A Simplification Algorithm for Visualizing the Structure of Complex Graphs

Daniel Hennessey, Daniel Brooks, Alex Fridman, David Breen
Department of Computer Science, Drexel University, Philadelphia, PA, USA
dph29@drexel.edu, dpb35@drexel.edu, af59@drexel.edu, david@cs.drexel.edu

Abstract

Complex graphs, ones containing thousands of nodes of high degree, are difficult to visualize. Displaying all of the nodes and edges of these graphs can create an incomprehensible cluttered output. This paper presents a simplification algorithm that may be applied to a complex graph in order to produce a controlled thinning of the graph. Using importance metrics, the simplification process removes nodes from the graph, leaving the central structure for visualization and evaluation. The simplification algorithm consists of two steps, calculation of the importance metrics and pruning. Several metrics based on various topological graph properties are described. The metrics are then used in a pruning process to simplify the graph. Nodes, along with their corresponding edges, are removed from the graph, while maintaining the graph's overall connectivity. This simplified graph provides a cleaner, more meaningful visual representation of the graph's structure; thus aiding the analysis of the graph's underlying data.

1 Introduction

Graphs are used to represent and analyze a variety of data sets, from computer network topologies, to cancer cell structures, to research paper citation trends. Graphs are a useful representation for these types of data, because they naturally provide a clear and distinct way of visualizing the data sets and the interrelationships contained within them. These data sets are becoming larger and larger, and thus their visual representations are becoming too complex and cluttered. Graphs frequently have hundreds or thousands of nodes. Displaying all of these nodes and edges to a user provides no real benefit, since the density and complexity of the graph overwhelms and fills standard displays due to resolution limitations. Graph simplification provides one way of making these graphs easily comprehensible while not removing the actual meaning or structure of the graphs.

In our approach a complex, cluttered graph is progressively thinned by the user until the underlying structure of the graph becomes apparent. The user may choose from a variety of weighting functions that quantify the importance of the nodes to be removed. The importance metrics can be based on the topology of the graph or may be provided by an external computational process unrelated to the graph structure itself. One important feature of the simplification process is that we maintain the connectivity of the graph during pruning. If the graph is singly connected at the outset, we ensure that all simplified versions are also fully connected. This feature, which the user may choose to employ or not, is included to be consistent with our goal of providing a tool for making a complex graph's underlying structure more evident. We do not want the simplification process to change the graph's connectivity, which may misinform the user's interpretation of the graph's structure.

The simplification process involves two steps. The first step generates weight values for all nodes in a given graph based on one of our importance metrics. The second step takes the weighted graph and prunes out less important nodes based on these weight values. Node removal can be done incrementally, so as to provide different graph resolutions to the user. The user may also control the connectivity maintenance feature, deciding to preserve the initial graph connectivity during simplification or not. The simplification algorithm is described in detail in Section 3. Results from applying several of the importance metrics to three types of graphs are presented in Section 6.

2 Related Work

Related work has been conducted on simplifying graphs, but primarily as a secondary process for manipulating or matching graphs. Qiu and Hancock [14, 15] develop three graph simplification algorithms. The first algorithm decomposes a large graph into smaller non-overlapping sub-graphs. The remaining two carry out simplification based on commute times within the graph. The algorithms produce a multilayer-ed representation and spanning trees. All three algorithms use their simplified representations to perform matching between the input graphs.

Frishman and Tal [7] and Walshaw [19] use simplification algorithms to generate simplified graphs for input into their graph layout algorithms. The simplified graphs are not actually visualized and presented to the user as a way to help them better comprehend the graph data. Instead, a series of progressively simplified graphs are used to guide the positioning of the nodes in a dense, complex graph.

Additional simplification algorithms have been proposed to assist in robot path planning [17], classifying the topology of surfaces [3], speech recognition [13], and improving the computational complexity and memory requirements of dense graph processing algorithms [12, 18, 6].

Simplification may also be accomplished through the use of graph clustering. Girvan and Newman [9] define one such clustering-based algorithm for better visualizing the community structure of network graphs. The kind of graphs that they are targeting with this method are those that would have a naturally occurring community structure.

Rafiei and Curial [16] present another approach to visually simplifying large scale graphs. They have developed different methods for randomly sampling a graph and using that sampling to construct the visual representation of the graph. Their general approach is similar to ours in that they are using characteristics of the graphs that frequently occur, but differs in that they reduce their large graphs down to a handful of nodes and then "grow" their simplified graph out from those nodes.

Gilbert and Levchenko [10] focus on providing metrics for simplifying graphs that represent specific network topologies. The goal of this work is to simplify and visualize complex network graphs while maintaining their semantic structures. Although network topologies are certainly reasonable candidates for these visualization techniques, there are other sets of graph data that could greatly benefit from simplification techniques for visualization, e.g. the cancer cell graphs produced by Demir et al. [5] and the citation graphs of Chen [4]. These graphs do not consist of the same properties and structures as network topologies, so the metrics of [10] may not be valid for these types of graph data. Our goal is to develop importance metrics that are based on the general topological properties of different types of graphs. This will allow for the simplification and visualization of any large graph in order for the user to better understand and comprehend its visual representation.

3 Simplification Algorithm

The first step of our simplification algorithm performs an importance metric calculation on the graph, generating a scalar weight value for every node in the graph. Several importance metrics have been implemented and evaluated, including number of N-ring neighbors, number of shortest paths, node eccentricity, distance to the graph's center node(s), and distance to a leaf node. It is possible to assign an importance values to the graph's nodes based on external factors and calculations not related to the graph's topology. In this case the calculations of the first step may be skipped, since pruning (the second step) only needs some type of importance value for the nodes, regardless of its origin.

The second step in the algorithm prunes the graph, removing nodes, and associated links, with importance values below a user-specified threshold. If desired by the user, nodes whose removal would change the graph's connectivity are not pruned from the graph. Adjusting the threshold value allows the user to control the resolution and complexity of the simplified graph. The higher the threshold, the more nodes are removed, and the coarser the graph becomes. The user may choose to remove nodes from the graph incrementally up to the defined threshold value. Here, a number of pruning steps are applied with successively higher threshold values up to the user define maximum threshold.

3.1 Importance Metrics

The weight values assigned to the nodes of a graph are computed with different importance metrics. These metrics are derived from known properties of graph structures. Currently, none of our metrics are scaled or normalized. Therefore the range of the resulting values will vary depending on the metric used, the size of the graph, as well as the topology of the graph.

3.1.1 Number of N-ring Neighbors

The first metric is based on the number of N-ring neighbors around a node. An N-ring neighbor is a node that can be reached in *n* unique hops from the original node. When calculating the number of $N-$ring neighbors where *n* is greater than one, the metric may be computed in two ways. The metric may count only those neighbors that lie exactly *n* hops away from the original node. The metric may also be cumulative, i.e. it may count all the node that are *n* or fewer hops away from the original node. In this paper we use the first non-cumulative option in calculating the number of N-ring neighbors greater than one.

3.1.2 Number of Shortest Paths

The shortest path metric is based on the graph theory principle of betweenness. A node's betweenness is determined by how many shortest paths pass through a given node [20]. This metric finds the shortest path between all nodes (u, v) belonging to graph G using a variation of Dijkstra's shortest path algorithm [11]. For a shortest path between nodes (u, v), each node that belongs to the path has its weight incremented by one. The endpoints of the path do not have their weights incremented. Note that when there are two or more shortest paths of the same length between two nodes one of these is selected at random during the node increment process. This metric is similar to the shortest path based importance metric designed by Gilbert and Levchenko [10].

3.1.3 Eccentricity

This metric assigns a node's importance equal to the value of its eccentricity in the graph. A node's eccentricity is defined as the maximum distance within the set of shortest paths from the node to all other nodes in the graph [11]. Higher values in this metric mean that the given node is a significant distance away from the farthest component in the entire graph, or in other words it is closer to the periphery of the graph than the center. Therefore the higher the weight value, the less important the node should be. This is opposite to the definitions of the other metrics. Thus to be consistent and to lead to proper pruning, the eccentricity importance metric is defined as $eccentricity_i - max(eccentricity)$, where $eccentricity_i$ is the actual eccentricity of a particular node i and $max(eccentricity)$ is computed over all nodes in the graph.

3.1.4 Shortest Distance to a Center Node

This metric is a variation on the eccentricity metric mentioned previously. All nodes with the minimum eccentricity value are defined as center nodes of the graph [11]. These center nodes are important because they have a minimum distance to the farthest limits of the entire graph; thus placing them in the central interior of the graph. Once the center nodes of a graph have been identified, an importance metric can be computed based on the distance to one of the center nodes (there can be more than one). Nodes with shorter paths are considered more important, because they are closer to the graph's center. As with the previous metric, this is opposite to the desired importance property. Thus the metric is computed as $distance_i - max(distance)$, where $distance_i$ is the actual shortest distance for a particular node i and $max(distance)$ is computed over all nodes in the graph.

3.1.5 Shortest Distance to a Leaf Node

This metric calculates the shortest distance from a node to a leaf node in the graph. It is assumed that the further a node is from a leaf node, the more centrally located within the graph it is, making that node more important. Ultimately this metric assumes that the input graph will have leaf nodes. If leaf nodes are not present then this metric cannot be applied during simplification.

3.1.6 Network Flow

Importance metrics do not have to be based solely on quantities derived from the graph's topology. They can be externally provided by other computations, e.g. network flow calculated for a graph representing a communications network. A widely-used model for ad-hoc networks is a capacitated directed graph $G(V, E)$ [1], where an edge $(u, v) \in E$ indicates a potential link of communication between transmitter i and receiver j. The weights $c(u, v)$ on the edges indicate the capacity of the transmission channel, or the maximum amount of information that can be transferred on that channel in a pre-specified amount of time. An important goal in the context of a network is to maximize the amount of information "flowing" from a *source* node to a destination (*sink*) node. A good measure of a node's importance given this objective is the outgoing maximum flow from a node on a fully active network. The maximum flow is computed by solving a flow optimization problem on the graph [8].

3.2 Graph Pruning

The pruning process visits all nodes and removes those nodes, and their associated edges, with importance values below a user-specified threshold. Removing a node can cause the graph to fragment. In order to maintain the semantic structure of the graph we do not prune nodes whose removal would split the graph into multiple components and destroy its connectivity. Producing a simplified graph that has been split into multiple components would convey incorrect structural information about the graph. Gilbert and Levchenko [10] do not maintain the connectivity of a graph during pruning. They reconnect the simplified graph as a post-process.

The connectivity check of the pruning process utilizes our variation on Dijkstra's single source shortest path computation mentioned previously. The candidate node is removed from the graph. It is then determined if all of the candidate's neighboring nodes are still connected to each other through the graph. This check is performed by attempting to calculate the shortest path between all pairs of neighboring nodes. If unable to complete this computation, the graph has been disconnected by the candidate node's removal, and it is then re-inserted in the graph.

The pruning process can be done incrementally. When pruning a weighted graph we use an increment value along with a target threshold value. The initial threshold is set to 0. The increment is added to this value and the pruning process is performed. The increment is then added to the current threshold and the pruning is performed again on the previously pruned graph. This continues until the current threshold becomes greater than or equal to the target threshold. Incrementally removing nodes will typically provide better results than a non-incremental removal. One reason for this is that it tends to remove less important nodes before more important nodes. When taking large threshold increments it is possible, since nodes are randomly accessed, that a more important node can be processed and pruned before a less important node. Removing important nodes first may leave less important nodes in a

position in which they become impossible to remove due to the connectivity check. As the increment is decreased the chance of pruning a more important node before a less important node diminishes. Once the increment is equal to the smallest difference between node weight values, incremental pruning effectively produces a removal of nodes in a sorted order, furnishing better results.

4 Test Data

Our graph simplification algorithm and all of its metrics have been applied to a number of graphs. The algorithm has been tested and evaluated on three different types of graphs. The first graph was automatically generated using the Inet tool [21], a power-law-based graph generator that creates graphs representing internet topologies. The Inet graph we used in our testing had 4500 vertices and 15308 edges.

The second graph was generated from data produced by the citation trends research of Chen [4]. Each node of the citation graph is a research paper and every edge represents a citation of that paper. Originally this was a directed graph, but since most of our metrics do not utilize the direction associated with an edge, we assume that the citation graph is undirected. Although we lose some of the graph's information, it still provides a unique real-world graph structure for evaluation. This citation graph had 1025 vertices and 15430 edges.

We also include a graph generated from the OMAN ad-hoc network simulator, a simulation system developed at Drexel University. The nodes of this graph are pre-weighted using the OMAN-computed flow metrics discussed previously. We imported the graph along with its weight values and processed it with our pruning mechanism to demonstrate that the pruning operation is independent from our importance metrics calculations and is capable of handling different types of graphs.

In order to better visualize these graphs, they have each been pre-processed with the neato graph layout tool which is a part of Graphviz [2]. This applies a spring model layout to our graphs prior to applying our simplification algorithm.

5 Metrics Properties

Once the various importance metrics were implemented and applied to the first two graphs, general properties of their resulting values were observed. In evaluating how each metric performed we first define what we believe makes an effective importance metric in terms of using it to simplify the graph for visualization.

First, a metric should provide a wide range of weight values across the nodes. A wider range allows for finer control of the resolution of the simplified graph. This property can be evaluated by the ratio of the range of weight values to the number of nodes in the graph. An example is metric A generates weights in the range of 0 to 400 on a graph G with 1000 nodes. The weight-range-to-node-count ratio would then be $400/1000 = 0.4 : 1$. This provides a consistent way to compare each metric's characteristics across a single graph. We have captured this data for all of our metrics and test graphs in Table 1.

An effective metric should also create a even distribution of these weight values across the entire range. An even distribution will allow for a more controlled simplification of the graph as the user increases the threshold value. This allows the user to control the resolution of the resulting simplified graph. This distribution can be measured by observing the rate at which nodes are removed from the graph while simplifying over the range of weight values. This data has been illustrated in Figure 1. The data in these graphs was generated by taking the range of the weight values of each metric on a given graph and dividing it into equal increments. We tried to use 10 equal increments for all metrics, but for some of the metrics that generated larger weight ranges we increased that up to 18 to get a better sample. The simplification process was then run at each increment. The number of nodes remaining after the simplification was then recorded.

The final criteria for an effective metric is that these two properties should be capable of being utilized to produce a meaningful visual representations of the original graph. Simplifying a graph using a particular metric should reveal something about the graph's connectivity or underlying structure.

N-ring Neighbors. For N-Ring neighbors we implemented three variations of the metric. We tested one, two, and three ring neighbors as metrics. Overall N-ring neighbors metric generates reasonable weight ranges. A comparison of the weight-range-to-node-count ratios for each of the three variations can be seen in Table 1. We can see that as we increase N, the ratios approach $1 : 1$. These metrics of course are dependent on the connectivity of the original graph; but since we are primarily focusing on large complex graphs, these metrics provide good ranges for these types of graphs.

The node drop off rates for these metrics also showed a similar relationship as N was increased. In Figure 1 the three N-Ring metrics start off with a quick drop off at the beginning of the weight range and as N increases, the node drop off rate becomes approximately linear. This indicates that the distribution of weight values across the entire range becomes more evenly spread out as N is increased.

Overall all the properties exhibited by the N-Ring neighbors metrics define them as effective metrics. The range of weights across the nodes and the distribution of

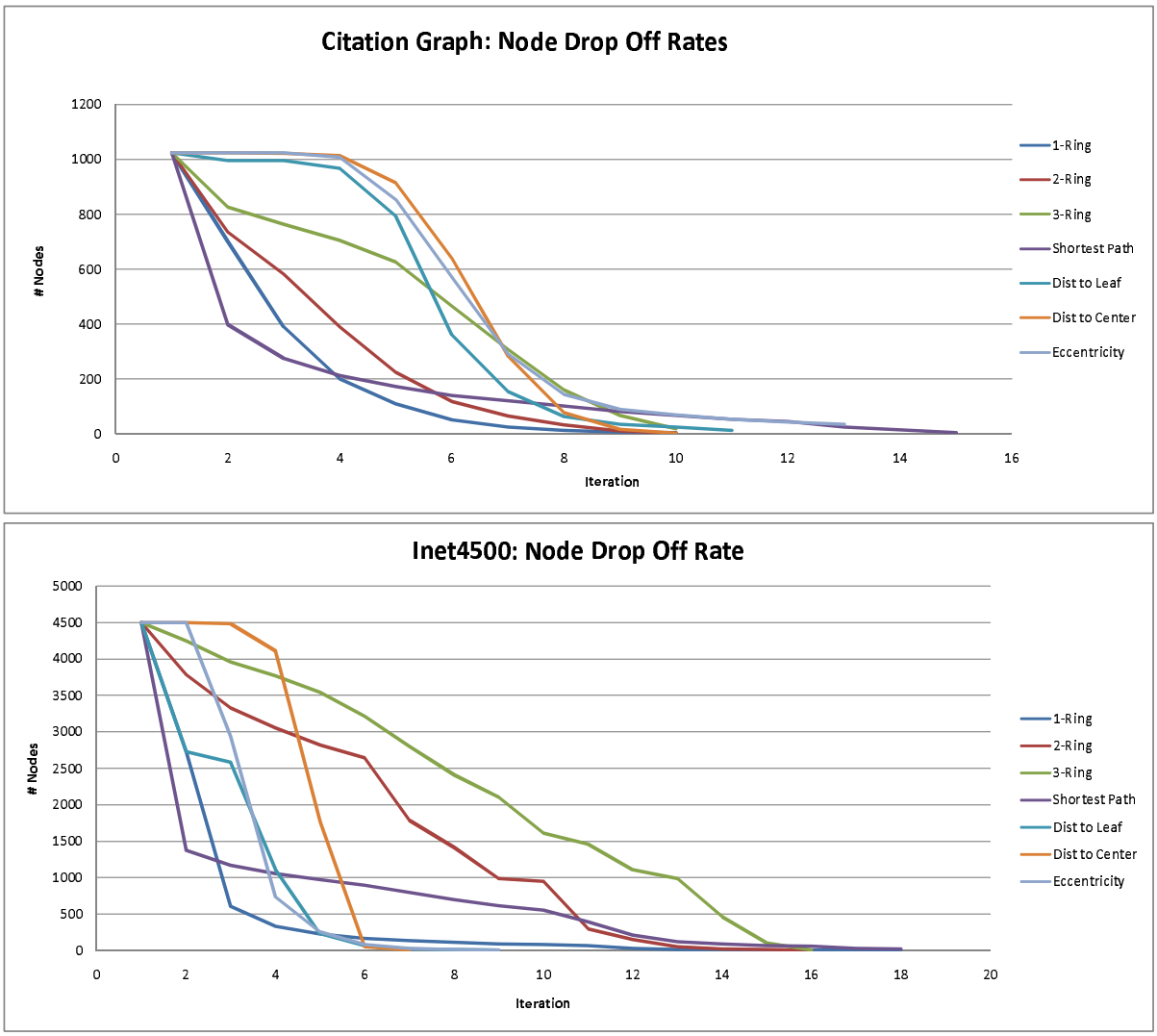

Figure 1: Two graphs showing the node drop off rate of each metric against the Citation graph [4] (top) and an Inet graph (bottom). The min and max weight were determined for each metric on the particular graph. This range was then divided into equal increments and the simplification process was run at each increment with the threshold value being equal to each increment. The number of nodes remaining at each increment was recorded.

	Citation Graph		Inet4500	
Metric	Weight Range	Ratio	Weight Range	Ratio
1-Ring	1 to 75	0.072195122	1 to 942	0.209333333
2-Ring	2 to 380	0.368780488	2 to 3585	0.796666667
3-Ring	3 to 763	0.741463415	3 to 4432	0.984222222
Shortest Path	0 to 85284	83.20390244	0 to 12926842	2872.631556
Dist to Leaf	0 to 7	0.006829268	0 to 4	0.000888889
Dist to Center	1 to 10	0.008780488	1 to 6	0.001111111
Eccentricity	9 to 17	0.008780488	5 to 9	0.000888889

Table 1: This table lists the Weight range to # of Nodes ratios for each metric and each graph they were run against.

those weight values are both in line with our definition of a good metric.

Number of Shortest Paths. The Shortest Path metric calculates very high weight values for frequently visited nodes in the center of the graph. There is a large number of paths through these central nodes since they allow the outer nodes to connect most directly with each other. This can be seen in its node drop off rate curve in Figure 1. Initially there is a very quick drop off of nodes. These removed nodes would exist more towards the periphery of the graph. This initial drop off tends to eliminate approximately half of the nodes in the graph within a few pruning steps. Once these nodes are removed the curve becomes much more linear in its rate of decent. This shows that the distribution of weights is very biased towards the lower end of the range.

The ratio of weight range to number of nodes on the other hand is quite large. For the citation graph the Shortest Path metric generated a ratio of $83.2 : 1$ and for the inet graph a ratio of $2872.6 : 1$ was generated. With such large weight-range-to-node-count ratios the difference between simplifying with two distinct threshold values may be only one or two nodes.

Overall the metric does not fit very well into our definition of an effective metric. The weight distribution tends to be heavily biased to the lower end of what is typically a very large range of weight values. Such a large range of weight values actually gives too much granularity between each distinct threshold value in the higher weighted nodes. The removal of one or two nodes rarely makes any distinguishable visible difference in the graphs. This metric could be made more effective by scaling or normalizing the generated weight values. Doing this should even out the distribution and contract the overly large range of weight values.

Eccentricity. The Eccentricity metric provides some interesting results. This metric depends on the underlying connectivity of the original graph. We can observe this by looking at the weight-range-to-node-count ratios. For both graphs this ratio we extremely low, $0.00088 : 1$ for the inet graph and $0.00878 : 1$ for the citation graph. The actual range of weight values for the inet graph was from 0 to 4. This means that the longest shortest path within this graph is length 4. This implies one of two things, that the graph is very well connected or it reveals a radial graph structure with everything filtering into a few central nodes. In our sample images we actually see that it is the latter of the two possibilities in Figure 7. Although these ratios are low, the metrics can still reveal interesting structural details in the graph not previously observed.

The distribution of these weight values is also highly dependent on the underlying structure of the graph. We see that for the citation graph this metric provides a relatively even drop off rate while the inet graph has a very sharp drop off over the middle weight values in its range. Although this metric does not fit our criteria for an effective metric, it can still produce some interesting visual information about the underlying structure and connectivity of a graph.

Shortest Distance to Center Node. Since this metric is based on the graph's eccentricity, it displays many of the same properties as the Eccentricity metric. It maintains very low weight-range-to-node-count ratios, $0.0011 : 1$ and $0.0087 : 1$ for the inet and citation graphs respectively, and the distribution is again dependent on the structure of the graph being processed.

While this metric and Eccentricity have some similarities, there are some differences in the resulting weights. In the Eccentricity metric a node relatively close to a center node may have a lower metric value because of the variety of paths available to reach all other nodes. While that same node may have a high Center Node metric value simply because it is near a center node. In a network analogy a given node may not need to reach the edges of the network directly. Instead it connects to a more centrally located node which handles the distribution. In this case the Distance to Center metric will capture this property. The real benefit with this metric is that it provides a different and still relevant visual representation of the simplified graph.

Shortest Path to Leaf Node. The Shortest Path to a Leaf Node metric is another that is highly dependent on the topological structure of the original graph. Firstly, it assumes that the given graph does contain leaf nodes. If it does not then this metric cannot be computed. In relating this metric to our standards for an effective metric, this one is a bit weak. Its weight-range-to-node-count ratios are fairly small. The citation graph's ratio was $0.0068 : 1$ while the inet graph's was $0.00088 : 1$. These values are about the same as the Eccentricity and Distance to Center Node metrics. The node drop off rates follow a trend similar to that of the Eccentricity and Distance to Center Node metrics as seen in Figure 1.

Overall this metric is designed to give a good measure of how well connected the leaf nodes of a graph may be. It can also offer insight into how well the non-leaf nodes are connected amongst each other. If the metric's values tend to be high, then the non-leaf nodes are not very well-connected and need to follow longer paths to get to the periphery of the graph. Although it does not meet the first two criteria of an effective metric is does produce some

acceptable visual output of the graphs.

6 Results

Figures 2 to 3 present the results from applying our simplification algorithm to the three graphs described in Section 4. Applying the pruning algorithm to the flow graph of Figure 2 presents a steady thinning of the graph, while maintaining its connectivity. The T-6000 graph shows the drop out of all the low flow nodes in the network. The T=18000 graph continues to simply the display of the network so we can start to see a more defined structure to the higher flowing nodes. The transition from the T=18000 graph to the T=50000 graph then reveals more of the "backbone" of the flow graph.

Applying the simplification algorithm, with its five different metrics, to the Inet graph in Figures 3 to 7 reveals the hierarchical structure of this Internet-like graph. In the first and second graph of Figure 3 we can see the radial dispersal of nodes, but the density of the edges still hides many of underlying nodes. The last two graphs reveal the underlying central structure of these radial layouts. Figure 4 very quickly reveals the central node structure that looks similar to Figure 3. It is also interesting to note that all five metrics converge on a slightly different set of core nodes.

Applying the simplification algorithm to the less structured citation graph provides different kinds of results. The 3-Ring Neighbors metric in Figure 8 reveals the structure underneath the heavily connected portion of the graph. The Shortest Path and Distance to Center metrics in Figures 9 and 11 reveal a small set of core nodes within the main body of the graph. While the Distance to Leaf and Eccentricity metrics in Figures 10 and 12 disclose a simplified version of the entire graph structure as opposed to just the central nodes.

Looking over Figures 6 to 12 visually reinforces our observations on the properties of the individual metrics. Those metrics that produce a narrow range of values (Distance to Leaf, Distance to Center, Eccentricity) provide less control for uniformly pruning the tree with fewer resolution options. The remaining metric (N-Ring Neighbors and Shortest Path) provide the user with many more threshold values to choose from; thus providing the ability to more smoothly simplify the input graph.

The computation time needed to perform simplification is not significant. Each graph simplification requires about a second to a few tens of seconds (with a rare simplification using the Number of Shortest Paths metric needing about a CPU-minute) to compute, with most of the graph processing requiring only a few CPU-seconds on an Intel Dual Core 1.73 GHz processor. These times were to both compute the given metric and prune the graph incrementally while preserving connectivity. Times varied slightly based on the metric calculated, graph size, and threshold and increment values.

7 Conclusion

Complex graphs, ones containing thousands of nodes of high degree, are difficult to visualize. Displaying all of the nodes and edges of these graphs can create an incomprehensible cluttered output. We have presented a simplification algorithm that may be applied to a complex graph in order to produce a controlled thinning of the graph. The simplification of the graph provides an approach to visualizing the fundamental structure of the graph by displaying the most important nodes, where importance may be based on the topology of the graph or external factors. The simplification algorithm consists of two steps, calculation of importance metrics and pruning. We have described several weighting functions and have presented the simplified graphs produced by applying them. Future work includes developing additional importance metrics, as well as testing and evaluating our approach on other types of graphs.

Acknowledgments

We would like to thank Dr. Chaomei Chen of Drexel's iSchool for providing the citation graph data set. This work is funded by the U.S. Army Communications–Electronics Research, Development and Engineering Center (CERDEC), under contract #DAAB-07-01-9-L504.

References

[1] AHUJA R. K., MAGNANTI T. L., ORLIN J. B.: *Network Flows : Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] AT&T LABS RESEARCH: *Graphviz*. http://www.graphviz.org.

[3] BAN T., SEN D.: Graph based topological analysis of tessellated surfaces. In *Proc. Eighth ACM Symposium on Solid Modeling and Applications* (2003), pp. 274–279.

[4] CHEN C.: Citespace II: Detecting and visualizing emerging trends and transient patterns in scientific literature. *Journal of the American Society for Information 57*, 3 (2006), 359–377.

[5] DEMIR C., GULTEKIN S., YENER B.: Augmented cell-graphs for automated cancer diagnosis. *BIOINFORMATICS 21*, 2 (2005), 7–12.

[6] FEDER T., MOTWANI R.: Clique partitions, graph compression, and speeding-up algorithms. *Journal of Computer and System Sciences 51*, 2 (1995), 261–272.

[7] FRISHMAN Y., TAL A.: Multi-level graph layout on the GPU. *Transactions on Visualization and Computer Graphics 13*, 6 (Nov. 2007), 1310,1319.

[8] FRIDMAN A., WEBER S., DANDEKAR K., KAM M.: Distributed multicommodity flow on ad hoc wireless networks of cognitive radios. In *Proc. 40th Annual Conference on Information Sciences and Systems*. to be published in March 2008.

[9] M. GIRVAN, M.E.J. NEWMAN: Community Structure in Social and Biological Networks *PNAS 99*, 12 (2002), 7821–7826

[10] GILBERT A., LEVCHENKO K.: Compressing network graphs. In *Proc. LinkKDD 04* (2004).

[11] HARARY F.: *Graph Theory*. Westview Press, 1969.

[12] KAO M., OCCHIOGROSSO N., TENG S.-H.: Simple and efficient graph compression schemes for dense and complement graphs. *Journal of Combinatorial Optimization 2*, 4 (1998), 351–359.

[13] LIU Y., HARPER M., JOHNSON M., JAMIESON L.: The effect of pruning and compression on graphical representations of the output of a speech recognizer. *Computer Speech and Language 17*, 4 (2003), 329–356.

[14] QIU H., HANCOCK E. R.: Spectral simplification of graphs. *Lecture Notes in Computer Science 4* (2004), 114–126.

[15] QIU H., HANCOCK E.: Graph simplification and matching using commute times. *Pattern Recognition 40*, 10 (2007), 2874–2889.

[16] DAVOOD RAFIEI, STEPHEN CURIAL: Effectively Visualizing Large Networks Through Sampling. *16th IEEE Visualization 2005 (VIS 2005)* (2005), pp. 48

[17] RIZZI S.: A genetic approach to hierarchical clustering of euclidean graphs. *Pattern Recognition 2* (1998), 1543–1545.

[18] SUEL T., YUAN J.: Compressing the graph structure of the web. In *Proc. Data Compression Conference* (2001), pp. 213–222.

[19] WALSHAW C.: A multi-level algorithm for force-directed graph drawing. *Journal for Graph Algorithms and Applications 7*, 3 (2003), 253–285.

[20] WEISSTEIN E.: Graph theory. *MathWorld – A Wolfram Web Resource* (Oct. 2007).

[21] WINIK J., JAMIN S.: *Inet 3.0: Internet Topology Generator*. Tech. Rep. CSE-TR-456-02, University of Michigan, 2002.

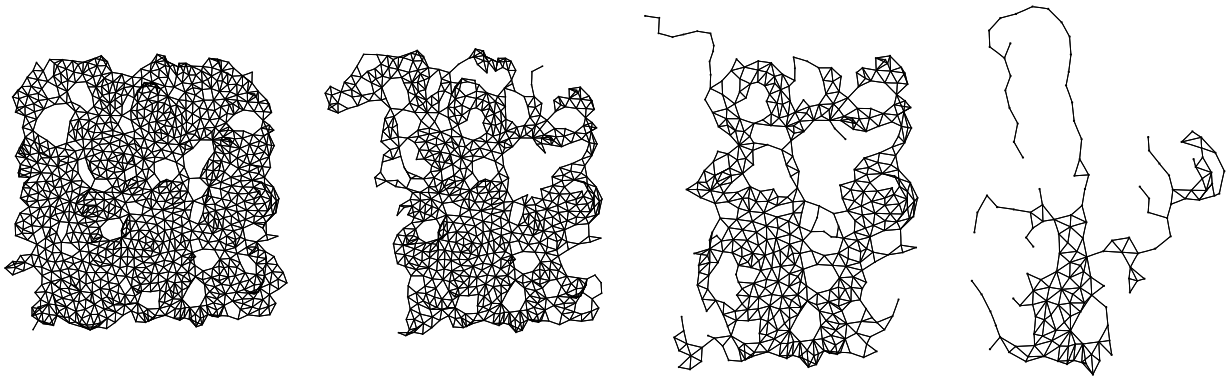

Figure 2: Ad hoc network graph with 1000 nodes: Network flow value with threshold values 0, 6000, 18000, 50000.

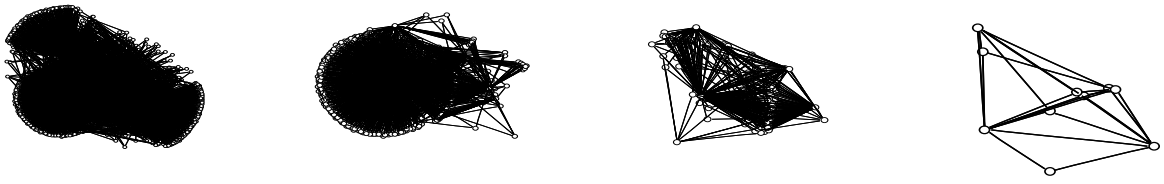

Figure 3: Inet graph with 4500 Nodes: 2-Ring Neighbors metric with threshold values 300, 700, 1200, 1800.

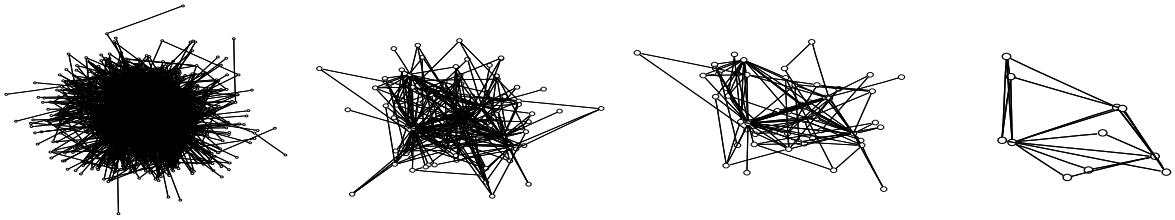

Figure 4: Inet graph with 4500 Nodes: Shortest Paths metric with threshold values 1000, 40000, 100000, 645000.

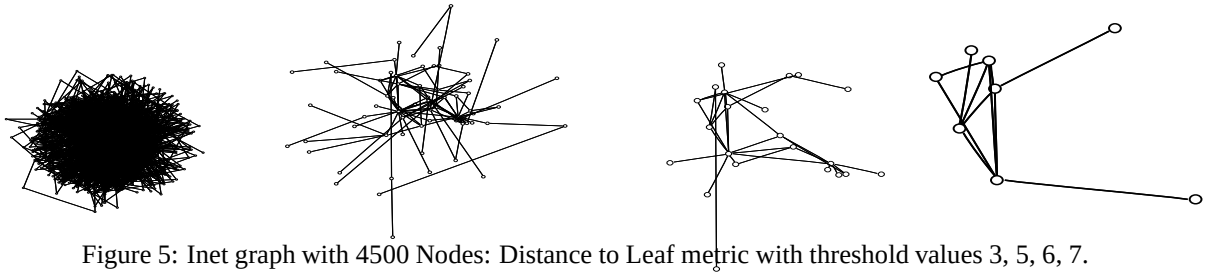

Figure 5: Inet graph with 4500 Nodes: Distance to Leaf metric with threshold values 3, 5, 6, 7.

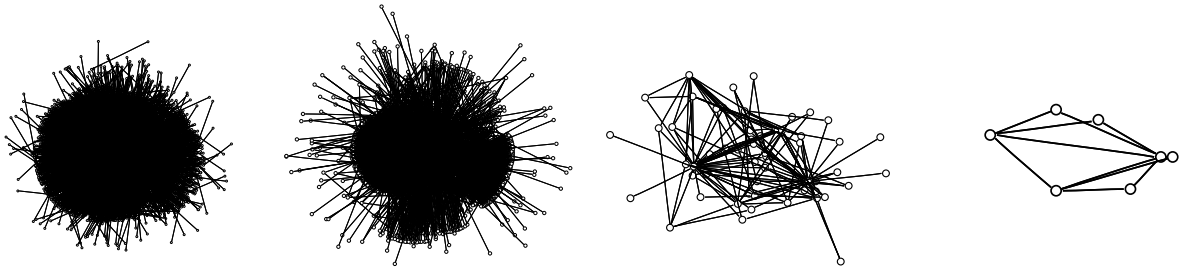

Figure 6: Inet graph with 4500 Nodes: Distance to Center metric with threshold values 3, 4, 5, 6.

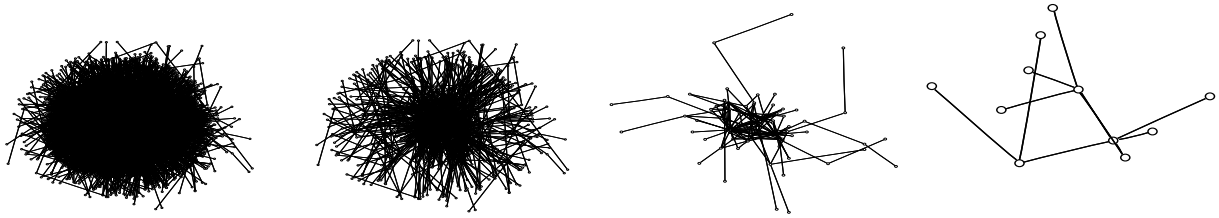

Figure 7: Inet 4500 Nodes: Eccentricity metric with threshold values 7, 8, 10, 12.

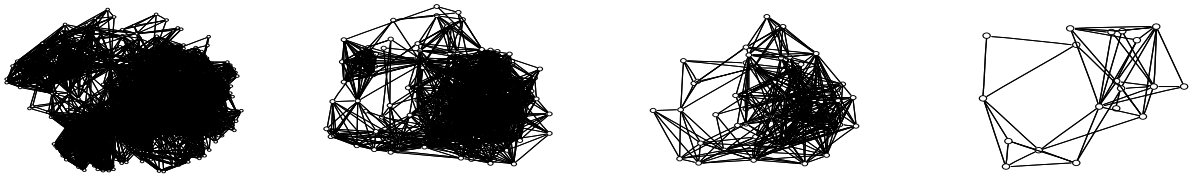

Figure 8: Citation Graph with 1025 Nodes: 3-Ring Neighbors metric with threshold values 360, 504, 576, 648.

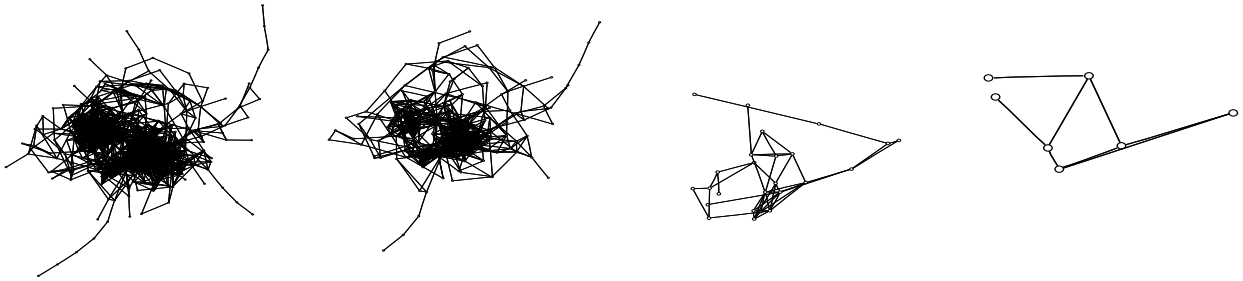

Figure 9: Citation Graph with 1025 Nodes: Shortest Paths metric with threshold values 1404, 4212, 28828, 49828.

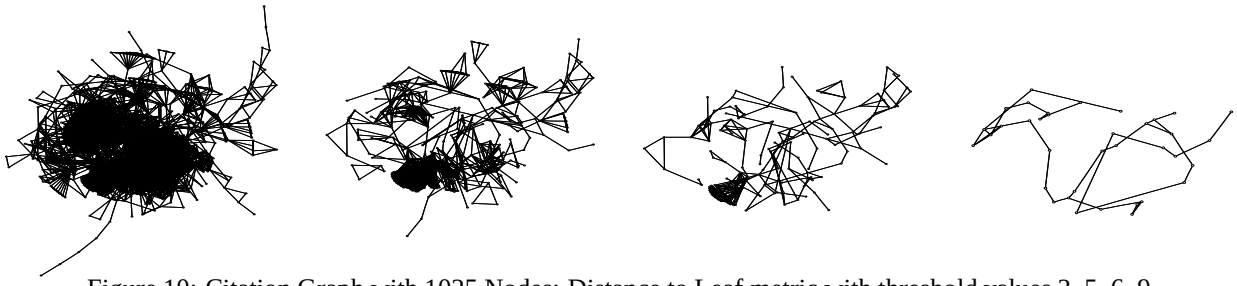

Figure 10: Citation Graph with 1025 Nodes: Distance to Leaf metric with threshold values 2, 5, 6, 9.

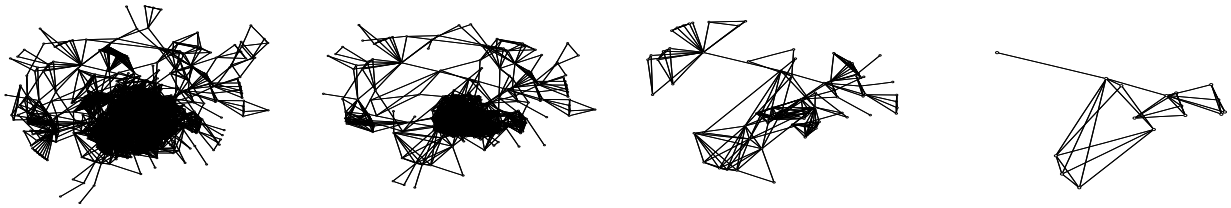

Figure 11: Citation Graph with 1025 Nodes: Distance to Center metric with threshold values 5, 6, 7, 8.

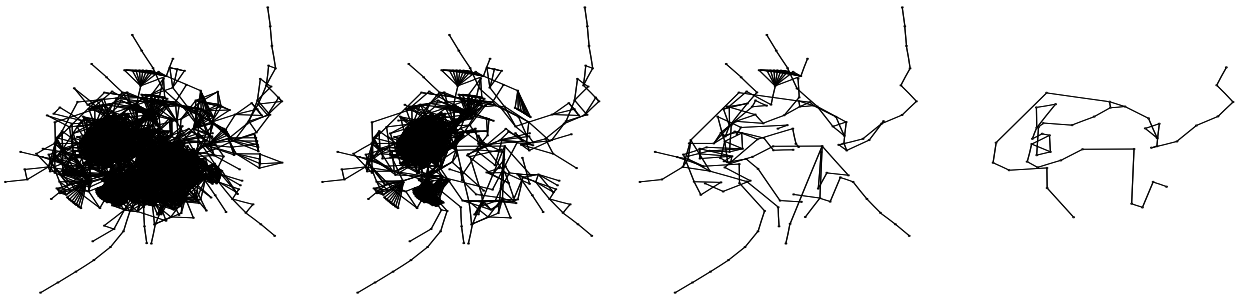

Figure 12: Citation Graph with 1025 Nodes: Eccentricity metric with threshold values 10, 13, 15, 19.