Compliance Graph Analysis Using Network Science and Structure Variations

Noah L. Schrick
Tandy School of Computer Science
The University of Tulsa
Tulsa, USA
noah-schrick@utulsa.edu

Peter J. Hawrylak
Tandy School of Computer Science
The University of Tulsa
Tulsa, USA
peter-hawrylak@utulsa.edu

Brett A. McKinney
Tandy School of Computer Science
The University of Tulsa
Tulsa, USA
brett-mckinney@utulsa.edu

***Abstract*—Compliance graphs are generated graphs (or networks) that represent systems' compliance or regulation standings at present, with expected changes, or both. These graphs are generated as directed acyclic graphs (DAGs), and can be used to identify possible correction or mitigation schemes for environments necessitating compliance to mandates or regulations. DAGs complicate the analysis process due to their underlying graph structures and asymmetry. This work presents network science centralities compatible with DAGs, and structure variations as a means to analyze three different example compliance graphs. Each centrality measure and structural change offers a unique importance ranking that can be used for prioritizing correction or mitigation schemes.**

***Index Terms*—Attack Graph; Compliance Graph; Cybersecurity; Compliance and Regulation; Network Theory; Centrality;**

I. Introduction

Compliance graphs are an alternate form of attack graphs, utilized specifically for examining compliance and regulation statuses of systems. Like attack graphs, compliance graphs can be used to determine all ways that systems may fall out of compliance or violate regulations, or highlight the ways in which violations are already present. These graphs are notably useful for cyber-physical systems due to the increased need for compliance. As the authors of [1], [2], and [3] discuss, cyber-physical systems have seen greater usage, especially in areas such as critical infrastructure and Internet of Things. The challenge of cyber-physical systems lies not only in the demand for cybersecurity of these systems, but also the concern for safe, stable, and undamaged equipment. The industry in which these devices are used can lead to additional compliance guidelines that must be followed, increasing the complexity required for examining compliance statuses. Compliance graphs are promising tools that can aid in minimizing the overhead caused by these systems and the regulations they must follow. The state-space explosion and large-scale nature of compliance graphs leads to additional overhead for analysis approaches. Simplistic, initial approaches for compliance graph analysis quickly results in difficulties in terms of spatial and runtime complexities. Comparing every edge of every node in a graph containing upwards of hundreds of millions of nodes and edges makes analysis techniques with exponential complexities in either spatial or runtime terms largely infeasible. Brute-force tactics, manual evaluations, and consideration of all permutations will yield lackluster output, or may fail to complete in a reasonable manner of time. To reduce the problem space, prioritization of nodes can be performed as a pre-processing step for further analysis works. A compliance graph can undergo an initial analysis process to determine which nodes or edges should undergo a more rigorous investigation. In addition, the structure of the compliance graph can be altered to limit or refine the goal of the analysis process. This work will proceed as follows: Section IV will present the prioritization metrics through a Network Science lens. Section V will present graph transformations for altering and refining compliance graphs to other structures. Section IV-H will present and discuss the centrality results, and Section V-C will present and discuss the transformation results.

II. Related Works

Compliance graphs have yet to be formally investigated for analysis purposes. However, compliance graphs share many similarities to attack graphs. As Section I discusses, attack and compliance graphs are both directed acyclic graphs (DAGs) that exhaustively walk through all changes in a system or set of systems. Attack graphs examine the cybersecurity postures, while compliance graphs examine compliance or regulation standings. These graphs are generated and processed similarly, but are focused and refined on different fields of interest. Many researchers have developed or applied analysis techniques to attack graphs in order to analyze various features and reveal information regarding common trends or possible corrections. These techniques, though applied to attack graphs, are capable of being applied to compliance graphs if slight modifications were made. This Section highlights a few of the research routes that have been undertaken in order to accomplish these goals. This Section categorizes these related investigations to highlight the availability and novelty of the research methods for this dissertation.

After the generation of graphs, it is reasonable to visualize possible violation paths an environment may endure. However, assuming not all paths can be removed, deciding which paths to choose is a cumbersome difficulty. One analysis technique to overcome this difficulty is through using minimization. Minimization can be employed to determine if a given security countermeasure increases the security or regulatory standing of a network [4]. Given a security countermeasure or correction scheme and the generated graph, if the proposed option prevents a transition from one graph state to another, the connecting edge is removed. After repeating for all possible edge removals, if the number of attacker or violation goal states has decreased, then the security countermeasure or correction scheme does improve the network. If the number of goal states remains the same, then the security countermeasure or correction scheme is not sufficient enough to improve the network's standing with regard to security or compliance.

Another technique for minimization analysis is identifying the smallest subset of security countermeasures or correction schemes that produce a desired network threshold [4]. However, the authors of [4] discuss that determining this is an NP-complete problem. This approach becomes increasingly infeasible as the size of the graphs grow with high numbers of possible attack vectors or regulatory violation conditions. If the minimum subset was known, then a set of countermeasures or correction schemes could be processed to identify the smallest number of resolutions that would prevent all attacks or violations in the minimum set.

Though the minimum subset is an NP-Complete problem, approximations can still be derived, and various works have taken approaches toward the cost minimization problem. The authors of [5] presented an approach utilizing disjunctive normal forms. In this approach, countermeasures and correction schemes are represented as disjunctive clauses. The authors of [6] developed a heuristic approach for attacks of multiple steps. In this approach, a cost is associated with the beginning nodes, with partial costs being distributed and propagated through state transitions. The authors of [7] implemented a graphical approach for cost minimization analysis. Boolean functions and Shannon decomposition were leveraged with the use of source and sink nodes.

Regarding network science approaches, the authors of [8] use betweenness centrality specifically for logical attack graphs. Using the importance derived from the centrality results, the authors were able to employ a correction scheme with greater efficiency as compared to prioritizing a shortest-path approach. The author of [9] presents three centrality measures that were applied to various attack graphs. The centrality measures implemented were Katz, K-path Edge, and Adapted PageRank, with the authors of [10] expanding on the Adapted PageRank approach. Each of these centrality measures are applicable to the directed format of compliance graphs, and conclusions were drawn by the author of [9] regarding patching schemes for preventing exploits in attack graphs. As an approach for avoiding complex eigenvalues, the authors of [11] present work examining directed, undirected, and mixed graphs using its Hermitian adjacency matrix. Other works, such as that discussed by the author of [12], include mathematical manipulation of directed graph spectra (originally presented by the author of [13]) with Schur's Theorem to bound eigenvalues and allow for explicit computation, which can then be used for additional analysis metrics.

III. Example Networks

IV. Identifying Correction Priority Through Network Centralities

In order to generate a correction scheme from a compliance graph, correction priorities first need to be obtained. For a correction scheme to be useful, it is imperative to tailor it around the most important concerns for a system or set of systems. Given a prior knowledge network and a compliance graph consisting of nodes and edges, it is possible to ascertain importance based on various information. Though nodes flagged as "in violation" have importance, compliant upstream nodes and edges may have greater importance. Figure 1 illustrates an example subgraph of a compliance graph. An initial approach could be to assign nodes 67, 71, 74, and 75 (the shaded, "in violation" nodes) the highest priority. If nodes 74 and 75 were to be independently addressed, both edge "e" and edge "b" would need to be prevented. However, though node 73 is compliant, removing this node from generation would prevent nodes 74 and 75 from their generations. Based on constraints and other factors present in a prior knowledge network, edge "a" could have insignificant impact on available resources. Preventing edge "a" would additionally prevent node 71 (another shaded, "in violation" node) from its generation. Obtaining a priority list using a prior knowledge network and topographical information makes a network science approach very appealing. This Section discusses using centrality and graph transformations as a means of obtaining a correction priority. Figure 2 displays this approach as it relates to obtaining the correction priority.

A. Introduction to Network Centralities

Within the field of network science, centralities are often used to determine the importance of a node or edge of a graph (or network). Various centrality metrics assign importance based on differing characteristics of a graph or network, such as the ability a node or edge has for transferring information, or the influence they may have on other, local or global nodes or edges. The author of [14] provides a survey of centrality measures, and discusses how various centrality measures have been implemented in order to determine

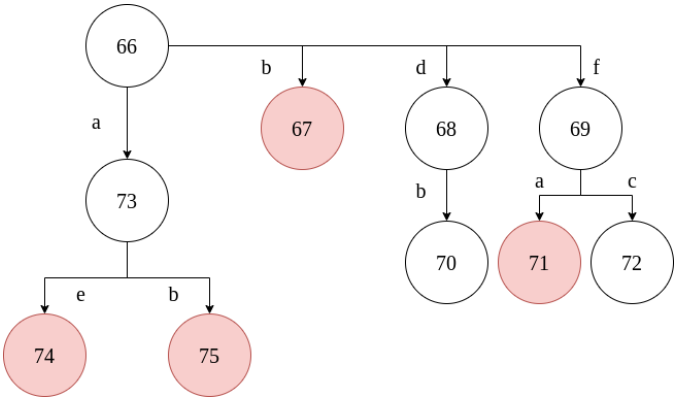


Fig. 1. Example Subgraph of a Compliance Graph. Prioritization of flagged nodes (shaded nodes) is one approach at minimizing severity in a system or set of systems. However, accounting for topographical information and upstream nodes that are in compliance can serve as better approaches for minimizing severity.

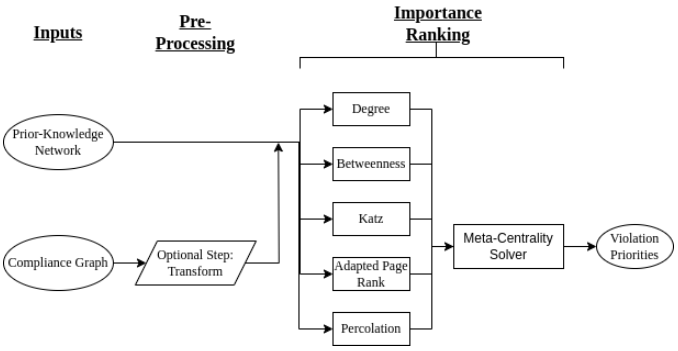


Fig. 2. Obtaining a Correction Priority. A correction priority can be obtained through a compliance graph and a prior knowledge network. By using graph transformations and network centralities, importance can be assigned to nodes to serve as a correction priority.

node importance in networks. By determining the importance of nodes, various conclusions can be drawn regarding the network, and how to identify noteworthy hubs. In the case of compliance graphs, conclusions can be drawn regarding the prioritization of patching or correction schemes. If one node directs to many other nodes, a mitigation enforcement may be considered imperative to prevent further opportunities for compliance violation. This work discusses centrality measures across various structural changes, and contextualizes their applications to compliance graphs.

1) Network Centralities for Directed Graphs: Compliance graphs, like attack graphs, are directed acyclic graphs, and analysis of directed graphs is notably more involved compared to their undirected counterparts. The primary contributor to the increased difficulty is due to the asymmetric adjacency matrix present in directed graphs. Figure 3 displays an undirected graph with a symmetric adjacency matrix, and Figure 4 displays a directed form of the same graph with an asymmetric adjacency matrix. With undirected graphs, simplifications can be made in the analysis process both computationally and conceptually. Since the "in" degrees are equal to the "out" degrees, less work is required both in terms of parsing the adjacency matrix, but also in terms of determining importance of nodes. The author of [15] discusses that common analysis techniques such as eigenvector centrality is often unapplicable to directed acyclic graphs. As the author of [12] discusses, the difficulty of directed graphs also extends to the graph Laplacian, where the definition for asymmetric adjacency matrices is not uniquely defined, and is based on either row or column sums computing to zero, but both cannot. The author of [12] continues to discuss that directed graphs lead to complex eigenvalues, and can lead to adjacency matrices that are unable to be diagonalized. These challenges require different approaches for typical clustering or centrality measures.

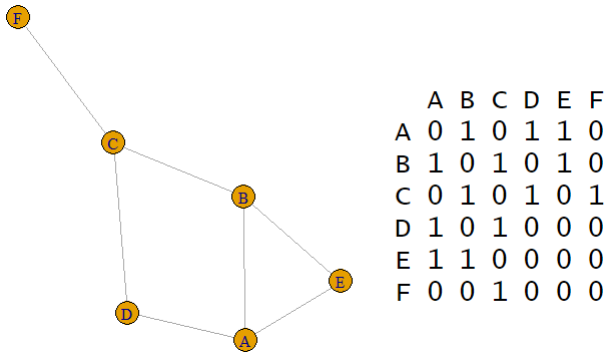
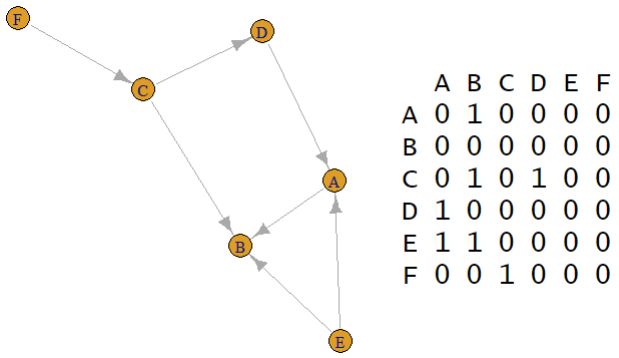


Fig. 3. For undirected graphs, the resulting adjacency matrix is symmetric. For all nodes that have a connecting edge, the corresponding cell in the matrix is marked with a "1". This value is present both when traversing by row and by column. All nodes that do not have a connecting edge have a value of "0" in their corresponding cell. This value is also present both when traversing by row and by column. Therefore, the halves of the matrix across its diagonal are mirrored.

While Section II discusses a few options for approaching the challenges with directed graphs, this work opted for centrality metrics that are natively compatible with compliance graphs. Specifically, this work employed centrality metrics that are compatible with directed graphs and assign importance to nodes, rather than edges. Recall that for compliance graphs, nodes represent the state of an environment, with all relevant asset information encoded within the node. Edges (exploits) represent the changes that can be made to a system state. When generating a compliance graph, the edge specification is performed in advance, where the exploit is known, or the behavior (such as for zero-day modeling) can be described. The edges can have probability values assigned through likelihood analysis, can be linked to known CVEs, or through historical data. Since the unique set of exploits is orders of



	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	0	0	0	0
C	0	1	0	1	0	0
D	1	0	0	0	0	0
E	1	1	0	0	0	0
F	0	0	1	0	0	0

Fig. 4. For directed graphs, the resulting adjacency matrix is asymmetric. Node "A" has a directed edge to Node "B", and therefore a value of "1" is found in the corresponding cell. However, since this edge is not reciprocated, there is a value of "0" when examining the cell between Node "B" and Node "A". This behavior is repeated for other, similar node relationships in this example graph. The halves of the matrix across its diagonal are not mirrored.

magnitudes less than the total number of edges in a graph, coupled with existing techniques for assigning edge values, and given the contextualization of edges in a compliance graph, assigning importance to edges is far less beneficial than examining prioritization at the node level. Since the goal of compliance graph analysis is securitization and limiting regulation violation seen at a node level, centrality metrics that assign importance to nodes were chosen over metrics that assign importance to edges. Section VI discusses future avenues that include assigning importance to edges in addition to nodes.

B. Degree

Degree centrality is a trivial, localized measure of node importance based on the number of edges that a node has. In an undirected graph, the degree centrality is predicated solely on the number of edges. However, in the case of a directed graph, a distinction is drawn with a degree centrality oriented on the number of edges entering a node, and another measure focused on the number of edges leaving a node. Both of these cases provide useful information for compliance graphs. When a node has a large number of other nodes it directs to, this node may be prioritized since it creates further opportunity for violation. When a node has a large number of edges pointing to it, this node may be prioritized since the probability that systems may enter this state is higher due to the increased number of possibilities that a system change could lead to this state.

Degree centrality for the example networks presented in Section III was implemented in R. The attack and compliance graph generator, RAGE, outputs the graph in a Graphviz [16] DOT file. R's igraph [17] package includes functionality to import from a DOT file into an igraph network, and this functionality was used for this purpose. Once the graph was imported, the built-in igraph function for degree centrality was called, and its output was stored in a list. With standard approaches, degree centrality has a spacial and time complexity of $\mathcal{O}(n^2)$. Standard approaches compute degree centrality through the adjacency matrix representation of a graph, which is a *n* x *n* matrix. Apart from the ease-of-use that igraph provides, it also includes optimizations for commonly-used graph operations. Due to the compliance graph structure, the graph can be stored in a column-compressed format as a sparse matrix. This sparse matrix is able to significantly reduce the memory footprint. Rather than storing an integer (or Boolean) value for each node-to-node connection (in the standard a *n* x *n* matrix), a series of reduced columns can be used, which reduces the spatial complexity to $\mathcal{O}(K)$, where K is the number of nonzero elements. Operations on a sparse matrix likewise have reduced time complexities, including degree centrality. When using a sparse matrix with igraph's degree centrality function, time complexity is bound by $\mathcal{O}(n * d)$, where d is the average degree.

C. Betweenness

Betweenness centrality ranks node importance based on its ability to transfer information in a network. For all pairs of nodes in a network, a shortest path is determined. A node that is in this shortest path is considered to have importance. The total betweenness centrality is based on the number of shortest paths that pass through a given node. For compliance graphs, the shortest paths are useful to identify the quickest way (least number of steps) that systems may fall out of compliance. By prioritizing the nodes that fall in the highest number of shortest paths, correction schemes can be employed to prolong or prevent systems from falling out of compliance.

Betweenness centrality is given in Equation 1, where *i* and *j* are two different, individual nodes in the network, σ_{ij} is the total number of shortest paths from *i* to *j*, and $\sigma_{ij}(v)$ is the number of shortest paths that include a node *v*.

$$\sum_{i \neq i \neq v} \frac{\sigma_{ij}(v)}{\sigma_{ij}} \tag{1}$$

The implementation details for betweenness centrality are largely similar to the details described in Section IV-B for degree centrality. Betweenness centrality was computed using igraph's betweenness function, due to the ease-of-use and graph algorithm optimizations offered by the library. For the applications described in Section III, the compliance graphs are unweighted. The igraph library implements Brande's algorithm [18] for the centrality computation, which provides drastic improvements over other algorithms for computing

betweenness. Using the igraph package, and coupled with the unweighted nature of the given compliance graphs, the time complexity of betweenness centrality is $\mathcal{O}(|n| * |e|)$, where *n* is the number of nodes, and *e* is the number of edges, and the spatial complexity is $\mathcal{O}(n + e)$.

D. Katz

Katz centrality was first introduced by the author of [19], and measures the importance of nodes through all paths in a network. Katz centrality varies in that its centrality measure is not limited to solely the shortest path between any two given nodes. The original work by the author defines Katz as seen in Equation 2, where *i* and *j* are nodes in the network, *n* is the total number of nodes in the network, *A* is the adjacency matrix, and α is an attenuation factor and has a value between 0 and 1. A value of 1 is assigned if node *i* is connected to node *j*.

$$C_{\text{Katz}}(i) = \sum_{k=1}^{\infty} \sum_{j=1}^{n} \alpha^k (A^k)_{ji} \tag{2}$$

Later works have expanded on the original Katz to include a β vector that allows for additional scaling in the instance that prior knowledge of the network exists. The modified equation implemented by the authors of [20] can be seen in Equation 3.

$$\vec{x} = (I - \alpha A)^{-1} \vec{\beta} \tag{3}$$

For compliance graphs, Katz centrality represents the total number of paths that exist from a given node to any other downstream nodes, and is scaled based on the attenuation factor as well as the prior knowledge vector β. When the Katz centrality of a given node is high, prioritizing a correction scheme for the node would be useful to prevent opportunity of future compliance violations that may be many steps ahead, but still reachable from the current state. Additional weighting and scaling can be applied to nodes known in advance to have greater importance through the β vector, and through tuning the attenuation factor to give greater weight to local or global reach of nodes.

For Katz centrality, difficulties can be encountered when computing the eigenvalues. When using the igraph "eigen" function, n^2 memory is required. This function calls upon a row summing helper function, which requires an intermediate matrix to be held in memory. For large-scale graphs, this quickly becomes problematic. In addition, computing eigenvalues is bound by a time complexity of $\mathcal{O}(n^3)$, which is cumbersome for the large-scale compliance and attack graphs that are generated [21]. igraph does make use of the LAPACK [21] routines, which reduces the time complexity to $\mathcal{O}(n^2)$. However, for the compliance graphs that are generated in this work, there are a few ways to work around the time and spatial complexity. Directed acyclic graphs (DAGs) have special properties in their spectral analysis. As the authors of [22] and [23] discuss, the eigenvalues of the adjacency matrix of a DAG are 0. As a result, a column vector of 0s of size *n* can be initialized in place of an eigenvalue computation. For computing Katz centrality, a custom method that follows the original approach of [19] was created. Though the eigenvalue computation can be omitted, Katz centrality is still bound by matrix multiplication, which has a direct definitional time complexity of $\mathcal{O}(n^3)$ [24]. Recent works have shown that the time complexity can theoretically be reduced to $\mathcal{O}(n^{2.371552})$ [25], however the implemented, tested, and confirmed algorithm for a time complexity of $\mathcal{O}(n^{2.3728596})$ [26] is more common in practice.

For this implementation, since sparse matrices are employed, the time complexity can be reduced even further. Though the eigenvalue vector is initialized to zero, its values are updated to match the adjacency matrix values each iteration. The adjacency matrix will be denoted as *A*, and the eigenvalue vector will be denoted as *B*. For each of these matrices, *a* shall represent the number of nonzero elements in *A*, and *b* shall represent the number of nonzero elements in *B*. Rather than the definitional time complexity being represented as $\mathcal{O}(n*n*n)$, the definitional time complexity for sparse matrices can be represented as $\mathcal{O}(a*b*n)$. Regarding parameters, α was set to 0.5 to allow for a balance in short and long distance edge traversals. The β vector was trivially set. If a node was in violation of a mandate, regulation, or some other form of compliance requirement, a value of 5.0 was assigned. Otherwise, the node had a value of 1.0.

E. Adapted PageRank

The original PageRank algorithm was first designed by the authors of [27] for the Google prototype for ranking web pages. The authors of [28] later introduced an Adapted PageRank algorithm that was designed to measure both the number and quality of connections specifically for an urban network. Equation 4 displays the PageRank algorithm, where γ is a damping factor with a value between 0 and 1, *n* is the total number of nodes in the network, *A* is the adjacency matrix of the network, *i* and *j* represent the row and column of the adjacency matrix, *x* is a given node in the network, and *k* is the row sum out degree. Since the Adapted PageRank algorithm measures the quality of connections, there is increased application to directed networks such as compliance graphs. As seen in Equation 4, the k_j term is a penalizing factor. Importance is based on the in degree of a node, with a penalty for the out degree. If many nodes point to a given node, then that node is considered important due to its accessibility.

$$x_i = \frac{1-\gamma}{n} + \gamma \sum_{j=1}^{n} \frac{A_{ij}}{k_j} x_j \tag{4}$$

The adapted PageRank algorithm includes additional data that may be present in an urban network, such as geographical position, resource availability, and proximity to facilities. This data is user-defined, and may not be present in the network. Equation 5 displays the Adapated PageRank algorithm in matrix form where *D* is the user-defined data matrix, *I* is the identity matrix, and $\mathbb{1}$ is a column matrix comprised of 1s.

$$(I - \gamma AD)\vec{x} = \frac{1-\gamma}{n}\mathbb{1} \tag{5}$$

For compliance graphs, the Adapted PageRank algorithm is useful for a few reasons. First, it is able to include user-defined data regarding the network. This could include scaling certain nodes to have greater weight, such as those known to be in a noncompliant state. Second, since nodes are penalized for pointing to other nodes, this algorithm is useful for determining nodes that are likely to be visited. If a state has a greater in-degree, it may require greater prioritization since the system has a higher likelihood of falling into this state.

The implementation details for PageRank centrality are largely similar to the details described in Section IV-B for degree centrality. PageRank centrality was computed using igraph's "pagerank" function, due to the ease-of-use and graph algorithm optimizations offered by the library. For the applications described in Section III, the time complexity of the Adapted PageRank centrality when using igraph is $\mathcal{O}(|E|)$, where *e* is the number of edges, and the spatial complexity is $\mathcal{O}(n)$ for the results vector, plus the spatial requirements of holding the graph object. The igraph computation features improvements over traditional implementations - the traditional time complexity is often $\mathcal{O}(n+e)$, where *n* is the number of nodes, and *e* is the number of edges.

F. Percolation Centrality

Percolation centrality was originally presented by the authors of [29], and has continued to see usage in works such as that presented by the authors of [30] for percolation centrality approximation, and in the work presented by the authors of [31] for parallel programming approaches. Percolation centrality aims to measure importance of nodes through their topographical connectivity, as well as through using percolation theory. As a contagion travels through a network, it has the capacity to alter the state of each node. This alteration, and any residual effects, can cause nodes to become percolated, which can then themselves cause other nodes to also become percolated. Equation 6 displays the formal definition for percolation centrality, where *x* is the percolated state, *s* is a source node, *v* is a different, unique node, *N* is the number of nodes, σ_{ij} is the total number of shortest paths from *i* to *j*, and $\sigma_{ij}(v)$ is the number of shortest paths that include a node *v*.

$$PC^t(v) = \frac{1}{(N-2)} \sum_{s \neq v \neq r} \frac{\sigma_{s,r}(v)}{\sigma_{s,r}} \frac{x_s^t}{[\sum x_i^t] - x_v^t} \tag{6}$$

For compliance graphs, percolation centrality is able to examine and consider the dependencies that violations may have. Some compliance or regulation mandates rely on the statuses of other mandates. When nodes are flagged as "at risk" of a violation or are actively violating a mandate, this percolation will spread to surrounding nodes. This measure is able to prioritize nodes based on their surrounding connections and their standings in regard to a mandate.

Percolation centrality required additional pre-processing before computing centrality values. Though NetworkX [32] includes a percolation centrality function, percolation attributes need to be embedded within the graph object prior to calling the function. For this centrality metric, all work was performed in Python, and the final centrality vector was passed back to R through the Reticulate library [33], which acts as an interface between R and Python.

The pre-processing component of percolation centrality required the parsing of the prior-knowledge network. This file was opened and parsed within Python in order to determine which edge labels marked a violation in a compliance requirement. The graph was processed to determine which nodes were considered "in violation", which was identifiable through an "in-edge" with a label that marked a violation state. In addition, the pre-processing identified nodes that were in contact with a violation node through *n*-step reachability. A node was considered to have exposure to a percolated state if it was *n* steps away from a node in violation. This work made use of a 2-step reachability scheme. As the graph was processed, node attributes were assigned with a "percolation" label. Nodes in violation were assigned a percolation value of 0.99, exposed nodes were assigned a value of 0.50, and all other nodes were assigned a percolation value of 0.01. This approach can be seen in Algorithm 1, which expands the algorithm into an unoptimized format to showcase the process for simplicity.

Identifying violation nodes is bound by a time complexity $\mathcal{O}(e)$, where *e* is the number of edges. This is due to iterating through the graph's edge labels. Identifying exposed nodes has a time complexity of $\mathcal{O}(v * k * \log v)$, where *v* is the number of nodes in violation and *k* is the *n*-step reachability cutoff

Algorithm 1: Expanded, Unoptimized Approach for Pre-Processing the Network for Percolation Centrality

Input : Prior-Knowledge Network (PKN), Network, n (Reachability Step)
Output: Network

STEP 1: Parse PKN and identify exploits that denote a violation.
violationEdges = array[];
for *exploit in PKN* **do**
  if *exploit causesVio* **then**
    do add *exploit* ID or label to violationEdges array;
STEP 2: Identify nodes in violation.
violationNodes = array[];
for *node in Network* **do**
  if *any in-edge is in the violationEdges array* **then**
    do add *node* to violationNodes array;
    do set *node*[percolation attribute] = 0.99;
  else
    do set *node*[percolation attribute] = 0.01;
STEP 3: Identify nodes exposed to violation nodes.
exposedNodes = array[];
if *n is not provided* **then**
  do set n = 2;
for *node in violationNodes array* **do**
  if ***find*** *nodes that are within n path length away from node and not in violationNodes array* **then**
    do add to exposedNodes array;
STEP 4: Update percolation label for exposed nodes.
for *node in exposedNodes array* **do**
  do set *node*[percolation attribute] = 0.50;

length. Since the adjacency matrix has already been obtained through the graph object and through edge labels, there is no edge exploration cost incurred. The NetworkX implementation of percolation centrality uses the algorithm presented by the authors of [29], which also includes Brande's algorithm [18]. For the applications described in Section III, the NetworkX time complexity for percolation centrality is $\mathcal{O}(|n|*|e|)$, where *n* is the number of nodes, and *e* is the number of edges. The spatial complexity is $\mathcal{O}(e)$.

G. Centrality Aggregation

Each centrality metric assigns importance on various features of a network. Each approach focuses and highlights on different aspects of the network's topographical properties, with some centrality approaches also relying on external data matrices and prior-knowledge networks. Due to the utility and strengths of each approach, an aggregation of scores in a meta-centrality fashion allows importance to be obtained as a collection of all centrality approaches, rather than choosing a single centrality metric to use. The aggregation of centralities has been investigated in the works seen by the authors of [34], [35], and [36].

To aggregate the importance scores, a few approaches are possible. The authors of [37] use a Borda-count based aggregation system to study the effect of super-spreaders in a network through a meta-centrality approach. Other approaches include the Kemeny-Young method [38], [39], however this approach requires a greater computation requirement that yields this unfeasible for the large-scale compliance graphs. Though the Borda-count approach would be feasible, this work opted for a mean-based rank method for simplicity. Equation 7 displays the approach for computing an aggregated centrality score. In this approach, a proportion is computed for each node in relation to the overall centrality score for that metric. That proportion is then adjusted based on a weighting for the metric, where the weighting is a value between 0.0 and 1.0, where all weightings sum to a value of 1.0. This approach allows for each centrality metric to contribute to the aggregated centrality score, but additional tuning can be employed to assign greater contributions to metrics that may utilize prior-knowledge of the embedded network information (such as Katz or Percolation Centrality).

$$
\begin{aligned}
importance_i = &(\frac{\frac{degree_i}{\Sigma degree} * weight_{degree}}{length(CentralityMetrics)})+ \\
&(\frac{\frac{betweenness_i}{\Sigma betweenness} * weight_{betweenness}}{length(CentralityMetrics)})+ \\
&(\frac{\frac{Katz_i}{\Sigma Katz} * weight_{Katz}}{length(CentralityMetrics)})+ \\
&(\frac{\frac{PageRank_i}{\Sigma PageRank} * weight_{PageRank}}{length(CentralityMetrics)})+ \\
&(\frac{\frac{percolation_i}{\Sigma percolation} * weight_{percolation}}{length(CentralityMetrics)})
\end{aligned}
\tag{7}
$$

Post-processing is performed on the aggregated centrality vector. Since prior-knowledge networks are implemented for all example applications, it is useful to further tune the aggregated scores. In order for a graph node to be fully mitigated, all "in-edges" must be prevented. Within the prior-knowledge network, all exploits (edges) have information to specify the cost of mitigation if possible. This prior-knowledge network is parsed in order to identify exploits that cannot be prevented. The graph is then processed to identify nodes that have an unpreventable exploit as an in-edge, and

are therefore unpreventable nodes. Since these nodes cannot be removed from the network or mitigated, their centrality value is removed from the aggregated vector since analysis computation will yield no beneficial results for these nodes. The removed aggregated value is equally distributed to all other nodes in the aggregated vector that have a nonzero value. This process is shown in Algorithm 2.

Algorithm 2: Redistribute Aggregated Centrality Scores

Input : Prior-Knowledge Network (PKN), aggregatedCentralityVector
Output: aggregatedCentralityVector

```
STEP 1: Parse PKN and identify unpreventable exploits.
STEP 2: Identify unpreventable nodes and gather/reset their centrality score.
redistributeValue = 0;
for node in Network do
    if any in-edge is unpreventable or unable to be mitigated then
        do redistributeValue += aggregatedCentralityVector[node];
        do aggregatedCentralityVector[node] = 0;
STEP 3: Redistribute Centrality Scores
redistributeValue /= number of nonzero aggregated scores;
for score in aggregatedCentralityVector do
    if score is nonzero then
        do score += redistributeValue
```

H. Centrality Results and Analysis

Table I displays the statistical properties of the aggregated centrality scores. In all example networks, by performing post-processing, the number of states that can be analyzed is drastically reduced. In all cases, due to unpreventable exploits in the network, severities or importances can be set to 0 to prevent any further analysis on the given states. This reduction in state space has implications that require contextual understanding of the input data; it could be considered beneficial, a negative indication, or neutral. Though it reduces the state space and alleviates additional computation strain on future analysis work, it can be indicative of insufficient mitigation information or of a large set of zero-day or critical issues with no known remedy. Alternatively, the pruned nodes could be neutral states. In each network, there are flag-setting states, states that progress time, and states that reflect normal, expected behavior. In the prior-knowledge network, these states have no mitigations since their execution is expected or required. For all three example networks in this work, all pruned nodes are neutral states. Section III describes the networks in more detail, and describes how all exploits that are known to cause a violation have at least one mitigation.

TABLE I: Properties of the Aggregated Centrality Scores for the Three Example Networks

	Automobile Maintenance	HIPAA	OSHA 1910H
Number of Nonzero Elements	4245	9215	4603
Percent of Total Elements that are Nonzero	6.341%	1.481 x 10^1 %	9.516%
Number of Zero Elements	6270 x 10^1	5300 x 10^1	4377 x 10^1
Nonzero Minimum	2.214 x 10^{-4}	9.723 x 10^{-5}	2.076 x 10^{-4}
Maximum	1.936 x 10^{-3}	2.713 x 10^{-4}	2.592 x 10^{-4}
Nonzero Mean	2.356 x 10^{-4}	1.085 x 10^{-4}	2.172 x 10^{-4}
Nonzero Element Standard Deviation	6.248 x 10^{-5}	5.428 x 10^{-6}	2.855 x 10^{-6}

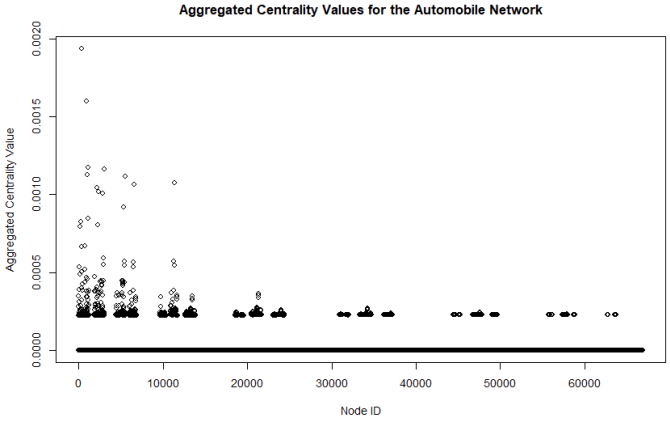


Fig. 5. Aggregated Centrality Score Distribution for the Automobile Maintenance Network. The resulting distribution of the aggregated centrality scores when using the centrality metrics presented in Section IV-A.

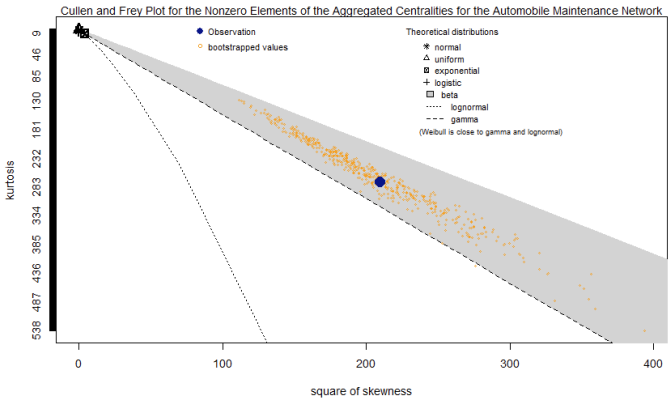


Fig. 6. Cullen and Frey Plot for the Nonzero Elements of the Aggregated Centralities for the Automobile Maintenance Network. 500 bootstrap values (random selection with replacement) are used. This Figure displays skewness^2 versus kurtosis to characterize the aggregated centrality scores with various distributions.

Figures 5, 7, and 9 display the distribution of the aggregated centrality scores. These Figures depict a distribution of

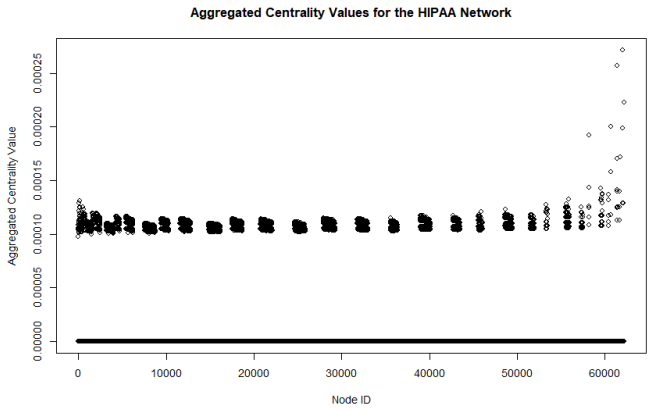


Fig. 7. Aggregated Centrality Score Distribution for the HIPAA Network. The resulting distribution of the aggregated centrality scores when using the centrality metrics presented in Section IV-A.

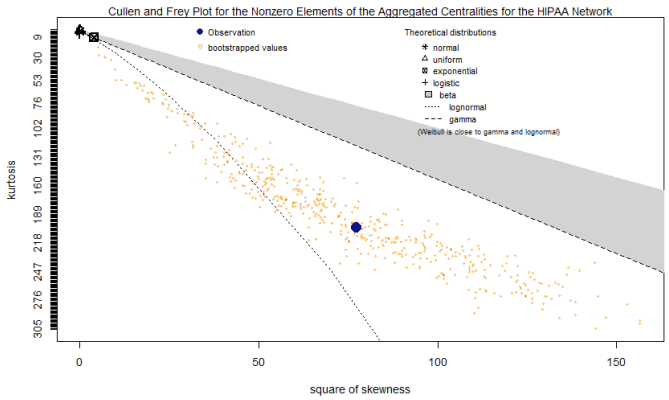


Fig. 8. Cullen and Frey Plot for the Nonzero Elements of the Aggregated Centralities for the HIPAA Network. 500 bootstrap values (random selection with replacement) are used. This Figure displays skewness^2 versus kurtosis to characterize the aggregated centrality scores with various distributions.

all elements, including the elements with a score of zero. Though no future analysis was conducted with the aggregated centrality scores in this work, additional insight or analysis could be performed regarding these results. To add insight, Cullen and Frey plots (Figures 6, 8, and 10) were generated using only the nonzero elements of all three example networks' aggregated centrality scores. Each plot uses 500 bootstrap values in the generation, which randomly select and replace values from the aggregated centrality vector to aid in the potential uncertainty of the data set. Though this work makes no direct analysis of these plots other than displaying their distribution characterizations, they could yield promising results for new techniques or approaches using statistical analysis. This is discussed further in Section VI.

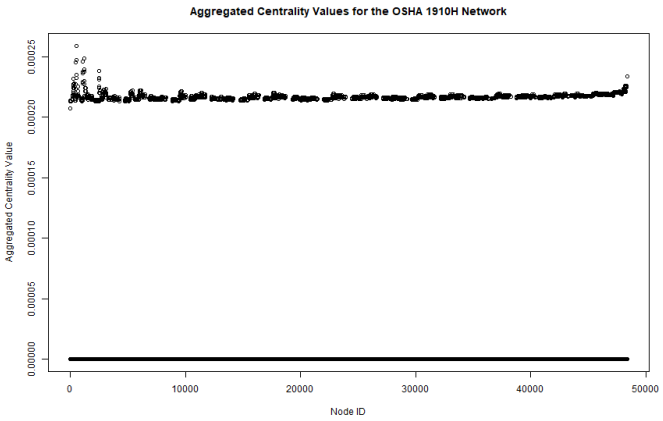


Fig. 9. Aggregated Centrality Score Distribution for the OSHA 190H Network. The resulting distribution of the aggregated centrality scores when using the centrality metrics presented in Section IV-A.

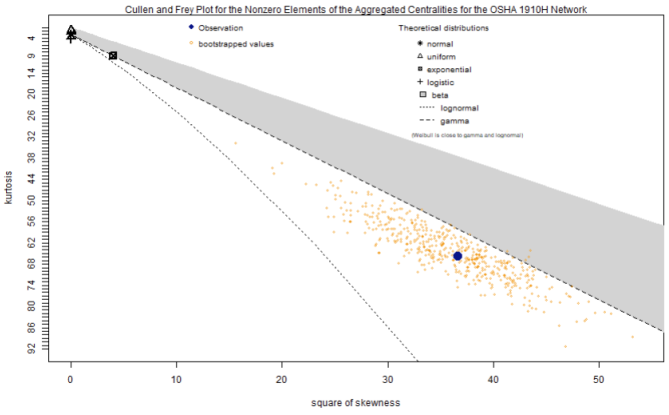


Fig. 10. Cullen and Frey Plot for the Nonzero Elements of the Aggregated Centralities for the OSHA 1910H Network. 500 bootstrap values (random selection with replacement) are used. This Figure displays skewness^2 versus kurtosis to characterize the aggregated centrality scores with various distributions.

I. Validation

In order to validate the aggregated centrality scores, the following characteristics were examined, and test cases were created to compare against expected behavior. The results of these tests are not included in this work, since the test results were a boolean "pass" or "fail". If a failed test was encountered, the validation process failed, and the methodology was flawed and in need of correction. For the work presented, each test returned a successful outcome.

- The sum across the aggregated centrality scores vector is 1.0.

- All individual centrality metric scores are greater than or equal to 0.0.
- All aggregated centrality scores are greater than 0.0.
- All individual centrality metric scores contain nonzero values.
- Select a random node and check the PKN to determine if this node is preventable:
  - If preventable, ensure that the aggregated centrality score is nonzero.
  - If unpreventable, ensure that the aggregated centrality score is zero.
- The aggregated centrality score for the root node is zero.
- All individual centrality metric score vectors match in length to the number of nodes in the network.
- The aggregated centrality metric score vector matches in length to the number of nodes in the network.

V. Graph Transformations

Generating compliance graphs as a DAG is done so purposefully. DAGs are useful representations for relationships and dependencies, and the authors of [22] reaffirm this standing. DAGs and their traversals reveal deeper understandings of causal relationships between nodes and events, and can aid in the analysis and prediction of known or expected events. However, for compliance graphs, it may be useful to transform the DAG into an alternate structure for additional analysis. It is still important to generate the compliance graph as a DAG initially to obtain the relationships of the network, and only after its initial generation is the graph transformation investigated. These transformations can be useful for determining which nodes are most important when an adversarial action can be considered to have infinite time and resources to perform changes to the original system. Alternatively, they can be useful for determining which nodes are most important from an information flow perspective, where adversarial actions must pass though a series of nodes to reach any other node in the network. This Section presents transformation options and contextualizations for compliance graphs to aid in the analysis process. Section V-A presents the Transitive Closure, and Section V-B presents the Dominant Tree.

A. Transitive Closure

Transitive closure represents a transitive relation on a given binary set, and can be used to determine reachability of a given network. Figure 11 displays an example output when performing transitive closure. In context of compliance graphs, it is useful to consider that an adversary (whether an internal or external malicious actor, poor policy execution by an organization, accidental misuse, or any other adversarial occurrence) could have no time constraints. That is, for any given state of the system or set of systems, an adversarial act could have "infinite" time to perform a series of actions. If no prior knowledge is known about the network, it can be assumed that all changes performed on the systems are equally likely. In practice, specifying a probability that a change can occur has been performed through a Markov Decision Process, such as that seen by the authors of [40] and [41]. When under these assumptions, it is useful to then consider which nodes are important, assuming they have 1-step reachability to any downstream node they may have a transitive connection to.

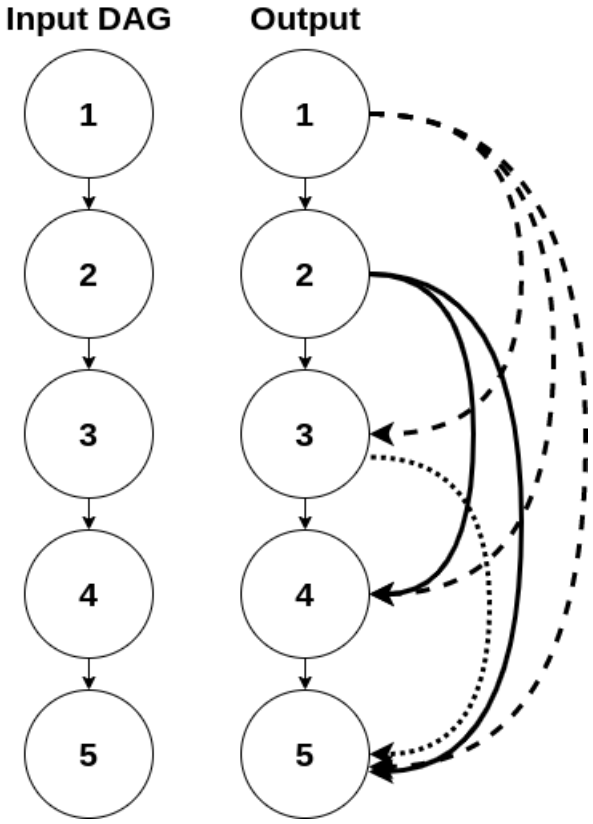


Fig. 11. Illustration of an Example DAG and its Transitive Closure. Each node in the original DAG has 1-step reachability to any downstream node it has a transitive connection to in the resulting transitive closure.

B. Dominant Tree

Dominance, as initially introduced by the author of [42] in terms of flow, is defined as a node that is in every path to another node. If a node i is a destination node, and every path to i from a source node includes node j, then node j is said to dominate node i. Figure 12 displays an example starting network. With node 1 as the source node, it is evident that node 2 immediately dominates nodes 3, 4, 5, and 6, since all messages from node 1 must pass through node 2. By definition, each node must also dominate itself, so node 2 also dominates node 2.

Following the properties of dominance, a dominator tree can be derived. In a dominator tree, each node has children that it immediately dominates. Immediate dominance is referred to nodes that strictly dominate a given node, but do not strictly dominate any other node that may strictly dominate a node. Figure 13 displays the dominant tree of the network seen in Figure 12.

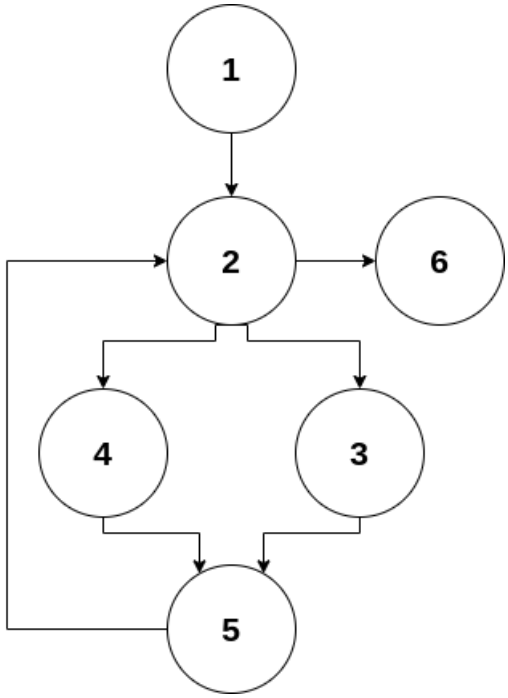


Fig. 12. Example "Base" Network for Illustrating Dominance. An arbitrary DAG that has not yet undergone transformation.

Dominant trees do alter the structure of compliance graphs, and lead to leaf nodes and branches that do not exist in the original network. As a result, some nodes that have directed edges to other nodes may be moved to a position where the edge no longer points to the original nodes. However, in dominant trees, all node parents dominate their children. In this format, the information flow is guided predominantly by the upstream nodes, and all parents in the dominant tree exist as upstream nodes in the original compliance graph. While some downstream nodes may be altered, the importance of nodes can be reexamined in the dominant tree to see how importance differs when information flow is refined.

C. Results and Analysis

To analyze the changes in the original DAG, network properties were collected for the transitive closure and dominant tree representations. Table II displays the properties for the automobile maintenance example, Table III displays the properties for the HIPAA example, and Table IV displays

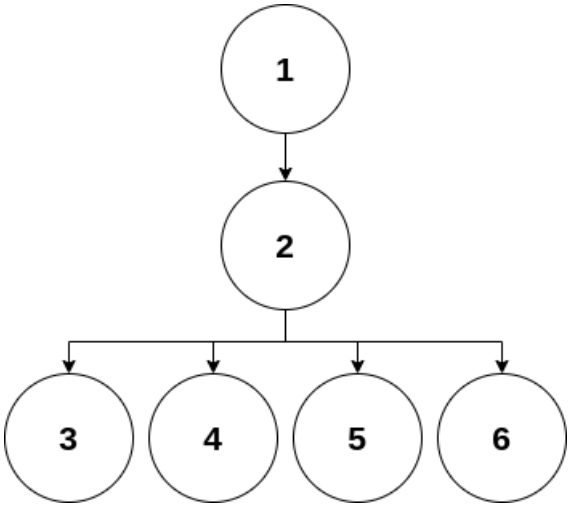


Fig. 13. Dominant Tree Derived from the Network Displayed in Figure 12. Node 1 dominates Node 2, and Node 2 possesses immediate dominance over Nodes 3, 4, 5, and 6.

the properties for the OSHA 1910H example. For each of the graphs, the number of nodes and number of edges were collected to examine how the quantity of the network structures changes. In all examples, it was expected that the number of edges for the transitive closure would substantially increase, and it was expected that the number of edges for the dominant tree would decrease. Efficiency in terms of network science was introduced by the authors of [43]. Efficiency of a graph is measured by its ability to exchange information, whereas the distance between nodes increases, the efficiency decreases. Global efficiency is a measure of communication exchange within the entire network. Local efficiency of a node is quantified through the impact of information exchange if that node was removed from the network, and is therefore a measure of fault tolerance. For the transitive closures, it is expected that the removal of any node has minimal impact on communication efficiency. Since each node has a connecting edge to all downstream nodes, the removal of any one midstream node should not degrade the ability to exchange information throughout the network. For dominant trees, the opposite is expected. The dominant tree network is generated through the concept of dominance, which is a measure of how information is passed through nodes. Since the dominant tree is hierarchical based on communication exchange properties, the removal of a node will have more severe impacts on the communication efficiency.

The radius and diameter of a graph are computed through eccentricity. Each node in a network will have an eccentricity value, which is the shortest path from that node to the farthest reachable node. The radius of a graph is the smallest eccentricity value, and the diameter is the largest eccentricity value. For transitive closures, since new edges are drawn from

TABLE II: Network Property Comparisons of the Original DAG, Transitive Closure, and Dominant Tree for the Automobile Maintenance Network

	DAG	Transitive Closure	Dominant Tree
Number of Nodes	6695 x 10^1	6695 x 10^1	6695 x 10^1
Number of Edges	4682 x 10^2	3958 x 10^4	6694 x 10^1
Global Efficiency	1.541 x 10^{-3}	8.831 x 10^{-3}	2.465 x 10^{-5}
Average Local Efficiency	1.515 x 10^{-1}	2.164 x 10^{-1}	0.000
Radius	13.00	1.000	7.000
Diameter	18.00	1.000	8.000
Density	1.045 x 10^{-4}	8.831 x 10^{-3}	1.494 x 10^{-5}

TABLE III: Network Property Comparisons of the Original DAG, Transitive Closure, and Dominant Tree for the HIPAA Network

	DAG	Transitive Closure	Dominant Tree
Number of Nodes	6222 x 10^1	6622 x 10^1	6222 x 10^1
Number of Edges	4009 x 10^2	2475 x 10^4	6222 x 10^1
Global Efficiency	1.417 x 10^{-3}	6.394 x 10^{-3}	1.935 x 10^{-5}
Average Local Efficiency	1.225 x 10^{-1}	1.866 x 10^{-1}	0.000
Radius	17.00	1.000	6.000
Diameter	19.00	1.000	6.000
Density	1.036 x 10^{-4}	6.394 x 10^{-3}	1.607 x 10^{-5}

all upstream nodes to all reachable downstream nodes, it is expected that the radius and diameter decrease. No analysis or conclusions were drawn from the dominant trees, since the dominance of the original DAG may vary, which could result in either an increase or decrease in the radius and diameter. The density of a graph is a proportion of actual edges and theoretical edges. Due to the addition of edges in the transitive closure, the density is expected to increase compared to the original DAG. No analysis or conclusions were drawn from the dominant trees, since the dominance of the original DAG may vary, which could result in either an increase or decrease in the density.

D. Validation

Since no direct analysis is conducted on the transformed compliance graphs, validation of the transformation is limited. Though centrality metrics can be collected on the transformed graphs, the same validation techniques employed by the original network will be used. In order to validate the transformed graphs, there are a few network properties that were examined, and test cases were created to compare against expected behavior. The results of these tests are not included in this work, since the test results were a boolean "pass" or "fail". If a failed test was encountered, the validation process failed, and the methodology was flawed and in need of correction. For the work presented, each test returned a successful outcome.

TABLE IV: Network Property Comparisons of the Original DAG, Transitive Closure, and Dominant Tree for the OSHA 1910H Network

	DAG	Transitive Closure	Dominant Tree
Number of Nodes	4837 x 10^1	4837 x 10^1	4837 x 10^1
Number of Edges	4083 x 10^2	3584 x 10^4	4837 x 10^1
Global Efficiency	3.187 x 10^{-3}	1.532 x 10^{-2}	3.533 x 10^{-5}
Average Local Efficiency	1.616 x 10^{-1}	2.200 x 10^{-1}	0.000
Radius	7.000	1.000	4.000
Diameter	25.00	1.000	5.000
Density	1.745 x 10^{-4}	1.532 x 10^{-2}	2.067 x 10^{-5}

- The root node in the original DAG is the root node of the Transitive Closure and Dominant Tree representations.
- The number of nodes in the Transitive Closure and Dominant Tree representations do not exceed the number of nodes in the original DAG.
- The number of nodes in the Dominant Tree representation is equal to the number of nodes in the original DAG.
- The number of edges in the Dominant Tree representation do not exceed the number of edges in the original DAG.
- The number of edges in the Transitive closure representation do exceed the number of edges in the original DAG.
- For the Transitive Closure representation, the root node should have a number of edges equal to 1 minus the number of nodes.
- The diameter and radius of the Transitive Closure representation are both 1.

VI. Future Work

VII. Conclusions

This work presented and implemented a methodology for obtaining violation priorities in a compliance graph. Specifically, this work analyzed and validated the results on three example networks: an automotive maintenance example, a HIPAA example, and an OSHA 1910H example. Each network centrality metric provides unique insight on the topological information in a compliance graph, and three of the metrics (Katz (Section IV-D), Adapted PageRank (Section IV-E), and Percolation (Section IV-F)) are also able to work with the embedded information of the compliance graph. Each unique scoring of each compliance graph node from the centrality metrics are then aggregated and processed as part of the work presented in Section IV-G. The results were validated as part of the validation process shown in Section IV-I. These results are provided in the form of violation priorities under constraints, and showcase significant savings in terms of computations due to the implemented solutions for each centrality metric. Additional computational savings will be carried forward in future analysis techniques due to the significant reduction of nonzero elements for each example network.

In addition, this work presented and implemented two transformation options for compliance graphs. The transitive closure transformation presented in Section V-A is useful

for determining all possible routes for noncompliance given unlimited time and resources. By reducing all chains of events to single-step reachability, new analysis techniques could be investigated to focus on immediate importance. The dominant tree transformation presented in Section V-B is useful for providing a graph structure based on information flow. This transformation alters the original structure of the compliance graph, leading to a new hierarchy of nodes based on dominance. Using this transformation, the importance of nodes can be reexamined to determine how importance differs when information flow is refined. The results were validated as part of the validation process shown in Section V-D. The results in Section V-C showcase notable differences in network properties, which allow for new investigations to uncover further information from future analysis techniques.

REFERENCES

[1] J. Hale, P. Hawrylak, and M. Papa, "Compliance Method for a Cyber-Physical System." U.S. Patent Number 9,471,789, Oct. 18, 2016.

[2] N. Baloyi and P. Kotzé, "Guidelines for Data Privacy Compliance: A Focus on Cyberphysical Systems and Internet of Things," in *SAICSIT '19: Proceedings of the South African Institute of Computer Scientists and Information Technologists 2019*, (Skukuza South Africa), Association for Computing Machinery, 2019.

[3] E. Allman, "Complying with Compliance: Blowing it off is not an option.," *ACM Queue*, vol. 4, no. 7, 2006.

[4] S. Jha, O. Sheyner, and J. M. Wing, "Two formal analyses of attack graphs," *Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15*, pp. 49–63, 2002.

[5] L. Wang, S. Noel, and S. Jajodia, "Minimum-cost network hardening using attack graphs," *Computer Communications*, vol. 29, pp. 3812–3824, nov 2006.

[6] T. Islam and L. Wang, "A heuristic approach to minimum-cost network hardening using attack graph," *2008 New Technologies, Mobility and Security*, pp. 1–5, 2008.

[7] F. Chen, L. Wang, and J. Su, "An efficient approach to minimum-cost network hardening using attack graphs," in *Proceedings of the 2008 The Fourth International Conference on Information Assurance and Security*, IAS '08, (USA), pp. 209–212, IEEE Computer Society, 2008.

[8] T. Gonda, T. Pascal, R. Puzis, G. Shani, and B. Shapira, "Analysis of attack graph representations for ranking vulnerability fixes," in *GCAI-2018. 4th Global Conference on Artificial Intelligence* (D. Lee, A. Steen, and T. Walsh, eds.), vol. 55 of *EPiC Series in Computing*, pp. 215–228, EasyChair, 2018.

[9] M. Li, P. Hawrylak, and J. Hale, *A System for Attack Graph Generation and Analysis*. PhD dissertation, The University of Tulsa, Tulsa, OK, 2021.

[10] M. Li, P. Hawrylak, and J. Hale, "Strategies for practical hybrid attack graph generation and analysis," *Digital Threats*, Oct 2021.

[11] K. Guo and B. Mohar, "Hermitian Adjacency Matrix of Digraphs and Mixed Graphs," *Journal of Graph Theory*, vol. 85, 2017.

[12] P. V. Mieghem, "Directed graphs and mysterious complex eigenvalues," 2018. Delft University of Technology.

[13] R. A. Brualdi, "Spectra of Digraphs," *Linear Algebra and Its Applications*, vol. 432, pp. 2181–2213, 2010.

[14] M. Ashtiani, A. Salehzadeh-Yazdi, Z. Razaghi-Moghadam, H. Hennig, O. Wolkenhauer, M. Mirzaie, and M. Jafari, "A systematic survey of centrality measures for protein-protein interaction networks," *BMC Systems Biology*, vol. 12, p. 80, July 2018.

[15] M. Newman, *Networks: An Introduction*. Oxford University Press, 2010.

[16] T. G. Authors, *Graphviz: Graph Visualization*, 2023. Graphviz release 8.1.0 [Online]. Available:https://www.graphviz.org/.

[17] G. Csárdi, T. Nepusz, V. Traag, S. Horvát, F. Zanini, D. Noom, and K. Müller, *Igraph: Network Analysis and Visualization in R*, 2023. R package version 1.5.1.

[18] U. Brandes, "A faster algorithm for betweenness centrality*," *The Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[19] L. Katz, "A New Status Index Derived From Sociometric Analysis," *Psychometrika*, vol. 18, pp. 39–43, March 1953.

[20] M. Ogura and V. M. Preciado, "Katz centrality of markovian temporal networks: Analysis and optimization," *2017 American Control Conference (ACC)*, pp. 5001–5006, 2017.

[21] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 3rd ed., 1999.

[22] L. Stankovic, M. Dakovic, A. B. Bardi, M. Brajovic, and I. Stankovic, "Fourier analysis of signals on directed acyclic graphs (dag) using graph zero-padding," 2023. arXiv:2311.01073.

[23] B. Seifert, C. Wendler, and M. Püschel, "Causal fourier analysis on directed acyclic graphs and posets," 2023. arXiv:2209.07970.

[24] H. D. Macedo, "Gaussian elimination is not optimal, revisited," *Journal of Logical and Algebraic Methods in Programming*, vol. 85, no. 5, Part 2, pp. 999–1010, 2016.

[25] V. V. Williams, Y. Xu, Z. Xu, and R. Zhou, "New bounds for matrix multiplication: From alpha to omega," 2023.

[26] J. Alman and V. V. Williams, "A refined laser method and faster matrix multiplication," 2020.

[27] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1, pp. 107–117, 1998. Proceedings of the Seventh International World Wide Web Conference.

[28] T. Agryzkov, J. L. Oliver, L. Tortosa, and J.-F. Vicent, "An algorithm for ranking the nodes of an urban network based on the concept of pagerank vector," *Appl. Math. Comput.*, vol. 219, pp. 2186–2193, 2012.

[29] M. Piraveenan, M. Prokopenko, and L. Hossain, "Percolation centrality: Quantifying graph-theoretic impact of nodes during percolation in networks," *Plos One*, vol. 8, pp. 1–14, 01 2013.

[30] S. De, M. S. Barik, and I. Banerjee, "A percolation-based recovery mechanism for bot infected p2p cloud," in *Proceedings of the 20th International Conference on Distributed Computing and Networking*, ICDCN '19, (New York, NY, USA), pp. 474–479, Association for Computing Machinery, 2019.

[31] A. Chandramouli, S. Jana, and K. Kothapalli, "Efficient parallel algorithms for computing percolation centrality," in *2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, pp. 111–120, 2021.

[32] A. Hagberg, P. J. Swart, and D. A. Schult, "Exploring network structure, dynamics, and function using networkx," Available: https://www.osti.gov/biblio/960616.

[33] K. Ushey, J. Allaire, and Y. Tang, *Reticulate: Interface to 'Python'*, 2023. R package version 1.28. Available: https://CRAN.R-project.org/package=reticulate.

[34] G. Audrito, D. Pianini, F. Damiani, and M. Viroli, "Aggregate centrality measures for iot-based coordination," *Science of Computer Programming*, vol. 203, p. 102584, 2021.

[35] C. Li, L. Wang, S. Sun, and C. Xia, "Identification of influential spreaders based on classified neighbors in real-world complex networks," *Applied Mathematics and Computation*, vol. 320, pp. 512–523, 2018.

[36] H. Mo and Y. Deng, "Identifying node importance based on evidence theory in complex networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 529, p. 121538, 2019.

[37] A. Madotto and J. Liu, "Super-spreader identification using meta-centrality," *Scientific Reports*, vol. 6, Dec. 2016.

[38] J. G. Kemeny, "Mathematics without numbers," *Daedalus*, vol. 88, no. 4, pp. 577–591, 1959.

[39] H. P. Young and A. Levenglick, "A consistent extension of condorcet's election principle," *SIAM Journal on Applied Mathematics*, vol. 35, no. 2, pp. 285–300, 1978.

[40] M. Li, P. Hawrylak, and J. Hale, "Combining OpenCL and MPI to support heterogeneous computing on a cluster," *ACM International Conference Proceeding Series*, 2019.

[41] K. Zeng, "Cyber Attack Analysis Based on Markov Process Model," Master's thesis, The University of Tulsa, Tulsa, OK, 2017.

[42] R. T. Prosser, "Applications of boolean matrices to the analysis of flow diagrams," in *Papers Presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '59 (Eastern), (New York, NY, USA), p. 133–138, Association for Computing Machinery, 1959.

[43] V. Latora and M. Marchiori, "Efficient behavior of small-world networks," *Phys. Rev. Lett.*, vol. 87, p. 198701, Oct 2001.