A Novel Method for Multilevel Autonomous Clustering (MAC) for Anomaly Detection in Distributed Systems

by

Mira Anita Partha

S.B., Massachusetts Institute of Technology (2019)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2020

Author . . . .

Department of Electrical Engineering and Computer Science

May 12, 2020

Certified by . . . .

Marija D. Ilić

Senior Research Scientist, MIT LIDS

Thesis Supervisor

Certified by . . . .

Colin V. Ponce

Computational Mathematician, Lawrence Livermore National Laboratory

Thesis Supervisor

Accepted by . . . .

Katrina LaCurts

Chair, Master of Engineering Thesis Committee

A Novel Method for Multilevel Autonomous Clustering (MAC) for Anomaly Detection in Distributed Systems

by

Mira Anita Partha

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 2020, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Anomaly detection in networks is crucial to detecting security threats. Network anomalies are often not localized to a single point, but spread over a range of nodes. In this case of distributed anomalies, the anomalies are typically too subtle to detect at an individual-node level, and so require examining groups of nodes together. But it is usually not known in advance on which subset of nodes to focus; and it is infeasible to check all 2^N subsets of nodes in a network. This renders distributed anomaly detection extremely challenging. An emerging strategy for detecting such anomalies is to apply a detection technique to a hierarchy of clusters of nodes in the network. However, developing such a hierarchy is challenging in large, decentralized networks with no central controller. Here, we present Multilevel Autonomous Clustering (MAC), a novel local algorithm for self-organized, hierarchical clustering in distributed networks. MAC enables individual devices in a distributed system to determine their cluster membership at multiple levels, without centralized computation or information about the entire network. The result is an approach to hierarchical network clustering that is both practical to use in large, real-world systems, as well as effective for distributed anomaly detection. The algorithm is evaluated on both synthetic and real-world networks. Its effectiveness for anomaly detection is demonstrated on various test problems. In particular, we examine the MAC algorithm's effectiveness for anomaly detection in electric power systems. Utilizing power flow balance equations, we generate anomalies that satisfy power conservation laws (and are therefore difficult to detect by normal means). Using MAC to cluster these power networks, we apply hierarchical anomaly detection on the resultant clusters.

Thesis Supervisor: Marija D. Ilić
Title: Senior Research Scientist, MIT LIDS

Thesis Supervisor: Colin V. Ponce
Title: Computational Mathematician, Lawrence Livermore National Laboratory

Acknowledgments

To Lawrence Livermore National Laboratory (LLNL), which has become a second home to me since I first joined the lab in 2016, I owe an infinite amount of gratitude. I conducted this research at LLNL under MIT's VI-A program. Thank you to Beth McCormick (Engineering Directorate, LLNL) and Kathleen Sullivan (MIT VI-A), without whom this could not have been possible.

For my mentors and colleagues in Global Security's Cyber and Infrastructure Resilience Group, as well as my colleagues in LLNL's Center for Applied Scientific Computing, I am eternally grateful. Thank you to Jovana Helms, for welcoming me to E Program and encouraging me in my academic and professional endeavors. Thank you to Eisha Nathan and Tim La Fond, who provided feedback and suggestions for improving the MAC algorithm. A huge thank you to Alyson Fox, whose endless positivity, encouragement, and technical insight not only improved this work, but made me a better researcher. Many thanks as well to my mentors and colleagues in the Computational Electromagnetics Group - Sean Lehman, Joseph Koning, Robert Ferencz, and Vanessa Gucho - for their unending support.

At MIT, I am tremendously grateful to my thesis advisor, Marija Ilic, who has supported me steadfastly these past two years. It has been an incredible privilege to work with her and learn from her - not only through our meetings and conversations, but also in her graduate course at MIT, 6.247 (Principles of Modeling, Simulations, and Control of Electric Power Systems). It was in this course that I learned to connect my research on network clustering, to electric power systems. I am so blessed to have been her student and advisee. I also owe so much to the patient tutelage of Rupamathi Jaddivada, my 6.247 TA. Thanks also to Le Xie, John Tsitsiklis, and all members of the Electric Energy Systems Group in MIT's Laboratory for Information and Decision Systems, for guiding the direction of my research. Many thanks as well to my academic advisor, Peter Hagelstein, for supporting me throughout my time at MIT.

To my family - my mom, dad, grandparents, and my sister Nina - thank you for unconditionally loving me, teaching me, and encouraging me all these years. I would not be here without you. To my best friends - Sunayana Rane, Rachel Green [née Levy], and René Garcia - thank you for sticking with me every step of the way.

Finally, this thesis would not exist without the expert guidance of my advisor at LLNL, Colin Ponce. As the principal investigator on the Time Warp project, Colin helped me determine my research focus, first from detection of GPS spoofing attacks on the power grid, to the work on hierarchical network clustering for anomaly detection presented here. It was during our weekly meetings, calls, and countless emails, that this work began to take shape; and it was on the whiteboard in his office, that the MAC algorithm was first formulated. Beyond shaping this work, Colin has also had tremendous impact on my academic trajectory, from my first conference paper to my imminent pursuit of a PhD; and beyond this still, he has become a cherished mentor and friend. (It is apt, therefore, that the 'MAC' acronym for our algorithm also stands for 'Mira and Colin'!)

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Contents

List of Figures

Chapter 1

Introduction

Today, distributed systems are everywhere, with a wide variety of applications ranging from electric power systems to the World Wide Web. With their ubiquity, it becomes increasingly critical for distributed systems to be able to rapidly detect unusual behavior. Anomalies may occur as the result of physical events, malfunctioning equipment, or even malicious cyber-attacks. Distributed anomalies, which impact anywhere from tens to thousands of nodes, have the potential to cause serious damage to a system if they remain undetected and unmitigated. If a distributed anomaly is locally very subtle - that is, the measured data appears very close to normal - such an anomaly will be difficult to detect by any individual device on the network. On the other hand, if the number of impacted devices is small relative to the size of the network as a whole, anomalies will be missed when considering large swaths of the network, or the network in its entirety. Therefore, to effectively detect a distributed anomaly, one needs to focus on the subset of nodes on which the anomaly actually exists, or a subset with high Jaccard similarity. As this subset is not known a priori, this is quite challenging.

An emerging strategy to detect such anomalies is hierarchical anomaly detection [4, 5, 8, 10, 13, 14, 19, 20]. One can apply a certain anomaly detection technique or ensemble of techniques starting with individual nodes, and then to successively larger and larger groups of nodes, moving through a full hierarchy of groupings. The purpose of this method is to provide a variety of intermediate granularities at which an anomaly that is not detectable on a smaller or larger scale may be found.

This approach works well because anomalies typically impact closely-related groups of devices. Therefore, in order to use hierarchical anomaly detection, we require a method of organizing devices in a distributed system into a hierarchy of clusters that captures these relationships. This differs from the task of simply finding a single level with the most ‘natural’ communities, or from the task of finding cluster(s) around a single vertex or set of seed vertices; instead, we want to obtain clusterings of the entire network at a multitude of levels, from individual devices all the way up to a single cluster containing the entire network.

An additional challenge that must be addressed in creating effective hierarchies of clusters in real networks is that many networks are both highly decentralized and constantly changing. For this reason, gathering an entire network topology in a single, central location and performing clustering analysis there is generally not feasible. Instead, practical hierarchical clustering requires a decentralized approach; this enables hierarchical clustering to proceed in a manner that befits the network’s control structure, and also enables easier altering of the clustering as the topology changes over time.

In this work, we propose Multilevel Autonomous Clustering (MAC): a local, decentralized algorithm for hierarchical clustering of distributed networks that is highly effective for hierarchical anomaly detection. Chapter 2 reviews related work on anomaly detection in wireless sensor networks, and provides references on graph clustering. Section 2 of Chapter 2 discusses our specific problem setting, as guided by the motivating application of this work, and lays out operating constraints. In Chapter 3, our algorithm is presented with its intuitive interpretation, its formal derivation, and its tuning parameters. In Chapter 4, we demonstrate the performance of the MAC algorithm on various networks (both synthetic and real-world), with regards to measures of cluster fitness. In Chapter 5, we apply the clusterings produced by the MAC algorithm to the problem of anomaly detection, comparing it with the Louvain modularity algorithm, a well-known algorithm for global, hierarchical clustering. In Chapter 6, we revisit our motivating application of electric power systems, and briefly note how MAC can be used for anomaly detection in such topologies. Lastly, Chapter 7 concludes with general discussion of the MAC algorithm, as well as extensions for future work.

Chapter 2

Context and Background

Behavior-based threat detection is a growing trend within network cybersecurity [9]. In behavior-based detection methods, rather than searching for signatures of specific attackers or kinds of attacks, the network is monitored for anomalous behavior on individual nodes or subsets of nodes. In large, distributed networks, it becomes necessary to delegate much of the responsibility for anomaly detection to individual nodes. This necessitates the hierarchical approach to anomaly detection previewed in Chapter 1.

2.1 Related Work

We begin by examining prior work in the area of hierarchical anomaly detection. Such research has been conducted primarily in the application area of wireless sensor networks, and is typically approached from the perspective of network communications. The use of graph theory to tackle the problem of anomaly detection in distributed networks, however, has rarely been investigated. We therefore briefly pivot to studying the fundamentals of graph theory, and graph clustering in particular, which will provide the necessary background to appreciate the novel Multilevel Autonomous Clustering Algorithm presented in this work.

2.1.1 Anomaly Detection in Wireless Sensor Networks

In the context of wireless sensor networks, there is some work exploring frameworks for hierarchical anomaly detection [4,5,8,10,13,14,19,20]. These frameworks generally fall into a few different categories: one, they assume an already hierarchical network architecture, consisting of sources and sinks on multiple levels; two, they rely on the deployment of a minority of more powerful nodes to act as 'cluster heads,' to aggregate and analyze data; or three, they propose a dynamic scheme for rotating the duty of cluster head among relatively homogeneous nodes. The construction of hierarchical architectures, and specifically the employment of graph clustering algorithms to construct such architectures, for the purpose of recursively applying anomaly detection methods is unexplored.

2.1.2 Graph Clustering

Graph clustering is a well-explored topic. The goal of clustering is to partition a graph into groups of nodes, where nodes sharing a cluster are densely connected, and nodes in separate clusters are sparsely connected. While cluster analysis is an NP-hard problem, a number of algorithms have been proposed to obtain reasonable clusterings. These algorithms fall broadly into three categories: divisive algorithms, in which the graph is recursively split into smaller and smaller clusters (these methods are, by necessity, global); agglomerative algorithms, which begin with each node in its own cluster, and then recursively merge clusters to form larger and larger groups; and optimization methods, which seek to maximize a previously defined objective function such as modularity.

The work of Bagrow and Bollt [1], Clauset [6], Schaeffer [15–17], and Blondel, et al. [2], provides insight into existing algorithms for graph clustering, both local and global.

2.2 Problem Setting

In order to understand the specific operating constraints that our solution is required to satisfy, and to understand why existing graph clustering algorithms fail to meet these needs, we must first examine our motivating application, which is to detect GPS spoofing attacks on the power grid.

2.2.1 Motivating Application

Conventional graph clustering algorithms have been highly successful in a variety of domains. However, we are motivated by the need to detect anomalies on decentralized, cyber-physical systems such as the electric grid. In particular, this work was conducted as part of an effort to detect (and subsequently mitigate) GPS spoofing attacks on the distribution power grid.

In the transmission power grid, it has long been accepted that safety, security, and reliability are paramount; the consequences of failure of the transmission grid can be catastrophic. Accordingly, electric utility companies are willing to invest in equipment and security solutions for the transmission grid that are expensive, both financially and computationally, in order to mitigate risk. The distribution grid, however, has not been treated with the same consideration. Only recently, as distributed energy resources become widespread, and more and more "smart" (IoT) devices (such as smart solar inverters and microPMUs) are brought online, has the security of devices on the distribution grid also emerged at the forefront of infrastructure resilience. As techniques for grid analysis advance, it has become apparent that attacks directly targeting the distribution grid also have the potential for dire consequences, such as the loss of considerable electric power produced by distribution-level sources (like solar power), damaged equipment, and even cascading power failures.

Recent events in Ukraine and other areas around the world have demonstrated that malicious attacks against electric grids and other infrastructure are becoming increasingly frequent threats to national security. GPS spoofing is one such kind of malicious attack. It has considerable potential to cause widespread harm, as it requires minimal equipment and knowledge, but can easily destabilize location and timing systems. As growing numbers of devices on the distribution grid turn to GPS as an accurate, low-cost source of precision timing, the distribution grid becomes an increasingly vulnerable target for adversaries seeking to damage the synchronization of electric power systems by spoofing GPS signals.

While certain kinds of GPS spoofing attacks, such as intrusion or jamming attacks, are easy to detect, sophisticated attacks, such as meaconing attacks, are much more difficult to defend against. In meaconing attacks, devices are tricked into locking onto very gradually shifting timing signals. These spoofed signals stray off the correct time so slowly, that systems may not recognize that they are under attack until it is too late to prevent damage. At any point during a meaconing attack, devices on the grid are likely to read measurements that are very close to normal, and well within reason given recent readings. By an individual device, this would be difficult to recognize, even using the most sophisticated moving-average time-series models.

Our goal, therefore, is to develop the capability to detect such extremely subtle attacks on networks like the distribution grid. Because of cost constraints and the decentralized nature of the distribution grid, we want to develop detection methods that can be easily performed by individual devices on the network, using the principles of collaborative autonomy and hierarchical anomaly detection. With this broad vision in mind, we can now model our specific problem setting, and determine its operating constraints.

2.2.2 Problem Constraints

We consider networks of heterogeneous, low-power devices (such as network routers or power grid sensors) that must autonomously organize into clusters rather than relying on a centralized data acquisition and control system. In such networks, a single device will not be aware of the complete network topology; rather, a device will typically only be aware of its neighboring devices. Global graph-clustering algorithms require a fully available graph. In our setting, however, we require devices to self-organize into clusters based on only partial availability of the full system. Thus, we need a decentralized, local clustering algorithm.

Additionally, as devices may come online and go offline at any time, we must account for a dynamic network topology. Instead of recalculating the cluster hierarchy for each change, our algorithm should easily, but intelligently, adapt to changes in the network. Given our expectation of computationally constrained devices, and a possibly highly latent or unreliable communications network, we require an algorithm that relies solely on relatively simple calculations, which unknown individual devices can reasonably be expected to perform. In the following sections, we endeavor to demonstrate that the novel Multilevel Autonomous Clustering (MAC) aglorithm satisfactorily addresses this tradeoff between accuracy and computational cost.

Chapter 3

MAC: Multilevel Autonomous Clustering

Here, we propose the MAC algorithm for multilevel autonomous clustering. It proceeds essentially in three steps. First, each node computes a local affinity score, dubbed ω, with each of its neighbors. Second, each node decides, based on this affinity score, which of its neighbors it believes it shares a community with. Third, the nodes collaboratively determine, based on the nodes' individual community beliefs, which nodes to finally group into communities. One can consider MAC through either a node-based interpretation, which is perhaps more intuitive, or an edge-based interpretation, which enables application to weighted graphs.

3.1 A Node-Based Interpretation

First, to calculate its affinity score with each neighbor, a node computes the Jaccard similarity, ω, of its neighborhood with the neighborhood of each of its neighbors:

$$\omega(v, w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{|\Gamma(v) \cup \Gamma(w)|}$$

where $\Gamma(v)$ denotes the neighborhood of node v, including v itself. Note that nodes v and w are counted as shared neighbors, so the numerator has a minimum value of two.

Second, to decide on its community beliefs, the node computes the arithmetic mean of ω over all its neighbors:

$$\bar{\omega}(v) = \frac{1}{|\Gamma(v)|} \sum_{\omega \in \Gamma(v)} \omega(v, w)$$

and sets this as its ω-threshold. For each neighbor w in $\Gamma(v)$, node v believes it shares a community with node w if:

$$\omega(v, w) \geq \bar{\omega}(v)$$

Finally, given each node's beliefs, each connected pair of nodes collectively decide whether or not they truly belong in a community together using one of two algorithmic cases, the bilateral ('AND') case and the unilateral ('OR') case. If the user chooses to enforce reciprocity, in the bilateral case, then both nodes are required to agree that they share a community in order for them to be assigned to the same cluster. If the user does not require reciprocity, the unilateral case allows two nodes to be assigned to the same cluster if at least one of them believes it shares a community with the other.

To summarize, the conditions for final cluster assignments in the two cases are as follows:

Bilateral ('AND'):

$$\omega(v, w) \geq \bar{\omega}(v) \wedge \omega(v, w) \geq \bar{\omega}(w)$$

Unilateral ('OR'):

$$\omega(v, w) \geq \bar{\omega}(v) \vee \omega(v, w) \geq \bar{\omega}(w)$$

Community inclusion is transitive, so if v and w agree to share a community, and w and u agree to share a community, then v, w, and u all belong to the same community. This describes how clustering is decided at a single level.

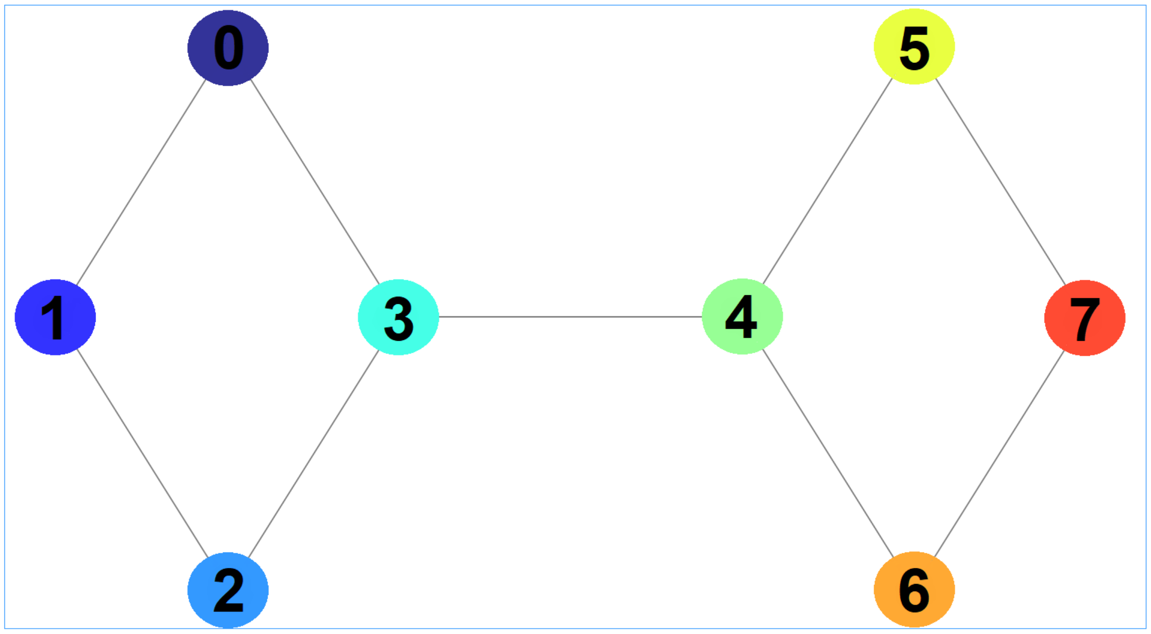


Figure 3-1: A simple graphical example for calculating ω: Consider node 3. Node 3 calculates an ω of 0.4 with nodes 0 and 2, and an ω of $0.\bar{3}$ with node 4. Thus, it believes it shares a cluster with nodes 0 and 2.

To produce a hierarchy of clusters, we simply coarse-grain the network from one level to the next; that is, the clusters that are formed at one level become single nodes in a new network for the next level. The algorithm is applied again to the new graph, producing new clusters. These two steps are repeated to obtain a full hierarchy of clusters. The hierarchy culminates with the entire network included in a single cluster.

While in this work we demonstrate the mathematical properties of the MAC algorithm using a serial implementation, MAC is designed to operate as a decentralized algorithm on distributed networks. In a decentralized implementation, the computation of nodes (individual devices) to determine their lowest-level cluster membership remains the same. For higher levels of the hierarchy, we envision all nodes in each coarse-grained cluster collaborating to determine their cluster's neighborhood Jaccard similarity with other clusters on the same level. This would entail message-passing between nodes within a coarse-grained cluster, in addition to communication between neighboring nodes on cluster boundaries. Computation could be carried out by a single nominated 'cluster head' that aggregates cluster-wide information; however, we prefer a decentralized computational approach, wherein all nodes within a cluster conduct this computation collaboratively. While this means that, at a higher level in the hierarchy, a single node may be utilizing more information than just its one-hop neighbors to compute higher-level cluster information, each cluster – a single coarse-grained node in the new network – determines its Jaccard similarity to other clusters, using only information that is local to that cluster (that is, it does not use network-wide information); and computation is still conducted internally to the cluster, rather than with the aid of a centralized computer that oversees the entire network or is external to the cluster. Thus, we still consider the MAC algorithm to be a local, decentralized algorithm.

3.2 Extension to Weighted Graphs

To extend the node-based algorithm to handle weighted graphs, we derived an equivalent formulation using edge weights. In this formulation, ω is calculated as follows:

$$\omega(i,j) = \frac{2e_{i,j} + \frac{1}{2}\sum_{k \in N_i \cap N_j}(e_{i,k} + e_{j,k})}{\frac{1}{2}\sum_{k \in N_i \cap N_j}(e_{i,k} + e_{j,k}) + \sum_{k \in N_i \cap \bar{N_j}} e_{i,k} + \sum_{k \in N_j \cap \bar{N_i}} e_{j,k}}$$

This exactly matches the node-based algorithm described in Section 3.1.

This initial formulation succeeds on highly structured networks. In graphs with less significant community structure, however, this method can result in very uneven cluster sizes (one excessively large cluster, surrounded by singletons or small groups) at higher levels in the hierarchy. (This is a similar problem to the "rich get richer" behavior exhibited by agglomerative clustering; see Section 4.1.) We deduced that this problematic behavior was a result of disregarding clusters' internal weights upon coarse-graining.

To fully understand this problem, let us examine the problem of clustering at a hierarchical level greater than one. Illustratively, a node i connected to two nodes j and k, each representing a cluster that was coarse-grained from the previous level, would have no preference for one or the other if $x(i,j) = x(i,k)$, even if j represented a very dense cluster and k represented a very sparse cluster. Intuitively, however, cluster j is quite insular; so unless the coarse-grained node i has many mutual neighbors with the coarse-grained node j, good clustering would suggest that cluster i ought to prefer joining with cluster k over cluster j. The original formulation of ω discards this information about cluster density when coarse-graining the network, allowing for the formation of giant components like those that appear in the Erdős-Rényi random graph model [3]. Thus, in order to preserve the information regarding clusters' internal density and preclude the formation of a single large cluster[1], we modify the denominator of the formula for ω to normalize affinity based on internal community

[1]Also referred to as a cluster of unusual size.

weight. Our revised formulation of ω is as follows:

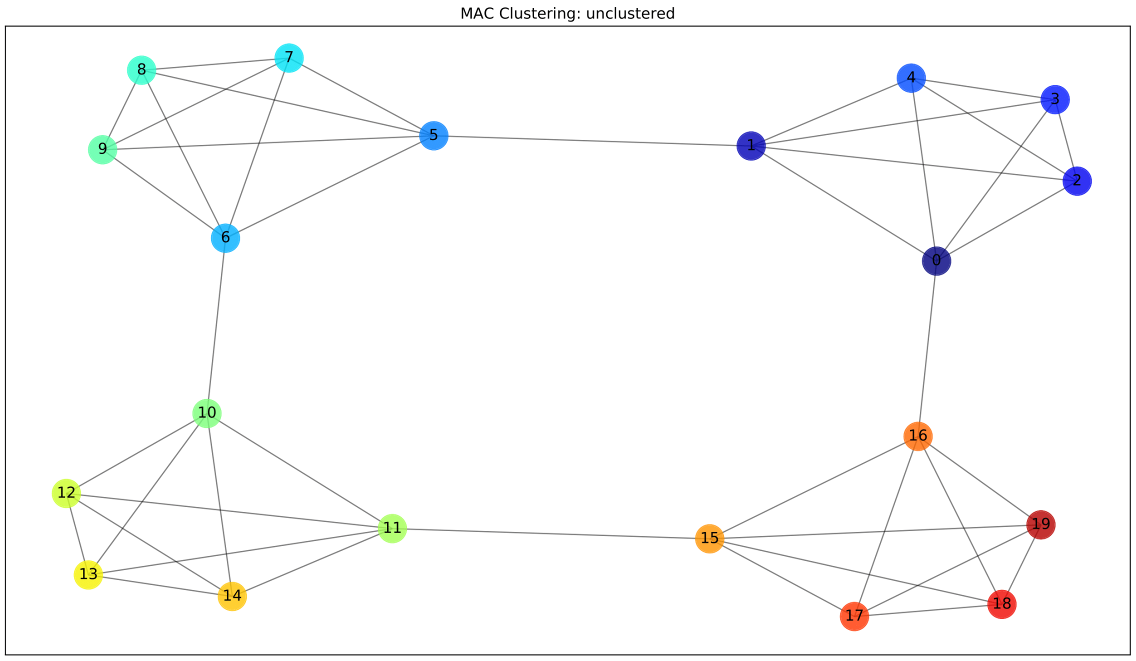
$$\omega(i,j) = \frac{2e_{i,j} + \frac{1}{2}\sum_{k\in N_i\cap N_j}(e_{i,k}+e_{j,k})}{a_i + a_j + \frac{1}{2}\sum_{k\in N_i\cap N_j}(e_{i,k}+e_{j,k}) + \sum_{k\in N_i\cap \bar{N_j}} e_{i,k} + \sum_{k\in N_j\cap \bar{N_i}} e_{j,k}}$$

where a_i denotes the sum of the weights of all edges internal to the cluster represented as node i in the coarse-grained network, or 0 if node i is a first-level node.

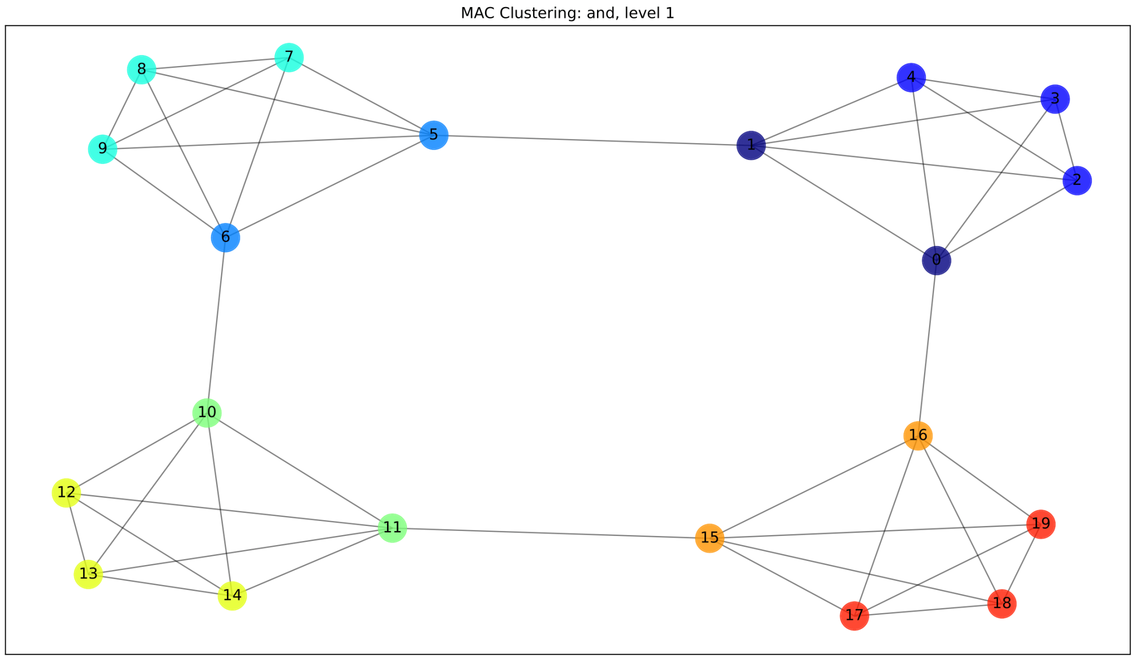
3.3 Tuning Parameters

The MAC algorithm includes a small number of tunable parameters.

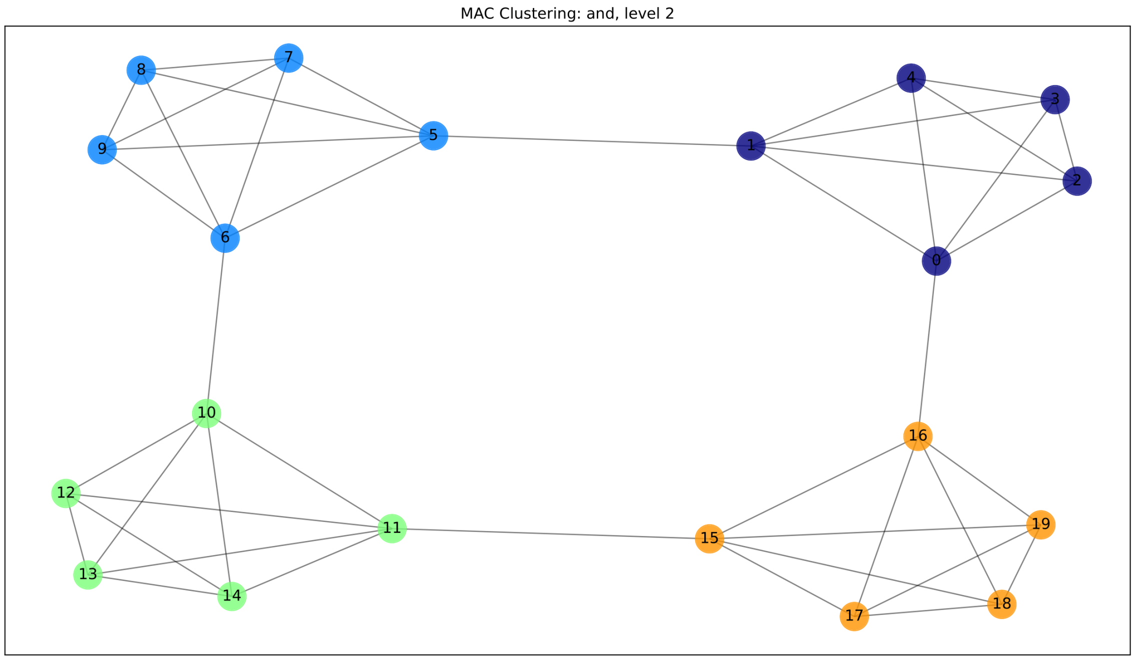
First, as discussed in Section 3.1, there are two agreement options, the bilateral ('AND') and unilateral ('OR') cases, which arise from the choice as to whether to require reciprocity in determining shared communities. The unilateral case, being a weaker threshold for allowing two nodes to share a community, naturally results in the formation of larger communities. The bilateral case yields many of the same clusterings as the unilateral case, but in an increased number of levels, as it additionally produces intermediate clustering steps that are not present in the unilateral case. As a result, the bilateral case carries a higher computational cost, and occasionally yields an insensible number of levels when attempting to cluster graphs with minimal community structure. However, since its hierarchy is frequently a superset of the hierarchy produced by the unilateral case, the bilateral case generally provides higher anomaly detection rates than the unilateral case. It is therefore set as the default for the MAC algorithm.



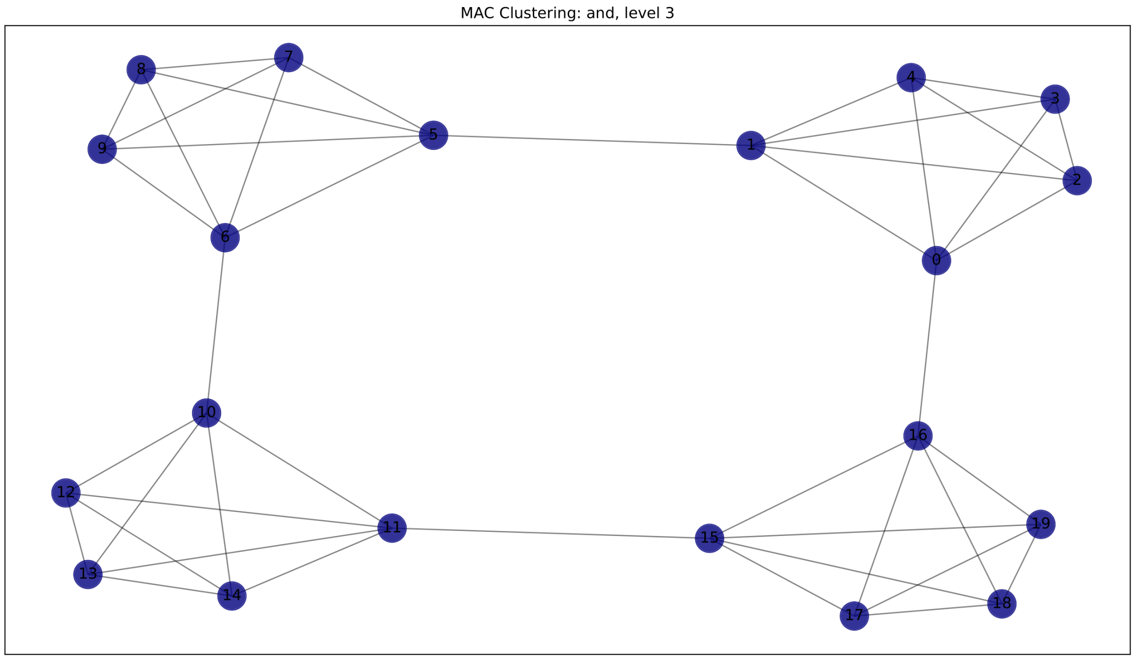
(a) Every node is in its own cluster.



(b) Each clique is split into two clusters: one of three nodes, and one of two nodes.



(c) Each clique is in its own cluster.



(d) The entire network is in a single cluster.

Figure 3-2: The hierarchy of clusters produced by bilateral MAC on a ring of cliques. The unilateral case is identical except for the omission of the clustering stage shown in figure (b).

Within the determination of an individual node's community beliefs, there is an optional parameter regarding a node's ω-threshold. Rather than setting the ω-threshold as $\omega(v, w) \geq \bar{\omega}(v)$, one can shift this threshold by a chosen factor, α, times the standard deviation of the ω values observed by that node. The ω-threshold for node v then becomes:

$$\omega(v, w) \geq \bar{\omega}(v) + \alpha \times \sigma_{\omega}(v)$$

A low value of α (below zero) allows nodes to join more easily, thus producing larger communities. As α increases above zero, the threshold for community formation becomes more stringent, resulting in a greater number of smaller communities. While one would reasonably expect α to vary between $[-3, 3]$ if ω values around a node are normally distributed, in practice we have found that ω values appear to belong to a lighter-tailed distribution. Thus, one typically selects α to be in the range $[-1.5, 1.5]$; values of α outside this range usually result in either all nodes joining a single cluster, or in no clusters forming at all. If a round sees no new cluster formation, α is presumed too high, and is subsequently reduced by 0.5. We typically initialize α with a value of zero. In applying MAC to real-world networks, studying the distributions of ω values around nodes may provide insight into the selection of the optimal α parameter for a given network architecture.

Alternatively, rather than setting an ω-threshold, a node can simply elect to share a community with its neighbors with whom it shares its k highest ω values, where k is a user-selected integer that is at least one. (This is independent of the 'AND' and 'OR' cases, which are still used to determine the final cluster assignments.) As expected, higher values of k lead to larger clusters.

It is important to note that, though these parameters are available for refining the performance of MAC on particular network architectures, the ability of the MAC algorithm to detect natural communities is reasonable with the default settings; parameter tuning is not required. Notably, specifying a known number of clusters is not necessary; nor is any global information about the network. Individual nodes, possessing only knowledge of their one-hop neighbors, can autonomously determine the clusters to which they belong. Working collaboratively in a decentralized fashion, the entire network can be clustered from the ground up. While these features enable MAC to be used in the constrained problem setting described in Section 2.2, they are not without consequence. In Chapters 4 and 5, we shall study the effect of these characteristics on clustering and anomaly detection performance.

Chapter 4

Clustering Performance on Test Graphs

4.1 Global Graph Clustering Algorithms for Comparison

Consider the ring of four cliques with five nodes each, shown as clustered by the MAC algorithm in Figure 3-2. We can examine the different clusterings of this same graph that are produced by five of the more well-known global clustering algorithms: k-means, agglomerative clustering, spectral clustering, affinity propagation, and the Louvain modularity algorithm.

The k-means algorithm [12] is typically used to cluster data. It does so by separating samples into groups of equal variance, while minimizing the within-cluster sum-of-squares criterion. To apply k-means to the problem of graph custering, the set median graph is used to represent the center of each cluster. K-means requires the number of clusters to be pre-specified.

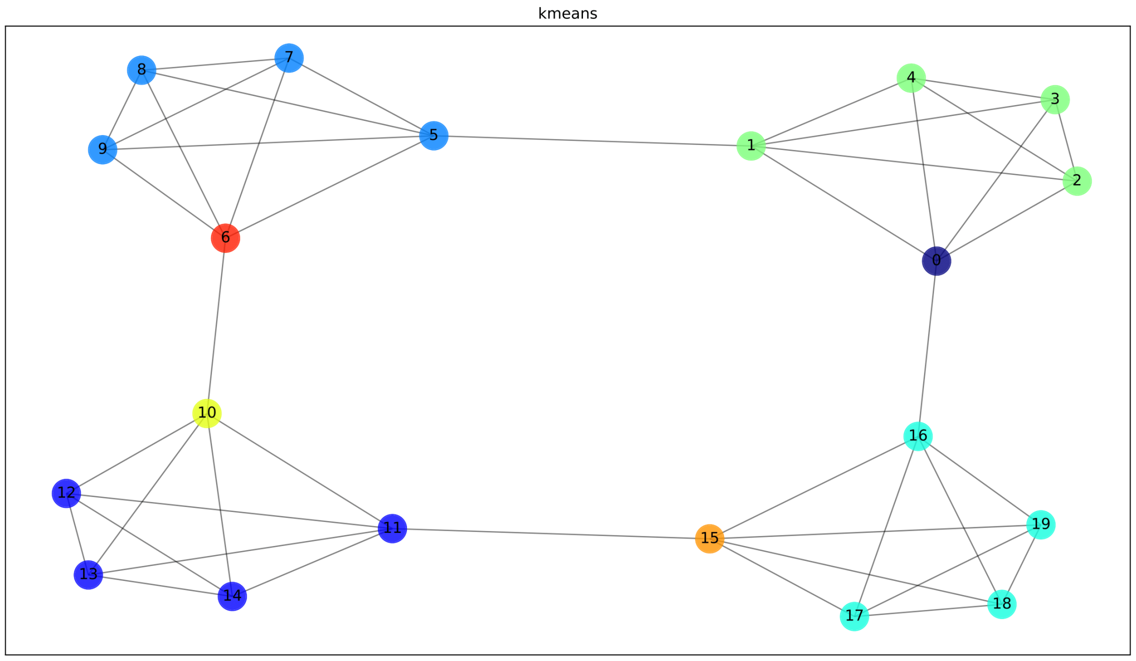
Agglomerative clustering [12] begins with each node in its own cluster, and then successively merges clusters together. Different criteria can be used to determine when and which clusters to merge; depending on the criteria chosen, it can be varyingly effective on different network topologies. Agglomerative clustering works well when used jointly with a connectivity matrix; but it has a high computational cost if no no connectivity constraints are given, because it considers all possible merges at each step. One pitfall of agglomerative clustering is that it tends towards a "rich get richer" behavior, which leads to uneven cluster sizes.

Spectral Clustering [12] clusters by embedding the affinity matrix between nodes in a low-dimensional space, and then grouping the components of the eigenvectors in the low-dimensional space. Like k-means and agglomerative clustering, spectral clustering requires the number of clusters to be pre-determined.

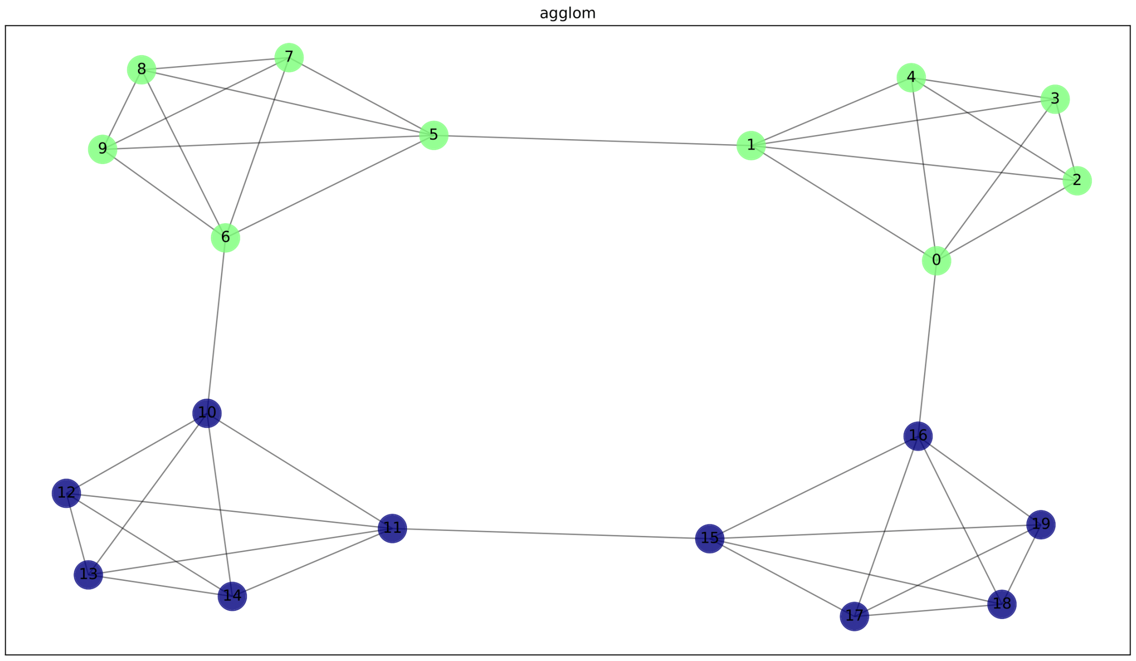
Affinity propagation [12] creates clusters by iteratively passing messages between pairs of nodes until it converges to a final clustering. Unlike k-means, spectral, or agglomerative clustering, affinity propagation chooses the number of clusters based on the network provided.

The Louvain method [2] is a global algorithm for detecting communities in networks. It works by recursively merging clusters in order to maximize the modularity scores for the final clusters. Modularity, a popular measure of cluster fitness, quantifies the quality of each node's current cluster assignment. It is calculated by evaluating how much more densely connected the nodes within a cluster are, compared to how connected they would be in a random network.

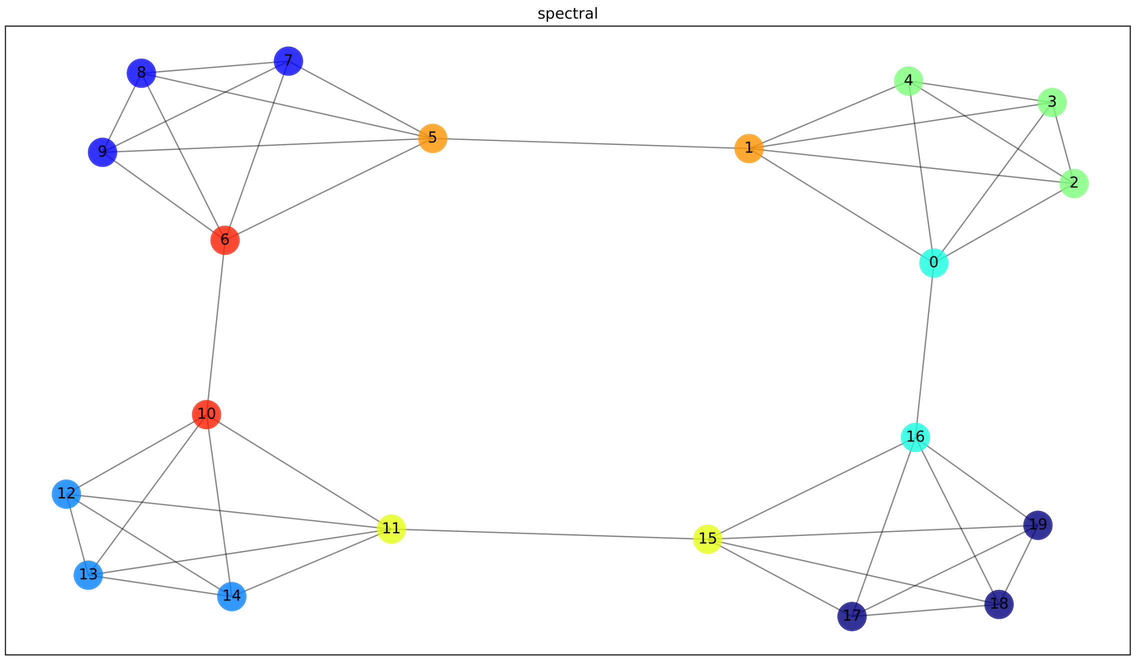
Unlike the MAC algorithm, all of these clustering algorithms are global: meaning that they require information about the entire network (information that would not be available to a single device on a distributed network in our problem setting), in order to compute cluster assignments. Additionally, most of them are not hierarchical algorithms; they produce only a single "optimized" clustering, rather than multiple clusterings which can be used for hierarchical anomaly detection.



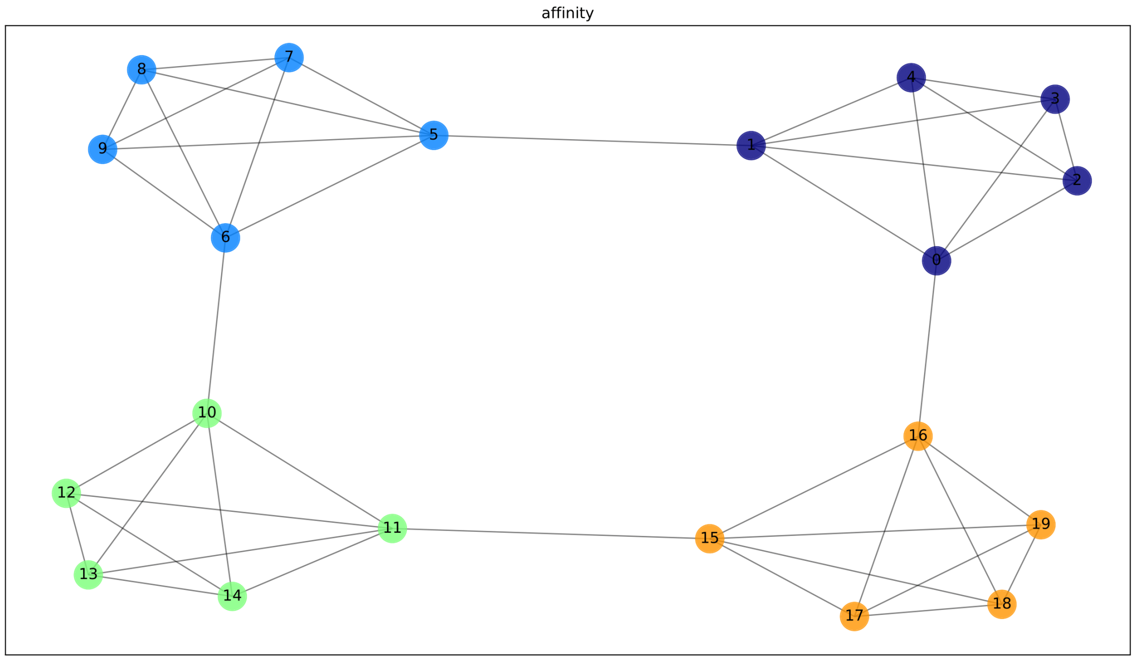
(a) Clustering produced by the k-means algorithm. Without a specified number of clusters, it resorted to its default value of 8 clusters (as in the scikit-learn implementation [12]).



(b) Clustering produced by agglomerative clustering. Without a specified number of clusters, it resorted to its default value of 2 clusters (as in the scikit-learn implementation [12]).



(c) Clustering produced by spectral clustering. Without a specified number of clusters, it resorted to its default value of 8 clusters (as in the scikit-learn implementation [12]).



(d) Both affinity propagation (implemented by scikit-learn [12]) and the Louvain modularity (implemented by python-louvain [2]) algorithms produced the same clustering.

Figure 4-1: Ring of cliques (4,5) as clustered by various global clustering algorithms.

4.2 Metrics for Measuring Clustering Performance

Clustering performance was quantitatively measured using two metrics: modularity, and the product of local and relative densities averaged over all clusters. Modularity, a widely used metric, is defined as follows:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{i,j} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

"where $A_{i,j}$ represents the weight of the edge between i and j, $k_i = \sum_j A_{i,j}$ is the sum of the weights of the edges attached to vertex i, c_i is the community to which vertex i is assigned, the δ-function $\delta(u, v)$ is 1 if $u = v$ and 0 otherwise, and $m = \frac{1}{2} \sum_{i,j} A_{i,j}$" [2].

Our other clustering metric is the objective function used by Schaeffer et al. The metric, $f(C)$, equals the product of the local density, $\delta_\ell(C)$, which is defined in terms of the internal degree of a cluster, and the relative density, $\delta_r(C)$, which is defined in terms of the internal and external degrees of a cluster. This clustering metric is detailed in "Stochastic Local Clustering for Massive Graphs" [17].

4.3 Compared Clustering Performance

Clustering performance of both the MAC algorithm and the global algorithms described in Section 4.1 has been studied for a variety of synthetic and real-world networks, containing anywhere from 20 to ~15,000 nodes, and ranging from 9 to ~90,000 edges. Performances were compared using the two metrics described in Section 4.2. Results on a ring of cliques are shown in Section 4.3.1, and results on a network modelling a 13,659 bus power system (from the University of Washington Power Systems Test Archive [21]) are shown in Section 4.3.2. The clustering approach (computing Jaccard similarity with other nodes/clusters to determine cluster membership beliefs) is the same for the MAC algorithm whether applied to synthetic or real-world networks.

4.3.1 Performance on Synthetic Graphs

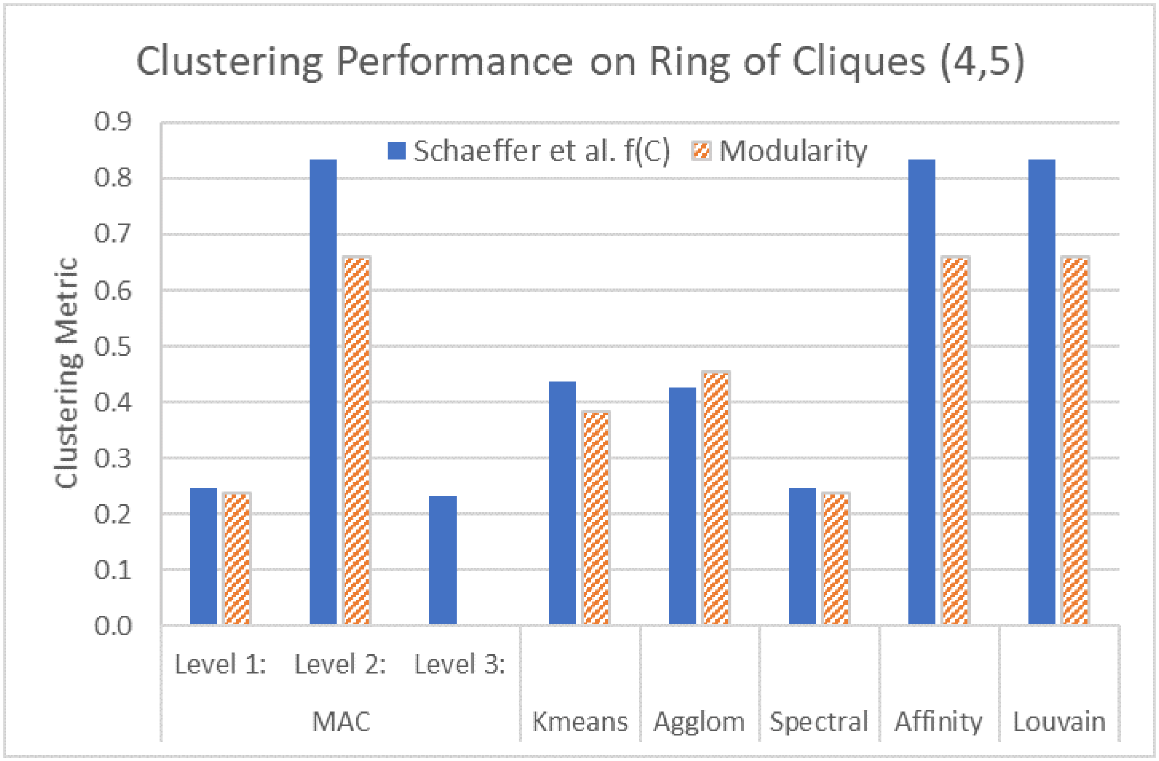


Figure 4-2: Quantitative clustering metrics compared across the full hierarchy of clusters produced by MAC, and clusterings produced by k-means, agglomerative clustering, spectral clustering, affinity propagation, and Louvain. MAC achieves the maximum scores on both clustering metrics at an intermediate clustering level. Note that MAC's modularity at level three of the cluster hierarchy is equal to zero.

4.3.2 Performance on Real-World Networks

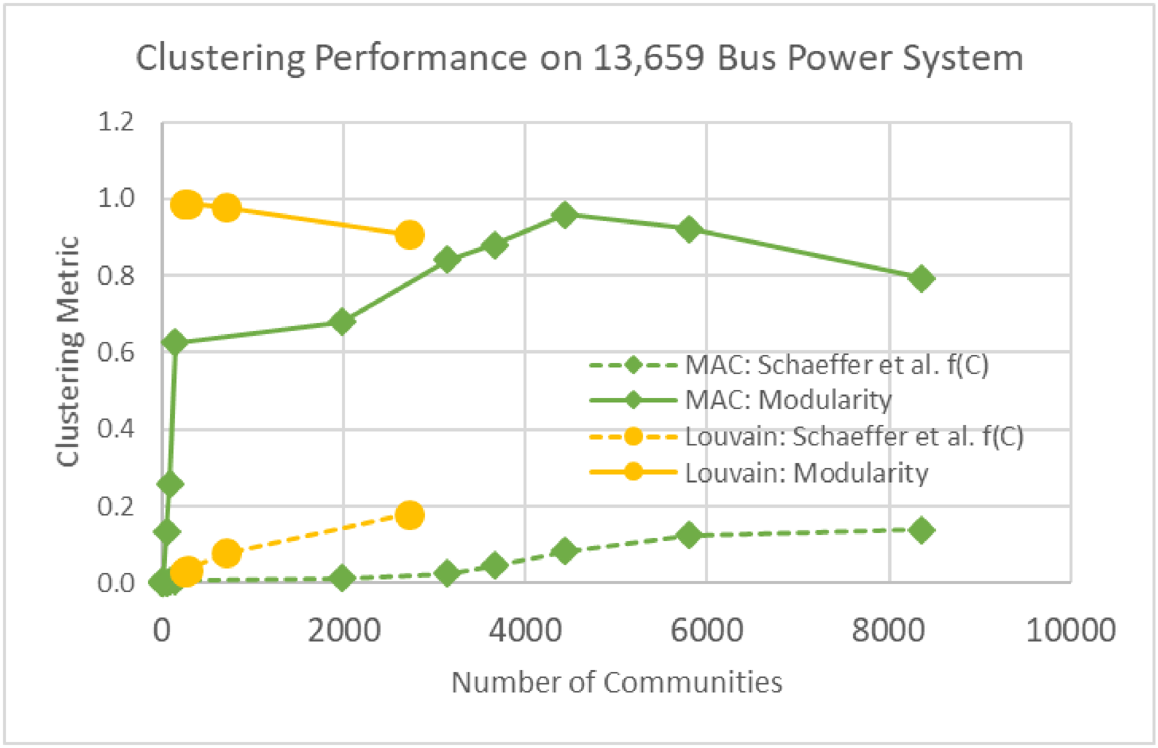


Figure 4-3: Clustering performance of MAC and hierarchical Louvain compared on a 13,659 bus power system. Both algorithms are agglomerative, so hierarchical clustering progresses from right to left in the plot.

4.4 Overall Clustering Results

Generally, MAC performs poorly on both clustering metrics at high and low granularities, but approaches the performance of global clustering algorithms at intermediate levels in the hierarchy.

Since the Louvain method is the global algorithm most analogous to the (local) MAC algorithm, it is of particular interest to compare the two in more detail. While MAC's intermediate modularity values approach those achieved generally by Louvain (a global, hierarchical algorithm which is designed to maximize modularity as its objective function), it produces qualitatively different clusterings. In MAC, the comparable modularity scores appear at levels with significantly more clusters than any clusterings produced by Louvain. Comparing levels with similar numbers of clusters, MAC's modularity scores are substantially lower than those achieved by Louvain; however, until the very highest levels in the hierarchy (wherein the entire network is partitioned into fewer than one hundred clusters), modularity is still well over 0.3, an accepted rule-of-thumb indicating that there is significant community structure [7].

While MAC's clustering performance is not as strong as that achieved by global algorithms, it is sufficient for the purpose of decentralized anomaly detection. This is demonstrated in Chapter 5.

Chapter 5

Benchmarking for Anomaly Detection

The effectiveness of the MAC algorithm for anomaly detection was compared against hierarchically applied Louvain, a global algorithm for graph clustering[14]. Various generated and real-world networks were clustered using both MAC and Louvain.

5.1 Creation and Detection of Anomalies

We employed a simple anomaly creation and detection method to abstractly represent anomalies and local detection algorithms. In practice, one could use any user-chosen local anomaly detection technique on a given cluster. Nodes in the network were modelled as sampling their data from a standard normal distribution (representing the output of device-level data and anomaly detection algorithms). A seed node chosen uniformly at random and all of its k-hop neighbors were "infected," drawing their data instead from a normal distribution with a shifted mean. This captures the idea that cyber intrusions spread from an entry point to neighboring nodes. To test for anomalies, each cluster produced by one of the two clustering algorithms used a statistical Z-test on the mean of its data to flag readings as either normal or anomalous. If a hierarchy correctly flagged a cluster containing one or more infected nodes on at least one level, a successful detection was marked. False positives, wherein a cluster that did not contain any infected nodes flagged an anomaly, were also recorded. For a variety of values of k and anomalous mean-shifts, a large number of Monte Carlo trials were run, to measure the detection rates and false positive rates produced by each algorithm. P-value thresholds were chosen in order to minimize Type I errors. Here, we utilized uncorrected p-values in flagging anomalies; however, to address the multiple comparisons problem that arises with testing hierarchically nested clusters, corrected p-values (using either the Holm-Bonferroni or the Benjamini-Hochberg correction) should be used. This would likely allow for the selection of higher p-value thresholds, lowering the rate of Type II errors.

5.2 Results

Anomaly detection results on the ring of cliques from Chapters 3 and 4 are shown in Section 5.3.1. Results of detection experiments on a real-world dataset of 10 ego-networks from Facebook (taken from the SNAP Network Library [18]) are shown in Section 5.3.2. The complete Facebook graph contains 4,039 nodes and 88,234 edges. Since social networks have vastly different graph characteristics from networks in other domains, results are also demonstrated on a 13,659 bus power system (from the University of Washington Power Systems Test Archive [21]).

5.2.1 Results on Generated Networks

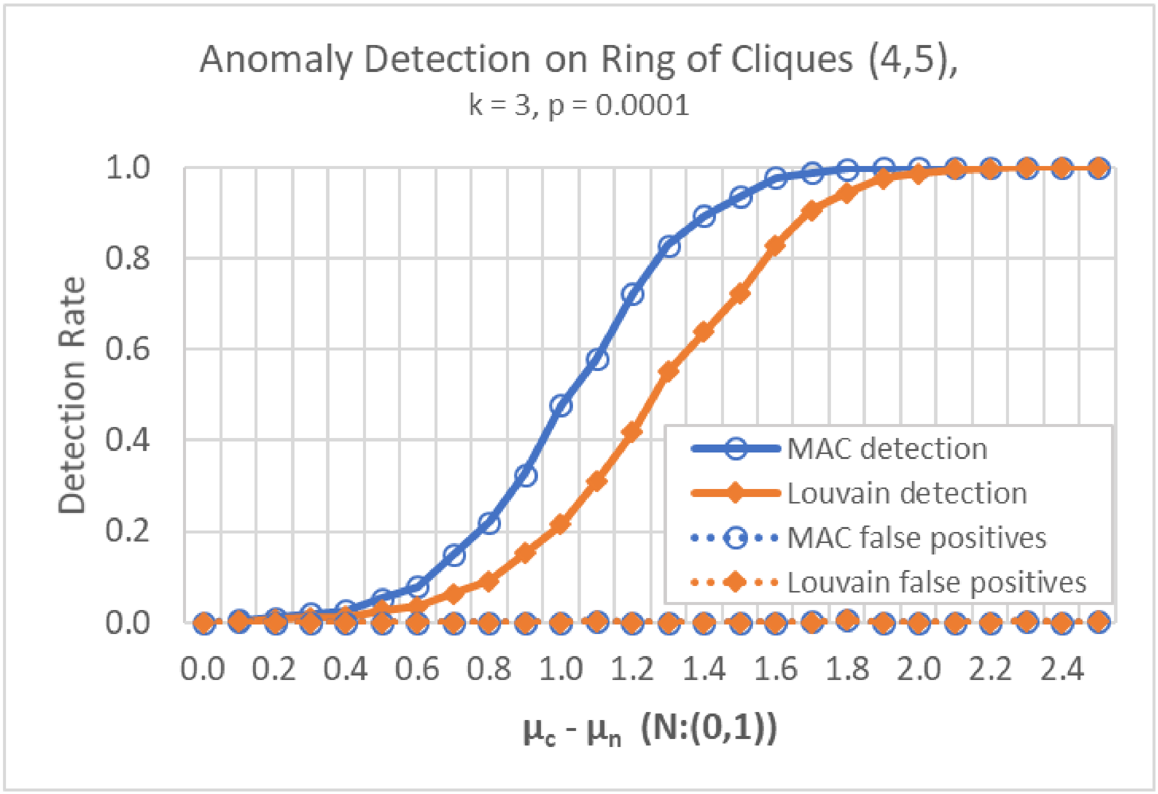
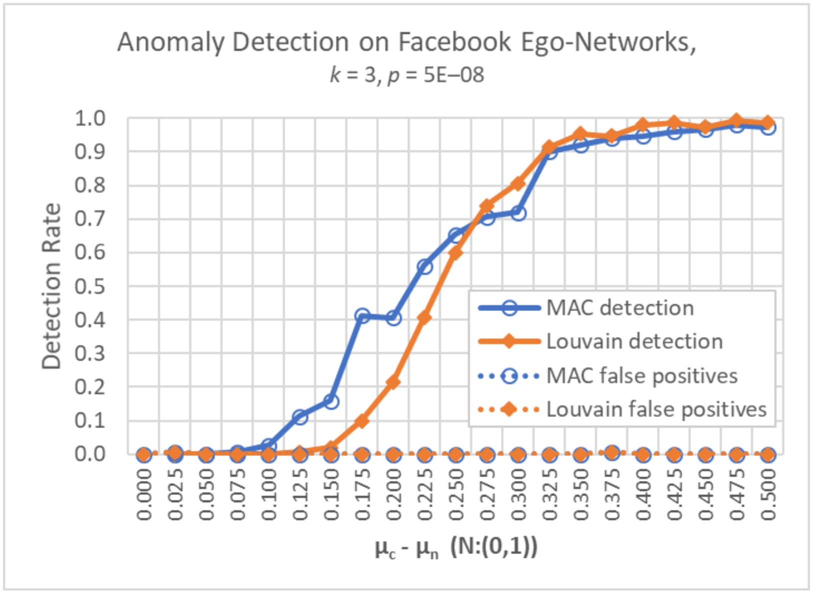
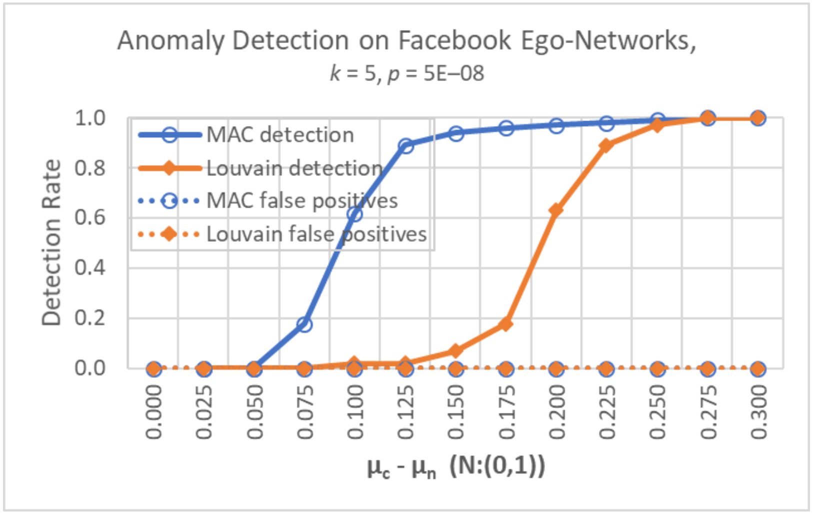


Figure 5-1: Hierarchical anomaly detection on a ring of cliques using the hierarchies of clusters produced by MAC and Louvain. This test was run with 1000 trials per mean-shift. With a value of $k = 3$, the anomaly covers approximately 75% of the network.

5.2.2 Results on Real-World Networks



(a) $k = 3$. The anomaly covers approximately 30% of the network.



(b) $k = 5$. The anomaly covers approximately 80% of the network.

Figure 5-2: Hierarchical anomaly detection on a set of 10 Facebook ego-networks using the hierarchies of clusters produced by MAC and Louvain, for $k = 3$ and $k = 5$. These tests were run with 150 trials per mean-shift.

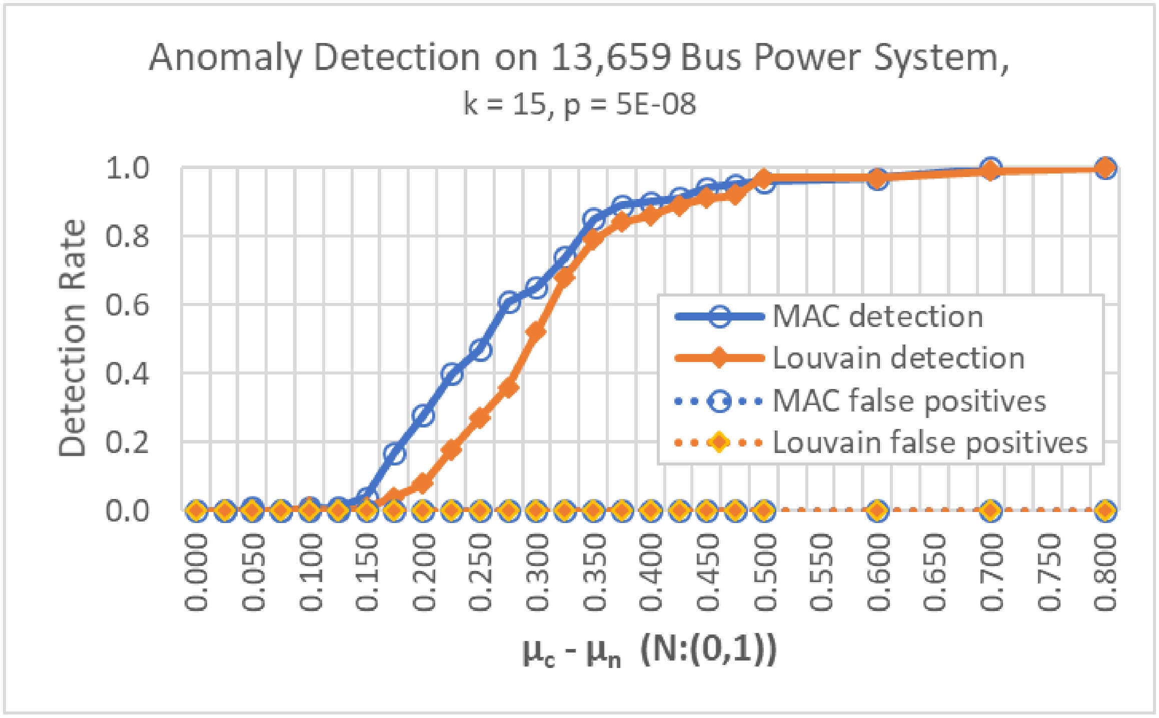


Figure 5-3: Hierarchical anomaly detection on a 13,659 bus power system, using the cluster hierarchies produced by MAC and Louvain, for $k = 15$. This test was run with 100 trials per mean-shift. With a value of $k = 15$, the anomaly covers approximately 15% of the network.

5.2.3 Overall Anomaly Detection Results

As expected, both algorithms perform poorly on very small anomalies; and both approach 100% detection rates as anomalies grow very large. However, for moderately sized anomalies, the clusterings produced by MAC yield a measurably higher anomaly detection rate. This difference is greatly amplified for higher values of k, that is, when larger portions of the network are affected. This is likely because Louvain is constrained to only produce clustering steps that increase modularity. Since even minimally modular clusterings can be useful for anomaly detection, MAC tends to outperform Louvain for the purpose of hierarchical anomaly detection. Both MAC and Louvain produce roughly equal false positive rates.

As mentioned in Chapter 3, for the implementation of MAC as a decentralized algorithm, we envision the computations necessary for the anomaly detection of each cluster to be conducted collaboratively (in a decentralized fashion, using message-passing and reduce operations) by the nodes within that cluster.

Chapter 6

Application to Anomaly Detection in Electric Power Systems

6.1 Motivation

As mentioned in Chapter 2, our motivating application is the detection of GPS spoofing attacks on the distribution power grid. Typical spoofing attacks, which disrupt the timing and synchronization of the grid, would produce subtle, distributed anomalies such as those simulated in Chapter 5. Our goal, therefore, is to utilize hierarchical anomaly detection to examine successively larger and larger groups of nodes, in order to detect these subtle, distributed anomalies.

Given this motivating application, we are particularly interested in a more accurate evaluation of the MAC algorithm's effectiveness for anomaly detection specifically on power networks. To achieve this fidelity, we present a number of ways to render the anomaly benchmarking more realistic, accurate, and customized to power systems.

6.2 Modeling and Representations

The distribution power grid is a representative example of a distributed network of small, variable devices. To be able to apply a clustering algorithm to such a network requires first representing a power system network as a weighted graph. We begin with a fairly simplistic representation, wherein buses (generators or loads) are modelled as nodes, and branches (overhead lines, underground lines, and transformers) are modelled as edges. Devices that are closely electrically connected or geographically proximal are likely to share similar physical environments and have strongly correlated measurements; accordingly, they are also more likely to simultaneously be under attack and/or measure anomalous readings. We therefore choose to weight the edges using the approximate branch admittances, since admittance is a reasonable proxy for electrical (or geographical) distance. (With additional information about a particular network, a more sophisticated choice of edge weights would become possible; however, our algorithm is designed to be domain-agnostic.)

Once we have a weighted graph representation of the system, we can apply the MAC algorithm, producing a hierarchy of clusters. This clustering can then be tested using our anomaly benchmarking setup, described in Chapter 5.

6.3 Implementation

To render the anomaly detection tests more meaningful for electric power systems, we propose the following changes.

6.3.1 Normal Profile Data

Our first proposed change is to conduct anomaly detection on realistic data that could feasibly have been generated by a power system. For each node's data, instead of simply drawing values from random standard normal distributions (and utilizing the same distribution for all nodes in the network), we can use optimal power flow solvers in MATPOWER to obtain the power generation (or load consumption) values at all the buses. With more sophisticated power system simulators, such as GridLAB-D, we can obtain multiple measured data streams to use for anomaly detection. With integrated communications modelling using simulators like ns-3, we can even integrate timestamped communications.

Initially, we begin by using the results of the optimal power flow solver as static values. With the nodes' normal data profile being defined with such extreme precision, it is possible for even a very simple, unsophisticated anomaly detection technique to successfully detect anomalies that result in buses seeing data that is off normal.

Next, we progress from static normal profiles, to building time-series models for normal bus data that are trained on publicly available load information. We are investigating the use of ARIMA, Holt-Winters, and Kalman filter-based models to see which works best for the context of power system network loads. This builds a much more accurate normal profile of the data, that takes into account seasonality within twenty-four hour periods. (For now, we ignore seasonality across periods of time longer than a day.) However, this also renders the issue of anomaly detection more challenging, because normal data for nodes is now defined in a much wider range, with much less precision. Accordingly, we switch from using simple Z-tests, to more sophisticated time-series algorithms for anomaly detection.

6.3.2 Anomaly Creation and Distribution

To seed anomalies within the network, we again begin by randomly selecting a node to be the "source" of the anomaly. (In the context of a GPS spoofing attack, for example, this would be the device located closest to the spoofing signal generator.) Since we now use real power generation values obtained from solving the power flow balance equations as the data for individual nodes, we can no longer use fixed anomaly sizes. Correspondingly, the size of the anomaly at the source node is determined in proportion to the range of normal data at that node. We spread the anomaly to neighbors of the source node in the same way as before, by selecting all $1 \ldots k$-hop neighbors. However, rather than preserving the same anomaly magnitude across all anomalous nodes, the anomaly magnitude is determined individually for each node, weakened in proportion to the full electrical distance (one could use the sum of the branch admittances along the path between the two nodes) to the source node. This more accurately represents the gradual spread of an anomaly through a network, starting from its source.

The prescribed perturbations of the anomalous nodes are input to the optimal power flow solvers, to yield data values for the entire network that still maintain power conservation. This represents the inherent resiliency in power systems, wherein there are changes in generation to exactly counterbalance any changes in load. Because the system still satisfies the power flow balance equations, traditional checks for unstable flows will overlook such anomalies. In these cases, hierarchical anomaly detection is necessary.

Eventually, our goal is to gather actual power system data from systems undergoing anomalous events (fires, earthquakes, cyber-attacks, or others) and assess the anomaly detection performance of the MAC algorithm's hierarchical clustering on this real-life data.

Chapter 7

General Discussion and Future Work

In this paper, we have presented the novel Multilevel Autonomous Clustering (MAC) algorithm for hierarchical clustering of distributed networks. A key aspect of the MAC algorithm is that it is decentralized; each node determines the clusters it belongs to, using purely locally available information. The MAC algorithm is based on the simple principle of Jaccard similarity, which is then extended to weighted graphs and further adapted to suit hierarchical clustering. Without requiring extensive parameter tuning, the algorithm produces hierarchies with fitnesses at intermediate clusterings that approach those produced by global algorithms.

More notably, as demonstrated in Chapter 5, the clusters produced by the MAC algorithm are shown to be highly effective for the purpose of hierarchical anomaly detection in distributed networks. In particular, cluster hierarchies produced by MAC achieve significantly better anomaly detection performance than those produced by a comparable global clustering algorithm (hierarchical Louvain) in scenarios where anomalies have spread to large portions of the network (higher values of k). Cluster hierarchies produced by MAC also vastly outperform those produced by the Louvain algorithm in cases where anomalies are very subtle, and achieve comparable performance for more obvious anomalies. With a sophisticated meta-detection technique that intelligently exploits the correlations between the nested communities, anomaly detection performance can be significantly improved.

As mentioned previously, the implementation of MAC used here to demonstrate its mathematical properties was a serial implementation. Future expansion of this work will involve a decentralized implementation of the MAC algorithm operating on a distributed network of devices. A wider variety of power system test networks, more sophisticated anomalies (eventually utilizing data from real-life events), and more sophisticated anomaly detection techniques will be used to explore and refine MAC's capability as a local, decentralized, hierarchical clustering algorithm for distributed networks, particularly electric power systems. Once this algorithm has been implemented in hardware, we plan to test it first on a cyber-physical testbed, before eventually migrating to testing facilities hosted by electric utility companies.

Our hope is that, someday, anomaly detection systems utilizing the hierarchical clustering of the MAC algorithm can be used to mitigate disasters like the power grid failures in Puerto Rico after Hurricane Maria, or the Ukraine power grid cyberattack in 2015. As grids grow ever more connected, and threats to network security increase, clustering-based hierarchical anomaly detection methods may prove to be the answer.

Bibliography

[1] James Bagrow and Erik Bollt. A local method for detecting communities. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 72:046108, 11 2005.

[2] Vincent Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics Theory and Experiment*, 2008, 04 2008.

[3] Béla Bollobás. *The Evolution of Random Graphs—the Giant Component*, page 130–159. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2 edition, 2001.

[4] R. A. Bridges, J. P. Collins, E. M. Ferragut, J. A. Laska, and B. D. Sullivan. Multi-level anomaly detection on time-varying graph data. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 579–583, 2015.

[5] V. Chatzigiannakis, S. Papavassiliou, M. Grammatikou, and B. Maglaris. Hierarchical anomaly detection in distributed large-scale sensor networks. In *Proceedings of the 11th IEEE Symposium on Computers and Communications*, ISCC '06, page 761–767, USA, 2006. IEEE Computer Society.

[6] Aaron Clauset. Finding local community structure in networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 72 2 Pt 2:026132, 2005.

[7] Aaron Clauset, M Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 70:066111, 01 2005.

[8] A. De Paola, S. Gaglio, G. L. Re, F. Milazzo, and M. Ortolani. Adaptive distributed outlier detection for wsns. *IEEE Transactions on Cybernetics*, 45(5):902–913, 2015.

[9] J. Hong, C. Liu, and M. Govindarasu. Integrated anomaly detection for cyber security of the substations. *IEEE Transactions on Smart Grid*, 5(4):1643–1653, 2014.

[10] Themistoklis Palpanas, Dimitris Papadopoulos, Vana Kalogeraki, and Dimitrios Gunopulos. Distributed deviation detection in sensor networks. *SIGMOD Rec.*, 32(4):77–82, December 2003.

[11] M. A. Partha and C. V. Ponce. Mac: Multilevel autonomous clustering for topologically distributed anomaly detection. In Hocine Cherifi, Sabrina Gaito, José Fernendo Mendes, Esteban Moro, and Luis Mateus Rocha, editors, *Complex Networks and Their Applications VIII*, pages 747–760, Cham, 2020. Springer International Publishing.

[12] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[13] Murad A. Rassam, Anazida Zainal, and Mohd Aizaini Maarof. An efficient distributed anomaly detection model for wireless sensor networks. *AASRI Procedia*, 5:9 – 14, 2013. 2013 AASRI Conference on Parallel and Distributed Computing and Systems.

[14] Jason Robinson, Margaret Lonergan, Lisa Singh, Allison Candido, and Mehmet Sayal. Shard: A framework for sequential, hierarchical anomaly ranking and detection. In Pang-Ning Tan, Sanjay Chawla, Chin Kuan Ho, and James Bailey, editors, *Advances in Knowledge Discovery and Data Mining*, pages 243–255, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[15] Satu Schaeffer. Graph clustering. *Computer Science Review*, 1:27–64, 08 2007.

[16] Satu Schaeffer and Stefano Marinoni. Dynamic local clustering for hierarchical ad hoc networks. volume 2, pages 667 – 672, 10 2006.

[17] Satu Elisa Schaeffer. Stochastic local clustering for massive graphs. In Tu Bao Ho, David Cheung, and Huan Liu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 354–360, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[18] Rok Sosic and Jure Leskovec. Large scale network analytics with snap: Tutorial at the world wide web 2015 conference. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 1537–1538, New York, NY, USA, 2015. Association for Computing Machinery.

[19] Sapon Tanachaiwiwat and Ahmed Helmy. Correlation analysis for alleviating effects of inserted data in wireless sensor networks. In *Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, MOBIQUITOUS '05, page 97–108, USA, 2005. IEEE Computer Society.

[20] M. Xie, S. Han, and B. Tian. Highly efficient distance-based anomaly detection through univariate with pca in wireless sensor networks. In *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 564–571, 2011.

[21] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on Power Systems*, 26(1):12–19, 2011.