Analysis of Graph Layout Algorithms for Use in Command and Control Network Graphs

THESIS

Matthew Stone, First Lieutenant, USAF

AFIT-ENG-MS022-S-039

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS022-S-039

Analysis of Graph Layout Algorithms for Use in Command and Control Network Graphs

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Matthew Stone, B.S.E.E.

First Lieutenant, USAF

August 31, 2022

AFIT-ENG-MS022-S-039

Analysis of Graph Layout Algorithms for Use in Command and Control Network Graphs

THESIS

Matthew Stone, B.S.E.E.
First Lieutenant, USAF

Committee Membership:

Mark Reith, Ph.D
Chair

Lt Col Wayne Henry, Ph.D
Member

Maj Richard Dill, Ph.D
Member

AFIT-ENG-MS022-S-039

Abstract

This research is intended to determine which styles of layout algorithm are well suited to Command and Control (C2) network graphs to replace current manual layout methods. Manual methods are time intensive and an automated layout algorithm should decrease the time spent creating network graphs. Simulations on realistic synthetically generated graphs provide information to help infer which algorithms perform better than others on this problem. Data is generated using statistics drawn from multiple real world C2 network graphs. The three algorithms tested against this data are the Spectral algorithm, the Dot algorithm, and the Fruchterman-Reingold algorithm. The results include a multiple objective statistics designed to inform on the algorithms performance in both aesthetic characteristics defined in literature, as well as some characteristics defined by the research sponsor. The results suggest that the Dot algorithm performs better with respect to the sponsor defined characteristics, whereas the Fruchterman-Reingold algorithm performs better on aesthetic characteristics. Due to the immediate need for an improved layout method, the Dot algorithm is recommended for use in C2 network graph applications as it best replicates the current manual layout style.

Table of Contents

Page

List of Figures

List of Tables

Analysis of Graph Layout Algorithms for Use in Command and Control Network Graphs

I. Introduction

This chapter includes an overview of the research including objectives and hypothesis as well as brief overviews of the contents of following chapters

1.1 Research Overview

Graphical visualizations of Command and Control (C2) network data are created to disseminate and analyze intelligence on adversary systems. The current process involves the manual creation of graphs using off-the-shelf software, thus limiting the number of graphs produced and the amount of data expressed. The C2 network problem is challenging because of overlapping complex issues. Analysts have immediate needs for a new visualization tool that can automate generation of graphs from existing data sources in a style suitable for existing customers, however any solution will also need to account for future data scaling. Additionally, analysts desire the ability to harness non-traditional analysis tools such as querying or AI/ML in the future. This design study of available layout algorithms will address some of these problems. As a part of this design study, an evaluation benchmark system was created for an objective comparison of visualizations. The benchmark tool and the layout algorithm that most fits the sponsor's needs will be recommended for future C2 network graph creation.

1.1.1 Research Objectives

The primary objective is improving the processes of creating C2 network visualizations with an appropriate automated layout algorithm. Additional objectives of this research are to design an unclassified benchmark evaluation process for C2 Network layout algorithms and perform an evaluation of available layout algorithms. The creation of an evaluation tool and benchmark data will allow for completion of the primary objective while also resulting in an efficient method for evaluating layouts in the future.

1.1.2 Hypotheses

The first hypothesis asserts that automated layout algorithms will significantly improve C2 Network creation speed while providing an acceptable approximation of human generated structures. Creation speed is improved by automation because manual methods can take an analyst hours to complete.

The second hypothesis is that the Dot algorithm will be more appropriate than Spectral or Fruchterman-Reingold algorithms for the data and sponsor's desires. The reason for this is the dot algorithm prioritizes showing hierarchical relationships, which are prevalent in C2 network data. Other algorithms tested focus on other data characteristics that may not be as useful when visualizing C2 network data.

1.2 Document Overview

This document is organized as follows. Chapter II, Background, provides an overview of relevant background information, including a description of C2 Network Data, and a review of graph visualization research. Chapter III, Methodology, details the process of developing a platform for generating benchmark graphs, creating an evaluation system for resultant layouts, and the test of 3 different layout algorithms

using the evaluation system and data developed. Chapter IV, Results, presents the evaluation results for the layout algorithms tested. Finally, Chapter V discusses the conclusions drawn from the results.

1.2.1 Background and Literature Review

Chapter II contains a basic background of the graph domain before investigating research on layout methods and characteristics. The C2 network data is hierarchical, therefore, additional focus is placed on researching hierarchical graph layout methods. Layout methods described include explicit node based layouts, matrix layouts, hybrid layouts, among others. A review of evaluation methods in literature is included as well. The chapter concludes with a background on the C2 network data that will be used for this research.

1.2.2 Methodology

Chapter III is separated into three sections to include the creation of benchmark data, the creation of the evaluation methodology, and the experimentation using available layout algorithms. This chapter describes why synthetic data is necessary to avoid classified information leakage, and how that synthetic data will be sufficiently representative of the real data. Chapter III also includes a description on how the metrics evaluate the layout algorithms chosen. Finally the experimentation section shows how the layout algorithms are implemented and tested.

1.2.3 Results

Chapter IV includes the results from testing three layout algorithms against benchmark data of varying complexity. The results section describes the data for each benchmark data complexity level, as well as interpretations of the data to understand

which algorithms are most well suited to the C2 network data problem.

II. Background and Literature Review

This chapter contains a basic analysis of the graph domain before investigating research on layout methods and characteristics. Because the C2 network data is hierarchical, additional focus is placed on researching hierarchical graph layout methods. Layout methods described include explicit node based layouts, matrix layouts, hybrid layouts, among others. Because this research involves creating evaluation methodology background of literature on evaluation of graph layouts is also included. The chapter concludes with a background on the C2 network data that will be used for this research.

2.1 Graph Domain

A graph is often defined as a set of nodes an edges written as:

$$G = \{V, E\} \tag{1}$$

G consists of a finite set of V that is not empty and set E consisting of ordered or unordered pairs in the form of (a,b) such that a element V and b element of V. The elements of V are vertices or nodes, and the elements of E are called edges. This interpretation of graphs represents data sets that include information on relationships and connections between data within the set. A graph can be undirected or directed, meaning the edges can be omni-directional relationships between a and b or a relationship that has a distinct start a and end b.[1][2]

Graphs can represent a wide spectrum of data relationships. Nodes can be any number of entities such as locations, people, or resources. Edges can show relationships including distances, familial relationships, price, and much more. McCulloch et al. defines a network graph as "a graph with a finite set of actors and the relation

or relations defined between them". A social network is an example of a network graph. In a social network the finite set of nodes can be a group of people and the relationships can be whether or not they are friends on a social network site. [1]

2.1.1 Graph Representations

Visualizations of graph data take many forms. Two of the most simple ways of visualizing graphs are shown in Figure 1. The link node diagram shows the directed edges as arrows pointing between dots which represent nodes. The same information is contained in the adjacency matrix A. An adjacency matrix is a $n \times n$ matrix where n is the number of nodes in a graph, in a graph each matrix element signifies the presence of an edge between vertices indexed at n A row is all the connections starting from a specific node, and a column is all the connections ending in a specific node. In Figure 1, the edges for the node labeled u are seen in the first row and first column. It is important to note that in an undirected graph edges will be represented twice because both nodes are start and endpoints. Additional methods to visually show these relationships are described in Section 2.2.3.

One of the unique aspects of graphs is that nodes do not have a defined order,

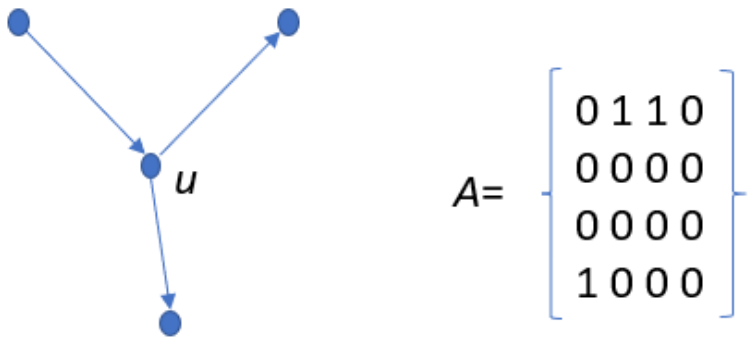

Figure 1: A directed graph shown in a simple link node diagram(left) and an adjacency matrix(right) node u corresponds to the first row and column in adjacency matrix A

meaning that the same nodes and edges seen in Figure 1 could be described in an adjacency matrix where u is the fourth row and column, or a link node diagram where u is in the same location as the top right node. The lack of defined ordering in graphs presents a wide array of possibilities in visualization layouts.

A variety of layouts can draw explicit link and node graphs. Different layouts are better suited to different domains. Potential layout style choices include radial vs grid based layouts, freedom of node placement, the density of the graph, whether the data is static or dynamic, and if the representation will be in three dimensions.[3] Furthermore aesthetic considerations may be important for clarity and readability. Some of these considerations are [3][4]:

- Minimal number of edge crossings: avoid letting edges intersect one another
- Minimal drawing area or efficient use of space: keep all nodes and edges within a tight cluster
- Minimal number of bends: edges are linear
- Uniform and short edge lengths: edge lengths are consistent and not longer than necessary
- Maximizing symmetry: distribution of nodes and edges follows a pattern
- Minimal overlapping nodes: nodes do not overlap one another

These considerations are not rules, often exhibiting trade-off relationships. For example, observe two illustrations of the same graph with minimized edge crossings in Figure 2a or minimized edge lengths and drawing area in Figure 2b. Both are acceptable drawing solutions and the choice made is dependent upon the needs of the user and the complexity of data represented. In Figure 2b the two black nodes are

placed closer together which may be important because the nodes have similar meaning and a very close relationship, whereas Figure 2a may be more useful to illustrate the centrality of the node in the middle of the graph.

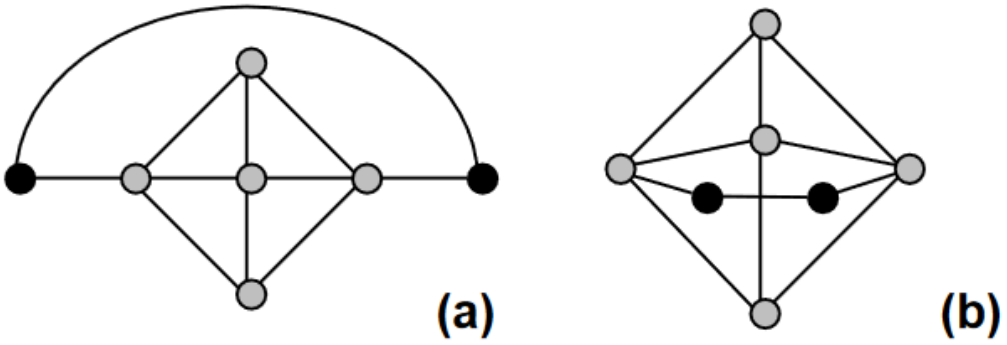

Figure 2: Two drawings of the same graph with different aesthetic considerations in mind

2.2 Hierarchy Layouts

Hierarchies, trees, multi-layer graphs, or *directed acyclic graphs* all describe the same style of drawing layout. In this style, nodes are oriented into *levels* defined by the number of edges from the root node. In foundational research an acyclic graph G of this style was written as:

$$G = \{V, I, A\} \tag{2}$$

In this equation, the set of edges is divided into both I and A. The subset I represents the inclusion edges or the edges between a nodes one level apart, and A represents the adjacency edges that connect nodes that are not one level apart. [5]

Certain types of data are well suited to a hierarchy style. Directed acyclic graphs where the edges represent a relationship of subordination are ideal for hierarchy styles because they have definitive start and end points without cycles. An advantage of hierarchy visualization is that additional information is often included in the layers of the graph. A familiar example of hierarchies is a family tree, in which each layer of the graph conveys generational information based on distance from the source or root. Even within the bounds of a hierarchy, different styles of visualization exist, the traditional link and node charts, matrix formats, a combination of styles, and others.[6]

2.2.1 Hierarchal Clustering

Many processes and algorithms can form hierarchies from graph data using features such as clusters, shortest path distances, and more. Hierarchal layout algorithms determine the layers that nodes belong in, or simply minimize the edges crossings in a layout. In this section, prominent algorithms are introduced to better understand the current landscape of hierarchy layouts.

Several graph layout algorithms can determine hierarchical clusters or layers, effectively deciding which layer nodes belong in. In a directed graph without cyclical adjacency edges determining hierarchical clusters can be simple process starting from a source node and building edge by edge, layer by layer. However often graphs exhibit cycles or are not conveniently directed graphs, these algorithms attempt to solve this problem.

Sugiyama proposed a hierarchal layout algorithm that focuses on both determining hierarchy from a data set and laying out the nodes in a way that shows that hierarchy. In hierarchization steps, the algorithm assigns levels to nodes based on the inclusion edges, when an adjacency edge prevents level assignment i.e. creates a cycle, the direction of the edge is reversed for layer assignment steps. [5]

Eiglsperger improves the Sugiyama algorithms by accounting for adjacency edges that span multiple layers by adding a 'dummy vertex' to the layers in between, in the final rendering these dummy vertices are turned into bent edges. In order to maintain drawing conventions the number of dummy vertices is minimized. [7]

The Dynadag algorithm developed by North, applies a different methodology with the same steps. Instead of reversing edges to determine level the algorithm applies a cost for the drawing different types of adjacency edges. This algorithm attempts to minimize this cost in it's layering steps by forcing nodes to span multiple layers or moving a node up or down layers to change a few adjacency edges into inclusion edges and vice versa. Ultimately the final layering will be one that has the most inclusion edges and least adjacency edges that increase the distance of edges, bends in edges, or violate other drawing constraints. [8]

2.2.2 Node Based Layouts

A trivial hierarchical representation is a tree structure where the top layer is commonly referred to as a root and the lower layers as leaves. It is possible to form Most graph data sets into multiple trees, by picking a root node and separating the rest of the nodes into layers based on the number of edges between them and the root. An example of a basic edge and node tree is shown in Figure 3. In this hierarchy node 22 serves as the root and the other nodes below expand into leaves. This is a simple tree layout without any of the adjacency edges discussed in Section 2.2.[5][6][9]

A node based layout has a set of algorithms that attempt to follow the drawing constraints of minimizing edge crossings, edge lengths, and bends. We see similar algorithms in Sugiyama, Eipsleberger and North. These algorithms first order the nodes in a way that minimizes edge crossings and then will space the nodes in order to keep edge lengths as short as possible. [5][7][8]

A tree graph is the simplest way to show a hierarchy with a small number of adjacency edges. However as the number of adjacency edges increases the tree becomes quite cluttered and difficult to visualize. [10]

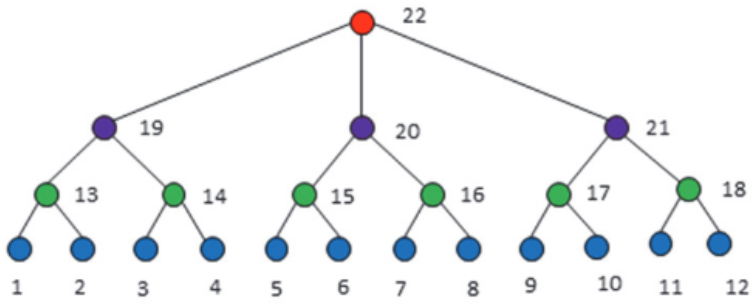

Figure 3: A simple example of a tree graph without any adjacency edges

2.2.2.1 Bundling

A solution to cluttered hierarchies involves edge bundling. This method reduces clutter by forcing the adjacency edges to bend and overlap where possible. The trade off for violating the number of bends constraint is a potential increase in readability. In Figure 4 we can see the process for bundling edges. In Figure 4(a) an example edge is shown, in Figure 4(b) the path that the bundled spline will take is shown and finally the spline is drawn in Figure 4(c). This spline will be followed similarly for any adjacency edge that passes along any of the edges included in this spline. An example of multiple bundled adjacency edges is shown in Figure 4(d) and (e); this example demonstrates how edge bundling can reduce the clutter of adjacency edges significantly.[10] [11]

2.2.3 Other Layouts

In addition to the simple trees described by explicit edges and nodes other methods can represent hierarchies. One method is depicting inclusion edges by simply drawing one node within the other, another is through adjacency matrices, and other approaches use hybrid techniques. The Sugiyama compound graph layout is a stan-

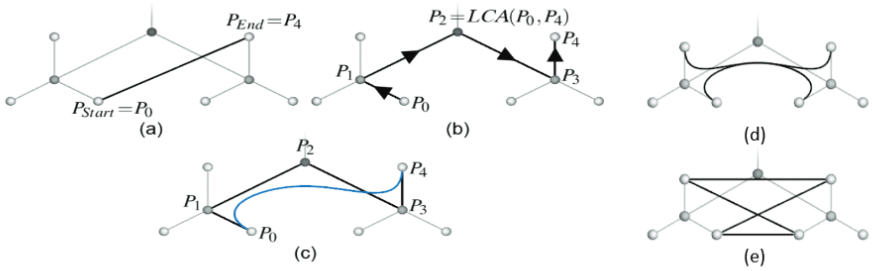

Figure 4: (a) An example adjacency edge to be bundled (b) The path of edges that the bundle will follow (c) The bundled adjacency edge (d) An example set of adjacency edges to be bundled (e) The resulting bundled splines

dard layout where inclusion edges show nodes within one another. The adjacency edges are drawn between the nodes that they connect. An example of the Sugiyama layout is shown in Figure 5. The rectangles represent a node and the larger rectangles are the nodes at the top of the hierarchy, the smaller ones are lower in the hierarchy. This method shows inclusion relationships effectively, but can sometimes make it difficult to compare nodes that are on the same layer, and can become very complex with massive graph data sets. [5] [6]

Another layout style is called implicit edge representation or icicle diagrams. In these maps an inclusion edge is not drawn but rather shown by the child nodes being smaller and attached to their parent nodes. Figure 6 shows a complex icicle diagram from Beck et al. The icicle layout has challenges visualizing adjacency edges and often may require edge bundling or an interactive nested layering. [12] [13]

Adjacency matrices show a large amount of edge data in a concise area. In order to employ adjacency matrices in a hierarchy a hybrid layout may be necessary. An example of a matrix hybrid layout is the Tree Matrix developed by Rufiange et al[14]. shown in Figure 7. This layout style shows the inclusion edges in a nested compound graph similar to the Sugiyama layout, while depicting many of the complex edge connections in the adjacency matrix format. Matrix format may be an organized way to view edges, but limits in how the hierarchy is interpreted in analysis.

Modern layout formats can employ elements from link and node diagrams, nested compound graphs, icicle diagrams, and hybrid techniques. Most layout research is quite domain and problem specific. Finding the correct layout style depends on the user needs, the intent of the visualization, and the complexity of the graph data. [6]

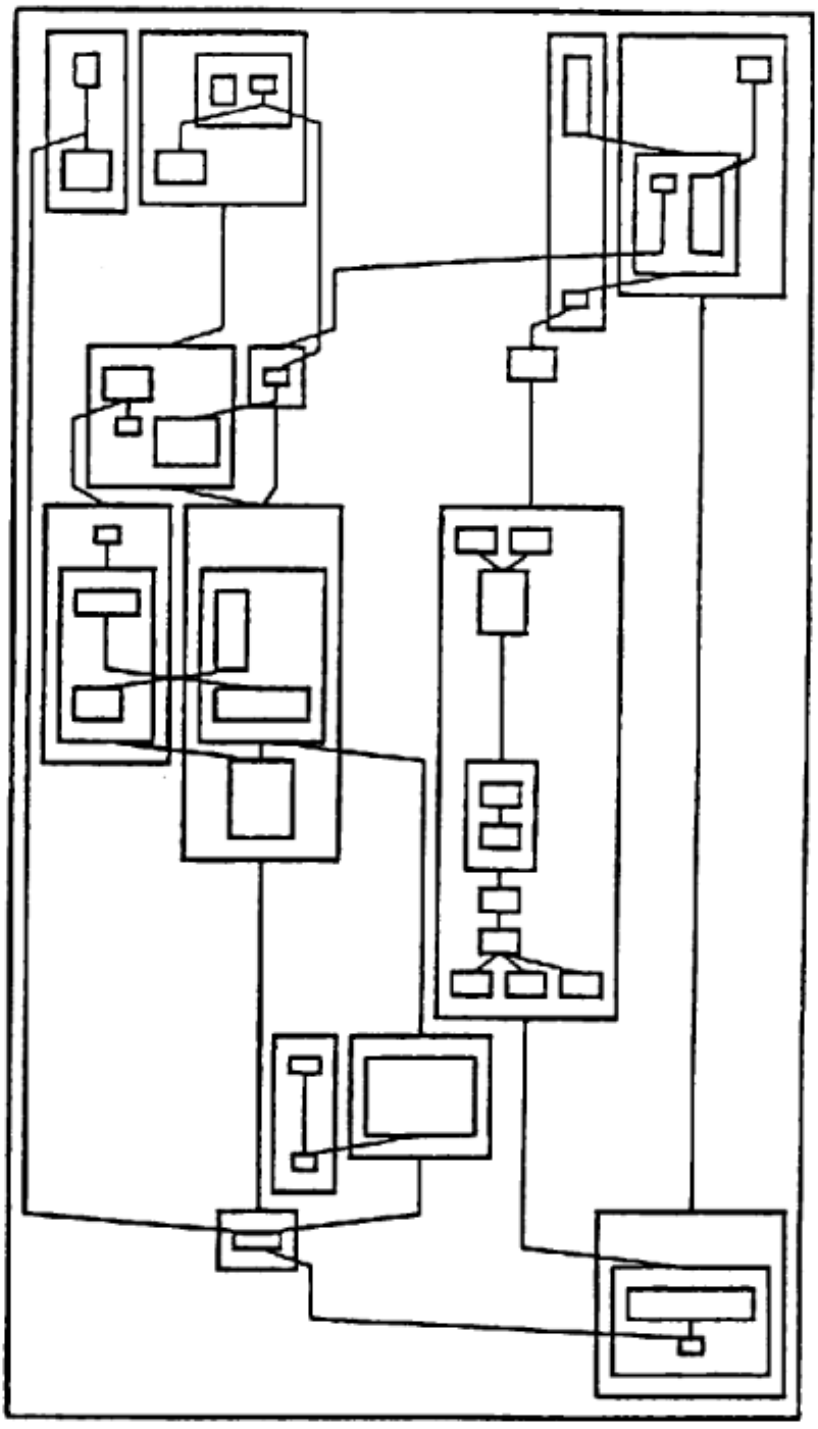

Figure 5: A multilayer hierarchy drawn in the Sugiyama layout style

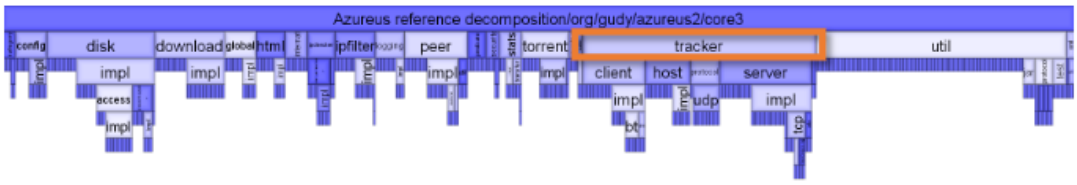

Figure 6: An example Icicle diagram

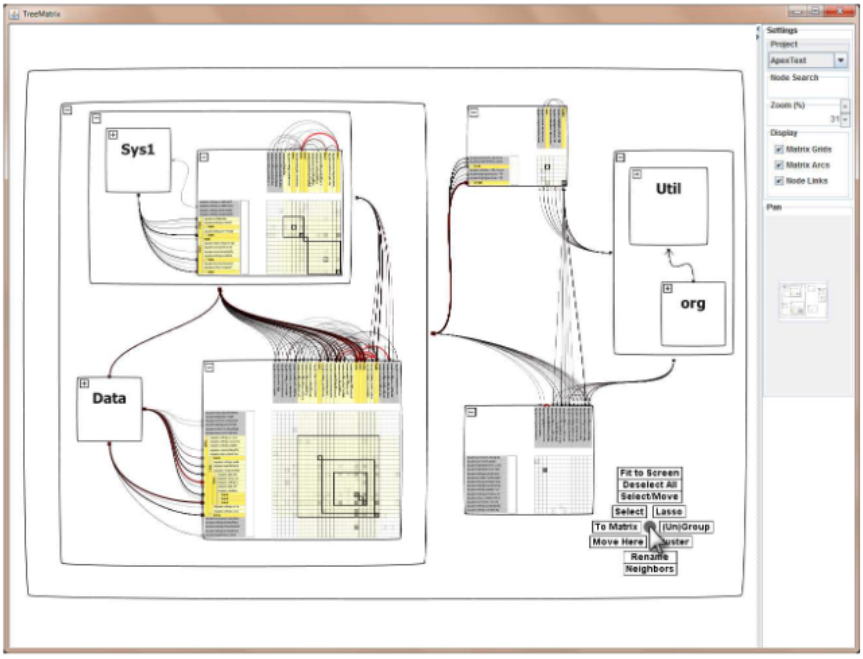

Figure 7: An example hybrid layout using adjacency matrices

2.3 Related Work

Related works include research on command and control (C2) and the analysis of command and control systems with visualization. Related works on hierarchy visualization in other domains include medicine, genetics, and machine learning.

2.3.1 Command and Control Visualization

Command and Control (C2) often expanded in military uses to include communications, computers, intelligence, surveillance and reconnaissance (C4ISR) is the study of the systems that enable information flow and understanding between enti-

ties. These systems are often complex with many entities and interconnections[15]. Graphs are a method for visualizing complex C2 systems. Command and control systems can include a simple home thermostat with user inputs and thermometer sensors, to complex military supply chains and logistics.[15]

It is useful to create visualization of C2 systems to analyze the effectiveness, determine weaknesses, or because they are quite large and complex. Gonzalez et al. provides an example of C2 specific visualizations with a study on visualization of law enforcement C2 paired with crime statistics. The Colombian police force segmented into regional controls all with their own crime monitoring, creating a C2 domain that is dynamic and geographically complex. The researched visualization tools allowed for criminal fluctuations to be better analyzed by law enforcement.[16]

Another C2 visualization project is a social network visualization of all the people involved in an emergency response in England. Similar to the previous study this example intends to find ways to improve the current response of law enforcement and other emergency services by analyzing the C2 flow from the local emergency controller to individuals within fire and police departments. Similar to the previous study this focuses primarily on the mathematical analysis of the graph data, however to best understand these analyses graph visualizations show all the connections between the different agents. [17] [18]

Most command and control studies are focused on the analysis of systems with the intention of improving current information flow and effectiveness. However, a few researchers analyze the value of command and control visualization for intelligence applications. Kang et al. presents a study in which participants were pretending to be a government intelligence analyst identifying a terrorist plot. All participants were given a set of documents and some were also given access to a visualization tool that allowed graphical representations of relationships in the documents. The participants

given the visualization tool generally outperformed the other participants even given the fact that they were unfamiliar with the tool before the study. Notably in this study the tool that most participants had success with was the "list view" shown in Figure 8 which shows the data in a view similar to a matrix hybrid layout. This shows the potential effectiveness of representing C2 data in graphical format.[19]

2.3.2 Existing C2 Hierarchy Visualization Tools

Much of the related work in the C2 field does not focus on hierarchy visualisation. Hierarchies have been used heavily in other fields because they provide an effective way to show graphical data that has a parent/child relationship. Domains that have already begun use of hierarchies are medicine, biology, and machine learning. [20][21]

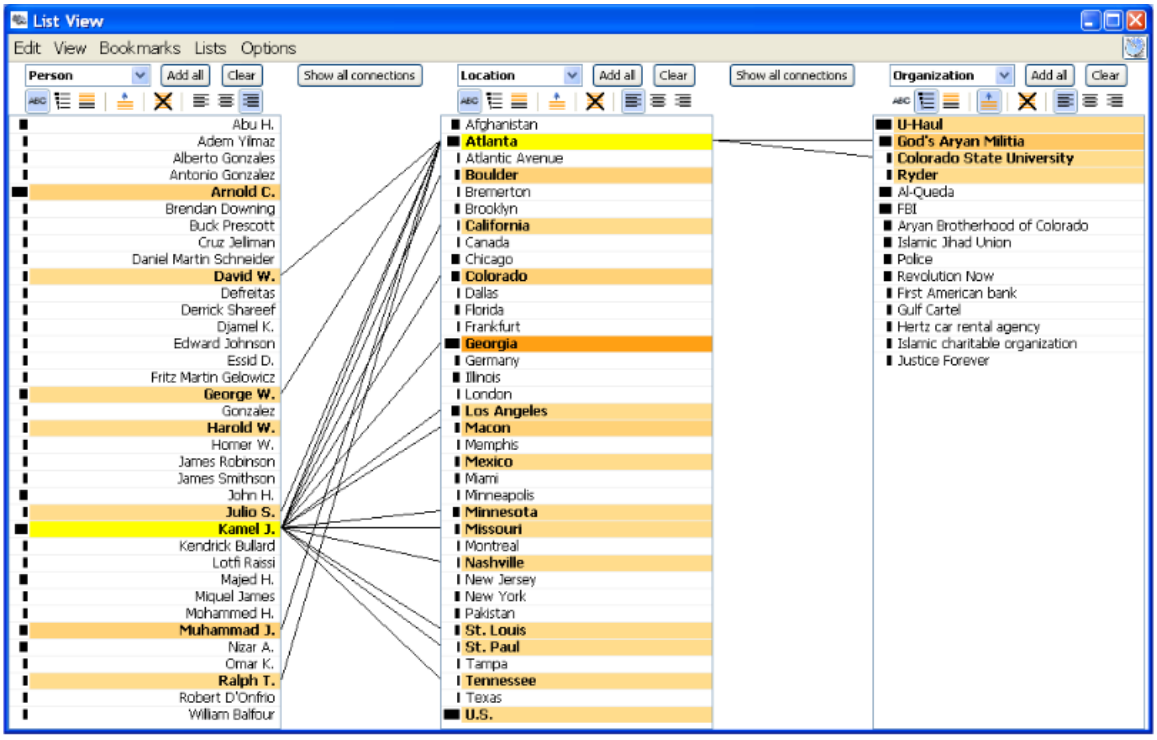

Figure 8: The list view tool used in the study by Kang et al: determined to be one of the most effective visualization tools from the study

2.4 Evaluation Techniques

Graph visualization researchers employ a variety of techniques when evaluating the effectiveness of a layout. One common technique is through evaluation of the layout via a usage study. Bourqui et al., Jing et al., and Zhang et al. utilise this method. These studies evaluate their tool or algorithm against a use case to demonstrate that the features they intended to employ work. These types of studies do not examine the effectiveness of a layout.[11][21][9].

Other research includes user based evaluation techniques such as a survey or an employment test. One example of a user based technique is Rufiange et al.'s test providing users with the new layout and gathering data on the time it takes a user to complete a set of tasks [14]. Other research that employed a user evaluation technique includes Holten and Kang et al.[10][19].

Fewer studies focus on simulation based research. Huang et al. analysed their PLANET algorithm by simulating against large datasets, and measuring execution time, and a few aesthetic characteristics. [22] Methods such as this one allow for a quick comparison of new techniques.

In 2019 Giovannangeli et al. proposed a new technique for evaluations. They cited the difficulties in avoiding bias, and gathering a statistically relevant amount of data from user studies as the reason for their techniques. Instead of having users evaluate a layout or visualization, deep neural networks could do the task more quickly and without bias. This research is the trend towards an objective repeatable visualization evaluation. Their methods involve generating data of varying complexity and testing the layout with their computer vision model to reproduce results from previous user evaluation style studies.[23]

2.5 Available Tools

Several tools implement a variety algorithms for their graph visualizations. A few open source tools include Neo4J, GraphViz and Gephi. These tools utilize specific layout algorithms alongside options to query the data. Neo4J utilises a force directed layout algorithm to visualise the user's data query. GraphViz is well known for their dot algorithm for directed acyclic graphs. Finally tools such as gephi can utilise multiple different layout algorithms to visualize the data in a variety of node-based styles.

2.6 C2 Network Graphs

C2 network graphs are created by the sponsor to visualize their C2 network data and perform additional analysis, share findings with customers, and collaborate across the DoD. These graph representations are created manually in Microsoft Visio taking on average 8 hours to complete using upwards of 3 GB of memory. An analysis of the classified data alongside discussions with the sponsor shows that a C2 network graph is connected by three types of edges, parent to child edges similar to traditional tree graphs, adjacent edges or edges connecting nodes within the same level, and skip-echelon edges, connecting a node to another at least 2 levels below in the hierarchy. The equation of a graph is modified here to include skip echelon edges (S) as a subset of adjacent edges.

$$G = \{V, I, A, S\} \tag{3}$$

Figure 2.6 shows an example C2 network graph created in Visio. This graph shows a skip echelon edge from the source node to a node at Level 2, an adjacent edge between a node in Level 1 to the adjacent node, and multiple parent to child

edges throughout.

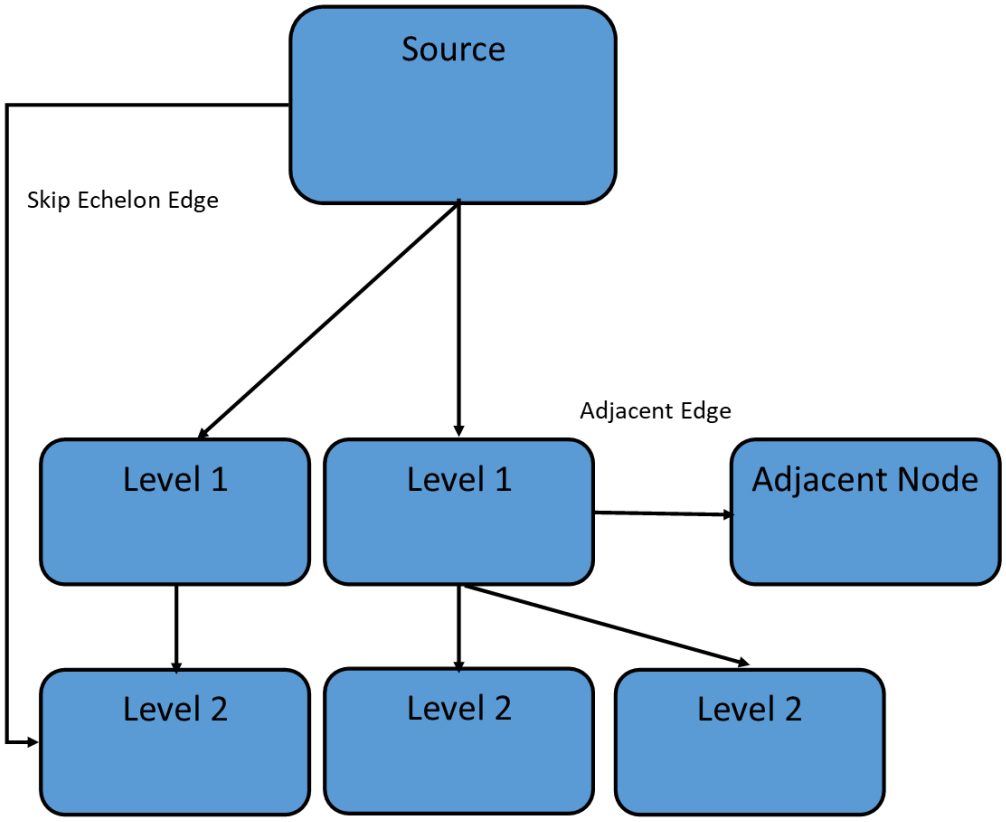

Figure 9: An example C2 network graph with skip echelon and adjacnet edges labeled, as well as the level for each node defined.

III. Methodology

This chapter describes the three major activities supporting this research to include synthetic data generation as described in Section 3.1, synthetic data evaluation as described in Section 3.2 and layout implementation as described in Section 3.3. The following sections elaborate on each of these respectfully.

The research methodology for this thesis is best illustrated in Figure 10. Figure 10 shows the beginning background research required, emphasized by the blue boxes, the tools and methods developed for this research shown in the yellow shapes, and the experimentation and analysis steps in the green and purple shapes. Partial sub-steps are shown via branches off of the main path.

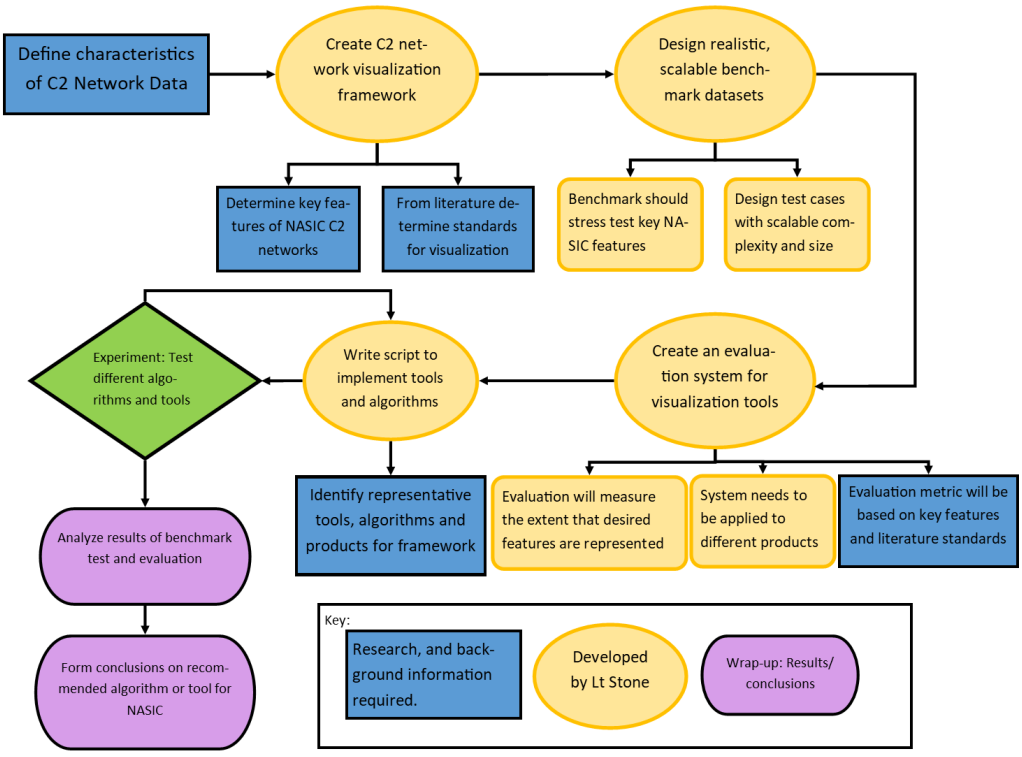

Figure 10: Research Methodology, including background research, Benchmark Data Generation, Evaluation System Design, Implementation of layout algorithms, the experiment, and final analysis

3.1 Data Generation

Synthetically generated data is necessary for two reasons: testing different benchmark features and protecting classified information. One key factor in creating the benchmark data is ensuring that the generated data represents the sponsor's data set. Using measurements and features of the sponsor's data will allow for a representative generated data that is unclassified.

A set of five C2 Network diagrams of varying size will be analyzed to create a baseline framework for data generation. Limitations of current visualization tools ensure the measurements must be taken by hand. From these five diagrams, three

types of edges are measured with respect to the source node's level in the hierarchy. The three edge types are children, adjacent edges, and skip-echelon edges. A function can generate a random representative graph which will be used as the baseline for future data set generation.

The preliminary data forms the basis for creating a graph. The process in Algorithm 1 describes how to use an $m \times n$ probability matrix to create a directed graph. Section 4.1 depicts statistics measured by level for each node in the sponsor's data. The probability matrix represents the likelihood of a node in each level to have a certain number of children. In the matrix, m represents the level and n is the possible number of children for each node in that level. Each node in a level has a probability that it will have $0, 1, 2, 3, ...(n-1), n$ children. By creating first a number, u, of nodes and looping through each node assigning edges based on the probability in the probability matrix, a graph will form. The number of edges is not set because of the randomness in the graph, however if all u nodes are used there will be at least v number of edges where $u = v$. The data generation algorithm should have a complexity O(N) as each node is only looped through once, with a maximum of 5 children.

Different graph complexities simulate different scenarios for analysis. A *simple*

Algorithm 1 Data Generation Algorithm

```
1: function DATA_GENERATION(Probability_Matrix)
2:     nodes[1 : u]
3:     counter ← 0
4:     for node in nodes do
5:         level ← shortest_path_length(node, nodes[1])
6:         #ofChildren ← random_selection(Probability_Matrix[Level, :])
7:         children ← range(counter + 1, counter + #ofchildren + 1)
8:         for child in children do
9:             assign edge(node, child)
10:            increment counter
11:
```

complexity will represent the average graph case as shown in Table 1. The *moderate complexity* will simulate the maximum graph case as shown in Table 1. Finally, the *high complexity* will simulate a use case with twice as many nodes, and edges as the current maximum to demonstrate the scaling potential of the algorithms tested.

The methods in this section can create graphs with approximately u nodes, a adjacency edges, and b skip-echelon source nodes. The graphs generated are based from the probability matrices measured from the sponsor's data. Measuring probability from the sponsor's data should cause the generated graphs to be representative of real classified data.

3.2 Data Evaluation

To quantitatively measure the effectiveness of a layout against the generated data calculations are necessary. Measurements will include euclidean distances between children and parents, euclidean distance between adjacently connected nodes, euclidean distance between skip-echelon connections, a validation check of each graph that the longest node is a skip-echelon connection, a validation check for the source node being identifiable, an r-coefficient for a line drawn through all nodes in the same level, and a validation that the nodes are stratified correctly by level.

Additionally, to compare the automated layouts to the current Microsoft Visio layouts, the execution time and memory usage will be measured. Acknowledging that all automated methods may have comparable execution times and memory usage, the comparison will primarily be between automated methods and Visio methods.

Measuring the distance between children and parents is intended to test how well the parent child relationships are shown. If a child is a great distance from its parent when compared to other similar edges in the graph, the reader may not understand it has the same relationship as other children. Similarly, measuring the distance between

two adjacent nodes could also inform how well a graph shows two nodes adjacent relationship. Additionally, if a graph contains a significant number of long edges the likelihood of edge crossings also may increase, which is a key aesthetic characteristic identified from the literature in Section 2.1. Acknowledging that different layouts may use different scales, it will be necessary to compare values using coefficient of variation, σ/μ, or the standard deviation normalized by the mean. This research uses coefficient of variation to compare values between layouts which use different scales for node placements. The algorithm for computation of values in one graph is shown in **??** which creates sets of edge distances from which max, averages, and coefficient of variation is calculated. The complexity of this algorithm is O(N) as it is directly related to the number of edges in all graphs tested.

Validating the hierarchy structure is also important to the sponsor. If a layout has skip-echelon edges and chooses to place nodes at *higher levels* (further from the source hierarchically) physically close to the source, then the graph would not be representative of current by hand layout methods, and could have potential to confuse a viewer. The following list of measures provides information on the hierarchy structure of a layout. Validating that the maximum edge is a skip echelon edge,

Algorithm 2 Layout Distance Evaluation Algorithm

function DISTANCEEVALUATION(Graph, Positions)

edgelist = *Graph.edges* ▷ edgelist is a set of (a,b) node pairs

for edge in edgelist **do**

if *edge_style* = *child*

child_list = append.distance.euclidean(position(a),position(b))

if *edge_style* = *adj*

adj_list = append.distance.euclidean(position(a),position(b))

if *edge_style* = *se*

se_list = append.distance.euclidean(position(a),position(b))

validating that the source is a maximum or central value, checking the regression fit of each level, and validating the stratification of each nodes by distance from the source, will provide enough information to infer whether or not the layout is adequately representing the hierarchy structure.

Algorithm 3 Layout Hierarchy Evaluation Algorithm

function HIERARCHYEVALUATION(Graph, Positions, child_list, adj_list, se_list)

source = Graph.nodes where level = 0

if max(se_list) ¿ max(child_list) and max(se_list) ¿ max(adj_list)

max_is_se = true

if (Positions(source) = Positions(max(y)) or Positions(min(y)) or Positions(min(x)) or Positions(max(x)) or Positions(median(x,y))

source_is_max_central = true

for level in Graph.nodedata.levels **do**

$R^2 = math.R^2(linear_regression(nodesinlevel)$

for node in level **do**

source_dist(node) = distance.euclidean(positions(node), positions(source))

level_dist(level) = avg(source_dist)

sorted_levels = sort(Graph.nodedata.levels, level_dist)

Stratification = % of sorted_levels = level

3.3 Implementation of Layouts and Experiment

This research will evaluate three primary layout styles, a tree style, a force directed style, and a radial style. The intent of testing these different layout algorithms is to verify which style of algorithm is most appropriate for C2 network visualization. In order to measure the metrics required for evaluation, the positional data for each node needs to be created using each layout algorithm. The algorithms used are implemented with Python using the *networkx* library. With a Python script, the algorithms can all be implemented on synthetic data to generate a dictionary of positions keyed by nodes.

The implementation script executes through the data generation, layout positioning, and data evaluation. The block diagram design for these processes is shown in Figure 11. In the block diagram the input signals for the experiment will be complexity and number of graphs. The complexity will inform the data generator how many nodes, the number of adjacent edges, and number of skip echelon source nodes as described in Section 3.1. The number of graphs input describes the size of the test data set, or how many example graphs should be generated for layout and evaluation. Once n graphs are created, each graph will be sent to three different layout algorithms which return n dictionaries keyed by node with an (x, y) coordinate pair (the node's position) as the value. Using the dictionaries of node positions and the edge data from the associated networks the data evaluation block will calculate the summary performance metrics as described in Section 3.2.

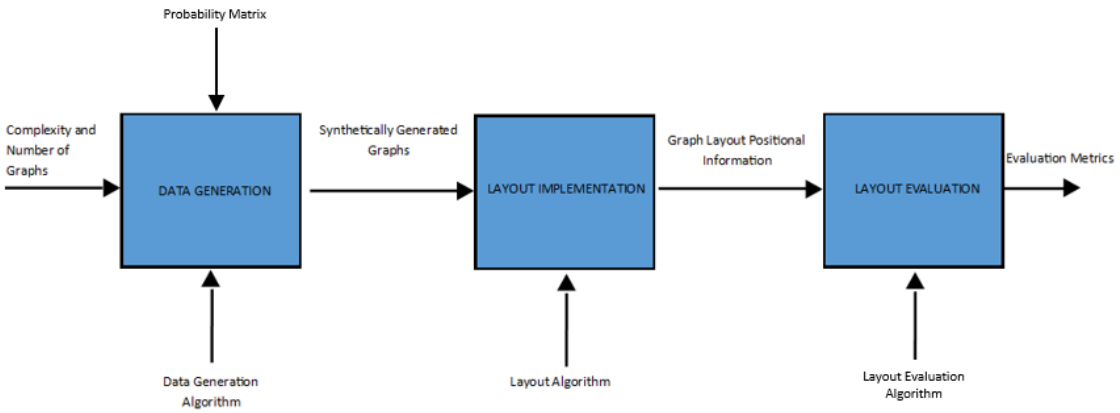

Figure 11: IDEF0 design for experiment implementation. Input signals are the desired number of graphs, and the complexity, the output is a series of evaluation metrics outlined in Section 3.2

IV. Results and Analysis

Preamble

Chapter IV includes the results from testing three layout algorithms against three levels of benchmark data. The results section describes the data for each level of benchmark data, as well as interpretations of the data to understand which algorithms are most well suited to the C2 network data problem.

4.1 Benchmark Generation Results

The benchmark generator is able to generate data for three different complexities. To create the benchmark generator, statistics were measured from a set of five real world graphs. Graphs were chosen in an attempt to sample a variety of C2 network graphs from one of the smaller graphs to the largest graph. The format of the existing graphs limits the number of graphs to sample because measurements are counted manually. A preliminary analysis of the data is necessary because the sponsor's data is sensitive. Tables 1 and 2 show the results of the preliminary analysis. In Table 1 the mean and maximum numbers of nodes and edges for a graph are shown, this information is a reference point for synthetically generated data. The Execution Time is an estimate agreed upon by subject matter experts who have created dozens of C2 network graphs in Visio. Memory usage is the average memory usage by Microsoft Visio with each of the 5 example graphs open. Execution time and memory usage are comparison points for the automation results against the manual methods.

Table 2 contains the probability matrix used to generate synthetic data as described in Section 3.1. Table 2 contains probabilities, rounded to the nearest .05 measured by counting the number of children of all nodes at each level. These probabilities contribute to generation of synthetic C2 network data.

Table 1: General Statistics of C2 Network Graphs from analysis of 5 real world graphs

Mean Number of Nodes	45.20
Mean Number of Edges	57.60
Maximum Number of Nodes	87
Maximum Number of Edges	105
Mean Adjacent Edges	10.4
Mean Skip Echelon Edges	4.2
Maximum Adjacent Edges	23
Maximum Skip Echelon Edges	13
Execution Time	8 hours
Memory Usage	3234 KB

Table 2: Probabilities for number of children by node level from analysis of 5 real world graphs

Number of Children / Node Level	0	1	2	3	4	5+
0	0	1	0	0	0	0
1	0	0.35	0.05	0.05	0.5	0.05
2	0	0	0.2	0.4	0.4	0
3	0.1	0.3	0.1	0.1	0.2	0.2
4	0.2	0.2	0.4	0	0.2	0
5	0.4	0.4	0.2	0	0	0
6	0.4	0.6	0	0	0	0
7	1	0	0	0	0	0

4.2 Benchmark Test Results

These results include three sets of 25 graphs each at simple, moderate, and high complexities as described in Section 3.1. The evaluation results against the simple graphs will primarily be used for comparison against hand-made Visio graphs for time and memory constraints. The performance results for moderate and high complexities compare algorithms to one another.

4.2.1 Simple Complexity Simulation Results

The execution time results for the three algorithms are in Table 3. The simple graphs are intended to represent the average C2 network graph created in Visio by the sponsor, so we compare the memory usage, and rendering time against analyst estimates for the average time and the average memory usage of a Visio representation of a C2 network. Table 3 shows that all of the automated algorithms are over 100,000x faster than manual creation in Visio, and operate with significantly less memory than the average Visio representation. While the automated layout algorithms all outperform manual creation in Visio, it is worth noting that at the simplest complexity the Spectral algorithm is 7.125x faster than the next fastest algorithm. The testing environment was Google Colab, which provides 12 GB of RAM.

Figures 12, 13, and 14 show a representation of a simple complexity example graph. In Figure 12, the characteristics of the dot layout are on display; the nodes are oriented into horizontal levels, and connected like a tree. Figure 13 shows the radial layout of the spectral algorithm. And Figure 14 shows the concise and efficient force directed layout of the Fruchterman-Reingold Algorithm.

4.2.2 Moderate Complexity Simulation Results

The results shown in Table 4 outline the performance metrics for each algorithm. In these tables the central column below each algorithm is most useful for comparing the performance of different algorithms. The average and maximum values are useful

Table 3: Evaluation Results for Dot, Spectral, Fruchterman-Reingold algorithms, and Microsoft Visio handmade representations using 25 graphs with simple complexity (approximately 45 nodes, 10 adjacent edges, 4 skip echelon edges)

Parameter	Dot	Spectral	F-R	Visio
Execution Time	0.060 s	0.008 s	0.055 s	12800 s
Memory Usage	69.289 kilobytes	192.703 kilobytes	359.255 kilobytes	3234 kilobytes

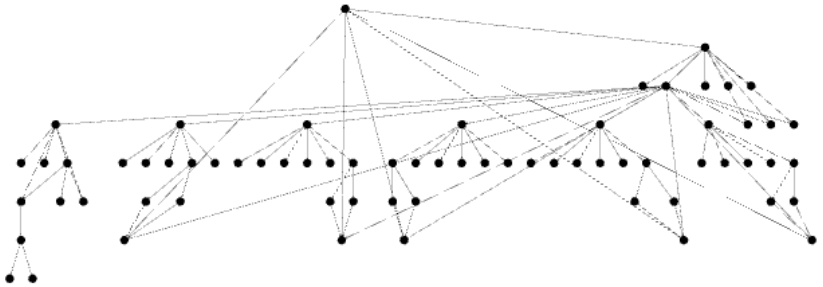

Figure 12: Simple Complexity example graph visualized using Gephi and the Dot layout algorithm. All edges Coefficient of Variation: 1.846

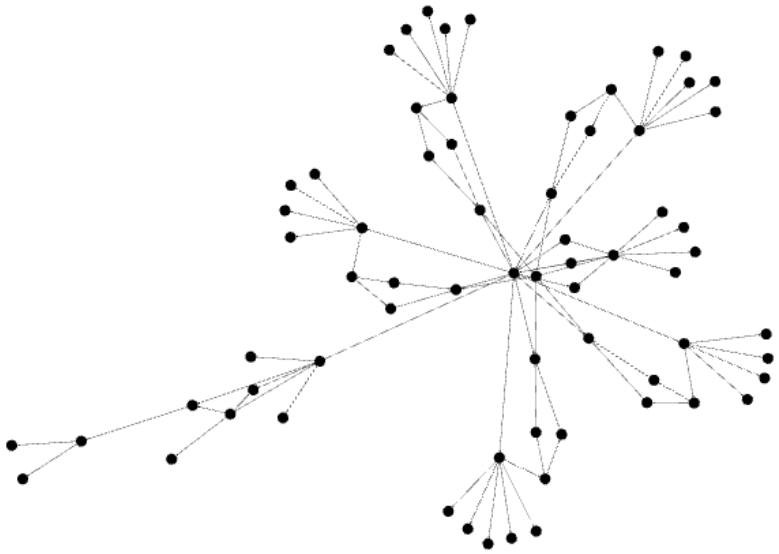

Figure 13: Simple Complexity example graph visualized using Gephi and the Spectral layout algorithm. All edges Coefficient of Variation: 0.9734

when comparing values within one algorithm, but since the graphs are all created on a different scale, the central column uses the normalized coefficient of variation or $\frac{\sigma}{\mu}$. Using the coefficient of variation allows for a comparison of how consistent the edge lengths are. A high coefficient of variation, for this research high references values greater than one, means that the edge lengths are inconsistent and also likely means that several edges are much longer than others which is an indicator of likely edge

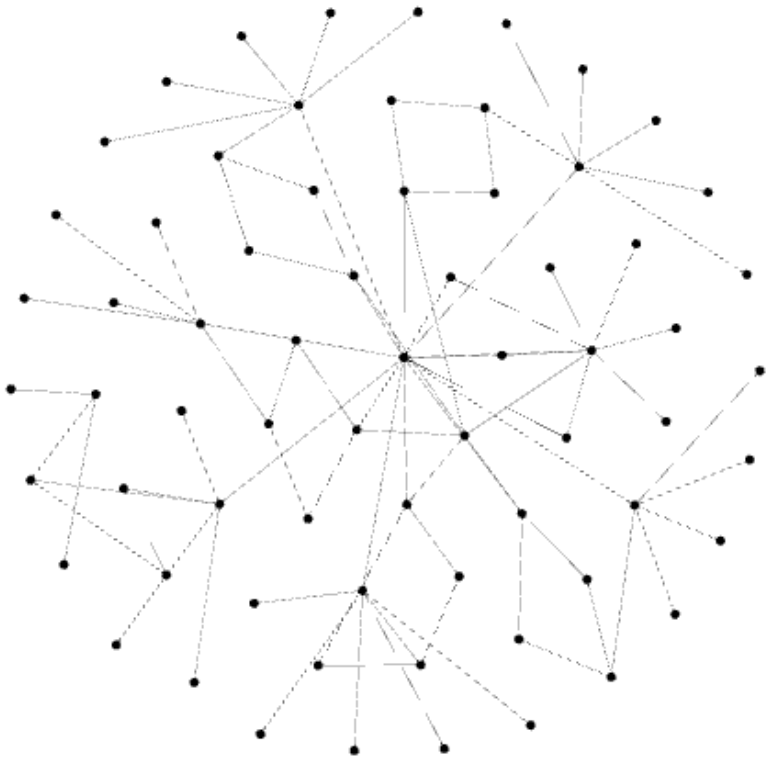

Figure 14: Simple Complexity example graph visualized using Gephi and the Fruchterman Reingold layout algorithm. All edges Coefficient of Variation: 0.562

crossings. For these results the tables are color coded to represent performance. At a coefficient of variations above one edge and node crossings become more prevalent, the space used for the graph increases, and edge length is inconsistent. For reference compare Figures 12,13, and 14, in which the Dot diagram posts a coefficient of variation for all edges of 1.846, with a significant amount of edge and node crossings, and the larges overall space used. The Fruchterman-Reingold algorithm layout has an overall coefficient of variation of 0.562, with much fewer edges crossed, and an overall more consistent edge length.

Table 4 shows that the Fruchterman-Reingold algorithm minimizes the variation of all edge lengths. Edge crossings are less likely in the Fruchterman-Reingold algo-

rithm, because of the limited variation in edge length. We can also observe that there is a large variation in length of child distances for the Dot and Spectral algorithm. This indicates that more than a few children are a great distance from their parents, an indicator that a parent child relationship may be difficult to infer based on the length of edges. Also, the majority of the network is comprised of child edges thus implying that the structure of the layout is highly variant. The data suggests that the aesthetic characteristics of consistent edge lengths and minimal edge crossings are not satisfied in the Dot and spectral algorithms. The maximum values are informative as well. The Dot algorithm performance suggests that skip echelon edges are consistently long from the low coefficient of variation and the high average value. We can infer from this data that the skip echelon edges in the Dot layout are likely spanning a large portion of the graph and crossing a significant amount of other edges. The final distance measurement is spacing, which measures the node's nearest neighbor, whether an edge exists or not. Spacing is a useful measure to understand how concise the graph is. The Dot algorithm has a significant coefficient of variation for spacing, and that means it is not using the space efficiently. Overall the Fruchterman-Reingold algorithm performs better in the aesthetic characteristics inferred from the distance measurements. Fruchterman-Reingold's algorithm efficiently spaces nodes for consistent edge lengths resulting in graphs that match many of the aesthetic characteristics from Section 2.1.1.

The results in Table 5 are intended to measure the layout's fit to C2 network graphs as defined by the sponsor. The first metric, "Max Edge Dist. is SE" is a check to see if the maximum edge distance is a skip echelon edge. The definition of a skip-echelon edge is that a connection from a superior node to another node that is at least two levels below; therefore the length of a skip echelon edge should be longer than the length of a child edge or adjacent edge. The dot algorithm performs better

because it prioritizes the child edges for layout purposes. Likewise the dot algorithm performs well in the next metric, "Source is Maximum or Central" which verifies if the source node is easily identified in the graph. The sponsor's desire to identify key nodes in the C2 network is often determined by the node's distance from the source; therefore an algorithm that presents an easily identifiable source on a (x, y) minimum, maximum or in the center of all nodes will allow for the quickest analysis. "Stratification" is a validation of all levels in each graph. Stratification checks how accurately the levels are distanced from the source such that each subsequent level is further from the source than the previous level. In the Dot algorithm, levels are defined by distance from the source, in the spectral algorithm levels are less defined by distance from the source, and in the Fruchterman-Reingold algorithm nodes in a level are not defined by distance from the source. The final measurement in Table 4 is the R^2 value for a regression drawn through all the nodes in a level. Similar to stratification, this metric helps understand how well a layout stratifies the nodes in a linear manner. Graphviz performs best in this metric because all nodes in the same level share the same y-coordinate. While Spectral satisfies the Stratification metric, it does not have a significant R^2 value because the nodes are spaced around the source not linearly. While the Fruchterman-Reingold algorithm satisfied many of the aesthetic characteristics identified in literature, the Dot algorithm has the most consistent performance for C2 network visualization styles. The spectral algorithm has moderate performance in both C2 network visualization metrics and aesthetic metrics. These results indicate the trade-off in aesthetic characteristics as defined by literature, and the C2 network visualization characteristics defined by the sponsor.

Table 4: Aesthetic Evaluation Results for Dot, Spectral and Fruchterman-Reingold Algorithms against 25 graphs with moderate complexity (Approximately 100 nodes, 25 adjacent edges, 10 skip echelon edges) Performance measured via coefficient of variation. Avg and Max values included for understanding results within layout. Red cells: Coefficient of variation above 1, yellow cells: coefficient of variation above or equal to 0.7, green cells: Coefficient of variation below 0.7

	Dot			Spectral			F-R		
Distance Metrics	Avg	C Var	Max	Avg	C Var	Max	Avg	C Var	Max
Child Dist.	199	1.22	1381.87	0.08	1.23	0.644	0.55	0.67	1.13
Adjacent Dist.	341	0.70	727.57	0.07	1.29	0.55	0.72	0.29	1.08
SE Dist.	818	0.47	1612.92	0.09	0.78	0.202	0.92	0.09	1.05
Spacing	103	1.85	1381.87	0.08	1.34	.644	0.64	0.3	1

Table 5: Sponsor's C2 network graph characteristic evaluation Results for Dot, Spectral and Fruchterman-Reingold Algorithms against 25 graphs with moderate complexity (Approximately 100 nodes, 25 adjacent edges, 10 skip echelon edges). Red cells: undesirable performance, values below 0.7 Yellow Cells: Moderate performance, values above 0.7 less than 0.9, Green Cells: Desired performance values above 0.9

	Dot	Spectral	F-R
Max Edge Dist. is SE	100%	12%	12%
Source is Maximum or Central	100%	24%	24%
Stratification	96%	72%	12%
Level R^2	1	0.23	0.004

4.2.3 High Complexity Simulation Results

The final test against high complexity graphs demonstrates how well each algorithm will scale against large numbers of nodes and complex edge connections. Recall from Section 3.1 the definition of high complexity includes those graphs, with approximately 200 nodes, 50 adjacent edges, and 20 skip echelon edges. High Complexity graphs are intended to simulate graphs that are too large for manual Microsoft Visio representations and provide an example for how the sponsor can utilise automated layouts to expand their visualizations. For this simulation, the previous level of complexity is a reference to understand how a change in complexity impacts the layout performance.

Table 6 demonstrates that the coefficient of variation varies slightly for all the algorithms when more nodes and edges are added to the graphs. Notably, the Dot algorithm continues to demonstrate poor performance on these characteristics and all of the metrics are increasing, aside from spacing which is understandable given that more nodes are on the graph and therefore a higher likelihood that nodes will have neighbors closer. Meanwhile, the Spectral and Fruchterman-Reingold algorithms show little change in coefficient of variation with additional nodes added. The variation of adjacent distances for the Spectral algorithm decreased significantly to fall closer to the other edge variations for this algorithm. The Dot algorithm shows no indication of worsening performance against C2 Network metrics, whereas the Spectral algorithm performance has begun to decline.

The high complexity results observed in Table 6 may be an indicator of a point where satisfying readability and aesthetic characteristics becomes more important. A graph with over 200 nodes will depict individual C2 relationships well, especially with the higher number of edge crossings, long edges, and inefficient space usage included in the Dot algorithm. With enough nodes added, the Dot algorithm will begin to deteriorate in readability, whereas the Fruchterman-Reingold and Spectral algorithms do not deteriorate as much. The other layouts can show clusters on large graphs in a way that GraphViz does not.

Table 6: Evaluation Results for Dot, Spectral and Fruchterman-Reingold Algorithms against 25 graphs with high complexity (Approximately 200 nodes, 50 adjacent edges, 20 skip echelon edges)

	Dot		Spectral		F-R	
Distance Metrics	Moderate	High	Moderate	High	Moderate	High
Child Dist.	1.22	1.65	1.23	1.21	0.67	0.73
Adjacent Dist.	.70	1.32	1.29	1.17	0.29	0.42
SE Dist.	0.47	0.68	0.78	0.91	0.09	0.31
Spacing	1.85	1.21	1.34	1.02	0.3	.25
Max Edge Dist. is SE	100%	100%	12%	20%	12%	4%
Source is Maximum or Central	100%	100%	24%	20%	24%	16%
Stratification	95%	100%	72%	51%	13%	4%
Level R^2	1	1	0.23	0.17	0.004	0.005

4.3 Feature Analysis

Layout performance is an indicator for what type of algorithm to use, however most graph visualization tools offer additional features that can improve overall readability of a layout. Many tools implement querying capabilities, color coding, shape variation, and other analytic measures to improve understanding of a graph. Features such as variation of color and shape are employed in a few cases for the sponsor's graphs.

Querying allows the user to filter certain nodes or edges based on a criteria. In this research the only node and edge data is the level and the edge style. Filtering out edge styles is useful for visualization of C2 network data. For example, the Dot algorithm performed poorly on aesthetic characteristics due to long skip echelon edges, and high variation of other edge types. If the user queries only edges between a parent and child on the Dot layout, the edge crossings and edge length variation would be reduced dramatically. In addition to these benefits, a user could include C2 network information data to allow for more advanced queries, such as unit function or physical location for example.

Varying the color or shape of nodes and edges can also provide a viewer with more information. For example may connections are associated with an analyst's confidence level. In C2 network graphs the colors red, yellow, and green edges signify low, medium, and high confidence. An analyst could use a visualization tool to color all edges of a certain confidence level automatically.

Finally many tools offer basic social network analysis measurements to improve understanding of a graph. Measurements such as centrality, clustering, and shortest paths could allow analysts a new avenue to explore their data. Microsoft Visio does not offer querying or social network analysis tools.

V. Conclusions

This chapter further discusses the results from chapter IV including a recommendation for an automated layout algorithm. The chapter also include some of the limitations of this research as well as some beneficial follow on work.

5.1 Layout Recommendations

The primary objective of this research was to identify an existing layout algorithm that is most well suited to the C2 network data visualization task. This research approached the problem by understanding of the data, needs of the sponsor, and the potential solutions. The data is a C2 network often in hierarchical format with skip echelon and adjacent edges. The sponsor's goal when creating C2 network graphs is to visualize the relationships between nodes in the hierarchy so analysts and customers can understand which nodes have the most impact on the C2 hierarchy. The sponsor emphasizes the need to show nodes that are connected to their direct subordinates as well as nodes that are connected to supporting units, and nodes well below in the hierarchy.

From the performance summarized in Section 4.2.1, the evidence supports that an automated layout algorithm will improve the efficiency with which C2 network visualizations are created. Section 4.2.2 allows us to compare which algorithm will best fit the sponsor's needs and the data representation. While GraphViz's did not perform the best in terms of aesthetic characteristics defined by the literature, it certainly outperformed the others in the metrics that measured the sponsor's needs and hierarchy structure. For this reason, the dot algorithm is the recommended layout algorithm to use for automation of future C2 network visualizations.

It is important to recognize the trade-off observed as layouts scaled in size. When

network graphs become larger other layouts may become more useful as visualization tools. In our high level complexity, the dot algorithm performed much worse on the readability and aesthetic metrics intended to measure edge crossings. This research shows that the Spectral algorithm is better for large-scale visualizations, because it does not significantly deteriorate in readability while still meeting the sponsor's needs.

5.2 Limitations

An initial limitation of this research comes from the format of the data. Current graph visualizations are created manually in Visio, which also ensures analysis of these graphs is also manual. Manual analysis of large graphs takes a significant amount of time, so only five real world graphs could be sampled for the creation of this data generator. While the graphs selected were varying in size and complexity, bias may have been introduced to the results because of the small sample size.

Another limitation is the number of algorithms tested. It was not feasible to test all possible layout algorithms in this research. For that reason, this research focused only on explicit link and node layouts, and implemented three algorithms of varying style. While the algorithms chosen were selected after a review of available options other algorithms may also perform well for use on C2 networks.

This research was also limited by the number of metrics tested. Solving the C2 network layout problem is potentially more complex than the eight metrics tested in this research, but potential tests were limited by subject matter expert understanding of C2 network graph characteristics. Potentially there are better ways to test visualizations of C2 networks, however this research only focused on the characteristics identified by literature and the sponsor. One example of an untested metric is counting the specific number of edge crossings. This research uses edge length variation to infer edge crossings because more aesthetic inferences can be made from this metric,

however inclusion of more explicit metrics could yield additional information

This research is also limited by sample size of graphs generated. Only 25 synthetic graphs were generated for each level of complexity. Due to the nature in which the graphs were generated, many graphs have similar compositions. Creating a system that allows for more randomness in data generation and a larger sample size to test from would improve the validity of future research.

5.3 Future Work

The results of this analysis open the door to future research on this topic:

- As more C2 network graphs are created with an automated layout, big data analysis can further improve the understanding of C2 layout structure.
- Testing a larger variety of layout algorithms. This research only tested algorithms of different styles, however algorithms within the same style could perform better than each other. Such as Elastic Tree, Pansy Tree, or Radial Tree in Bunches. [24][25][26]
- Implementation of AI/ML in all phases of research, generating data, evaluating the layout, and in analyzing the data. With the graph data in a format for automation it will lend itself better to machine learning analysis.
- A case study of algorithm usage in the future can provide more information on the performance of this benchmark test, as well as a better understanding on how C2 network graphs are visualized in practice.

Bibliography

1. Ian A. McCulloh, Helen L. Armstrong, and Anthony N. Johnson. Social network analysis with applications. *European University Institute*, 2013.

2. D. (Dieter) Jungnickel. *Graphs, networks, and algorithms*. Springer, 2005.

3. Hans-Jörg Schulz and Heidrun Schumann. Visualizing graphs-a generalized view, 2006.

4. Banafsheh Hajinasabrazlighi, Paul Davidsson, and Jan Persson. Visualization of data from transportation simulation systems supervisors: Examiner, 2011.

5. Kozo Sugiyama and Kazuo Misue. Visualization of structural information: Automatic drawing of compound digraphs. *IEEE Transactions on Systes, Man, and Cybernetics*, 21, 1991.

6. F. McGee, M. Ghoniem, G. Melançon, B. Otjacques, and B. Pinaud. The state of the art in multilayer network visualization. *Computer Graphics Forum*, 38:125–149, 9 2019.

7. Markus Eiglsperger, Martin Siebenhaller, and Michael Kaufmann. Lncs 3383 - an efficient implementation of sugiyama's algorithm for layered graph drawing, 2005.

8. Stephen C North and Gordon Woodhull. On-line hierarchical graph drawing, 2001.

9. Fangyan Zhang, Song Zhang, Christopher Lightsey, Sarah Harun, and Pak Chung Wong. Bgs: A large-scale graph visualization tool. pages 3781–3789. Society for Imaging Science and Technology, 2018.

10. Danny Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data, 2006.

11. Romain Bourqui, Dino Ienco, Arnaud Sallaberry, and Pascal Poncelet. Multilayer graph edge bundling. pages 184–188, 2016.

12. Fabian Beck, Franz-Josef Wiszniewsky, Michael Burch, Stephan Diehl, and Daniel Weiskopf. Asymmetric visual hierarchy comparison with nested icicle plots, 2014.

13. Willy Scheibel, Matthias Trapp, Daniel Limberger, and Jürgen Döllner. A taxonomy of treemap visualization techniques. volume 3, pages 273–280. SciTePress, 2020.

14. Sébastien Rufiange, Michael J Mcguffin, and Christopher P Fuhrman. Treematrix: A hybrid visualization of compound graphs, 2011.

15. David S Alberts and Richard E Hayes. Understanding command and control. 2006.

16. Mayra Salcedo-Gonzalez, Julio Suarez-Paez, Manuel Esteve, Jon Ander Gómez, and Carlos Enrique Palau. A novel method of spatiotemporal dynamic geovisualization of criminal data, applied to command and control centers for public safety. *ISPRS International Journal of Geo-Information*, 9, 2020.

17. Robert J Houghton, Chris Baber, Richard Mcmaster, Neville A Stanton, Paul Salmon, Rebecca Stewart, and Guy Walker. Command and control in emergency services operations: A social network analysis, 2006.

18. Menelaos Bakopoulos, Sofia Tsekeridou, Eri Giannaka, Zheng-Hua Tan, and Ramjee Prasad. Command control: Information merging, selective visualization and decision support for emergency handling, 2011.

19. Youn Ah Kang, Carsten Gorg, and John Stasko. How can visual analytics assist investigative analysis? design implications from an evaluation. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, 60:234, 2011.

20. Jia Li, Helen Meng, Yu Rong, Wenbing Huang, Hong Cheng, and Junzhou Huang. Semi-supervised graph classification: A hierarchical graph perspective. pages 972–982. Association for Computing Machinery, Inc, 5 2019.

21. Xia Jing, Matthew Emerson, David Masters, Matthew Brooks, Jacob Buskirk, Nasseef Abukamail, Chang Liu, James J. Cimino, Jay Shubrook, Sonsoles De Lacalle, Yuchun Zhou, and Vimla L. Patel. A visual interactive analytic tool for filtering and summarizing large health data sets coded with hierarchical terminologies (viads). *BMC Medical Informatics and Decision Making*, 19, 2 2019.

22. Ge Huang, Yong Li, Xu Tan, Yuejin Tan, and Xin Lu. Planet: A radial layout algorithm for network visualization. *Physica A: Statistical Mechanics and its Applications*, 539, 2 2020.

23. L. Giovannangeli, R. Bourqui, R. Giot, and D. Auber. Toward automatic comparison of visualization techniques: Application to graph visualization. 10 2019.

24. Yu Dong, Alex Fauth, Maolin Huang, Yi Chen, and Jie Liang. Pansytree: Merging multiple hierarchies, 2020.

25. Armando Arce-Orozco, Luis Camacho-Valerio, and Steven Madrigal-Quesada. Radial tree in bunches: Optimizing the use of space in the visualization of radial trees. pages 369–374, 2017.

26. Xin Yuan Yan and Yi Fang Ma. Elastic Tree Layouts for Interactive Exploration of Mentorship. 2021.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
26–03–2020	Master's Thesis	Sept 2018 — Mar 2020

4. TITLE AND SUBTITLE

Analysis of Graph Layout Algorithms for Use in Command and Control Network Graphs

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Stone, Matthew, 1st Lt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/ENG)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS022-S-039

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

NASIC/ACX
Building 858
WPAFB OH 45433-7765
COMM 937-522-4917
Email: kara.brenner@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research is intended to determine which styles of layout algorithm are well suited to Command and Control (C2) network graphs to replace current manual layout methods. Manual methods are time intensive and an automated layout algorithm should decrease the time spent creating network graphs. Simulations on realistic synthetically generated graphs provide information to help infer which algorithms perform better than others on this problem. Data is generated using statistics drawn from multiple real world C2 network graphs. The three algorithms tested against this data are the Spectral algorithm, the Dot algorithm, and the Fruchterman-Reingold algorithm. The results include a multiple objective statistics designed to inform on the algorithms performance in both aesthetic characteristics defined in literature, as well as some characteristics defined by the research sponsor. The results suggest that the Dot algorithm performs better with respect to the sponsor defined characteristics, whereas the Fruchterman-Reingold algorithm performs better on aesthetic characteristics.

15. SUBJECT TERMS

GraphViz, Dot, Layout, C2, Hierarchy, Visualization, Graphs, Fruchterman-Reingold, Spectral, Command and Control

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Mark Reith, AFIT/ENG
U	U	U	UU	56	19b. TELEPHONE NUMBER *(include area code)* mark.reith@afit.edu