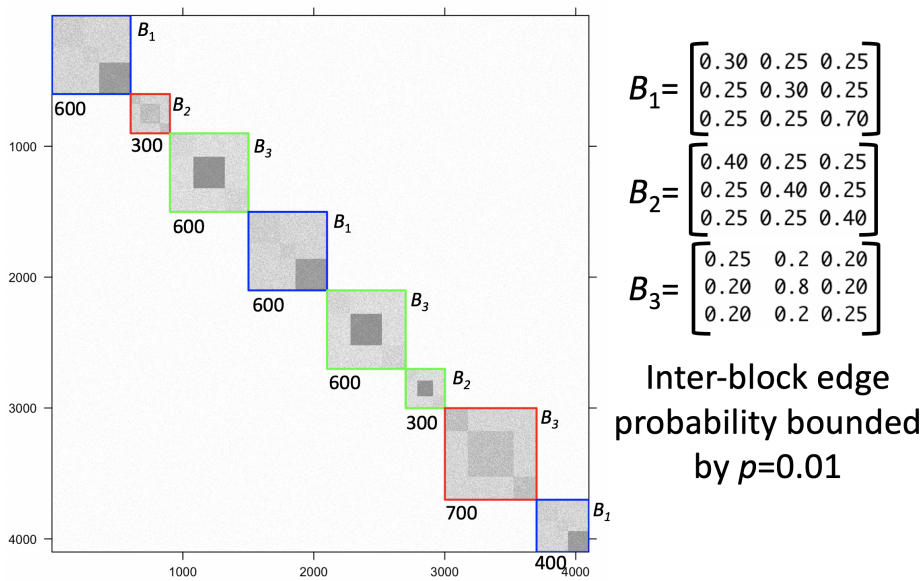


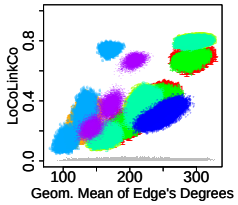
Figure 4.26: An example HSBM graph similar to the graph in [159].

arranged in a different order in the motif. That said, it is interesting to see the structure reflected in the plots.

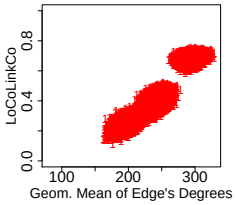
Claim 4.6.13. (CS) Hierarchical stochastic block models with motif substructures have patterns that can partially be observed by looking at the LoCoLinkCo weights verses the geometric mean of the vertex degrees of the corresponding edge.

4.6.8 Communities in the Co-authorship Network of Published Researchers

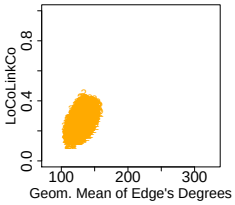
Without ground truth communities, the performance of community detection algorithms on G_{DBLP} needs to be evaluated in a different way. First, does G_{DBLP} have community structure? And if so, what type? The observations from Subsection 4.6.7 can be used in the analysis. It is as assortatively mixed rich-club network, which could mean anything. Figure 4.28 shows the G_{DBLP} graph, the LoCoLinkCo weights verses the geometric mean of the vertex degrees of its edges, and the MDCore of the graph using LoCoLinkCo weights (see Section 3.5)).



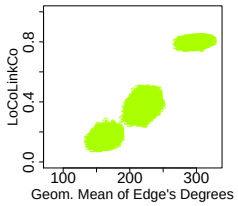
(a) Edges in the entire HSBM graph



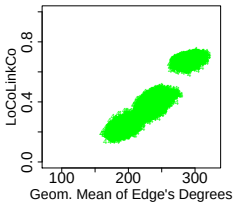
(b) Internal edges in predicted community 1



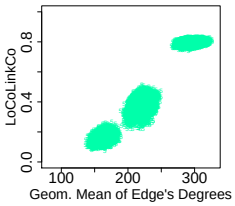
(c) Internal edges in predicted community 2



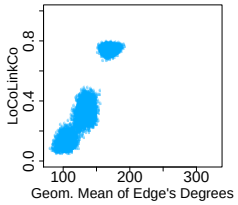
(d) Internal edges in predicted community 3



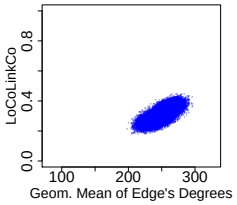
(e) Internal edges in predicted community 4



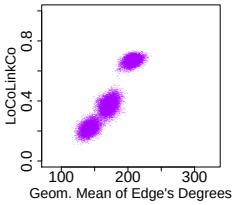
(f) Internal edges in predicted community 5



(g) Internal edges in predicted community 6



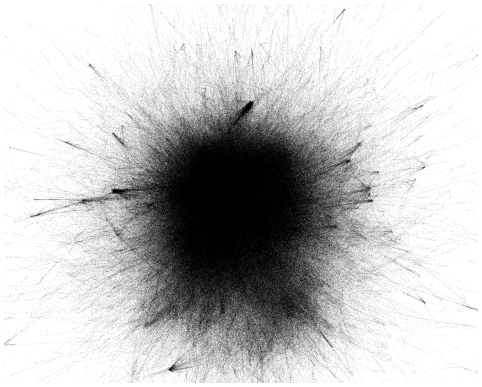
(h) Internal edges in predicted community 7



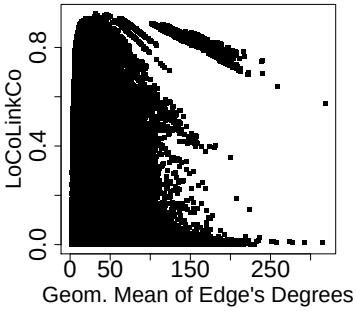
(i) Internal edges in predicted community 8

Figure 4.27: LoCoLinkCo verses geom. mean of an edge's degree for the HSBM graph from Figure 4.26. The first figure illustrates all of the edges, both internal community edges shown in rainbow colors and external community edges shown in gray. The remaining figures show the internal community edges for each predicted community, which aligns exactly with the blocks of the SBM.

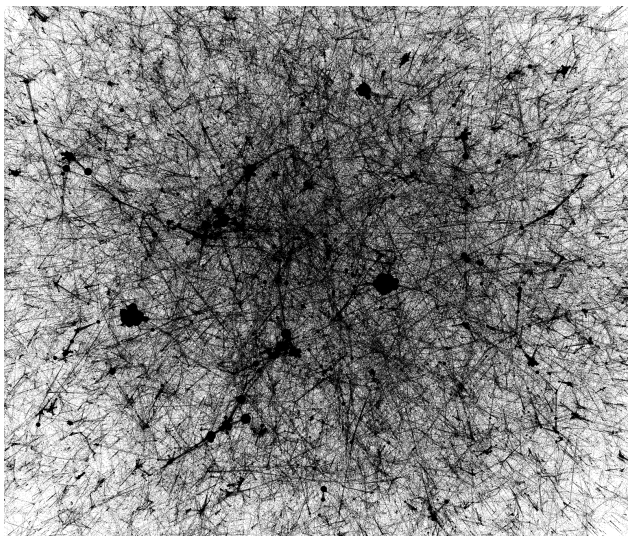
Claim 4.6.14. (CS) LoCoLinkCo weights of the graph G_{DBLP} have the distinctive pattern that edges connecting the highest degree vertices either have relatively high LoCoLinkCo weights or near zero LoCoLinkCo weights (see Figure 4.28b) expected for graphs with degree assortative community structure. Additionally, observing the MDCore of G_{DBLP}



(a) G_{DBLP} visualized using Gephi



(b) LoCoLinkCo verses the geometric mean of the vertex degrees of the edges in G_{DBLP}



(c) MDCore of G_{DBLP} visualized using Gephi

Figure 4.28: Community structure of the G_{DBLP} and its MDCore. (a) G_{DBLP} graph (b) LoCoLinkCo verses geom. mean of an edge's degree for G_{DBLP} graph (c) MDCore of G_{DBLP} with LoCoLinkCo.

LoCoLinkCo weights seem to highlight structure that does not seem random (see Figure 4.28c).

An additional interesting observation of G_{DBLP} made form analyzing the LoCoLinkCo weights is that 43,181 vertices have edges that are not supported by any nearby alternate paths of length 2 or 3. This was also observed in the LFR_BHLP_S graphs. When 10 LFR_BHLP_S graphs were generated with the same number of vertices as G_{DBLP}, the mean number of vertices that had edges that were not supported by any nearby alternate paths of

length 2 or 3 was 45,912. 6 of the 10 graphs had a value in the 70 thousands and 4 had a value of 1. The average number of communities in the benchmark graphs is 3,227 with a standard deviation of 72. The maximum community size in the benchmark graphs is 1,010 with a standard deviation of 4.

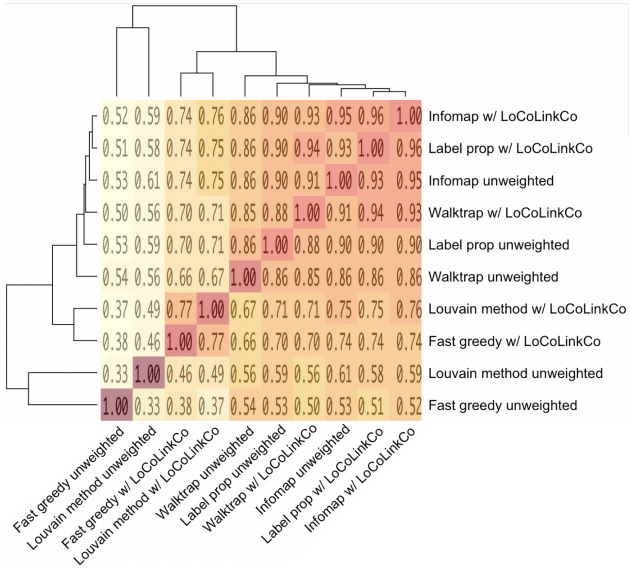
As was done in Paul Wagenseller III and Feng Wang in their 2018 paper analyzing communities without ground truth, predicted communities are judged based on their community size distributions. The community sizes are chosen to reflect research done by Robin Dunbar shown in Table 2.1. The results for the weighted and unweighted community detection methods are shown in Table 4.2. Figure 4.29a illustrates how the communities detected by different methods are related to each other by computing the NMI of each pair of community detection method.

Table 4.2: Predicted community size distribution in G_{DBLP} from different methods.

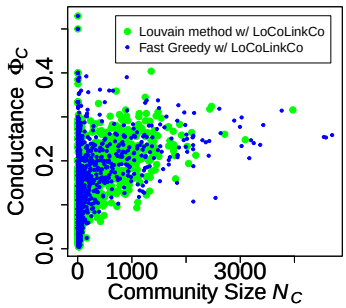
Method \Community size range	1-3	4-5	6-15	16-50	51-150	151-500	501-1500	1501-5000	5000+	Total
Louvain method unweighted	0	8	98	30	3	18	61	45	12	275
Louvain method w/ LoCoLinkCo	518	717	798	159	65	151	226	22	0	2,656
Fast Greedy unweighted	274	767	1,475	426	84	36	23	12	10	3,109
Fast Greedy w/ LoCoLinkCo	574	888	1,298	380	131	139	122	54	0	3,586
Label Prop unweighted	3,142	4,962	10,635	4,340	488	40	1	1	1	23,610
Label Prop w/ LoCoLinkCo	6,570	8,612	16,769	4,231	146	6	0	0	0	36,334
Infomap unweighted	741	1,704	6,781	6,977	744	22	2	0	0	16,971
Infomap w/ LoCoLinkCo	3,288	3,972	10,159	6,922	304	2	0	0	0	24,647
Walktrap unweighted	10,036	6,978	10,183	2,864	323	31	6	1	1	30,45
Walktrap w/ LoCoLinkCo	69,024	11,519	14,125	1,787	54	4	0	1	0	96,513

Claim 4.6.15. (CS) As was seen in the benchmark graphs that were generated to be similar to G_{DBLP}, weighted and unweighted Infomap and Walktrap and weighted Label Prop detect significantly more communities than the weighted Louvain method and the weighted Fast Greedy algorithm (see Table 4.2), although they do not necessarily reflect true communities as shown in Figure 4.15.

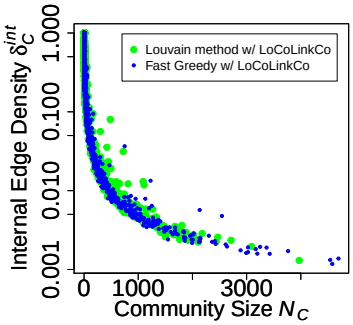
Note that the number of detected communities by the Louvain method with LoCoLinkCo and the Fast Greedy method with LoCoLinkCo are surprisingly close to the benchmark graphs, although the size distribution is different. The over count on the number of



(a) NMI comparison of detected communities from different methods for G_{DBLP}



(b) Conductance comparison



(c) Internal edge density comparison

Figure 4.29: Comparison of detected communities from different methods for G_{DBLP}. A comparison of the conductance and internal edge density of the unweighted methods can be found in [269].

communities by the other algorithms seems to be exacerbated by the fact that co-authors form cliques, which may be the reason why Label Prop on the unweighted network generates so many more communities than in the similar benchmark graphs where the correct number of communities was consistently detected. Note that the unweighted Label Prop also performed significantly different on real-world network compared to the related benchmark graphs in the previous chapter as well.

Because of this, a case can be made that the communities detected by the methods that maximize modularity pre-weighted with LoCoLinkCo may be best. While large communities in the range of 1501-5000 are created using LoCoLinkCo with the Louvain method and Fast Greedy algorithm, extremely large communities of size greater than 5,000 which are created by the unweighted modularity maximization algorithms are removed with the LoCoLinkCo weights are used.

Figure 4.29a shows that the Louvain method weighted with LoCoLinkCo provides communities that have higher NMI with the unweighted Infomap and the unweighted Label Prop which performed well in the benchmark graphs, though with such little data there is not a way to test for significance. The conductance of the communities (see Definition 2.3.3), shown in Figure 4.29b, have a higher mean value for the Louvain method weighted with LoCoLinkCo (0.1175) than the Fast Greedy algorithm weighted with LoCoLinkCo (0.1218) with a p-value of 0.02 that the true difference in means are not equal using a Welch two sample t-test. There is not a statistically significant difference in the internal edge densities (see Definition 2.3.3) shown in Figure 4.29c.

The conductance and the internal edge densities of the unweighted community detection methods on G_{DBLP} can be found in [269]. While Infomap yields higher internal edge densities than the Louvain method weighted with LoCoLinkCo, the conductance are worse providing more evidence that the Louvain method weighted with LoCoLinkCo may be best.

Claim 4.6.16. (CS) Making observations based on how algorithms perform on benchmark graphs with proposed similar characteristics to a network without ground truth may be used to make judgements as to which algorithm provides the best results on the real-world network (see Claims 3.7.9, 4.6.6, 4.6.3, 4.6.4, and 4.6.15)).

4.7 Conclusions

In this chapter, we make a case for studying LFR benchmark graphs as they are able to create degree assortative community structure which is not possible by nPSO benchmark graphs (see Claim 4.2.1). Many of the LFR benchmark graphs used in the literature have degree assortative community structure, but not all (see Claim 4.6.1). We suggest that the particular conditions that create degree assortative community structure in LFR benchmarks are the same cases in which algorithms that detect communities by maximizing modularity have the most difficulty (see Claim 4.2.2) and that by creating a new cycle-based (see Claim 4.4.8) link cohesion metric (see Definition 3.4.1) called Local Community Link Cohesion (see Definition 4.4.7) which incorporates formulas used for link prediction and link reliability (see Claim 4.4.6) can help resolve some of these issues. This was demonstrated to be true in small toy examples (see Claim 4.6.2) and in the a number of LFR benchmark graphs studied in the literature (see Claims 4.6.3 and 4.6.4). Additionally, pre-weighting a highly-connected LFR benchmark graph with LoCoLinkCo was shown to provide higher NMI values for all community detection methods (e.g. the Louvain method, the Fast Greedy algorithm, Label Prop, Infomap, and Walktrap) except HSBM (see 4.6.5). Note from the previous chapter that LoCoLinkCo did not improve highly connected nPSO benchmark graphs, so some effort should be taken to understand the expected community structure of a real-world network.

To investigate possible exploratory data analysis techniques that can be used to deter-

mine the expected community structure of a network, graph-level metrics from both nPSO graphs and LFR graphs were analyzed which were both found to be able to create assortatively mixed rich-club networks, disassortatively mixed rich club networks and assortatively and disassortatively mixed graphs that are not rich-club networks (see Claim 4.6.7). One interesting finding is that certain LFR benchmark graphs create anti rich-club networks with consistently high rich-clubness p-values which nPSO benchmark graphs are unlikely to create (see Claim 4.6.8). A number of claims are then made on the utility of analyzing LoCoLinkCo verses the geometric mean of the vertex degrees of an edge as graphs with degree assortative community structure, graphs influenced by Populairty$\times$Similarity optimization, random graphs, and hierarchical stochastic block models all seem have distinct patterns (see Claims 4.6.9,)4.6.10, 4.6.11, 4.6.12, and 4.6.13). As stated previously, the MDCore graph (see Definition 3.5.1 and Claims 3.5.2 and 3.7.10) can also be used as an exploratory data analysis technique to understand community structure.

These exploratory data analysis techniques are used to provide insights into whether the G_{DBLP} graph corresponding to the DBLP collaboration network from the SNAP datasets [148] has community structure or not, and if so which type. Note that G_{DBLP} does not have non-overlapping ground truth communities, so determining the type of community structure would otherwise be done solely based on speculation as was done in Section 4.2. The MDCore graph provided evidence that G_{DBLP} has community structure and the plots of LoCoLinkCo verses the geometric mean of the vertex degrees of an edge of G_{DBLP} were similar to the benchmark graphs with degree assortative community structure (see Claim 4.6.14). The community detections that were not modularity-based created a larger number of predicted communities (see Claim 4.6.15) as expected based on the performance of the algorithms on sparse LFR graphs (see Claims 4.6.6, 4.6.3, and 4.6.4). Therefore, finding or creating benchmark graphs similar to a real-world network has shown to be a valuable way of evaluating the performance of different community detection approaches (see Claims

3.7.9, 4.6.15, and 4.6.16).

While the community structure of G_{EU} and G_{DBLP} were easier networks to have insights into the type of community structure to expect, it is not so easy for other networks. For example, unlike the members of the research institute that make of the vertices in G_{EU} and the research scientists that make of the vertices of G_{DBLP}, a wide range of different types of users and bots can make up the vertices of online social networks with different purposes of using the network. Therefore, the underlying community structure could be degree assortative, could be hierarchical, or could be a mix or neither. To side step the issue, in the next chapter where online social networks are analyzed, both type of benchmark graphs are used as models of the online social network so that a preliminary analysis can be done on whether the vertex-level features generated using the link cohesion metric from this chapter and the previous chapter can be used to detect fake accounts.

Chapter 5: Interpretable Sybil Detection in Online Social Networks Using Semi-Local Link Cohesion Metrics

5.1 Introduction

Online social networks (OSNs) such as Facebook, LinkedIn, Twitter, and Instagram, are becoming increasingly prevalent across the world, as illustrated by Facebook's steady gain over a 12 year period to 2.8 billion active monthly users in the fourth quarter of 2020 [258]. Their popularity makes OSNs attractive environments to criminals and bad actors who can easily disguise themselves with fake accounts. In 2012, malicious actors created a fake Facebook account impersonating NATO's most senior commander, Admiral James Stavridis, and were able to deceive other high-ranking officials into accepting friendship requests from the account [149]. Personal information only shared with friends of friends on the network was then harvested, including email addresses, phone numbers, dates of birth, family pictures, and messages revealing beliefs and states of mind. Such information could be used to produce detailed profiles of potential targets, to blackmail individuals, or to create convincing deep fakes of people not typically in the public spotlight [24, 58]. In 2019, it was reported that more than 120 accounts were found impersonating three of the United States military's highest-ranking generals [204] with the problem only getting worse [142]. Scammers also create fake accounts with stolen pictures impersonating active-duty soldiers to target vulnerable women who may be tricked into sending them money [142, 204] or impersonating young women to seduce service members into sexual activities online for

the purpose of gathering evidence for blackmail [142, 190]. Additionally, malicious actors have been found to pose as recruiters or talent agents with enticing opportunities, targeting current and former government employees with access to classified material [164]. In April of 2021, the Centre for the Protection of National Infrastructure in the United Kingdom (UK) stated that at least 10,000 UK nationals have been approached by fake profiles tied to hostile states on LinkedIn over the past five years [74]. Those who connect to the fake accounts put themselves at risk of being solicited for sensitive information and put organizations they have work for more at risk by making the fake accounts look more legitimate to their employees since they have more shared contacts. With a goal of obtaining as much sensitive information as possible, this tactic has been successful on a number of the targeted individuals, some who have even gotten to the point of selling classified information to foreign contacts met in this way and are now serving prison sentences [275]. Successful fake accounts used in these scenarios are often created in small batches to make the identities more realistic, since the control of multiple identities on the network can be used convince benign users that the accounts are real. In their 2002 article, John R. Douceur defines the forging of multiple identities a *Sybil attack* on the system [88].

While it can be easy for maintainers of OSNs to block bot accounts or accounts that are attempted to be made in bulk, Sybil attacks are more sophisticated and harder to detect [88, 205]. Sybils can be effectively created at a smaller scale by making some initial connections among a small set of fake accounts, then sending requests for connections to benign users more likely to connect to anyone who makes a request [139]. Once the Sybils are created, they can be used on their intended targets or sold on web for varying prices [272]. While efforts are made to make the accounts look legitimate, researchers have shown that certain structural attributes are able to make different types of Sybils detectible [17, 107, 108, 139, 177]. Mimicking local structural attributes of benign users is easy, such as the number of connections the account has. Attributes built using semi-local informa-

tion (see Definition 2.3.2) are able to capture structural details encoding community related characteristics that would require much more time and effort to emulate with connections to real users. Creating enough fake accounts to copy semi-local patterns without depending on real users to accept requests would require a large number of fake accounts, which as stated previously can often be blocked [205]. If the maintainers of OSNs are able to more easily find Sybils based on their semi-local structural characteristics, then the asymmetric advantage that criminals and bad actors have by easily hiding in OSNs may be minimized.

In this chapter, an initial analysis on using Inversely Proportional Link Cohesion (see Definition 3.4.8) and Local Community Link Cohesion (see Definition 4.4.7) to create semi-local attributes for detecting Sybils in OSNs is conducted. Because of the ubiquity of the problem, Sybil detection is approached as an anomaly detection task on a whole network. It is assumed that one would not be able to create a reliable labelled dataset on a real OSN denoting which users are benign and which users are Sybils, therefore, an unsupervised approach is used. Specifically, an interpretable anomaly detection method recently proposed by Cetin Savkli and myself earlier this year, called Random Subspace Mixture Model (RSMM) [238], will be considered and compared to a number of baseline approaches.

The initial concept of RSMM was invented by Savkli, with certain design aspects determined jointly based off of experimental results run by myself. The initial implementation of the algorithm was written in Java by Savkli which I made callable from Python while additionally creating an implementation using purely Python code. Using Python allows for easier comparison to baseline anomaly detection algorithms. I designed and conducted the experiments and prioritized the interpretability of the model. Wojciech Czaja suggested using Inversely Proportional Link Cohesion and Local Community Link Cohesion for an anomaly detection problem that utilized the interpretability of RSMM, leading to the research in this chapter.

5.2 Motivation for Using Link Cohesion Metrics to Detect Sybils

There are a number of techniques criminals and bad actors can use to create Sybil accounts. While they are described more in depth in [139], a brief overview will be made here. Sybils can be made by fully connecting a set of fake accounts to create a tight-knit community, then making requests to highly connected people such as celebrities and politicians who often connect with people unknown to them, resulting in fake accounts with dense local communities. Alternatively, they can be made by creating some connections among a set of fake accounts, then randomly sending friend requests out without discrimination until a certain number of connections are made, creating fake accounts with sparse local communities. A third approach is to loosely connect a set of fake accounts, then randomly send out friend requests to users part of a few specific communities in hopes of making the accounts look more legitimate. Often, they are only successful in creating connections with real users that have more contacts, but nevertheless, they are able to create fake accounts with somewhat normal local communities. In each case, effort is typically made to have a number of contacts in the range of a benign user. That said, all of these approaches are likely to have semi-local structural patterns that are different than benign users.

Inversely Proportional Link Cohesion (see Definition 3.4.8) and Local Community Link Cohesion (see Definition 4.4.7) are two examples of link cohesion metrics which are used to assess how supported an edge is relative to other edges, accounting for nearby alternate paths (see Definition 3.4.3) and associated vertex degrees. By accounting for nearby alternate paths and associated vertex degrees, semi-local information is encoded into the link cohesion edge weights. These metrics correspond to different ways of measuring the importance of mutual friendship connections in a network. Connected pairs of users that have a large number of mutual friends or friends who are friend with each other will have higher link cohesion weights, particularly if those involved are not connected to many others in

the network. As a result, higher weights are a good indication of common community or mutual trust as demonstrated by their ability to improve the communities detected by traditional algorithms when used as a pre-weighting step (see Chapters 3 and 4). Vertex-level measures can be created by calculating their average strength when the edge weights from these metrics are used (see Definition 2.1.11). When used in combination with the local clustering coefficient (see Definition 2.2.6), another vertex-level metric used to measure the importance of mutual friendship connections, it is hypothesized that Sybil accounts will be easier to detect. Most benign users naturally have a majority of their connections with people who are also friends with each other or are friends of friends [139]. Sybils, on the other hand, are easiest to create by sending out random requests for friendship, which would yield connections to people who are likely not connected to each other. If effort is made to send requests to users that are part of particular communities, it may be more difficult to obtain a large number of accepted connections. It is expected that the Sybil accounts, independent of how they are created, would have different semi-local structural characteristics than benign users.

In order to test this hypothesis, Sybils that fit the patterns described above are injected into a network, then structural measures are extracted from the network and fed into unsupervised anomaly detection algorithms. The extracted structural measures, also called attributes, and the algorithms are compared based on their ability to single out the Sybils as anomalies. An OSN from SNAP datasets (the Stanford Large Network Dataset Collection) [148] is used as a reference, specifically, the Orkut social network which has 3,073,411 vertices and 117,185,083 edges, which will be referred to as G_{Orkut}. G_{Orkut} additionally has 6,288,363 overlapping communities that are not utilized in this analysis. The dataset was created by crawling a subset of the Orkut from October 3, 2006 to November 11, 2006 in which the researchers were able to obtain 11.3% of the user population at the time of the crawl [172]. One concern of partial web crawls is that high degree vertices tend to be

oversampled while low degree vertices tend to be under sampled. That said, one of the purposes of obtaining the dataset was so that researchers could understand the structure of OSNs enough to be able to develop techniques to defend against Sybil attacks [172].

Claim 5.2.1. (CS) While studying characteristics of the G_{Orkut} can provide guidance on how to create benchmark graphs with similar structure, injecting Sybils directly into G_{Orkut} and determining their detectability using traditional anomaly detection algorithms may lead to inconclusive results. The Orkut network may already include a number of anomalies, so assessing errors can be complicated. A complementary approach is to create benchmark graphs with similar characteristics to the real-world network and inject the Sybils into the benchmark graphs. While this could oversimplify the complexity of the real-world network, it may provide valuable insights that can be used to further understand the problem.

One issue is that it is hard to have an intuitive notion of what the community structure should look like for G_{Orkut}. To sidestep this issue, both LFR graphs (see Subsection 2.5.2) and nPSO graphs (see Subsection 2.5.4) are used as benchmark graphs to determine the ability to detect Sybils in networks with either community structure paradigm. For ease in analysis, benchmark graphs with 10,000 vertices are used. Since the G_{Orkut} is a social network, it is assumed to be extremely sparse (see Definition 2.2.2) [199] meaning the average degree would remain constant as the size of the network grows (or is scaled down). Based solely on the minimum degree, maximum degree, and the size of G_{Orkut}, a maximum degree for a similar graph with a smaller number of vertices is determined by assuming a power-law distribution and determining the matching power-law cut-off degree (See Definition 2.2.4). While the G_{Orkut} has overlapping communities, they do not necessarily provide insights into the sizes and numbers of non-overlapping communities. General assumptions are made about community sizes based on research from anthropologist Robin Dunbar (see Table 2.1). The parameters used for the LFR graphs and nPSO

graphs are described in Subsection 5.8.3.

5.3 Related Work

Sybil detection has been an active area of research over the past 15 years with a comprehensive review [226] and survey [8] published in the past few years. To demonstrate their ability to detect Sybils, some researchers use real-world datasets containing corresponding meta-data such as which accounts were suspended that are used as proxies for ground truth Sybil attacks [107, 108]. Others inject Sybils directly into real-world networks to measure their detectability [139, 177]. Another approach used is to merge datasets together that contain real accounts and known fake accounts [177]. For evaluation on purely synthetic networks, an approach used is to create synthetic benign network and a synthetic Sybil network, both with the Barabási-Albert (BA) model [20], and connect the two with random attack edges [107, 108, 177]. This approach assumes that a large number of Sybils can be created, specifically the synthetic Sybil network used in [107, 108] is 40% the size of the synthetic benign network and the synthetic Sybil network used in [177] is almost exactly the same size as the benign network. In this chapter, LFR graphs and nPSO graphs with known community structure are used generate synthetic benign networks instead of graphs generated from the BA model which generally have low clustering and no community structure. It is assumed that large synthetic Sybil networks are not as easy to create, as it has recently been stated that maintainers of OSNs are able to block such requests [205]. Instead, Sybils with dense local communities, sparse local communities, and normal local communities are injected into the synthetic benign networks similar to the approach used by [139].

A wide range of local, semi-local, and global structural attributes (see Definition 2.3.2) have been proposed to detect Sybils [8, 108, 139, 177, 226]. The attributes used in [139]

are most similar in nature to the semi-local attributes proposed in this chapter, and hence will be used as a baseline for comparison.

Using derived structural graph metrics and measures as attributes for detecting anomalies in an unsupervised way has been successfully done in various applications including intrusion detection and spam detection [7, 23, 85]. In [6], derived structural graph attributes from weighted graphs are analyzed in pairs so that interpretable observations can be made. Certain pairs were found to have common patterns which could be used for general anomaly detection purposes. In this chapter, structural attributes are studied in pairs, but there are not consistent patterns that are expected between all pairs of attributes. The method proposed in [238] is used as the main approach for unsupervised anomaly detection due to its ability to build arbitrary models between pairs of attributes and intrinsically determine which data points are the most anomalous compared to the rest. While other interpretable anomaly detection methods exist [11, 48, 80, 81, 158, 168, 252, 257, 293], they do not evaluate pairs of attributes in this way.

Human-in-the-loop interactions discussed in Subsection 5.8.6 is similar to semi-supervised learning [27, 67, 78]. It is possible that online social networks could be combined with outside datasets to better detect Sybils with data fusion techniques being a method to combine such information [66].

5.4 Local and Semi-Local Structural Attributes for Sybil Detection

A number of vertex-level structural metrics and measures have been proposed to be used as attributes for detecting Sybils in OSNs. The following 6 definitions introduce the attributes proposed by Naeimeh Laleh, Barbara Carminati, and Elena Ferrari in 2015 [139], which are the most similar in concept to the new attributes proposed in this chapter. Laleh et al. used the attributes to assign risk scores to the direct contacts of a user based on the

deviation of their divergency factors, whereas they will be used for network-wide anomaly detection in this chapter. The first 3 attributes are local attributes.

Definition 5.4.1. Let $G = (V, E)$ be a graph with $v \in V$ and let k_v represent its degree. The log of the degree is a vertex-level measure defined as

$$\text{logk}_v = \log k_v. \tag{5.1}$$

The attribute in general will be referred to as *logk*.

Definition 5.4.2. Let $G = (V, E)$ be a graph with $v \in V$ and let t_v represent its triangle count, the number of triangles v is a part of in G. The log of the triangle count is a vertex-level measure defined as

$$\text{logt}_v = \log t_v. \tag{5.2}$$

The attribute in general will be referred to as *logt*.

Definition 5.4.3. Let $G = (V, E)$ be a graph with $v \in V$ and let k_v represent its degree and t_v represent its triangle count. The ratio between the degree and triangle count is a vertex-level measure defined as

$$\text{ratekt}_v = \frac{k_v}{t_v}. \tag{5.3}$$

The attribute in general will be referred to as *ratekt*.

While Laleh et al. used degree and triangle count as their attributes, the log of the values are used in Definitions 5.4.1 and 5.4.2. The distributions of the raw values are highly skewed in a typical network which is mitigated by taking the log without any change to the ordering of the values. Both Definition 5.4.2 and Definition 5.4.3 capture community related information, as triangles tend to appear more often in communities [30, 222]. The next 3 attributes are semi-local attributes.

Definition 5.4.4. Let $G = (V, E)$ be a graph with $v \in V$ and let $avgk_v$ represent the average degree of all direct contacts of v. The log of the average degree is a vertex-level measure defined as

$$\text{logavgk}_v = \log avgk_v. \tag{5.4}$$

The attribute in general will be referred to as *logavgk*.

Definition 5.4.5. Let $G = (V, E)$ be a graph with $v \in V$ and let $avgt_v$ represent the average triangle count of all direct contacts of v. The log of the average triangle count is a vertex-level measure defined as

$$\text{logavgt}_v = \log avgt_v. \tag{5.5}$$

The attribute in general will be referred to as *logavgt*.

Definition 5.4.6. Let $G = (V, E)$ be a graph with $v \in V$ and let $avgk_v$ represent the average degree of all direct contacts of v and $avgt_v$ represent the average triangle count of all direct contacts of v. The ratio between the average degree and average triangle count is a vertex-level measure defined as

$$\text{avgratekt}_v = \frac{avgk_v}{avgt_v}. \tag{5.6}$$

The attribute in general will be referred to as *avgratekt*.

Similar to Definitions 5.4.1 and 5.4.2, the log of the values proposed by Laleh et al. are used in Definitions 5.4.4 and 5.4.5. Both Definition 5.4.5 and Definition 5.4.6 capture average community related information on all direct contacts of a vertex. Note that Definitions 5.4.5 and 5.4.6 contain information in a 2-hop neighborhood around a vertex v since a vertex x in a triangle of one of the direct contacts u of vertex v can be at most a distance 2 from the vertex v. Similarly, Definition 5.4.4 contains information in a 2-hop neighborhood since the degree of a direct contact u of vertex v can require knowledge of vertices 2-hops

away from the vertex v. The next two definitions make use of the link cohesion metrics defined in Chapters 3 and 4 and are semi-local attributes that contain information in a 3-hop neighborhood around a vertex.

Definition 5.4.7. Let $G = (V, E, \text{InvPrLinkCo})$ be a graph weighted using Inversely Proportional Link Cohesion and let $u, v \in V$. The average Inversely Proportional Link Cohesion weight is a vertex-level measure defined as

$$\text{iplc}_v^{avg} = \frac{1}{k_v} \sum_{u \in \mathcal{N}(v)} \text{InvPrLinkCo}(u, v). \tag{5.7}$$

The attribute in general will be referred to as *iplcavg*.

Definition 5.4.8. Let $G = (V, E, \text{LoCoLinkCo})$ be a graph weighted using Local Community Link Cohesion and let $u, v \in V$. The average Local Community Link Cohesion weight is a vertex-level measure defined as

$$\text{lclc}_v^{avg} = \frac{1}{k_v} \sum_{u \in \mathcal{N}(v)} \text{LoCoLinkCo}(u, v). \tag{5.8}$$

The attribute in general will be referred to as *lclcavg*.

Claim 5.4.9. (CS) Inversely Proportional Link Cohesion (see Definition 3.4.8) and Local Community Link Cohesion (see Definition 4.4.7) weight each edge in a network based on alternate 2-hop and 3-hop simple paths connecting their corresponding seed vertices (see Definition 3.4.2). Because they make use of semi-local information which corresponds to community structure, the link cohesion metrics are proposed to be good candidates to create vertex-level attributes that can be used to detect anomalies in a network.

An alternate 3-hop simple path connecting two seed vertices can contains a vertex at most 2-hops away from either seed vertex. For an intermediate vertex that is 2-hops away

from a seed vertex, determining its degree may require knowledge of vertices that are an additional 1-hop away from the seed vertex. Therefore, these link cohesion metrics contain information within a 3-hop neighborhood around a seed vertex. Unlike Definitions 5.4.5 and 5.4.6 that took an average of community related information on the direct contacts of a vertex, Definitions 5.4.7 and 5.4.8 contain more detailed community related information for a specified vertex, as described in Section 5.2. The following definition is for a well-known local attribute, which was originally defined in Definition 2.2.6.

Definition 5.4.10. Let $G = (V, E)$ be a graph with $v \in V$. The local clustering coefficient is a vertex-level measure defined as

$$\mathrm{cc}_v = \frac{2|\{(u,x) : u,x \in \mathcal{N}(v), (u,x) \in E\}|}{k_v(k_v - 1)}. \tag{5.9}$$

The attribute in general will be referred to as cc.

The local clustering coefficient captures community-related information within a 1-hop neighborhood, similar to Definitions 5.4.2 and 5.4.3. Specifically, the local clustering coefficient measures how similar the 1-hop neighborhood around a vertex is to a complete graph, which would create a score of 1, verses a star graph, which would create a score equal to 0.

The 9 definitions defined in this section are all vertex-level measures that can be used to detect Sybils in OSNs. Once these measures are calculated on a network, they can be extracted and placed into a matrix to be fed into traditional anomaly detection algorithms.

Definition 5.4.11. Let $G = (V, E)$ with $N = |V|$ and let $\mathcal{A}$ be a set of $D \in \mathbb{Z}^+$ vertex-level measures, also referred to as attributes. A dataset can be created containing the attributes

for each vertex in V, specifically,

$$X_{N\times D} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \dots & \dots & \dots & \dots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix}, \tag{5.10}$$

where $X_{N\times D} \subset \mathbb{R}^D$ and x_{ij} represents the jth attribute in $\mathcal{A}$ of the ith vertex. A state vector of X represents data for a specific vertex i, x_i.

There are 5 different combinations of attributes that will be analyzed in the Results section, shown in Table 5.1. AS1 and AS2 are used in [139]. Although the attributes were used in a different type of algorithm, the purpose was still to detect Sybils. Using AS2 was shown to obtain better performance results than AS1. AS3 are the three new attributes introduced in this chapter. AS4 and AS5 combines the new attributes with the previous two attribute sets.

Table 5.1: Attribute sets used for Sybil detection.

AS1	[logk; logt; ratekt; logavgk; logavgt; avgratekt]
AS2	[ratekt; avgratekt]
AS3	[iplcavg; lclcavg; cc]
AS4	[ratekt; avgratekt; iplcavg; lclcavg; cc]
AS5	[logk; logt; ratekt; logavgk; logavgt; avgratekt; iplcavg; lclcavg; cc]

There are many different techniques that can be used to detect anomalies given a set of attributes. To obtain results that are interpretable, a similar technique will be used that Leman Akoglu, Mary McGlohon, and Christos Faloutsos promoted in their 2010 conference paper [6] which was recently awarded the "Most Influential Paper Award" at the 2020 Pacific-Asia Conference on Knowledge Discovery and Data Mining. In their paper, carefully chosen attributes from weighted graphs were analyzed in pairs. Common patterns

were observed and formalized into new rules (in their case, power laws). In graphs where a majority of the vertices follow the rules they discovered, the few vertices that did not would be considered anomalies. Akoglu et al. also noticed some outliers occurred that also obeyed their rules, so they additionally used the density-based anomaly detection algorithm called LOF [42] and combined the two resulting scores by summing their normalized values. There was not an attempt to combine scores across different pairs of attributes.

In this chapter, all pairs of attributes in each carefully chosen attribute set will be scored. Unlike in [6], the pairs of attributes in our attribute sets do not have common patterns that can easily be described by a rule. Instead, Gaussian Mixture Models are used to fit 2-dimensional subspaces of the attribute sets. Certain design enhancements are made to mitigate singularity issues and automatically determine the number of components for Gaussian Mixture Models built with any input dimension, as will be discussed in the next section. The scores from each 2-dimensional subspace are combined into a single score in a way that is able to be heavily influenced by very low scores in a few subspaces while still accounting for the scores across all subspaces, as will be discussed in Section 5.6.

5.5 Noise-Enhanced Automatic Gaussian Mixture Model (NEAGMM)

In their 2021 Unifying Review of Deep and Shallow Anomaly Detection [233], Lukas Ruff, Jacob R. Kauffmann, Robert A. Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G. Dietterich, and Klaus-Robert Müller stated that "one of the most basic approaches to multivariate anomaly detection is to compute the Mahalanobis distance from a test point to the training data mean. This is equivalent to fitting a multivariate Gaussian distribution to the training data." Gaussian mixture models (GMMs) are similar in concept, but are able to model more complex distributions. A Gaussian mixture is a composition of multivariate Gaussian distributions where each component c has a weight

w_c proportional to the overall data that belongs to that component. Given that $X_{N\times D} \subset \mathbb{R}^D$ represents a dataset with N data points and dimension D, the combined probability distribution, $f^{\hat{C}}(x)$, is of the form:

$$f^{\hat{C}}(x) = \sum_{c=1}^{\hat{C}} w_c f_c(x) = \sum_{c=1}^{\hat{C}} \frac{w_c}{(2\pi)^{\frac{D}{2}} |\hat{\Sigma}_c|^{\frac{-1}{2}}} \exp\left\{ \frac{1}{2}(x-\hat{\mu}_c)^T \hat{\Sigma}_c^{-1}(x-\hat{\mu}_c) \right\}, \tag{5.11}$$

where $\hat{C}$ is the number of components, $\hat{\mu}_c$ is the D dimensional vector that represents the mean of component c, and $\hat{\Sigma}_c$ is the $D \times D$ covariance matrix of component c.

Given a set of training data, the maximum likelihood estimate (MLE) of the Gaussian mixture parameters cannot be directly computed. That said, the Expectation-Maximization (EM) algorithm can be used to estimate the model parameters from the training data [166] which is a well-established method, as illustrated by the fact that all of the main scientific computing languages have libraries with default implementations to compute GMM parameters through EM [13, 44, 54, 150, 214, 244]. While the EM algorithm is guaranteed to have likelihood estimates that monotonically increase with each iteration, a couple of drawbacks to using GMMs are that it can produce singular covariance matrices in certain situations and the number of components need to be set by the user. These issues have led to other approaches being adopted over GMMs [233]. To mitigate these issues, two modifications were introduced by Cetin Savkli and myself in a paper that will be published later this year [238]. GMMs that are built using both of these modifications will be referred to as Noise-Enhanced Automatic Gaussian Mixture Models (NEAGMMs).

The first modification handles the issue with singular covariance matrices. As the dimensionality of the data grows, sparsity of the distribution can result in pancake-like mixture components causing singularities, though singularities can also occur in 2 dimensions.

Claim 5.5.1. Adding a small amount of noise to the data after the data has been standard-

ized can prevent GMMs from erroring due to singularity issues.

The data can be standardized by transforming each attribute using a z-transformation,

$$z = (x - \tilde{\mu})/\tilde{\sigma}, \tag{5.12}$$

where $\tilde{\mu}$ and $\tilde{\sigma}$ represent the mean and standard deviation, respectively, of an attribute according to the global distribution. After this transformation, each attribute can be varied by adding a small amount of noise,

$$z \rightarrow z + \eta, \tag{5.13}$$

where η is distributed according to a normal distribution $N(0, \zeta)$. Because the noise is added in z space, ζ is a dimensionless and universal parameter of the model.

Most real-world data is noisy with measurements being only an estimate of the real underlying values. By acknowledging this, an assumption can be made that adding a small amount of noise to the data collected will not negatively impact the ability to model the true real-world patterns that one is trying model. In this chapter, it is assumed that $\zeta = 0.01$ can be used to represent the uncertainty in the collected data which was shown in [238] to be a value that can safely be used to avoid singularities without adding significant noise to the data. The concept of adding noise to enhance the performance of methods has been successful in numerous algorithms [1, 14, 53, 68, 207, 216]. Osonde Osoba and Bart Kosko proposed an approach that added noise directly into the steps of the EM algorithm to obtain better clustering in their 2013 paper [207]. Adding noise to the input data is similar in concept to the independence assumption in Chapter 3 and the modified degree computations of Definition 4.4.4 in Chapter 4 where it has been shown that allowing for a less accurate theoretical mathematical definition can yield the ability to obtain an overall better solution

in practice. An added benefit of creating noisy input data is that it allows for NEAGMM to build probabilistic models on categorical and mixed datasets, which is not typically possible with GMMs due to singularity issues. One hot encoding can be used to map categorical attributes onto binary numerical attributes which can then be used in NEAGMM.

The second modification is a simple way of automatically determining the number of components $\hat{C}$ without overfitting the training data, where it is assumed that a model containing more components is more likely to be a model that overfits the training data. For each proposed value of $\hat{C}$, an EM procedure is performed to converge to a solution $f^{\hat{C}}(x)$ using the available components provided yielding a corresponding maximum likelihood estimate $\hat{L}$. The quality of the mixture model solution is measured using the Bayesian Information Criterion (BIC, [243]),

$$\text{BIC} = \log(N)\nu - 2\log(\hat{L}), \tag{5.14}$$

where ν is the degrees of freedom of the model. Using BIC values with GMMs is common practice [13, 44, 150, 151, 214]. It is a measure used to prevent overfitting that, in addition to the degrees of freedom of the model, also takes into account the number of data points.

The number of components $\hat{C}$ is determined by searching for a minimum BIC with the lowest component number to prevent overfitting. The search is performed starting with $\hat{C} = 1$ and incrementing the number of components until the BIC score starts increasing. This is similar to the approach used in [150]. Since mixture models are dependent on initial conditions, the number of initiations to perform for each GMM is set to 3 to minimize influence of the initial state and to help ensure the minimum BIC is chosen appropriately. The measures taken to prevent overfitting with too many components can help the algorithm handle contaminated data in which the training data contains both normal and anomalous data points.

The combined steps used to fit a NEAGMM to a set of training data can be found in Algorithm 1. To obtain the anomaly score for a state vector, the state vector is standardized then the score of the trained NEAGMM model is returned where small values representing less probable and therefore more anomalous values.

Algorithm 1 The fit method for NEAGMM

Require: $X_{N\times D}$: The training data, ζ : Noise standard deviation (default 0.01), *n_init*: The number of initializations to perform (default 3)

```
Standardize X_{N×D}
Add Noise with ζ
Fit a GaussianMixture f^Ĉ(x) with the noisy standardized data and n_init for Ĉ = 1.
Set BestBIC to the BIC value from f^Ĉ(x) for Ĉ = 1
Set BestGaussianMixture to f^Ĉ(x) with Ĉ = 1
for Ĉ=2,3,... do
    Fit a GaussianMixture f^Ĉ(x) with the noisy standardized data and n_init
    Calculate BIC value from f^Ĉ(x)
    if (BIC < BestBIC):
        Set BestBIC to BIC value using f^Ĉ(x)
        Set BestGaussianMixture to f^Ĉ(x)
    else:
        return BestGaussianMixture as the neagmm(x)
```

5.6 Random Subspace Mixture Model using NEAGMMs

NEAGMMs are defined to build models on datasets with any number of attributes. Ruff et al. stated in their 2021 Unifying Review of Deep and Shallow Anomaly Detection that while algorithms like GMM "perform fairly well for low dimensional problems, they suffer notoriously from the curse of dimensionality: the sample size required to attain a fixed level of accuracy grows exponentially in the dimension of the [attribute] space." Many researchers have studied ways of using dimensionality reduction techniques to improve models [61, 65, 76, 77, 87, 224, 276]. One approach used to mitigate this issue is to use dimensionality reduction techniques in conjunction with GMMs [282, 294]. The RSMM

proposed by Cetin Savkli and myself in [238] uses an alternative approach.

Claim 5.6.1. The dimensionality of a model can be reduced by creating an ensemble of low-dimensional subspaces which naturally handles singularities due to high-dimensionality and allows for the ability to ascribe an anomaly to specific attributes within an anomalous state vector.

For a chosen subspace dimension $d < D$ and a specified number of desired subspaces $\hat{M}$, the algorithm first ensures that the number of subspaces would allow for an equal representation for each attribute which is needed to ensure correct statistical limits. Specifically, $\hat{M}$ must satisfy

$$\hat{M} \in \alpha \times \frac{LCM(d,D)}{d}, \tag{5.15}$$

where LCM represents the least common multiple and $\alpha \in \mathbb{Z}^{+}$. Additionally, $\hat{M}$ must satisfy

$$\hat{M} \leq \binom{D}{d} \tag{5.16}$$

$\binom{D}{d}$ represents 'D choose d' – the maximum number of distinct subspaces of D with dimension d. As an example, when constructing $d = 6$ dimensional subspaces for a $D = 20$ dimensional dataset, the available $\hat{M}$ values would be $\hat{M} \in \{10; 20; ...; 38,760\}$. If the specified number of desired subspaces does not equal one of these values, the next highest $\hat{M}$ is used, or the maximum $\hat{M}$ is used if the desired number is out of range.

Claim 5.6.2. Creating a balanced participation of each attribute in the subspace dimensions will lead to more interpretable results, as will be demonstrated in Subsection 5.8.5.

Once $\hat{M}$ is ensured to meet the criteria, $\hat{S}$ is defined to contain $\hat{M}$ randomly generated distinct subspaces of dimension d from D. Note that if the subspaces dimension is small, for example $d = 2$, and the dimension of the dataset is not too large, for example $D < 10$, all subspaces can be enumerated easily without the need for randomness. For each subspace

in $\hat{S}$, a NEAGMM is trained and stored for evaluation. The method to fit a RSMM to a set of training data is outlined in Algorithm 2.

Algorithm 2 The fit method for RSMM

Require: $X_{N\times D}$: The training data, d : Dimensionality of a subspace, $\hat{M}$: Number of desired subspaces, ζ : Noise standard deviation (default 0.01), *n_init*: The number of initializations to perform (default 3)

1: Set $\hat{M}$ to the next closest value that satisfies Equation 5.15
2: Set $\hat{S}$ to be a set of $\hat{M}$ randomly generate distinct subspaces of dimension d from D
3: **for** i=1,2,...,$\hat{M}$ **do**
4: Use **Algorithm 1** to build $neagmm_i(x)$ using $X_{N\times d}$ with $d = \hat{S}[i]$, ζ, and *n_init*

Once trained, scores across all of the subspaces can be combined using geometric averaging, specifically,

$$rsmm(x) \equiv \left(\prod_{i=1}^{\hat{M}} neagmm_i(x)\right)^{1/\hat{M}}, \tag{5.17}$$

which is able to be heavily influenced by very low scores in a few subspaces while still accounting for the scores across all subspaces.

This concept is illustrated in the two toy scenarios of Figure 5.1 (analogous to Figure 2 in [292]). Geometric averaging works best when scores close to 0 represent anomalies. In practice, to prevent a score of 0 from setting the combined result to 0, which would prevent the ranking of anomalies among the most anomalous subspaces, a machine epsilon is added to every subspace score. This approach was used in Equations 3.8 and 4.15 as well.

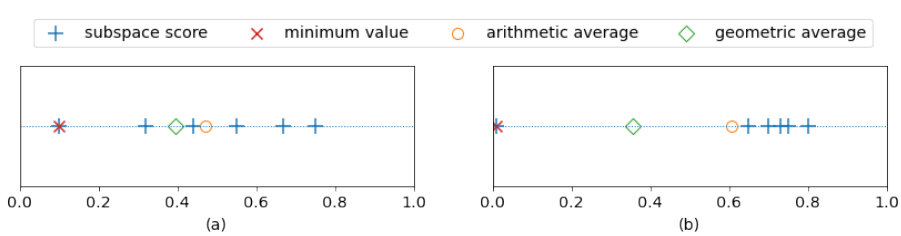


Figure 5.1: Toy scenarios demonstrating the combination of subspace scores (lower scores means more anomalous) that are (a) diverse and (b) clustered with a single near-zero value.

Claim 5.6.3. Geometric averaging is a desirable combination method when lower scores close to 0 correspond to anomalies (see Figure 5.1).

An ensemble of low-dimensional subspaces where each subspace has the same dimension d built with equal representation from the attributes provides a result that is interpretable. Given a specified anomaly rate, when the model labels a point to be anomalous, attributes can be ranked based on how often they are part of subspaces that characterize the point as anomalous compared to how often they are part of subspaces that characterize the point as normal. Additionally, the result can be explained by noting which subspaces yield the smallest percentile rankings of the subspace scores for the anomalous state vector. Both approaches will be illustrated in Subsection 5.8.5.

5.7 Implementation and Experimental Setup

One of the most appealing aspects of NEAGMM and RSMM is in the simplicity of their implementation. The Algorithm 1 can be implemented using any existing method to fit a GaussianMixture to a set of data and Algorithm 2 makes use of Algorithm 1. Both have easily been implemented in both java and python. The java implementation uses a custom Expectation Maximization algorithm initializing the clusters with k-means result from Apache commons math library and the python implementation has leveraged the GaussianMixture implementation provided by the scikit-learn toolkit [44, 214]. The python implementation also builds off of the PyOD toolkit [288].

The computational complexity of a single EM iteration with respect to the number of data points is $O(N)$. The different NEAGMM models fitted for each subspaces in Algorithm 2 is completely independent, so RSMM is easily parallelizable and therefore very fast to build.

All experiments were run on MacBook Pro laptops, either on my old laptop with a 2.9

GHz Quad-Core Intel Core i7 Processor and 16 GB of memory or my new laptop with a 2.4 GHz 8-Core Intel Core i9 Processor with 64 GB of RAM.

5.8 Results

The local and semi-local attributes from Section 5.4 are tested in 5 different combinations shown in Table 5.1 on their ability to detect Sybils in benchmark graphs similar to the Orkut network described in Section 5.2 and further outlined in Subsection 5.8.3. RSMM is used to detect Sybils, but also compared to baseline anomaly detection algorithms described in Table 5.2. All of the baseline algorithms except NEAGMM can be found in PyOD [288], a popular open-source Python toolbox for performing scalable outlier detection on multivariate data with a single, well-documented API. The NEAGMM algorithm introduced in Section 5.5 and used as the base algorithm for RSMM is also used as a baseline for comparison, but instead of using the subspace dimension d, a mixture model is built using all attributes with dimension D. All of the algorithms are run with default parameters, including RSMM whose parameters are selected in Subsection 5.8.1.

Table 5.2: Anomaly detection methods used as baselines for comparison to NEAGMM and RSMM. Baseline algorithms can be found in the PyOD toolkit [288].

Name	Type	Description	Initial Paper
CBLOF	Proximity-Based	Clustering-Based Local Outlier Factor	[120]
FB	Outlier Ensembles	Feature Bagging using LOF	[145]
HBOS	Proximity-Based	Histogram-based Outlier Score	[113]
IForest	Outlier Ensembles	Isolation Forest	[153]
KNN	Proximity-Based	k-Nearest Neighbor	[227]
LOF	Proximity-Based	Local Outlier Factor	[42]
OCSVM	Linear Model	One-Class Support Vector Machines	[242]
COPOD	Probabilistic	Copula-Based Outlier Detection	[152]
VAE	Neural Networks	Variational AutoEncoder	[132]

Before experiments are done to detect Sybils, in order to demonstrate the benefits of

RSMM standard anomaly detection benchmark datasets from the ODDS repository [229] and the UCI repository [89] are used to compare the performance of the anomaly detection methods. All multi-dimensional point datasets from the ODDS repository with at least 5,000 data points and a dimesion of at least 5 are used. Additionally, a real-world dataset from Smart Buildings introduced in [252] is also used. To test on categorical datasets, the 2 largest relevant categorical datasets from the UCI repository were also incorporated. Each dataset was constructed to have a relatively small number of anomalies, a, with the percentage of anomalies, $a\% = a/s$ range from 1% to 32% across the 12 datasets, summarized in Table 5.3. The Nusery dataset from the UCI repository is converted into an anomaly detection dataset by setting two smallest classes, *recommend* and *very_recom*, as the anomalous class. The Mushrooms dataset from the UCI repository is converted into an anomaly detection datset by keeping all the *edible* data points as the normal points and randomly selecting 792 of the 3,916 *poisonous* data points to be the anomalies[1]. Datasets with at least 5,000 data points were selected in order to test the performance of the algorithms on their ability to perform with a small percentage of training data (down to 5%), which required a larger starting size. To select the default parameters of the algorithms, 3 additional datasets from the ODDS repository were selected based on their varying range in dimensions.

The results are presented as ROC (i.e., True Positive vs. False Positive) Area Under the Curve (ROC-AUC). For the benchmark graphs with injected Sybils, the ROC-AUC values can be very similar from different attributes sets and different anomaly detection algorithms, so the average precision at the anomaly percent cut-off is also used as a metric, meaning if 1% of the data is anomalous then precision is measured on the top 1% most anomalous values. The average precision removes information as to if the remaining

[1] 10 different random samplings of the *poisonous* data points were conducted and there was not a statistically significant difference in the results

Table 5.3: Summary of anomaly detection datasets ordered by dimensionality.

Dataset	s	n	Attribute Type	a ($a\%$)
Mammography (MM)	11,183	6	Numerical	260 (2.3%)
Annthyroid (AT)	7200	6	Numerical	534 (7.4%)
Smart Buildings (SB)	60,425	8	Numerical	1,921 (3.2%)
Shuttle (SH)	49,097	9	Numerical	3,511 (7.2%)
Forest Cover (FC)	286,048	10	Numerical	2,747 (1.0%)
Pen Digits (PD)	6,870	16	Numerical	156 (2.3%)
Satellite 1 (S1)	6,435	36	Numerical	2,036 (32%)
Satellite 2 (S2)	5,803	36	Numerical	71 (1.2%)
Optdigits (OD)	5216	64	Numerical	150 (3%)
Mnist (MN)	7603	100	Numerical	700 (9.2%)
Nursery (NS)	12,960	27 (8)	Categorical	330 (2.5%)
Mushroom (MR)	5,000	117 (22)	Categorical	792 (16%)

anomalies are near the cut-off or far from the cut-off, therefore the maximum percentiles for each Sybil type is also presented for RSMM.

5.8.1 Subspace Selection Study

While random subspaces are used by the RSMM algorithm, the subspace dimension d was analyzed in conjunction with the number of subspaces $\hat{M}$. Selected datasets from the ODDS not used for benchmark comparisons, such as the Thyroid ($N = 6$), Breast Cancer ($N = 30$), and Arrhythmia ($N = 274$) datasets, were used for this study based on their varying dimensionalities. Candidate subspace dimensions were set to $d \in \{2, 3, 4, \sqrt{N}, 2\sqrt{N}, 3\sqrt{N}\}$ and candidate number of subspaces were set to $\hat{M} \in \{N, 2N, 3N, 4N, 50, 100, 500\}$. It was found that lower dimensional subspaces, even as small as $d = 2$, obtained the best results for many datasets and that selecting $\hat{M}$ as a function of N was generally better than an set integer, but not significantly. A benefit of using $d = 2$ is that the subspaces can be easily visualized which can help with interpretability. Therefore, the default parameters are set to $d = 2$ and $\hat{M} = 4N$.

Special considerations are made for categorical data in which one-hot encoding is used, for example the Nursery dataset and the Mushroom dataset. If a dataset with numerical attributes is set to $d = 2$, a dataset with one-hot encoded categorical attributes is set to $d = 2 * \overline{c}$ where $\overline{c}$ is equal to the average number of categories an attribute has. In future work, leaving categorical attributes unexpanded as an input to the RSMM algorithm and expanding within the subspaces so that handling mixed categorical and numerical attributes is even more seamless can be explored.

Claim 5.8.1. (CS) Default parameters can be chosen for RSMM which are expected to work well in a variety of anomaly detection applications where pair-wise interactions yield valuable information. Using such default parameters can lead to more interpretable results, interpretable methods can be transferred to higher dimensional subspaces which may be better for certain data types.

5.8.2 Small Percent Training Study on Other Benchmark Datasets

The next experiments compare the performance on the anomaly detection datasets in both the normal testing paradigm and with just a small percent of training data used. First, results are generated by randomly splitting the dataset described in Table 5.3 into 60% training data and 40% testing data with 10 iterations are run for each of the dataset. As shown in Table 5.4, RSMM is competitive with all of the other algorithms including the prominent Isolation Forest algorithm. The RSMM algorithm scores the highest in terms of ROC-AUC with a score of 85.23% which is 2.3% higher than IForest at 82.87%. RSMM ranks first in half of the datasets, more than any other algorithm. It is also interesting to observe that the performance of RSMM on the benchmark datasets did not degrade with an increase in the dimensionality of the datasets. Even when a small subset of the possible subspaces are selected, the RSMM algorithm scored well on the higher dimensional

datasets. RSMM also performs best on the categorical datasets.

Table 5.4: Mean and standard deviation of ROC-AUC scores as % for benchmark datasets (mean and standard deviation is calculated using 10 independent trials using 60% training data, highest mean highlighted in bold); Rank of mean is shown in parenthesis (lower is better). RSMM outperforms all baselines.

Data	CBLOF	FB	HBOS	IForest	KNN	LOF	OCSVM	COPOD	VAE	NEAGMM	RSMM
MM	82±2 (9)	75±2 (10)	84±2 (6)	86±2 (4)	84±2 (6)	73±3 (11)	87±2 (3)	**91±2 (1)**	89±1 (2)	84±2 (6)	86±2 (4)
AT	66±2 (10)	79±4 (4)	61±3 (11)	82±2 (3)	79±1 (4)	72±2 (7)	68±2 (8)	77±1 (6)	67±2 (9)	88±1 (2)	**90±1 (1)**
SB	80±1 (7)	48±4 (11)	93±1 (3)	81±4 (6)	70±1 (9)	55±2 (10)	85±0 (5)	**94±0 (1)**	90±0 (4)	80±9 (7)	**94±1 (1)**
SH	61±2 (9)	48±3 (11)	99±0 (2)	**100±0 (1)**	65±0 (8)	53±1 (10)	99±0 (2)	99±0 (2)	99±0 (2)	84±9 (7)	98±0 (6)
FC	94±0 (2)	57±2 (10)	71±0 (9)	88±2 (5)	78±1 (8)	54±1 (11)	**95±0 (1)**	88±0 (5)	93±0 (3)	90±1 (4)	87±1 (7)
PD	88±4 (7)	47±4 (10)	92±1 (5)	**95±1 (1)**	75±2 (9)	47±4 (10)	93±1 (3)	91±1 (6)	94±1 (2)	76±6 (8)	93±1 (3)
S1	75±1 (3)	56±2 (10)	76±1 (2)	71±1 (5)	68±1 (6)	56±2 (10)	66±1 (8)	63±1 (9)	68±5 (6)	73±2 (4)	**80±1 (1)**
S2	**100±0 (1)**	46±10 (10)	98±1 (6)	**100±0 (1)**	95±1 (8)	46±10 (10)	**100±0 (1)**	98±1 (6)	99±1 (5)	94±1 (9)	**100±0 (1)**
OD	77±5 (3)	45±6 (9)	**87±1 (1)**	71±5 (4)	37±2 (11)	45±6 (9)	50±2 (8)	68±1 (5)	51±1 (7)	55±5 (6)	86±1 (2)
MN	**85±1 (1)**	72±2 (8)	57±1 (11)	80±3 (6)	**85±1 (1)**	72±2 (8)	**85±1 (1)**	77±1 (7)	**85±1 (1)**	81±6 (5)	58±2 (10)
NS	50±8 (6)	55±5 (2)	49±20 (8)	51±8 (3)	49±8 (8)	51±4 (3)	49±3 (8)	46±8 (11)	50±19 (6)	51±6 (3)	**57±9 (1)**
MR	84±2 (8)	54±4 (10)	87±1 (5)	91±1 (3)	92±1 (2)	52±3 (11)	86±1 (7)	88±1 (4)	87±1 (5)	74±18 (9)	**94±1 (1)**
AVG	79±2 (7)	57±4 (10)	80±3 (6)	83±3 (2)	73±2 (9)	56±3 (11)	80±1 (5)	82±1 (3)	81±3 (4)	78±6 (8)	**85±2 (1)**

The next results shown in Table 5.5 are generated by randomly splitting the dataset described in Table 5.3 into 5% training data and 95% testing data with 100 iterations are run for each of the dataset. The performance of the algorithms did not drastically degrade and the ranking of most algorithms stays the same on average, only changing by one at most.

Table 5.5: Mean and standard deviation of ROC-AUC scores as % for benchmark datasets (mean and standard deviation is calculated using 100 independent trials using 5% training data, highest mean highlighted in bold); Rank of mean is shown in parenthesis (lower is better). RSMM outperforms all baselines.

Data	CBLOF	FB	HBOS	IForest	KNN	LOF	OCSVM	COPOD	VAE	NEAGMM	RSMM
MM	82±2 (9)	81±2 (11)	83±1 (8)	86±1 (4)	84±1 (6)	81±2 (10)	87±1 (3)	**90±0 (1)**	88±2 (2)	83±2 (7)	86±1 (5)
AT	70±5 (8)	78±6 (4)	69±5 (10)	82±2 (3)	76±5 (6)	74±5 (7)	70±4 (9)	78±0 (5)	69±3 (11)	89±2 (2)	**89±2 (1)**
SB	80±3 (7)	59±5 (11)	93±1 (2)	82±4 (6)	62±3 (10)	64±5 (9)	85±1 (5)	**94±0 (1)**	90±1 (4)	79±7 (8)	92±2 (3)
SH	65±5 (9)	52±7 (11)	99±0 (5)	**100±0 (1)**	76±2 (8)	52±5 (10)	99±0 (3)	99±0 (2)	99±0 (4)	90±10 (7)	97±2 (6)
FC	94±1 (2)	72±4 (10)	72±3 (11)	88±3 (7)	83±2 (8)	74±3 (9)	**95±0 (1)**	88±0 (6)	93±0 (3)	90±1 (4)	88±1 (5)
PD	92±6 (6)	90±5 (10)	92±1 (5)	**95±1 (1)**	93±5 (4)	91±5 (8)	93±2 (3)	90±0 (9)	94±1 (2)	81±7 (11)	91±2 (7)
S1	74±3 (4)	61±4 (10)	78±3 (1)	71±2 (5)	75±2 (3)	61±3 (11)	67±1 (6)	63±0 (9)	66±4 (8)	67±4 (7)	76±3 (2)
S2	**100±0 (1)**	100±0 (4)	98±0 (10)	99±0 (7)	100±0 (2)	100±0 (6)	100±0 (3)	97±0 (11)	98±0 (9)	98±1 (8)	100±0 (5)
OD	67±14 (7)	71±6 (4)	**86±2 (1)**	71±6 (5)	53±17 (8)	73±6 (3)	51±5 (11)	68±0 (6)	52±4 (9)	51±8 (10)	84±2 (2)
MN	84±3 (7)	86±2 (3)	57±2 (11)	79±3 (8)	**87±2 (1)**	87±7 (2)	85±2 (5)	77±0 (9)	85±2 (4)	85±2 (6)	65±2 (10)
NS	50±9 (6)	50±13 (6)	49±15 (9)	**51±11 (1)**	50±7 (4)	50±10 (5)	51±9 (2)	47±7 (10)	50±14 (8)	51±10 (3)	45±11 (11)
MR	83±4 (9)	82±7 (10)	86±1 (5)	**88±3 (1)**	86±3 (6)	80±9 (11)	84±3 (8)	88±0 (3)	85±2 (7)	87±3 (4)	88±2 (2)
AVG	78±5 (8)	74±5 (11)	80±3 (6)	83±3 (2)	77±4 (9)	74±5 (10)	80±2 (5)	82±1 (3)	81±3 (4)	79±5 (7)	**83±2 (1)**

While the remaining experiments will be done using the standard 60% training and 40%

testing, it is beneficial to know that the performance does not necessarily degrade when only a small percent training is used. This makes it possible for a human to be in the loop, checking the most anomaly data points in a set of training data using the interpretable plots shown in Subsection 5.8.5 and validating with potentially outside information. As will be seen in Subsection 5.8.6, it is possible to obtain improvements if anomalies of interest are removed from a training set. While this may be daunting if a large amount of training data is needed, it maybe possible with only a small amount of training data required.

Claim 5.8.2. (CS) RSMM obtains the highest average ROC-AUC value and highest average ranking compared to 10 baseline algorithms on all of the ODDS multi-dimensional point datasets with at least 5,000 data points and at least 5 dimensions, in addition to another recently introduced multi-dimensional point dataset [252], and a couple of the largest categorial datasets from the UCI repository both when 60% training data is used (see Table 5.4) and when 5% training data is used (see Table 5.5).

5.8.3 Benchmark Graphs with Injected Anomalies

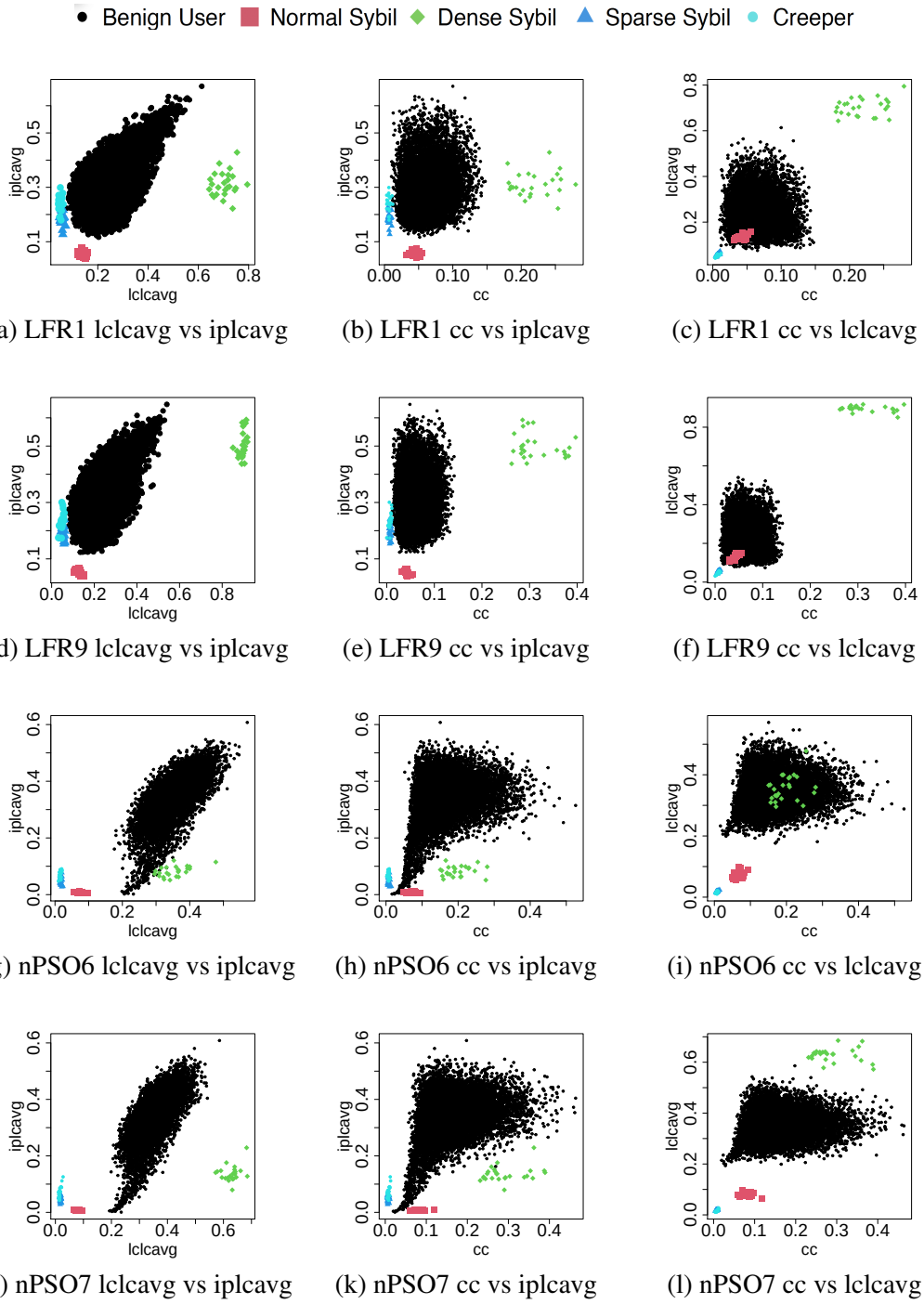
To generate graphs similar to G_{Orkut}, the LFR benchmark parameters used were $N = 10,000$, the average degree was set to $\langle k \rangle = 70$, the maximum degree was set to $k_{max} = 630$, the degree distribution scaling parameter was set to $\gamma = 3$, the community size distribution scaling parameter was set to $\beta = 1.5$, the minimum and maximum number of communities were set to $c_{min} = 38$ and $c_{max} = 630$ respectively, and the mixing parameter was set to $\mu = .6$ to create similar clustering. Similarly, the nPSO benchmark parameters used were $N = 10,000$, the degree distribution scaling parameter was set to $\gamma = 3$, the parameter equal to half the average degree of a vertex is set to $m = 35$, and 66 equidistant means around the circle are set with standard deviations equal to $\frac{1}{6} * \frac{2\pi}{66}$ (where * represents scalar multiplication). The mixing proportions are used to create unequal sized communities, specifically,

the proportions are set using a power law with exponent $\beta = 1.5$. The temperature was set to $T = 0.7$ to create similar clustering. 10 random graphs of each type were generated. For simplicity, they are referred to as LFRi and nPSOi where $i \in 1, .., 10$.

In each benchmark graph, four additional types of anomalous users are added: Sybils with normal local communities, Sybils with dense local communities, Sybils with sparse local communities, and Creepers, all originally introduced by Naeimeh Laleh, Barbara Carminati, and Elena Ferrari in 2015 [139]. Laleh et al. injected Sybils directly into G_{Orkut}, therefore the approach used to generate the anomalous users is different in a couple of ways. While 100 users of each type were injected into G_{Orkut} which has 3,073,411 vertices making the anomalies just 0.01% of the data, 25 users of each type were injected into the benchmark graphs making the anomalies 1% of the data. Additionally, the Sybils with normal local communities in this chapter leverage the ground truth communities that are available from the benchmark graphs.

Sybils with normal local communities, abbreviated as *normal Sybils*, are generated initially using an ER graph $N = 25$ and $p = 0.25$. Then a minimum of 3 ground truth communities and a maximum of 5 ground truth communities are selected and candidate benign users in those communities above the median degree are randomly selected proportional to their degree until each normal Sybil has a number of edges equal to a uniform random number between the average degree in the benchmark graph minus one tenth of a standard deviation to the average degree plus a standard deviation. The normal Sybils are shown as pink squares in the example benchmark graphs in Figure 5.2.

Sybils with dense local communities, abbreviated as *dense Sybils*, are generated by creating a complete graph with 25 vertices, then randomly making connections to the vertices with the 25% highest degrees proportional to the degree of the vertices until each dense Sybil has a number of edges equal to a uniform random number between the average degree in the benchmark graph to the average degree plus a standard deviation. The dense



(a) LFR1 lclcavg vs iplcavg (b) LFR1 cc vs iplcavg (c) LFR1 cc vs lclcavg

(d) LFR9 lclcavg vs iplcavg (e) LFR9 cc vs iplcavg (f) LFR9 cc vs lclcavg

(g) nPSO6 lclcavg vs iplcavg (h) nPSO6 cc vs iplcavg (i) nPSO6 cc vs lclcavg

(j) nPSO7 lclcavg vs iplcavg (k) nPSO7 cc vs iplcavg (l) nPSO7 cc vs lclcavg

Figure 5.2: Example attribute subspaces using link cohesion metrics. The top two rows show two of the LFR graphs and the bottom two rows show two of the nPSO graphs. Each subspace combination of lclcavg, iplcavg, and cc are shown.

Sybils are shown as green diamonds in the example benchmark graphs in Figure 5.2.

Sybils with sparse local communities, abbreviated as *sparse Sybils*, are generated by creating 25 isolated vertices, then randomly making connections to edges in the benchmark graph until each sparse Sybil has a number of edges equal to a uniform random number between the average degree in the benchmark graph to the average degree plus a standard deviation. The sparse Sybils are shown as blue triangles in the example benchmark graphs in Figure 5.2.

A last type of anomaly, called a *Creeper*, is added to represent real users with fake accounts that are not as threatening as Sybils. Similar to sparse Sybils, they are generated by creating 25 isolated vertices, then randomly making connections to edges in the benchmark graph until each sparse Sybil has a number of edges equal to a uniform random number between the average in the benchmark graph minus a standard deviation (or the minimum degree in the benchmark graph if it is larger) to the average degree. The Creepers are shown as light blue circles in the example benchmark graphs in Figure 5.2.

Two examples of the LFR graphs are shown in (a)-(c) and (d)-(f) in Figure 5.2 and two examples of the nPSO graphs are shown in (g)-(i) and (j)-(l). The first column shows a visualization of the subspace using lclcavg and iplcavg, the second column shows a visualization of the subspace using cc and iplc, and the third column shows a visualization of the subspace using cc and lclcavg. While the patterns are similar when they are generated using the same benchmark, the two different benchmarks look quite different. The two different example shown for each benchmark were chosen based on their extreme (highest and lowest) ROC-AUC scores using attribute set 5 (AS5 from Table 5.1). The injected anomalies show up in small clusters instead of randomly being spread across the subspace space. As observed by Meghanath Macha and Leman Akoglu in 2018 [160], almost all existing work in the anomaly detection literature assume anomalies to be scattered, but often times there are anomalous patterns that constitute interpretable “signatures”.

Claim 5.8.3. (CS) The injected anomalies into the benchmark graphs show up in small clusters (see Figure 5.2) similar to what has recently been observed of a number of anomalies in datasets [160].

The patterns in the anomalies shown in Figure 5.2 fit this newer way of looking at anomalies, and visualizing the attributes in low dimensional subspaces makes it more clear that it is the case. Therefore, it is expected that RSMM, which conducts anomaly detection on the low dimensional subspaces, will perform well.

5.8.4 Anomaly Detection on Benchmark Graphs with Injected Anomalies

To test the ability to detect the injected anomalies in the benchmark graphs, results are generated by randomly splitting the dataset described in Table 5.3 into 60% training data and 40% testing data with 10 iterations are run for each of the dataset. The average mean and standard deviation of the average ROC-AUC scores, average precision values, and maximum percentiles for each anomaly type from the 10 iterations are aggregated for each attribute set and benchmark type combination. Additionally, mean and standard deviation of the average ROC-AUC scores, average precision values, and maximum percentiles for each anomaly type from the 10 iterates are aggregated for each attribute set combining both benchmark types. As shown in Table 5.4, RSMM is competitive with all of the other algorithms in terms of the ROC-AUC scores, but all are so close it is hard to compare with many algorithms obtaining 100% (rounded) ROC-AUC. As shown in Figure 5.7, the Mean and standard deviation of the average precision is able to better differentiate the performance of the different algorithms on the 5 different attribute sets on each benchmark and the combined benchmark. It is clear that different algorithms perform better and worse in the two different benchmark types, so looking at the average and standard deviation of both combined is of interest. RSMM obtains the highest mean average precision score

when attribute set 5 (AS5) is used from Table 5.1.

Table 5.6: Mean and standard deviation of the average ROC-AUC scores as % for the benchmark graphs with injected Sybils (average ROC-AUC is calculated using 10 independent trials using 60% training data, highest mean of the average ROC-AUC highlighted in bold); Rank of mean across all attribute sets and anomaly detection algorithms (lower is better).

Data	CBLOF	FB	HBOS	IForest	KNN	LOF	OCSVM	COPOD	VAE	NEAGMM	RSMM
AS1_LFR	99±0 (12)	98±0 (24)	98±1 (23)	98±1 (19)	96±2 (32)	97±1 (28)	98±1 (22)	94±1 (37)	97±1 (30)	98±1 (25)	99±0 (15)
AS2_LFR	82±5 (50)	85±4 (46)	86±5 (43)	85±4 (45)	81±5 (52)	85±4 (47)	85±4 (44)	81±4 (51)	89±2 (41)	82±5 (48)	82±5 (49)
AS3_LFR	100±0 (8)	79±3 (53)	96±2 (33)	99±0 (11)	69±4 (54)	69±2 (55)	99±0 (9)	94±0 (36)	98±0 (21)	99±0 (14)	100±0 (3)
AS4_LFR	100±0 (6)	98±0 (20)	98±1 (26)	99±0 (17)	86±3 (42)	97±1 (29)	99±0 (13)	95±1 (35)	99±0 (18)	91±5 (39)	99±0 (10)
AS5_LFR	100±0 (2)	96±1 (31)	100±0 (4)	100±0 (7)	90±2 (40)	95±1 (34)	100±0 (5)	97±0 (27)	99±0 (16)	92±5 (38)	**100±0 (1)**
AS1_nPSO	91±3 (29)	93±2 (25)	90±2 (33)	92±2 (27)	98±1 (20)	88±4 (35)	90±2 (32)	88±2 (34)	87±2 (37)	99±0 (13)	96±2 (24)
AS2_nPSO	80±2 (39)	76±3 (47)	65±4 (54)	80±3 (41)	78±4 (43)	76±4 (48)	72±4 (50)	76±3 (46)	71±3 (52)	78±4 (45)	78±4 (44)
AS3_nPSO	**100±0 (1)**	72±2 (51)	98±1 (18)	100±0 (7)	54±3 (55)	69±2 (53)	100±0 (2)	98±1 (9)	99±0 (15)	92±3 (28)	99±0 (8)
AS4_nPSO	100±0 (3)	84±3 (38)	97±2 (22)	99±0 (11)	88±3 (36)	79±4 (42)	100±0 (4)	98±1 (17)	99±1 (14)	91±6 (30)	100±0 (5)
AS5_nPSO	100±0 (6)	80±3 (40)	97±1 (21)	98±1 (16)	93±3 (26)	75±3 (49)	99±0 (12)	97±1 (23)	98±1 (19)	91±5 (31)	99±0 (10)
AS1_Both	95±4 (28)	96±3 (26)	94±5 (30)	95±3 (27)	97±2 (22)	93±5 (31)	94±4 (29)	91±4 (36)	92±5 (32)	99±1 (16)	98±2 (18)
AS2_Both	81±2 (43)	80±5 (44)	76±11 (52)	82±3 (42)	80±3 (49)	80±5 (45)	79±7 (50)	78±4 (51)	80±10 (48)	80±4 (47)	80±4 (46)
AS3_Both	**100±0 (1)**	76±5 (53)	97±2 (21)	99±0 (9)	62±9 (55)	69±2 (54)	100±0 (4)	96±2 (23)	99±1 (13)	96±4 (25)	100±0 (6)
AS4_Both	100±0 (3)	91±8 (35)	97±1 (19)	99±1 (11)	87±3 (40)	88±10 (39)	99±0 (10)	96±2 (24)	99±1 (14)	91±6 (37)	100±0 (7)
AS5_Both	100±0 (2)	88±9 (38)	99±1 (15)	99±1 (12)	92±3 (33)	85±11 (41)	100±0 (8)	97±1 (20)	99±1 (17)	91±5 (34)	100±0 (5)

Table 5.7: Mean and standard deviation of the average precision scores as % for the benchmark graphs with injected Sybils (average precision is calculated using 10 independent trials using 60% training data, highest mean of the average precision highlighted in bold); Rank of mean across all attribute sets and anomaly detection algorithms is shown in parenthesis (lower is better).

Data	CBLOF	FB	HBOS	IForest	KNN	LOF	OCSVM	COPOD	VAE	NEAGMM	RSMM
AS1_LFR	62±8 (17)	45±8 (43)	53±4 (30)	55±9 (26)	53±2 (29)	40±8 (48)	55±11 (23)	45±2 (44)	46±6 (40)	46±3 (41)	68±8 (12)
AS2_LFR	48±8 (39)	42±6 (47)	52±4 (31)	55±8 (25)	44±3 (45)	40±5 (49)	48±4 (38)	45±2 (42)	49±6 (37)	51±8 (35)	51±8 (34)
AS3_LFR	63±14 (16)	49±2 (36)	27±10 (52)	55±4 (24)	26±1 (53)	44±3 (46)	54±11 (27)	25±1 (54)	28±5 (51)	56±6 (21)	74±5 (4)
AS4_LFR	69±2 (9)	61±4 (18)	57±8 (20)	71±3 (7)	52±3 (32)	54±3 (28)	71±2 (8)	65±4 (15)	69±4 (10)	25±6 (55)	65±4 (14)
AS5_LFR	76±3 (2)	57±4 (19)	74±6 (3)	72±5 (6)	51±3 (33)	55±5 (22)	74±2 (5)	68±5 (11)	67±5 (13)	30±4 (50)	**88±2 (1)**
AS1_nPSO	54±2 (29)	44±5 (49)	51±1 (34)	47±1 (45)	62±3 (16)	47±7 (46)	48±0 (42)	44±2 (48)	46±2 (47)	61±3 (17)	53±2 (31)
AS2_nPSO	49±0 (39)	24±3 (53)	57±9 (19)	48±0 (41)	49±0 (36)	24±4 (54)	49±0 (40)	44±1 (50)	47±2 (43)	49±0 (36)	49±0 (36)
AS3_nPSO	**93±6 (1)**	51±3 (32)	68±6 (14)	79±7 (6)	23±2 (55)	51±3 (33)	90±8 (3)	74±1 (13)	78±8 (8)	36±4 (52)	55±4 (27)
AS4_nPSO	90±8 (2)	57±3 (20)	56±5 (24)	78±6 (9)	56±5 (23)	54±2 (30)	84±10 (4)	66±2 (15)	78±7 (11)	47±11 (44)	79±5 (7)
AS5_nPSO	82±8 (5)	54±3 (28)	60±2 (18)	57±2 (22)	55±2 (25)	55±3 (26)	78±7 (10)	49±1 (35)	57±2 (21)	37±9 (51)	76±5 (12)
AS1_Both	58±7 (21)	45±7 (48)	52±3 (31)	51±8 (34)	58±5 (20)	44±8 (50)	52±8 (32)	44±2 (49)	46±4 (45)	53±8 (29)	60±10 (17)
AS2_Both	48±6 (40)	33±10 (53)	55±8 (25)	52±7 (33)	46±2 (44)	32±9 (54)	48±3 (39)	45±2 (47)	48±4 (41)	50±6 (38)	50±6 (37)
AS3_Both	78±18 (4)	50±3 (35)	48±22 (42)	67±14 (11)	25±2 (55)	47±5 (43)	72±20 (10)	50±25 (36)	53±26 (30)	46±11 (46)	64±11 (15)
AS4_Both	80±12 (2)	59±4 (19)	56±6 (22)	74±6 (7)	54±5 (26)	54±3 (27)	77±10 (5)	65±3 (13)	73±7 (8)	36±14 (51)	72±8 (9)
AS5_Both	79±6 (3)	56±4 (23)	67±9 (12)	64±9 (14)	53±4 (28)	55±4 (24)	76±5 (6)	59±10 (18)	62±7 (16)	34±7 (52)	**82±7 (1)**

As stated previously, the average precision scores have the disadvantage of removing information – how close are the remaining anomalies to the cut-off threshold? In Table 5.8, the average and standard deviation of the maximum percentile are shown. Here, it can

more clearly be seen that attribute set 2 (AS2) from Table 5.1 has a difficult time differentiating normal Sybils and sometimes dense Sybils, with average maximum percentiles much larger than the 1 percentile that is expected. The most difficult type of anomaly to detect using attribute set 5 (AS5) is the dense Sybil due to the difficulty in detecting it in nPSO graphs. This leads to a average maximum percentile across both benchmarks of 3 where it is expected to be 1, as is the case with the other types of anomalies using AS5.

Table 5.8: Average maximum percentile of different anomalies in the benchmark graphs with injected Sybils. The ranks in parentheses are for the row and column among similar benchmark type, respectively.

Data	Normal Sybil	Dense Sybil	Sparse Sybil	Creeper
AS1_LFR	8±2 (4,4)	2±2 (3,4)	1±0 (1,2)	1±0 (2,3)
AS2_LFR	89±26 (4,5)	4±4 (3,5)	2±1 (2,5)	1±0 (1,1)
AS3_LFR	1±0 (2,1)	1±0 (1,1)	2±0 (4,4)	2±0 (3,5)
AS4_LFR	4±1 (4,3)	1±1 (2,3)	1±0 (3,3)	1±0 (1,4)
AS5_LFR	2±0 (4,2)	1±0 (1,2)	1±0 (3,1)	1±0 (2,2)
AS1_nPSO	3±1 (3,4)	25±13 (4,4)	1±0 (1,1)	1±0 (2,2)
AS2_nPSO	95±3 (4,5)	38±16 (3,5)	1±0 (2,3)	1±0 (1,3)
AS3_nPSO	2±0 (2,3)	1±0 (1,1)	2±0 (4,5)	2±0 (3,5)
AS4_nPSO	1±0 (3,2)	2±1 (4,2)	1±0 (2,4)	1±0 (1,4)
AS5_nPSO	1±0 (3,1)	5±3 (4,3)	1±0 (2,2)	0±0 (1,1)
AS1_Both	5±3 (3,4)	13±15 (4,4)	1±0 (1,2)	1±0 (2,3)
AS2_Both	92±5 (4,5)	21±21 (3,5)	2±1 (2,4)	1±0 (1,1)
AS3_Both	2±1 (2,2)	1±0 (1,1)	2±0 (4,5)	2±0 (3,5)
AS4_Both	2±1 (4,3)	2±1 (3,2)	1±0 (2,3)	1±0 (1,4)
AS5_Both	1±0 (3,1)	3±3 (4,3)	1±0 (2,1)	1±0 (1,2)

Claim 5.8.4. (CS) Overall, the newly proposed attributes that include those derived from the link cohesion metrics provide benefit over the baseline attributes both when used independently and jointly with the baseline attributes.

Claim 5.8.5. (CS) Using all 9 attributes with the RSMM algorithm provided near 100% average ROC-AUC (see Table 5.6) and the highest average precision 5.7 when both LFR graph and nPSO graphs are considered as the base benchmark On the best performing

results, 3% of the anomalies would be observed to find all anomalies which make up 1% of the data.

As shown in Figure 5.8, Creepers are one of the easier anomalies to detect due to both their low degree and their complete randomness. Unfortunately, Creepers are the least important type of anomaly. As will be seen in the next section, it may be possible to visualize where these types of anomalies fall across the different subspaces. If there is a human in the loop that is looking to remove anomalies from a training set to make them more detectable by the algorithm, these anomalies can be left in to encourage the algorithm to not alert when they come up.

5.8.5 Interpretability of Anomaly Detection Results

To start, a benefit of RSMM is that each of the subspaces and corresponding densities can be visualized. FIgure 5.3 is a visualization of all of the subspaces for AS5 for LFR1 from (a)-(c) in Figure 5.2 and Figure 5.4 is a visualization of all of the subspaces for AS5 for nPSO6 from (g)-(i) in Figure 5.2. Since this is an academic study for learning about the attributes and the RSMM algorithm, the data points are plotted in different colors and shapes based on if they are true normal data points (TN) or true anomalous data points (TA) and whether they are from the training data or the test data. In both cases, the subspaces from 5.2 are outlined in a purple box. The value on top of each subspace corresponds to the number of gaussian components, $\hat{C}$ were used to build the NEAGMM. The red lines outline the decision boundary based on the anomaly rate on whether a data point should be considered anomalous, with data points falling inside being labelled as so. In each figure, the subspaces are ordered by their BIC.

While obtaining a high average ROC-AUC and high average precision is valuable for an anomaly detection algorithm, understanding *why* a data point is anomalous is extremely

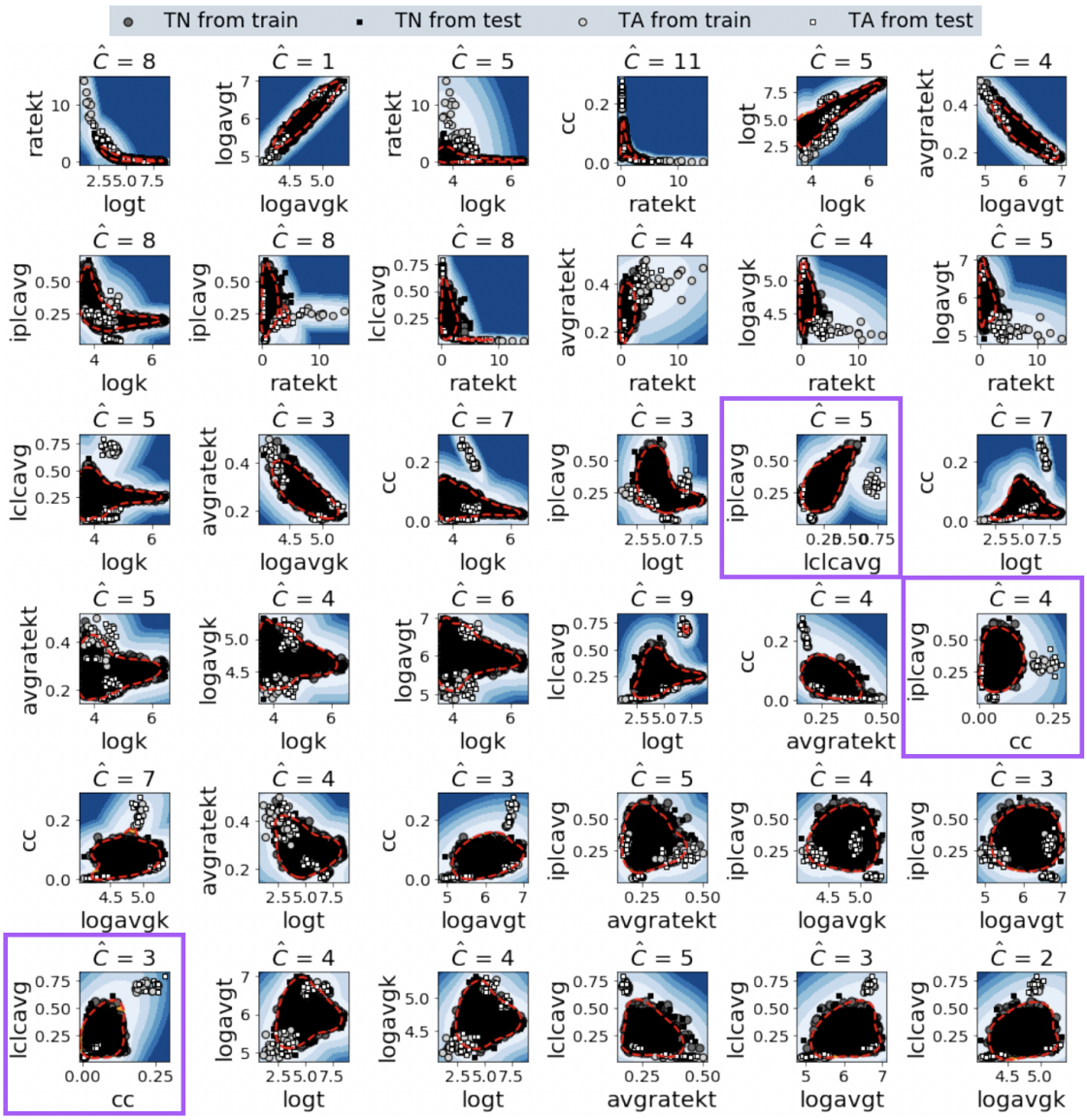


Figure 5.3: Subspaces of an example LFR benchmark graph with injected Sybils.

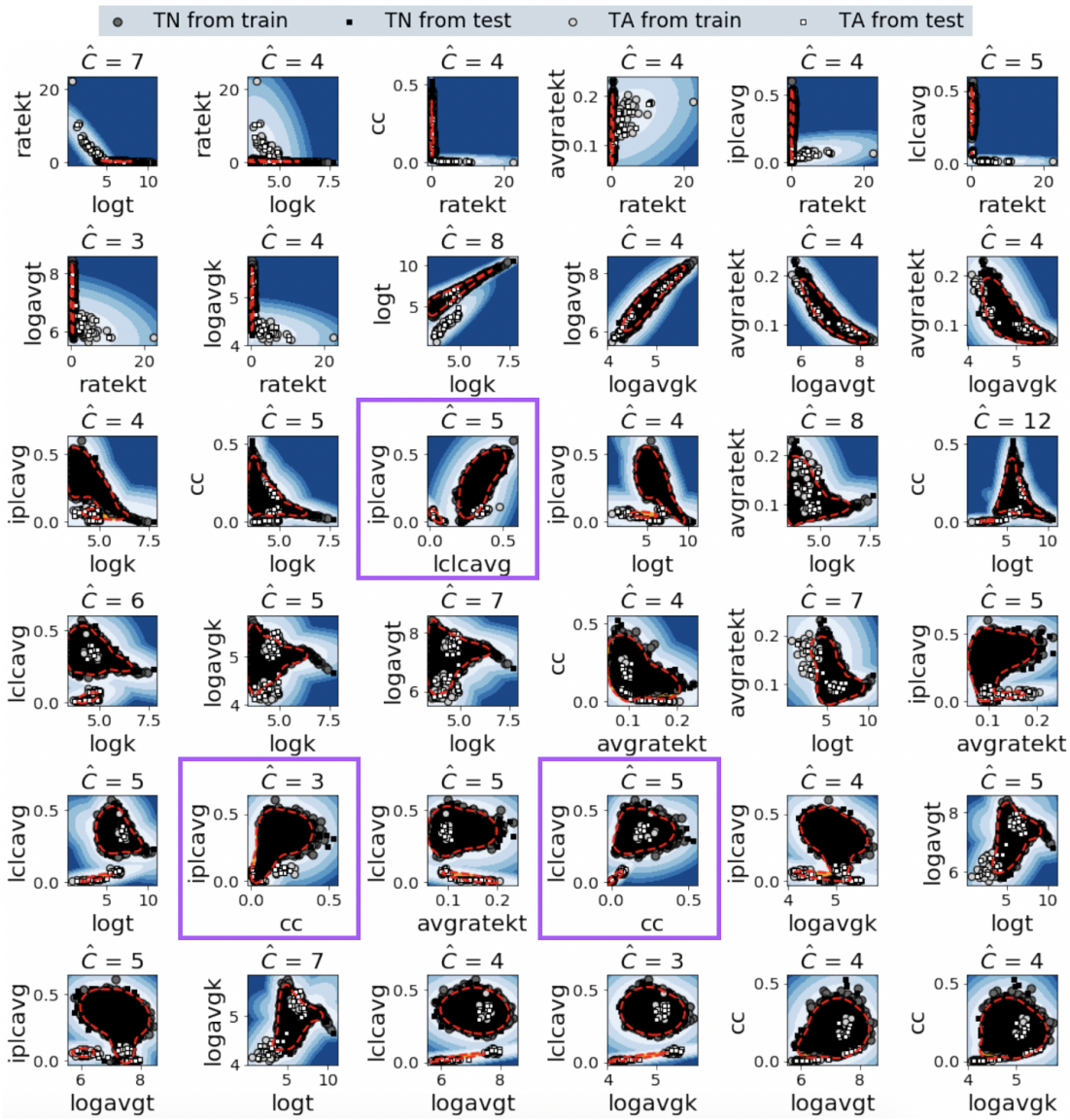


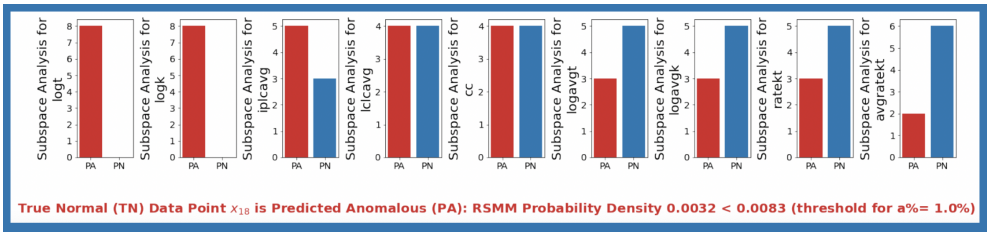
Figure 5.4: Subspaces of an example LFR benchmark graph with injected Sybils.

important in determining how to handle it.

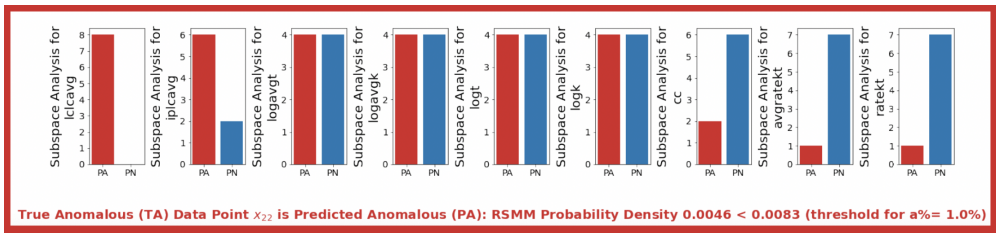
Claim 5.8.6. (CS) A benefit of RSMM is that analyses can be conducted across the subspaces to understand how much each attribute contributes to a given data point being anomalous. While each subspace has different ranges in scores, a decision threshold can be determined for each subspace based on the anomaly rate of the dataset a% (or assumed anomaly rate). For a given data point, the attributes can be ranked based on the number of subspaces it is a part of that predicts the point to be anomalous (PA) compared to the number of subspaces it is a part of that predicts the point to be normal (PN). The comparison of PA to PN across attributes is fair since all subspaces are generated randomly and have the same dimension k.

Note that for all attribute sets from Table 5.1, all possible combinations of subspaces are used for RSMM. Several examples of bar plots that contain aggregate information for each attribute can be found in Figure 5.5. Note that the $x_{\hat{a}}$ notation is such that $\hat{a}$ represents its anomaly ranking, where x_1 is the most anomalous point, x_2 is the second most anomalous point, etc, just for better understanding by the reader. Because for these benchmark graphs there is ground truth on what is an anomaly, this information is used in Figure 5.5 to better understand the RSMM algorithm. The color of the border corresponds to whether the corresponding data point is a true normal point (with a blue border) which would mean it was part of the LFR graph or the nPSO graph or if it is a true anomalous point (with a red border) which would mean it was one of the injected anomalies. The text within the figure is written in red if the data point is predicted anomalous and written in blue if the data point is predicted normal. The subfigures are increasing in how anomalous the point is predicted to be, specifically (a) is of x_{18}, (b) is of x_{22}, (c) is of x_{36}, (d) is of x_{38}, and (e) is of x_{41}.

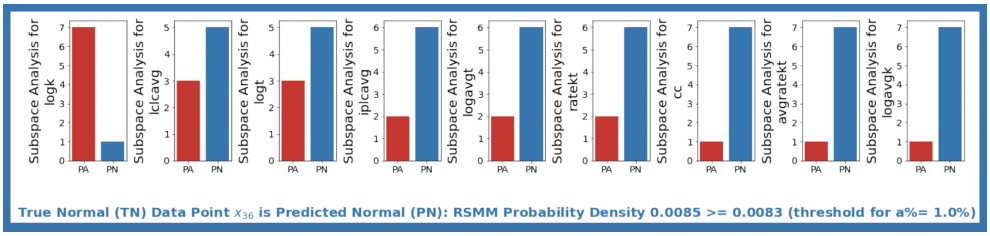
Note that FIgure 5.5a is a true normal point that is predicted to be anomalous. What is beneficial about the bar plots is that the two most anomalous attributes in Figure 5.5a are



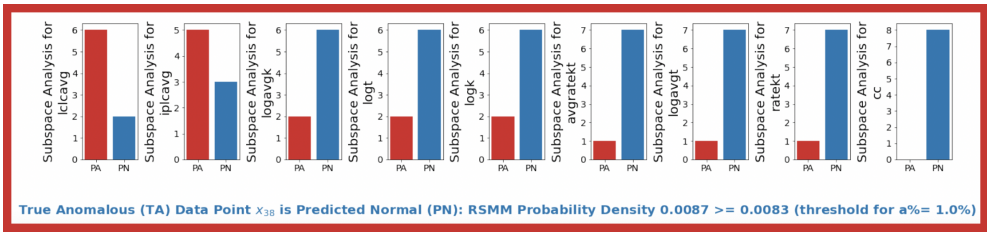
(a) LFR1 subspaces trained with anomalies included



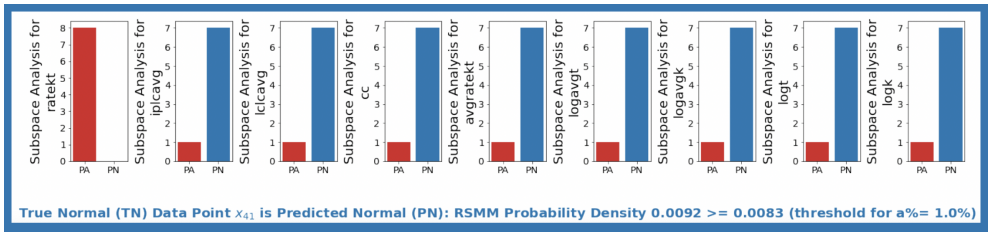
(b) LFR1 subspaces trained with anomalies included



(c) LFR1 subspaces trained with anomalies removed



(d) nPSO6 subspaces trained with anomalies included



(e) nPSO6 subspaces trained with anomalies removed

Figure 5.5: Interpretable bar plots showing attributes most anomalous for a state vector. See text for more details.

logt and logk, which corresponds to a high degree vertex. Since the networks are scale-free (see Definition 2.2.4), high degree vertices are expected to be not very common and therefore while it makes sense that one might show up to be anomalous, it is not very interesting or beneficial in detecting Sybils. Alternatively, a less anomalous (but still predicted anomalous) example of a true anomaly is shown in Figure 5.5b. Here, the most anomalous attributes are lclcavg and iplcavg which are expected to be important in detecting Sybils. Figures 5.6 and 5.7 highlight these two points in the different subspaces and also rank the subspaces based on the percentile of the data point relative to the training data, with lower percentiles shown first.

Claim 5.8.7. (CS) If a user is familiar with the attributes of a dataset, knowing which attributes are most anomalous across the subspaces may be informative (see Figure 5.5). Additionally, visualizing where a data point lies in the subspaces with that have the lowest percentile scores relative to the training data can be used to make additional insights (see Figures 5.6 and 5.7).

Seeing where each point lies in the subspaces can provide a user with more information as to how anomalous a data point may be. The underlying data is not actually a mixture of Gaussians, and attributes like logt and logk still are right-skewed even when the logarithm is used. By analyzing Figure 5.6, it can be seen that the true normal data point (the same as the one in Figure 5.5a) follows the general patterns in the dataset although the combined score makes it seem anomalous. Alternatively, the data point in Figure 5.7 corresponding to a true anomalous point in Figure 5.5b is part of a small cluster distinct from the majority of the other data points.

Back to the bar plots, third example is of a true normal point that is predicted normal, shown in Figure 5.5c, where one attribute (logk, which is less indicative of a Sybil by itself) is more anomalous across different subspaces than not, but the other attributes are charac-

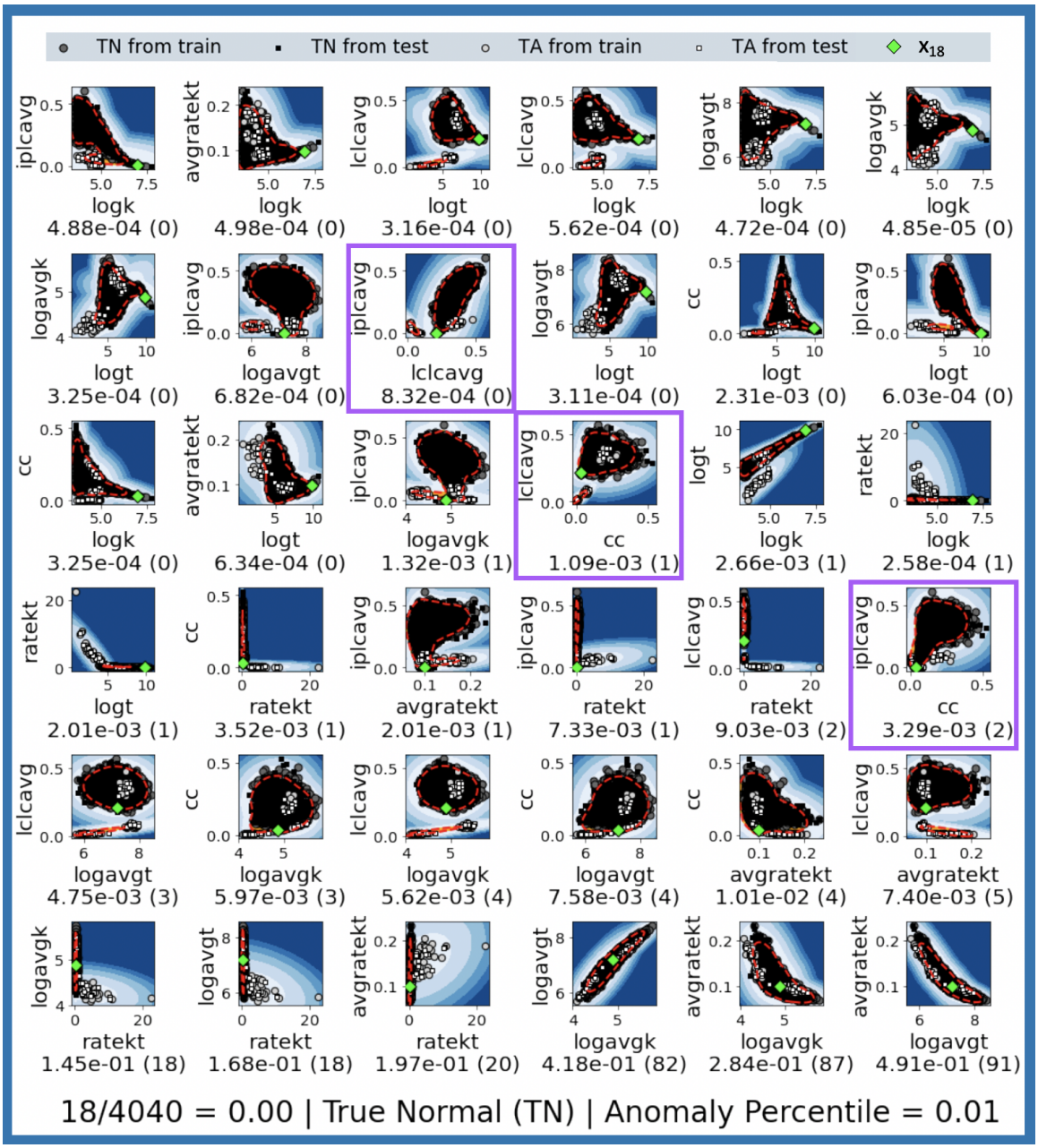


Figure 5.6: Subspaces of a true normal data point predicted anomalous.

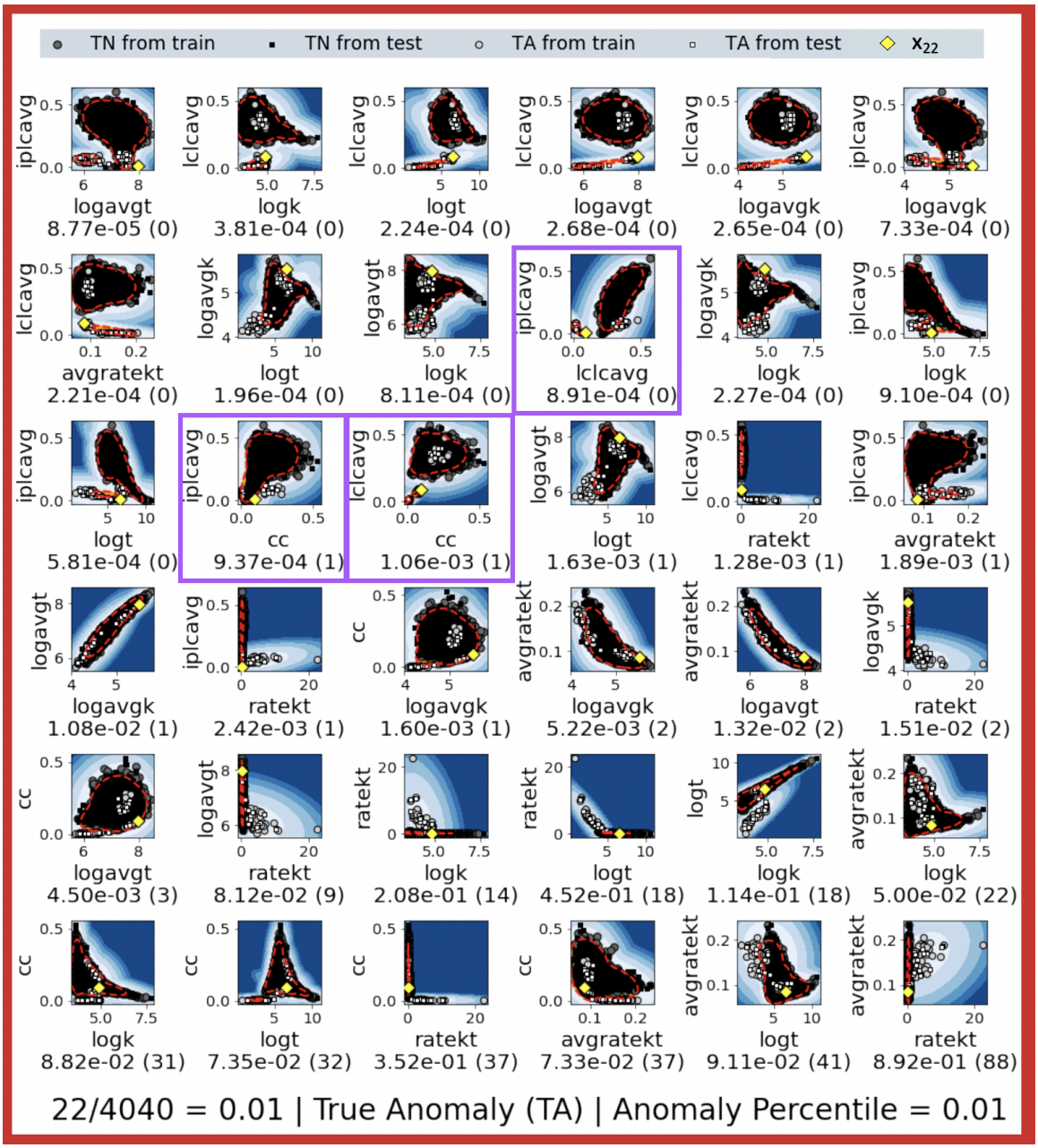


Figure 5.7: Subspaces of a true anomalous data point predicted anomalous.

terized as normal across most subspaces. Alternatively, Figure 5.5d is of a true anomalous point that is predicted normal where it can be seen that while it is predicted normal, two of the attributes expected to be important in detecting Sybils are more anomalous than not across the different subspaces. Lastly, in Figure 5.5e which represents a true normal point predicted as normal, one anomaly that may be indicative of a Sybil is anomalous in all subspaces, but none of the other attributes are anomalous in any other subspace.

Claim 5.8.8. (CS) The techniques used in this section to understand anomalies in the test set can be also used after an initial trained model to then process the outliers in the training dataset. Assuming the anomalies can be understood on whether they are desired anomalies or not, anomalies can be removed and the model can be retrained on the now pre-processed training set.

As shown in Subsection 5.8.2, anomaly detection algorithms can still perform well with a small amount of training data. While it was not necessary to do on a relatively small dataset with $N = 10,000$ vertices, on real-world networks with billions of vertices it may be very beneficial to do so. In the next section, results are generated assuming that a human was able to analyze the training data using the techniques described in this section with possible outside verification as to which users were benign and which were anomalous. It is assumed that the anomalous users were able to be removed from the training dataset. The question is, how would that affect the performance of the anomaly detection algorithms if this were to be achieved?

5.8.6 Sybil Detection with Human-in-the-Loop Interaction

Assuming that a human-in-the-loop is able to look through the training data after an initial RSMM is trained to determine which points in the training data are true anomalies and which are not using the techniques described in the previous section, the anomaly detec-

tion experiments are re-run on the pre-processed training data to compare the performance across different attribute sets and different anomaly detection methods. 10 independent trials are run using 60% training with the anomalies removed to represent the pre-processing and 40% testing with the average ROC-AUC shown in Figure 5.9, the average precision shown in Figure 5.10, and the average maximum percentile shown in Figure 5.11.

Table 5.9: Mean and standard deviation of the average ROC-AUC scores as % for the pre-processed benchmark graphs with injected Sybils (average ROC-AUC is calculated using 10 independent trials using 60% training data, highest mean of the average ROC-AUC highlighted in bold); Rank of mean across all attribute sets and anomaly detection algorithms (lower is better).

Data	CBLOF	FB	HBOS	IForest	KNN	LOF	OCSVM	COPOD	VAE	NEAGMM	RSMM
AS1_LFR	98±1 (35)	100±0 (14)	98±1 (36)	98±0 (33)	100±0 (18)	100±0 (15)	98±1 (32)	95±1 (44)	96±1 (41)	100±0 (19)	99±0 (25)
AS2_LFR	83±4 (50)	86±4 (46)	83±6 (52)	85±4 (48)	83±5 (51)	85±4 (47)	84±4 (49)	81±4 (55)	88±2 (45)	82±6 (53)	82±5 (54)
AS3_LFR	99±0 (26)	100±0 (9)	98±0 (39)	99±0 (30)	100±0 (6)	**100±0 (1)**	100±0 (22)	96±0 (42)	98±0 (31)	100±0 (4)	100±0 (6)
AS4_LFR	99±0 (28)	100±0 (5)	97±1 (40)	98±0 (38)	100±0 (17)	100±0 (10)	99±0 (27)	95±1 (43)	98±0 (34)	100±0 (13)	100±0 (16)
AS5_LFR	100±0 (23)	100±0 (2)	100±0 (21)	99±0 (24)	100±0 (11)	100±0 (2)	100±0 (20)	98±0 (37)	99±0 (29)	100±0 (12)	100±0 (8)
AS1_nPSO	91±3 (42)	100±0 (21)	92±1 (40)	92±2 (39)	99±0 (27)	100±0 (18)	92±2 (41)	90±2 (43)	88±2 (44)	100±0 (19)	97±2 (38)
AS2_nPSO	80±4 (48)	82±3 (46)	75±3 (55)	80±2 (47)	80±4 (49)	82±3 (45)	79±3 (52)	76±3 (54)	78±2 (53)	80±4 (50)	80±4 (51)
AS3_nPSO	100±0 (17)	100±0 (9)	98±1 (34)	99±0 (25)	100±0 (5)	**100±0 (1)**	100±0 (14)	99±1 (30)	99±0 (23)	**100±0 (1)**	100±0 (10)
AS4_nPSO	100±0 (22)	100±0 (5)	98±1 (37)	99±0 (28)	100±0 (12)	**100±0 (1)**	100±0 (20)	98±1 (31)	99±1 (29)	100±0 (5)	100±0 (16)
AS5_nPSO	99±0 (26)	100±0 (8)	98±1 (35)	98±1 (32)	100±0 (12)	**100±0 (1)**	99±0 (24)	98±1 (36)	98±1 (33)	100±0 (10)	100±0 (15)
AS1_Both	94±4 (42)	100±0 (17)	95±3 (41)	95±3 (39)	100±0 (21)	100±0 (15)	95±4 (40)	92±3 (43)	92±4 (44)	100±0 (19)	98±2 (33)
AS2_Both	81±3 (50)	84±3 (45)	79±6 (54)	83±3 (48)	81±3 (51)	84±3 (46)	82±3 (49)	79±4 (55)	83±5 (47)	81±3 (52)	81±3 (53)
AS3_Both	100±0 (22)	100±0 (9)	98±1 (34)	99±1 (26)	100±0 (6)	**100±0 (1)**	100±0 (18)	97±2 (37)	99±1 (27)	100±0 (3)	100±0 (8)
AS4_Both	99±0 (25)	100±0 (4)	97±1 (36)	98±1 (32)	100±0 (14)	100±0 (7)	99±0 (23)	97±2 (38)	99±1 (30)	100±0 (12)	100±0 (16)
AS5_Both	99±0 (24)	100±0 (4)	99±1 (29)	99±1 (28)	100±0 (10)	100±0 (2)	100±0 (20)	98±1 (35)	99±1 (31)	100±0 (10)	100±0 (13)

Unlike in the non-preprocessed example, RSMM does not perform best overall, although it does obtain better scores than the non-preprocessed example. Not only does its average precision increase, but all anomalies in the testing set can be found by looking through only 1% of the data which corresponds to the true anomaly rate (with rounding). Another difference is the best performing attribute set is AS3 which includes just the three new attributes proposed in this chapter. With the anomalies removed in the training data, which was possible due to the interpretability of RSMM, the best performing anomaly detection algorithm becomes LOF which obtains near perfect average precision values. To understand both why RSMM improves and why LOF obtains the best performance, Figure 5.8 shows what learned model looks like for RSMM on the data that contains anomalies in

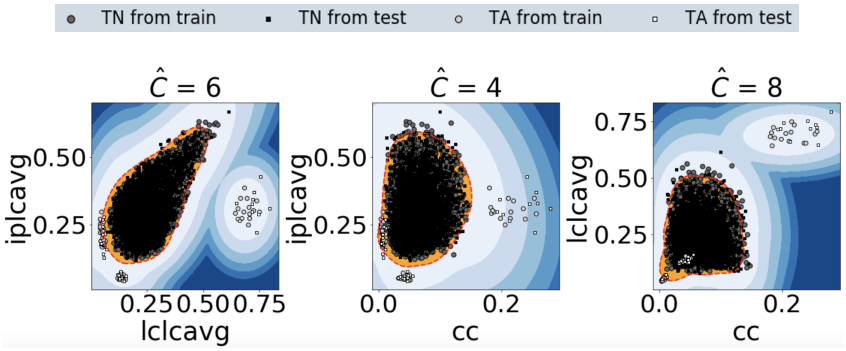
Table 5.10: Mean and standard deviation of the average precision scores as % for the pre-processed benchmark graphs with injected Sybils (average ROC-AUC is calculated using 10 independent trials using 60% training data, highest mean of the average precision highlighted in bold); Rank of mean across all attribute sets and anomaly detection algorithms (lower is better).

Data	CBLOF	FB	HBOS	IForest	KNN	LOF	OCSVM	COPOD	VAE	NEAGMM	RSMM
AS1_LFR	50±7 (42)	88±2 (12)	51±4 (41)	52±8 (40)	77±7 (18)	88±3 (11)	56±10 (38)	45±2 (49)	47±3 (46)	78±2 (17)	73±5 (21)
AS2_LFR	47±3 (45)	60±9 (34)	45±3 (50)	48±4 (43)	55±9 (39)	61±9 (33)	48±3 (44)	46±2 (48)	46±2 (47)	59±11 (36)	59±9 (37)
AS3_LFR	39±10 (51)	97±3 (2)	28±8 (54)	36±4 (52)	97±2 (5)	**99±1 (1)**	65±7 (32)	26±0 (55)	29±6 (53)	97±2 (3)	96±1 (6)
AS4_LFR	67±5 (31)	96±1 (8)	59±12 (35)	68±7 (28)	75±3 (19)	93±2 (10)	71±2 (24)	68±3 (27)	67±5 (30)	86±3 (15)	80±5 (16)
AS5_LFR	71±2 (23)	97±2 (4)	71±7 (22)	70±4 (25)	87±3 (14)	96±2 (7)	73±2 (20)	70±4 (26)	67±6 (29)	87±3 (13)	94±2 (9)
AS1_nPSO	49±0 (41)	80±5 (20)	48±1 (50)	45±1 (54)	70±3 (29)	84±5 (17)	49±0 (44)	45±1 (53)	49±0 (44)	78±3 (23)	56±3 (35)
AS2_nPSO	49±0 (44)	52±4 (37)	43±3 (55)	48±1 (51)	49±0 (42)	55±7 (36)	49±0 (44)	46±1 (52)	49±0 (44)	49±0 (44)	49±0 (42)
AS3_nPSO	83±10 (19)	98±1 (9)	67±3 (31)	74±3 (28)	99±2 (5)	100±0 (2)	91±7 (14)	75±1 (27)	78±8 (24)	100±1 (3)	96±4 (10)
AS4_nPSO	80±8 (21)	98±1 (7)	57±3 (34)	63±3 (32)	95±4 (12)	**100±0 (1)**	83±9 (18)	70±1 (30)	77±7 (25)	99±1 (6)	85±8 (16)
AS5_nPSO	75±4 (26)	98±1 (8)	52±2 (38)	49±1 (40)	95±4 (13)	99±1 (4)	79±8 (22)	51±2 (39)	59±2 (33)	96±1 (11)	88±9 (15)
AS1_Both	49±5 (45)	84±5 (16)	50±3 (44)	48±6 (47)	73±6 (23)	86±5 (14)	52±8 (41)	45±2 (54)	48±3 (50)	78±3 (18)	64±9 (28)
AS2_Both	48±2 (49)	56±6 (36)	44±3 (55)	48±3 (48)	52±7 (42)	58±6 (34)	48±2 (46)	46±2 (53)	48±2 (51)	54±8 (38)	54±8 (39)
AS3_Both	61±25 (31)	97±3 (5)	47±21 (52)	55±20 (37)	98±2 (3)	**100±1 (1)**	78±15 (19)	50±25 (43)	54±26 (40)	98±2 (2)	96±3 (9)
AS4_Both	73±9 (22)	97±2 (7)	58±8 (35)	65±6 (27)	85±11 (15)	96±4 (8)	77±9 (20)	69±2 (26)	72±8 (25)	92±7 (10)	82±7 (17)
AS5_Both	73±4 (24)	97±2 (6)	61±11 (30)	60±11 (33)	91±6 (12)	98±2 (4)	76±6 (21)	61±10 (32)	63±6 (29)	91±5 (11)	91±7 (13)

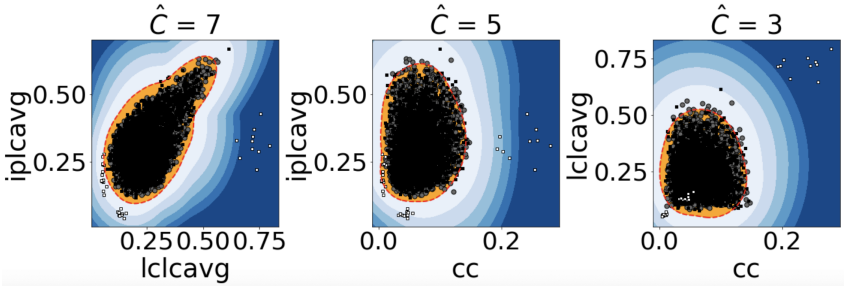
Table 5.11: Average maximum percentile of different anomalies in pre-processed benchmark graphs. The ranks in parentheses are for the row and column among similar benchmark type, respectively.

Data	Normal Sybil	Dense Sybil	Sparse Sybil	Creeper
AS1_LFR	7±2 (4,4)	1±1 (3,4)	1±0 (1,1)	1±0 (2,4)
AS2_LFR	90±26 (4,5)	3±4 (3,5)	3±1 (2,5)	1±0 (1,1)
AS3_LFR	1±0 (2,1)	0±0 (1,1)	1±0 (3,4)	1±0 (4,5)
AS4_LFR	2±0 (4,3)	1±0 (1,3)	1±0 (3,3)	1±0 (2,2)
AS5_LFR	1±0 (4,2)	1±0 (1,2)	1±0 (2,2)	1±0 (4,3)
AS1_nPSO	3±0 (3,4)	22±14 (4,4)	0±0 (2,1)	0±0 (1,3)
AS2_nPSO	93±4 (4,5)	35±20 (3,5)	0±0 (2,1)	0±0 (1,1)
AS3_nPSO	1±0 (3,3)	1±0 (4,1)	1±0 (2,5)	0±0 (1,5)
AS4_nPSO	1±0 (3,1)	2±1 (4,3)	0±0 (2,1)	0±0 (1,2)
AS5_nPSO	1±0 (3,2)	2±1 (4,2)	0±0 (2,1)	0±0 (1,3)
AS1_Both	5±3 (3,4)	12±15 (4,4)	1±0 (1,1)	1±0 (2,4)
AS2_Both	91±4 (4,5)	19±21 (3,5)	2±1 (2,5)	1±0 (1,1)
AS3_Both	1±0 (4,1)	1±1 (1,1)	1±0 (3,4)	1±0 (2,5)
AS4_Both	1±0 (3,3)	1±1 (4,3)	1±0 (2,3)	1±0 (1,2)
AS5_Both	1±0 (3,2)	1±1 (4,2)	1±0 (1,2)	1±0 (2,3)

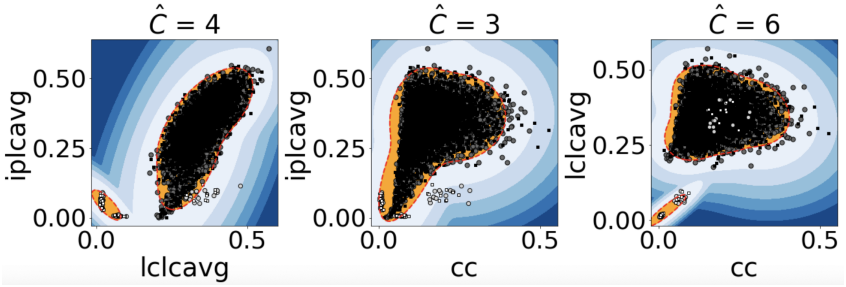
the training dataset and on the data in which the anomalies were removed.



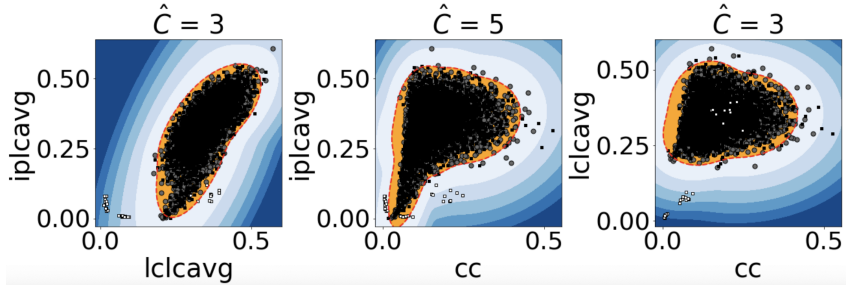
(a) LFR1 subspaces trained with anomalies included



(b) LFR1 subspaces trained with anomalies removed



(c) nPSO6 subspaces trained with anomalies included



(d) nPSO6 subspaces trained with anomalies removed

Figure 5.8: Subspace densities with and without pre-processing.

Figure 5.8a and Figure 5.8c visualize the probability densities of the RSMMs that are trained with the anomalies included. The dark blue represents less probable areas with the lighter blue representing more probable but still predicted anomalous areas. The orange represents areas inside of the red borders which represents predicted normal areas. In Figure 5.8a, it can be seen that small components are centered around several of the anomalies and in Figure 5.8c, the anomalies are distinct enough to create regions in which the data points would be predicted normal. To a user, these regions would look anomalous if the techniques were used that were proposed in Subsection 5.8.5. Figure 5.8b and Figure 5.8d visualize the probability densities of the RSMMs that are trained with the anomalies removed. Now the anomalous test data shown in the blue regions do not have gaussian components centered around them. That said, since the data is not truly Gaussian, the model is only an approximate fit. With the anomalies removed, LOF which a proximity-based model that does not assume a specific distribution is able to perform best.

Claim 5.8.9. (CS) Overall, the anomaly detection performance improves when the training set is pre-processed (see 5.9, 5.10, and 5.8 compared to 5.6, 5.7, and 5.8). The RSMM provides some visualizations that are interpretable (see Figures 5.3, 5.4, 5.5, 5.6 and 5.7)) and therefore can be used to assist the process. The algorithm has been shown to perform well on a small amount of training data (see Claim 5.8.2) making a potentially smaller amount of data for a user to verify.

5.9 Conclusions

In this chapter, we propose using benchmark graphs with community structure as a way to analyze possible attributes and anomaly detection algorithms that may be useful in detecting malicious fake accounts called Sybils so that performance can be evaluated more easily, though it may oversimplify the problem (see Claim 5.2.1). Because online social

networks are quite complicated, it is hard to determine which type of community structure it may have, therefore two different benchmark graphs with community structure are used to evaluation. Anomalies are injected into the benchmark graphs that are consistent with those studied in literature and they are found to create small disjoint clusters in some of the proposed attribute spaces 5.8.3 that make use of the link cohesion metrics proposed in Chapters 3 and 4 (see Claim 5.4.9 and Definitions 5.4.7 and 4.4.7).

A new anomaly detection method is introduced that builds off of Gaussian mixture models but adds a small amount of noise to the training data prevent singularity issues (see Claim 5.5.1) and determines the number of components automatically, yielding a Noise-Enhanced Automatic Gaussian mixture model (NEAGMM). A Random Subspace Mixture Model (RSMM) is an algorithm that reduces the dimensionality of a model by creating an ensemble of low-dimensional subspaces (see Claim 5.6.1) with balanced participation from each attribute (see Claim 5.6.2) and geometric averaging to combine subspace attribute scores (see 5.6.3). Default parameters are selected that provide good performance on datasets not part set used in the main experiments but also have an added benefit of being more interpretable than using subspaces of higher dimension, though once the model is understood higher dimensions would also provide interpretable results (see Claims 5.8.1 and 5.8.6). The algorithm was shown to obtain the highest mean ROC-AUC with corresponding highest average ranking on a wide range of standard multidimensional point datasets compared to 10 baseline community detection algorithms when both 60% training data was used and 5% training data was used (see Claim 5.8.2).

For detecting Sybils, the newly proposed attributes based on the link cohesion metrics from Chapters 3 and 4 were shown to provide an overall benefit compared to the baseline attributes (see Claim 5.8.4) with the best overall performing metric in terms of average precision using all 9 attributes and the RSMM anomaly detection algorithm. As stated previously, RSMM was designed to be interpretable (see Claims 5.8.1 and 5.8.6) both to

understand what attributes contribute most to a data point being consider anomalous (see Figure 5.5 where the balances attributes are useful from Claim 5.6.2) and to determine which subspaces a data point is most anomalous in relative to the training data (see Figures 5.6 and 5.7) and Claim 5.8.7). Given a user who understand the data and is able to interpret the results, it is possible that the techniques described could be used to process training data after an initial models is built to verify outliers that are expected and outliers that they would want marked as anomalies and therefore remove them from the training set (see Claim 5.8.8). Since small amount of training data has shown to yield anomaly detection results that are not severely degraded, validating a small fraction of the data may not be so daunting given visualization tools. If a user is able to correctly understand the data and remove anomalies, it is shown that the performance of the anomaly detection algorithms would significantly improve with this human-in-the-loop interaction (see Claim 5.8.9).

Bibliography

[1] R. Abdolazimi, S. Jin, and R. Zafarani. Noise-enhanced community detection. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, 2020.

[2] C. C. Aggarwal. Outlier ensembles: position paper. *ACM SIGKDD Explorations Newsletter*, 14(2), 2013.

[3] M. Ahmed, A. N. Mahmood, and M. R. Islam. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55.

[4] K. Aho, D. Derryberry, and T. Peterson. Model selection for ecologists: the worldviews of aic and bic. *Ecology*, 95(3), 2014.

[5] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*. Springer, 1998.

[6] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2010.

[7] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3), 2015.

[8] M. Al-Qurishi, M. Al-Rakhami, A. Alamri, M. Alrubaian, S. M. M. Rahman, and M. S. Hossain. Sybil defense techniques in online social networks: a survey. *IEEE Access*, 5, 2017.

[9] R. D. Alba. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, 3(1), 1973.

[10] L. Ana and A. K. Jain. Robust data clustering. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2. IEEE, 2003.

[11] L. Antwarg, R. M. Miller, B. Shapira, and L. Rokach. Explaining anomalies detected by autoencoders using shap. *arXiv preprint arXiv:1903.02407*, 2019.

[12] K. Appel and W. Haken. Every planar map is four colorable. *Bulletin of the American mathematical Society*, 82(5), 1976.

[13] T. L. Athey, B. D. Pedigo, T. Liu, and J. T. Vogelstein. Autogmm: Automatic and hierarchical gaussian mixture modeling in python. *arXiv preprint arXiv:1909.02688*, 2019.

[14] K. Audhkhasi, O. Osoba, and B. Kosko. Noise-enhanced convolutional neural networks. *Neural Networks*, 78, 2016.

[15] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

[16] D. Bader and J. McCloskey. Modularity and graph algorithms. *SIAM AN10 Minisymposium on Analyzing Massive Real-World Graphs*, 1216(2), 2009.

[17] L. Bahri, B. Carminati, and E. Ferrari. Coip—continuous, operable, impartial, and privacy-aware identity validity estimation for osn profiles. *ACM Transactions on the Web (TWEB)*, 10(4), 2016.

[18] L. Bai, J. Liang, H. Du, and Y. Guo. A novel community detection algorithm based on simplification of complex networks. *Knowledge-Based Systems*, 143, 2018.

[19] A. Barabási. Network science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987), 2013.

[20] A. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439), 1999.

[21] V. Barnett and T. Lewis. Outliers in statistical data, wiley and sons. *New York*, 1994.

[22] S. Battiston. Inner structure of capital control networks. *Physica A: Statistical Mechanics and its Applications*, 338(1-2), 2004.

[23] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. A. Baeza-Yates. Link-based characterization and detection of web spam. In *AIRWeb*, 2006.

[24] K. Bellware. Cheer mom used deepfake nudes and threats to harass daughter's teammates, police say. *The Washington Post*, 3, 2021.

[25] S. Ben-David, U. Von Luxburg, and D. Pál. A sober look at clustering stability. In *International Conference on Computational Learning Theory*. Springer, 2006.

[26] A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. In *Biocomputing 2002*. World Scientific, 2001.

[27] J. Benedetto, W. Czaja, J. Dobrosotskaya, T. Doster, K. Duke, and D. Gillis. Semi-supervised learning of heterogeneous data in remote sensing imagery. In *Independent Component Analyses, Compressive Sampling, Wavelets, Neural Net, Biosystems, and Nanoengineering X*, volume 8401. International Society for Optics and Photonics, 2012.

[28] J. J. Benedetto and M. Fickus. Finite normalized tight frames. *Advances in Computational Mathematics*, 18(2), 2003.

[29] H. Bengtsson. future.apply: Apply function to elements in parallel using futures. *cran.r-project.org*, 1, 2021.

[30] J. W. Berry, B. Hendrickson, R. A. LaViolette, and C. A. Phillips. Tolerating the community detection resolution limit with edge weighting. *Physical Review E*, 83(5), 2011.

[31] C. Bichot and P. Siarry. *Graph partitioning*. Wiley Online Library, 2011.

[32] N. Biggs, E. K. Lloyd, and R. J. Wilson. *Graph Theory, 1736-1936*. Oxford University Press, 1986.

[33] F. Bloch, M. O. Jackson, and P. Tebaldi. Centrality measures in networks. *Available at SSRN 2749124*, 2019.

[34] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), 2008.

[35] P. Bock. *Getting it right: R&D methods for science and engineering*. Academic Press, 2001.

[36] B. Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1(4), 1980.

[37] B. Bollobás and O. Riordan. Sparse graphs: metrics and random models. *Random Structures & Algorithms*, 39(1), 2011.

[38] S. P. Borgatti, K. M. Carley, and D. Krackhardt. On the robustness of centrality measures under conditions of imperfect data. *Social Networks*, 28(2), 2006.

[39] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2), 2007.

[40] L. Breiman. Random forests. In *Machine Learning*, 2001.

[41] L. Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3), 2001.

[42] M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 2000.

[43] A. D. Broido and A. Clauset. Scale-free networks are rare. *Nature Communications*, 10(1), 2019.

[44] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013.

[45] C. V. Cannistraci. Modelling self-organization in complex networks via a brain-inspired network automata theory improves link reliability in protein interactomes. *Scientific Reports*, 8(1), 2018.

[46] C. V. Cannistraci, G. Alanis-Lobato, and T. Ravasi. From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. *Scientific Reports*, 3(1), 2013.

[47] C. V. Cannistraci and A. Muscoloni. Latent geometry inspired graph dissimilarities enhance affinity propagation community detection in complex networks. *arXiv preprint arXiv:1804.04566*, 2018.

[48] M. Carletti, M. Terzi, and G. A. Susto. Interpretable anomaly detection with diffi: Depth-based feature importance for the isolation forest. *Submitted to the IEEE Transactions on Cybernetics*, 2020.

[49] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8), 2019.

[50] C. Castellano, F. Cecconi, V. Loreto, D. Parisi, and F. Radicchi. Self-contained algorithms to detect communities in networks. *The European Physical Journal B*, 38(2), 2004.

[51] J. M. Chambers. Greater or lesser statistics: a choice for future research. *Statistics and Computing*, 3(4), 1993.

[52] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3), 2009.

[53] H. Chen, L. R. Varshney, and P. K. Varshney. Noise-enhanced information systems. *Proceedings of the IEEE*, 102(10), 2014.

[54] M. Chen. Em algorithm for gaussian mixture model (em gmm). *MATLAB Central File Exchange*, 12, 2009.

[55] M. Chen, K. Kuzmin, and B. K. Szymanski. Community detection via maximization of modularity and its variants. *IEEE Transactions on Computational Social Systems*, 1(1), 2014.

[56] M. Chen, T. Nguyen, and B. Szymański. A new metric for quality of network community structure. *ASE Human Journal*, 2, 2013.

[57] T. Chen, P. Singh, and K. E. Bassler. Network community detection using modularity density measures. *Journal of Statistical Mechanics: Theory and Experiment*, 2018(5), 2018.

[58] B. Chesney and D. Citron. Deep fakes: A looming challenge for privacy, democracy, and national security. *California Law Review*, 107, 2019.

[59] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3), 1968.

[60] M. Ciglan, M. Laclavík, and K. Nørvåg. On community detection in real-world networks and the importance of degree assortativity. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013.

[61] C. A. Clark and K. A. Okoudjou. Nonlinear dimensionality reduction applied to the binary classification of images. 2013.

[62] C. A. Clark and K. A. Okoudjou. On optimal frame conditioners. In *2015 International Conference on Sampling Theory and Applications (SampTA)*. IEEE, 2015.

[63] A. Clauset, M. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6), 2004.

[64] A. Clauset, C. R. Shalizi, and M. Newman. Power-law distributions in empirical data. *SIAM review*, 51(4), 2009.

[65] A. Cloninger and W. Czaja. Eigenvector localization on data-dependent graphs. In *2015 International Conference on Sampling Theory and Applications (SampTA)*. IEEE, 2015.

[66] A. Cloninger, W. Czaja, and T. Doster. Multimodal data fusion via graph-and operator-based techniques. In *Hyperspectral Imaging and Sounding of the Environment*. Optical Society of America, 2015.

[67] Alex Cloninger and HN Mhaskar. A low discrepancy sequence on graphs. *arXiv preprint arXiv:2010.04227*, 2020.

[68] Alexander Cloninger and Timo Klock. A deep network construction that adapts to intrinsic dimensionality beyond the domain. *Neural Networks*, 2021.

[69] J. D. Cohen. Trusses: Cohesive subgraphs for social network analysis. *National Security Agency Technical Report*, 16, 2008.

[70] J. D. Cohen. Graph twiddling in a mapreduce world. *Computing in Science & Engineering*, 11(4), 2009.

[71] J. D. Cohen. Trusses and trapezes: Easily-interpreted communities in social networks. *arXiv*, abs/1907.09417, 2019.

[72] R. Cohen and S. Havlin. Scale-free networks are ultrasmall. *Physical Review Letters*, 90(5), 2003.

[73] V. Colizza, A. Flammini, M. A. Serrano, and A. Vespignani. Detecting rich-club ordering in complex networks. *Nature Physics*, 2(2), 2006.

[74] G. Corera. Mi5 warns of spies using linkedin to trick staff into spilling secrets. *BBC News*, 4, 2021.

[75] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems, 2006.

[76] W. Czaja, D. Dong, P. Jabin, and F. Olivier N. Njeunje. Transport model for feature extraction. *SIAM Journal on Mathematics of Data Science*, 3(1), 2021.

[77] W. Czaja, T. Doster, and A. Halevy. An overview of numerical acceleration techniques for nonlinear dimension reduction. *Recent Applications of Harmonic Analysis to Function Spaces, Differential Equations, and Data Science*, 2017.

[78] W. Czaja and M. Ehler. Schroedinger eigenmaps for the analysis of biomedical data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5), 2012.

[79] S. Dagstuhl. dblp. `https://dblp.uni-trier.de/`, 2021.

[80] X. H. Dang, I. Assent, R. T. Ng, A. Zimek, and E. Schubert. Discriminative features for identifying and interpreting outliers. In *2014 IEEE 30th International Conference on Data Engineering*. IEEE, 2014.

[81] X. H. Dang, B. Micenková, I. Assent, and R. T. Ng. Local outlier detection with interpretation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2013.

[82] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of statistical mechanics: Theory and experiment*, 2005(09), 2005.

[83] P. De Meo, E. Ferrara, G. Fiumara, and A. Ricciardello. A novel measure of edge centrality in social networks. *Knowledge-based Systems*, 30, 2012.

[84] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1), 1977.

[85] Q. Ding, N. Katenka, P. Barford, E. Kolaczyk, and M. Crovella. Intrusion as (anti) social communication: characterization and detection. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012.

[86] D. Donoho. 50 years of data science. *Journal of Computational and Graphical Statistics*, 26(4), 2017.

[87] T. Doster, J. Benedetto, and W. Czaja. Nonlinear dimensionality reduction for hyperspectral image classification. *Project Final Report*, 2011.

[88] J. R. Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*. Springer, 2002.

[89] D. Dua and C. Graff. UCI machine learning repository, 2017.

[90] R. Dunbar. Do online social media cut through the constraints that limit the size of offline social networks. *Royal Society Open Science*, 3(1), 2016.

[91] C. Durán, A. Muscoloni, and C. V. Cannistraci. Geometrical inspired pre-weighting enhances markov clustering community detection in complex networks. *Applied Network Science*, 6(1), 2021.

[92] M. Ehler and K. A. Okoudjou. Minimization of the probabilistic p-frame potential. *Journal of Statistical Planning and Inference*, 142(3), 2012.

[93] P. Erdős and A. Rényi. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, 5(1), 1960.

[94] P. Erdös and G. Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2, 1935.

[95] K. S. Ettabaa and M. B. Salem. Anomaly detection in hyperspectral imagery: an overview. In *Environmental Information Systems: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2019.

[96] L. Euler. The seven bridges of Königsberg. *The World of Mathematics*, 1, 1956.

[97] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *ACM SIGCOMM Computer Communication Review*, 29(4), 1999.

[98] A. Faqeeh, S. Osat, and F. Radicchi. Characterizing the analogy between hyperbolic embedding and community structure of complex networks. *Physical Review Letters*, 121(9), 2018.

[99] M. Fire, R. Goldschmidt, and Y. Elovici. Online social networks: threats and solutions. *IEEE Communications Surveys & Tutorials*, 16(4), 2014.

[100] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5), 2010.

[101] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1), 2007.

[102] S. Fortunato and D. Hric. Community detection in networks: A user guide. *Physics Reports*, 659, 2016.

[103] S. Fortunato, V. Latora, and M. Marchiori. Method to find community structures based on information centrality. *Physical Review E*, 70(5), 2004.

[104] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2), 2004.

[105] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 1977.

[106] J. Gao and P. Tan. Converting output scores from outlier detection algorithms into probability estimates. In *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006.

[107] P. Gao, N. Z. Gong, S. Kulkarni, K. Thomas, and P. Mittal. Sybilframe: A defense-in-depth framework for structure-based sybil detection. *arXiv preprint arXiv:1503.02985*, 2015.

[108] P. Gao, B. Wang, N. Z. Gong, S. R. Kulkarni, K. Thomas, and P. Mittal. Sybilfuse: Combining local attributes with global structure to perform robust sybil detection. In *2018 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2018.

[109] J. O. Garcia, A. Ashourvan, S. Muldoon, J. M. Vettel, and D. S. Bassett. Applications of community detection techniques to brain graphs: Algorithmic considerations and implications for neural function. *Proceedings of the IEEE*, 106(5), 2018.

[110] E. N. Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4), 1959.

[111] M. Girvan and M. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12), 2002.

[112] J. Glattfelder. Backbone of complex networks of corporations: The flow of control. In *Decoding Complexity*. Springer, 2013.

[113] M. Goldstein and A. Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *Poster and Demo Track of the 35th German Conference on Artificial Intelligence (KI-2012)*, 2012.

[114] B. H. Good, Y. De Montjoye, and A. Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4), 2010.

[115] N. Greggio. Anomaly detection in idss by means of unsupervised greedy learning of finite mixture models. *Soft Computing*, 22(10).

[116] D. Tenenbaum H. Wallig, S. Weston. doparallel: Foreach parallel adaptor for the 'parallel' package. *cran.r-project.org*, 10, 2020.

[117] S. Weston H. Wallig. foreach: Provides foreach looping construct. *cran.r-project.org*, 1, 2021.

[118] A. Hafftka, H. Celik, A. Cloninger, W. Czaja, and R. G. Spencer. 2d sparse sampling algorithm for nd fredholm equations with applications to nmr relaxometry. In *2015 International Conference on Sampling Theory and Applications (SampTA)*, 2015.

[119] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. In *Advances in Neural Information Processing Systems*, 2007.

[120] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10), 2003.

[121] D. O. Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[122] J. A. Henderson and P. A. Robinson. Geometric effects on complex network structure in the cortex. *Physical Review Letters*, 107(1), 2011.

[123] D. Hric, R. K. Darst, and S. Fortunato. Community detection in networks: Structural communities versus ground truth. *Physical Review E*, 90(6), 2014.

[124] H. Hu, T. Laurent, M. A. Porter, and A. L. Bertozzi. A method based on total variation for network modularity optimization using the mbo scheme. *SIAM Journal on Applied Mathematics*, 73(6), 2013.

[125] Y. Hu, H. Chen, P. Zhang, M. Li, Z. Di, and Y. Fan. Comparative definition of community and corresponding identifying algorithm. *Physical Review E*, 78(2), 2008.

[126] M. D. Humphries and K. Gurney. Network 'small-world-ness': a quantitative method for determining canonical network equivalence. *PloS one*, 3(4), 2008.

[127] B. Karrer and M. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1), 2011.

[128] R. Keller, E. Muller, and K. Bohm. Hics: High contrast subspaces for density-based outlier ranking. In *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 2012.

[129] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2), 1970.

[130] A. Khadivi and M. Hasler. A weighting scheme for enhancing community detection in networks. In *2010 IEEE International Conference on Communications*. IEEE, 2010.

[131] A. Khadivi, A. A. Rad, and M. Hasler. Network community-detection enhancement by proper weighting. *Physical Review E*, 83(4), 2011.

[132] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[133] C. Klymko, D. Gleich, and T. G. Kolda. Using triangles to improve community detection in directed networks. *ASE BigData/SocialCom/CyberSecurity Conference*, 2014.

[134] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. Adaptive Computation and Machine Learning. MIT Press, 2009.

[135] Paul J Koprowski. *Finite frames and graph theoretic uncertainty principles*. PhD thesis, 2015.

[136] B. Kovács and G. Palla. The inherent community structure of hyperbolic networks. *arXiv e-prints*, 2021.

[137] H. Kriegel, P. Kroger, E. Schubert, and A. Zimek. Interpreting and unifying outlier scores. In *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM, 2011.

[138] N. Kumar, D. Venugopal, L. Qiu, and S. Kumar. Detecting anomalous online reviewers: an unsupervised approach using mixture models. *Journal of Management Information Systems*, 36(4).

[139] N. Laleh, B. Carminati, and E. Ferrari. Graph based local risk estimation in large scale online social networks. In *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*. IEEE, 2015.

[140] A. Lancichinetti and S. Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, 84(6), 2011.

[141] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4), 2008.

[142] K. Lange. Social media scams are affecting the military. *Joint Base San Antonio*, 2019.

[143] R. Laxhammar. Anomaly detection for sea surveillance. In *2008 11th International Conference on Information Fusion*. IEEE, 2008.

[144] R. Laxhammar, G. Falkman, and E. Sviestins. Anomaly detection in sea traffic-a comparison of the gaussian mixture model and the kernel density estimator. In *2009 12th International Conference on Information Fusion*. IEEE, 2009.

[145] A. Lazarevic and V. Kumar. Feature bagging for outlier detection. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, New York, NY, USA, 2005. Association for Computing Machinery.

[146] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2006.

[147] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1), 2007.

[148] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, 2014.

[149] J. Lewis. How spies used facebook to steal nato chiefs' details. *The Telegraph*, 10, 2012.

[150] H. Li. Smile. `https://haifengl.github.io`, 2014.

[151] L. Li, R. J. Hansman, R. Palacios, and R. Welsch. Anomaly detection via a gaussian mixture model for flight operation and safety monitoring. *Transportation Research Part C: Emerging Technologies*, 64.

[152] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu. Copod: copula-based outlier detection. *IEEE International Conference on Data Mining (ICDM)*, 2020.

[153] F. T. Liu, K. M. Ting, and Z. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008.

[154] H. Long. Edge intensity-based community measurement in complex networks. *Physics Letters A*, 383(11), 2019.

[155] M. O. Lorenz. Methods of measuring the concentration of wealth. *Publications of the American Statistical Association*, 9(70), 1905.

[156] R. D. Luce. Connectivity and generalized cliques in sociometric group structure. *Psychometrika*, 15(2), 1950.

[157] R. D. Luce and A. D. Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2), 1949.

[158] S. Lundberg and S. Lee. A unified approach to interpreting model predictions. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[159] V. Lyzinski, M. Tang, A. Athreya, Y. Park, and C. E. Priebe. Community detection and classification in hierarchical stochastic blockmodels. *IEEE Transactions on Network Science and Engineering*, 4(1), 2016.

[160] M. Macha and L. Akoglu. Explaining anomalies in groups with characterizing subspace rules. *Data Mining and Knowledge Discovery*, 32(5), 2018.

[161] P. Marsden. Egocentric and sociocentric measures of network centrality. *Social Networks*, 24(4), 2002.

[162] S. Maslov and K. Sneppen. Specificity and stability in topology of protein networks. *Science*, 296(5569), 2002.

[163] S. Maslov, K. Sneppen, and A. Zaliznyak. Detection of topological patterns in complex networks: correlation profile of the internet. *Physica A: Statistical Mechanics and its Applications*, 333, 2004.

[164] K. McCallum. Think before you link. *CPNI*, 5, 2021.

[165] G. Mclachlan and K. Basford. *Mixture Models: Inference and Applications to Clustering*, volume 38. Marcel Dekker, New York, 1988.

[166] G. J. McLachlan, S. X. Lee, and S. I. Rathnayake. Finite mixture models. *Annual Review of Statistics and its Application*, 6, 2019.

[167] F. Menczer, S. Fortunato, and C. A. Davis. *A first course in network science*. Cambridge University Press, 2020.

[168] B. Micenková, R. T. Ng, X. H. Dang, and I. Assent. Explaining outliers by subspace separability. In *2013 IEEE 13th International Conference on Data Mining*. IEEE, 2013.

[169] M. Middendorf, E. Ziv, and C. H. Wiggins. Inferring network mechanisms: the drosophila melanogaster protein interaction network. *Proceedings of the National Academy of Sciences*, 102(9), 2005.

[170] R. Milo, N. Kashtan, S. Itzkovitz, M. Newman, and U. Alon. On the uniform generation of random graphs with prescribed degree sequences. *Technical Report*, 21, 2003.

[171] E. Minieka. The chinese postman problem for mixed networks. *Management Science*, 25(7), 1979.

[172] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, 2007.

[173] R. J. Mokken. Cliques, clubs and clans. *Quality & Quantity*, 13(2), 1979.

[174] J. Moody. Peer influence groups: identifying dense clusters in large networks. *Social Networks*, 23(4), 2001.

[175] B. Moradi-Jamei. Detecting communities and performing statistical inferences on networks through renewal non-backtracking random walks. *Kansas State University Doctor of Philosophy*, 2020.

[176] B. Moradi-Jamei, H. Shakeri, P. Poggi-Corradini, and M. J. Higgins. A new method for quantifying network cyclic structure to improve community detection. *Physica A: Statistical Mechanics and its Applications*, 2020.

[177] D. Mulamba, I. Ray, and I. Ray. On sybil classification in online social networks using only structural features. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2018.

[178] E. Müller, I. Assent, P. Iglesias, Y. Mülle, and K. Böhm. Outlier ranking via subspace analysis in multiple views of the data. In *2012 IEEE 12th International Conference on Data Mining*. IEEE, 2012.

[179] E. Müller, M. Schiffer, and T. Seidl. Statistical selection of relevant subspace projections for outlier ranking. In *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 2011.

[180] H. Muoz-Barona, J. Vettel, and A. Bohannon. Signal processing for time-series functions on a graph. Technical report, US Army Research Laboratory Aberdeen Proving Ground United States, 2018.

[181] J. Muschelli. matlabr: An interface for matlab using system calls. *cran.r-project.org*, 8, 2018.

[182] A. Muscoloni, I. Abdelhamid, and C. V. Cannistraci. Local-community network automata modelling based on length-three-paths for prediction of complex network structures in protein interactomes, food webs and more. *bioRxiv*, 2018.

[183] A. Muscoloni and C. V. Cannistraci. Rich-clubness test: how to determine whether a complex network has or doesn't have a rich-club. *arXiv preprint arXiv:1704.03526*, 2017.

[184] A. Muscoloni and C. V. Cannistraci. Leveraging the nonuniform pso network model as a benchmark for performance evaluation in community detection and link prediction. *New Journal of Physics*, 20(6), 2018.

[185] A. Muscoloni and C. V. Cannistraci. A nonuniform popularity-similarity optimization (npso) model to efficiently generate realistic complex networks with communities. *New Journal of Physics*, 20(5), 2018.

[186] A. Muscoloni and C. V. Cannistraci. Angular separability of data clusters or network communities in geometrical space and its relevance to hyperbolic embedding. *arXiv preprint arXiv:1907.00025*, 2019.

[187] A. Muscoloni and C. V. Cannistraci. Navigability evaluation of complex networks by greedy routing efficiency. *Proceedings of the National Academy of Sciences*, 116(5), 2019.

[188] A. Muscoloni, U. Michieli, and C. V. Cannistraci. Local-ring network automata and the impact of hyperbolic geometry in complex network link-prediction. *arXiv preprint arXiv:1707.09496*, 2017.

[189] A. Muscoloni, J. M. Thomas, S. Ciucci, G. Bianconi, and C. V. Cannistraci. Machine learning meets complex networks via coalescent embedding in the hyperbolic space. *Nature Communications*, 8(1), 2017.

[190] M. Myers. Sextortion: The u.s. military's dirty little secret is a growing national security concern. *Military Times*, 8, 2017.

[191] Z. P. Neal. How small is it comparing indices of small worldliness. *Network Science*, 5(1), 2017.

[192] M. Newman. Assortative mixing in networks. *Physical Review Letters*, 89(20), 2002.

[193] M. Newman. Mixing patterns in networks. *Physical Review E*, 67(2), 2003.

[194] M. Newman. Analysis of weighted networks. *Physical review E*, 70(5), 2004.

[195] M. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6), 2004.

[196] M. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46(5), 2005.

[197] M. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23), 2006.

[198] M. Newman. Equivalence between modularity optimization and maximum likelihood methods for community detection. *Physical Review E*, 94, 2016.

[199] M. Newman. *Networks*. Oxford University Press, second edition, 2018.

[200] M. Newman and A. Clauset. Structure and inference in annotated networks. *Nature Communications*, 7(1), 2016.

[201] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 2004.

[202] M. Newman and J. Park. Why social networks are different from other types of networks. *Physical Review E*, 68(3), 2003.

[203] H. V. Nguyen, H. H. Ang, and V. Gopalkrishnan. Mining outliers with ensemble of heterogeneous detectors on random subspaces. In *International Conference on Database Systems for Advanced Applications*. Springer, 2010.

[204] J. Nicas. Facebook connected her to a tattooed soldier in iraq or so she thought. *The New York Times*, 7, 2019.

[205] J. Nicas. Why can't the social networks stop fake accounts. *The New York Times*, 12, 2020.

[206] R. Noldus and P. Van Mieghem. Assortativity in complex networks. *Journal of Complex Networks*, 3(4), 2015.

[207] O. Osoba and B. Kosko. Noise-enhanced clustering and competitive learning algorithms. *Neural Networks*, 37, 2013.

[208] L. Page. Patent 6285999: Method for node ranking in a linked database, 1997.

[209] F. Papadopoulos, M. Kitsak, M. Á. Serrano, M. Boguná, and D. Krioukov. Popularity versus similarity in growing networks. *Nature*, 489(7417), 2012.

[210] A. Paranjape, A. Benson, and J. Leskovec. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017.

[211] V. Pareto. *Cours d'économie politique*, volume 1. Librairie Droz, 1964.

[212] S. Parthasarathy, Y. Ruan, and V. Satuluri. Community discovery in social networks: Applications, methods and emerging trends. In *Social Network Data Analytics*. Springer, 2011.

[213] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[214] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2011.

[215] L. Peel, D. B. Larremore, and A. Clauset. The ground truth about metadata and community detection in networks. *Science Advances*, 3(5), 2017.

[216] R. Peng, H. Chen, and P. K. Varshney. Noise-enhanced detection of microcalcifications in digital mammograms. *IEEE Journal of Selected Topics in Signal Processing*, 3(1), 2009.

[217] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. A review of novelty detection. *Signal Processing*, 99, 2014.

[218] P. Pons and M. Latapy. Computing communities in large networks using random walks. In *International Symposium on Computer and Information Sciences*. Springer, 2005.

[219] M. Porter, J. Onnela, and P. Mucha. Communities in networks. *Notices of the AMS*, 56(9), 2009.

[220] T. Pourhabibi, K. Ong, B. H. Kam, and Y. L. Boo. Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems*, 133:113303, 2020.

[221] D. S. Price. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American Society for Information Science*, 27(5), 1976.

[222] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences*, 101(9), 2004.

[223] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3), 2007.

[224] V. N. Rajapakse, W. Czaja, Y. G. Pommier, W. C. Reinhold, and S. Varma. Predicting expression-related features of chromosomal domain organization with network-structured analysis of gene expression and chromosomal location. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, 2012.

[225] E. Raju, Y. Ramadevi, and K. Sravanthi. Cilpa: a cohesion index based label propagation algorithm for unveiling communities in complex social networks. *International Journal of Information Technology*, 10(4), 2018.

[226] D. Ramalingam and V. Chinnaiah. Fake profile detection techniques in large-scale online social networks: A comprehensive review. *Computers and Electrical Engineering*, 65, 2018.

[227] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000.

[228] E. Ravasz and A. Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(2), 2003.

[229] S. Rayana. Odds library. `http://odds.cs.stonybrook.edu`, 2016.

[230] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1), 2006.

[231] M. Riveiro, G. Pallotta, and M. Vespe. Maritime anomaly detection: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(5), 2018.

[232] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4), 2008.

[233] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K. Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 2021.

[234] V. Satuluri, S. Parthasarathy, and Y. Ruan. Local graph sparsification for scalable clustering. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. ACM, 2011.

[235] C. Savkli, J. R. Carr, P. Graff, and L. Kennell. Bayesian learning of clique tree structure. In *Proceedings of the International Conference on Data Mining (DMIN)*. The Steering Committee of The World Congress in Computer Science, 2016.

[236] C. Savkli, R. Carr, M. Chapman, B. Chee, and D. Minch. Socrates: A system for scalable graph analytics. In *IEEE High Performance Extreme Computing*, 2014.

[237] C. Savkli, J. Lin, P. Graff, and M. Kinsey. Galileo: A generalized low-entropy mixture model. *Proceedings of the 2017 International Conference on Data Mining*, 2017.

[238] C. Savkli and C. Schwartz. Random subspace mixture model for interpretable anomaly detection. *The 23rd International Conference on Artificial Intelligence*, 7, 2021.

[239] C. Savkli, C. Schwartz, A. Galante, and J. D. Cohen. Novel edge and density metrics for link cohesion. 2019.

[240] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1), 2007.

[241] M. T. Schaub, J. Delvenne, M. Rosvall, and R. Lambiotte. The many facets of community detection in complex networks. *Applied Network Science*, 2(1), 2017.

[242] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7), 2001.

[243] G. Schwarz. Estimating the dimension of a model. *Annals of statistics*, 6(2), 1978.

[244] L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery. mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1), 2016.

[245] T. Seidl, E. Müller, I. Assent, and U. Steinhausen. Outlier detection and ranking based on subspace clustering. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

[246] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3), 1983.

[247] S. B. Seidman and B. L. Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6(1), 1978.

[248] C. Seshadhri, T. G. Kolda, and A. Pinar. Community structure and scale-free collections of Erdős-Rényi graphs. *Physical Review E*, 85(5), 2012.

[249] S. Shai, N. Stanley, C. Granell, D. Taylor, and P. J. Mucha. Case studies in network community detection. In *The Oxford Handbook of Social Networks*. 2017.

[250] H. Shakeri, P. Poggi-Corradini, N. Albin, and C. Scoglio. Network clustering and community detection using modulus of families of loops. *Physical Review E*, 95(1), 2017.

[251] N. Singh, B. A. Miller, N. T. Bliss, and P. J. Wolfe. Anomalous subgraph detection via sparse principal component analysis. In *2011 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE, 2011.

[252] J. Sipple. Interpretable, multidimensional, multimodal anomaly detection with negative sampling for detection of device failure. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[253] S. N. Soffer and A. Vazquez. Network clustering coefficient without degree-correlation biases. *Physical Review E*, 71(5).

[254] C. J. Stam. Modern network science of neurological disorders. *Nature Reviews Neuroscience*, 15(10), 2014.

[255] H. Sun, J. Huang, J. Han, H. Deng, P. Zhao, and B. Feng. gskeletonclu: Density-based network clustering via structure-connected tree division or agglomeration. In *2010 IEEE International Conference on Data Mining*. IEEE, 2010.

[256] J. J. Sylvester. Chemistry and algebra, 1878.

[257] N. Takeishi. Shapley values of reconstruction errors of pca for explaining anomaly detection. In *2019 International Conference on Data Mining Workshops (icdmw)*. IEEE, 2019.

[258] H. Tankovska. Number of monthly active facebook users worldwide as of 4th quarter 2020. *Statista*, 1, 2021.

[259] Q. K. Telesford, K. E. Joyce, S. Hayasaka, J. H. Burdette, and P. J. Laurienti. The ubiquity of small-world networks. *Brain connectivity*, 1(5), 2011.

[260] H. Thimbleby. The directed chinese postman problem. *Software: Practice and Experience*, 33(11), 2003.

[261] E. K. Towlson, P. E. Vértes, S. E. Ahnert, W. R. Schafer, and E. T. Bullmore. The rich club of the c. elegans neuronal connectome. *Journal of Neuroscience*, 33(15), 2013.

[262] J. W. Tukey. The future of data analysis. *The Annals of Mathematical Statistics*, 33(1), 1962.

[263] M. Tumminello, T. Aste, T. Di Matteo, and R. Mantegna. A tool for filtering information in complex systems. *Proceedings of the National Academy of Sciences*, 102(30), 2005.

[264] A. Ukil, S. Bandyoapdhyay, C. Puri, and A. Pal. Iot healthcare analytics: The importance of anomaly detection. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*.

[265] L. M. Vaquero and M. Cebrian. The rich club phenomenon in the classroom. *Scientific Reports*, 3(1), 2013.

[266] A. Vázquez, R. Pastor-Satorras, and A. Vespignani. Large-scale topological and dynamical properties of the internet. *Physical Review E*, 65(6), 2002.

[267] I. Vragović and E. Louis. Network community structure and loop coefficient method. *Physical Review E*, 74(1), 2006.

[268] S. I. Vrieze. Model selection and psychological theory: a discussion of the differences between the akaike information criterion (aic) and the bayesian information criterion (bic). *Psychological Methods*, 17(2).

[269] P. Wagenseller, F. Wang, and W. Wu. Size matters: A comparative analysis of community detection algorithms. *IEEE Transactions on Computational Social Systems*, 5(4), 2018.

[270] D. Wang and Q. Liu. Nonlinear stein variational gradient descent for learning diversified mixture models. In *International Conference on Machine Learning*, 2019.

[271] N. Wang, Z. Jin, and X. Peng. Community detection with self-adapting switching based on affinity. *Complexity*, 2019, 2019.

[272] Charlie Warzel. Here's how easy it is to buy fake facebook profiles, 2018.

[273] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world'networks. *Nature*, 393(6684), 1998.

[274] Q. Wei, Y. Ren, R. Hou, B. Shi, J. Y Lo, and L. Carin. Anomaly detection for medical images based on a one-class classification. In *Medical Imaging 2018: Computer-Aided Diagnosis*, volume 10575.

[275] R. Weiner. Former cia officer kevin mallory found guilty of selling secrets to china. *The Washington Post*, 9, 2018.

[276] D. P. Widemann. *Dimensionality reduction for hyperspectral data*. PhD thesis, 2008.

[277] J. Wu, Y. Hou, Y. Jiao, Y. Li, X. Li, and L. Jiao. Density shrinking algorithm for community detection with path based similarity. *Physica A: Statistical Mechanics and its Applications*, 433, 2015.

[278] X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger. Scan: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[279] J. Yang and J. Leskovec. Structure and overlaps of ground-truth communities in networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(2), 2014.

[280] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1), 2015.

[281] J. Yang, J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *2013 IEEE 13th international conference on data mining*. IEEE, 2013.

[282] X. Yang, K. Huang, J. Y. Goulermas, and R. Zhang. Joint learning of unsupervised dimensionality reduction and gaussian mixture model. *Neural Processing Letters*, 45(3), 2017.

[283] Z. Yang, R. Algesheimer, and C. J. Tessone. A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, 6(1), 2016.

[284] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[285] W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4), 1977.

[286] H. Zhang, C. Zhou, X. Liang, X. Zhao, and Y. Li. A novel edge weighting method to enhance network community detection. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2015.

[287] Y. Zhao and M. K. Hryniewicki. Dcso: dynamic combination of detector scores for outlier ensembles. *ACM SIGKDD Workshop on Outlier Detection De-constructed*, 2018.

[288] Y. Zhao, Z. Nasrullah, and Z. Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96), 2019.

[289] S. Zhou and R. J. Mondragón. The rich-club phenomenon in the internet topology. *IEEE Communications Letters*, 8(3), 2004.

[290] S. Zhou and R. J. Mondragón. Structural constraints in complex networks. *New Journal of Physics*, 9(6), 2007.

[291] T. Zhou, Y. Lee, and G. Wang. Experimental analyses on 2-hop-based and 3-hop-based link prediction algorithms. *Physica A: Statistical Mechanics and its Applications*, 564, 2021.

[292] A. Zimek, R. J. Campello, and J. Sander. Ensembles for unsupervised outlier detection: challenges and research questions a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1).

[293] A. Zimek and P. Filzmoser. There and back again: Outlier detection between statistical reasoning and data mining algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(6), 2018.

[294] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.