# Guacamole: An Evaluative Procedure for the Advogato Trust Metric

Jim Zheng
jimzheng@stanford.edu

Hongxia Zhong
hzhong62@stanford.edu

Aric Bartle
abartle@stanford.edu

December 10, 2014

## Abstract

We propose a procedure to measure and compare attack-resistant properties in real world graphs. Our procedure extends Raph Levien's Advogato Trust Metric of a single graph to a broader set of real world graphs and allows for comparison between the attack resistance of two graphs. We evaluate our procedure by devising and running a parameterized attack model on a trust graph. We conclude that under certain structural assumptions, our procedure can be applied to real world graphs.

## 1 Introduction

With the ever-increasing number of online social networks over the last decade, the issue of trust within networks is becoming increasingly relevant across real world networks. Companies such as AirBNB and Couchsurfing model their social graphs as trust networks to protect users from bad actors. Peer-to-peer distributed networks often deal with the problem of trust, as malicious nodes can be easily set up and added to the network. Recent academic work have focused on real world graphs such as Epinions, Slashdot, and Couchsurfing as models of trust and reputation Dandekar (2009). These studies aim to define and evaluate **trust metrics**, a measurement of the degree to which one node trusts another. Due to the subjectivity of trust within real networks, it is difficult to obtain a common definition of trust metric across domains.

An important property of good trust metrics is **attack resistance**, which reflects a network's resilience against agents who abuse the presumption of trust. The first application to study attack resistance was in the context of peer-to-peer distributed networks. Given a series of bad actors $M$ in a larger network graph $G$, we can ask the following question:

- Which trust metrics are able to detect the most members of $M$ as being untrustworthy?
- Can we quantify and bound the degree to which bad actors can influence good nodes?

The Advogato Trust Metric devised by Raph Levien in Levien (2004) answers these questions, but the paper only applies the procedure to a specific trust graph. More importantly, the metric does not provide a way to compare the attack resistance of two graphs. Our project extends this model to the set of real world trust graphs. We propose a procedure GUACAMOLE which leverages the Advogato trust metric to produce an overall resistance score for a graph. In particular, our contributions to the algorithm include an initialization step, an attack model based on the trust metric, and a tuning procedure to generate attack resistance scores. We use this procedure to compare the attack resistance of two graphs: the original Advogato network, and a generated graph that models real world graph properties. Finally, we investigate how our procedure stands up to graphs at different scales and input networks of varying attributes such as diameter, and conclude with remarks about applications to real world graphs.

## 2 Background

### 2.1 Graphs and Trust Metrics

In the realm of trust metrics, several have been proposed. The simplest trust metric is one that models a chain of trust; given a seed node $s$ and a target node $t$, can we find a path of trust from $s$ to $t$? Though easy to understand and implement, this definition of trust is naive and does not defend very against attacks, since an attacker simply needs to compromise a single node. Moreover, the source node has no way of determining the trustworthiness of a randomly chosen target node.

The first non-trivial trust metric comes from Reiter and Stubblebine [3]. This trust metric looks at the number of node-independent paths of bounded

length from the seed to the target; above a certain threshold, the seed node may trust the target. This definition is attack resistant because an attacker must add at least as many edges from a malicious node under his control to influence the metric. Though this method is NP-complete, it provides a first defintion of attack resistance.

More recent, more robust trust metric have been studied in the context of trust in distributed systems: the Eigentrust metric discussed in Kamvar et al. as well as the improved method in Xinxin Fan and Su, and the Advogato Trust Metric found in Levien (2004). We cover the Advogato Trust Metric in detail in the next section 2.2. The Eigentrust paper models trust by having the source node ask each node in the graph how much it trusts the target, and uses a procedure similar to Pagerank's Power Iteration to obtain a global trust score for each node. The Advogato Trust Metric models trust by assigning a maximum "trust" capacity to a set of seed nodes, and simulating the total trust flow through the graph. Both trust metrics leverage the global network structure in order to compute trust, and rely on other nodes to obtain information about the target node. In this paper, we focus on the less investigated of the two - the Advogato Trust Metric - and investigate ways to use the metric to compare attack resistance among real world trust graphs.

## 2.2 Advogato Trust Metric

The Advogato Trust Metric described in Levien (2004) models trust by assigning the notion of a total "capacity" for trust to a given source node. It uses the concept of network flow to model the propagation of trust among a node and its neighbors. The problem of finding trusted nodes is essentially a single-source, multi-sink problem, converted into a single-source, single-sink problem.

More formally, consider a directed graph $G = (E, V)$, an additional node $S_{sink}$ the supersink, and a seed source node $s$. Each node $x$ is assigned a capacity $c_x$ which is a non-increasing function of the distance away from a specified seed node. The graph is now modified in the following ways: each node $x$ is split into two nodes $x_+$ and $x_-$. For each pair of these split nodes $(x_+, x_-)$, an edge is created from the negative to the positive node with capacity $c_x - 1$. For each edge $(s, t)$ in the original graph, an infinite capacity edge is added between $(s_+, t_-)$. Finally, for the negative component of each node, an edge is formed to $S_{sink}$ with unit capacity. The max-flow algorithm is then run to find the flow across each edge. The set of trusted nodes is then the set of nodes for which there is flow from their negative components $x_-$ to $S_{sink}$.

An upper bound can be shown to exist for the max flow to any "bad" set of nodes. Briefly, assume that nodes are partitioned into three sets:

- "bad nodes", which are declared as malicious actors in disguise
- "good nodes", which point to no nodes in the "bad" group
- "confused nodes", which are good nodes that point to bad nodes

Now, consider the cut in Figure 1 and let the set of good and confused nodes be the set $GC$ and let the bad nodes be the set $B$. The flow across this cut from $GC$ to $B$ is the number of nodes selected. This comes from the fact that nodes are accepted if and only if they have a flow from $x_-$ to the supersink. Hence, the number of bad nodes selected is then the flow from $GC$ to $B$ subtracted from the total number of good and confused nodes selected. The flow from the confused nodes to the bad nodes is then (by similar reasoning) the number of bad nodes selected. For a given confused node, $x$, the max flow it can send to the bad node set is $c_x - 1$. Thus, the max flow that confused nodes can send and hence the max number of bad nodes selected is $\sum c_x - 1$ where the sum is over the edges from the confused nodes to the bad nodes.

There are two hidden difficulties with implementing the algorithm: the choice of seed nodes, and the total capacity distributed at each level. The algorithm requires a seed of "trusted" nodes to begin with; these are combined into the source from which all trust emanates. Furthermore, each set of nodes that are $k$ hops away from the source must receive

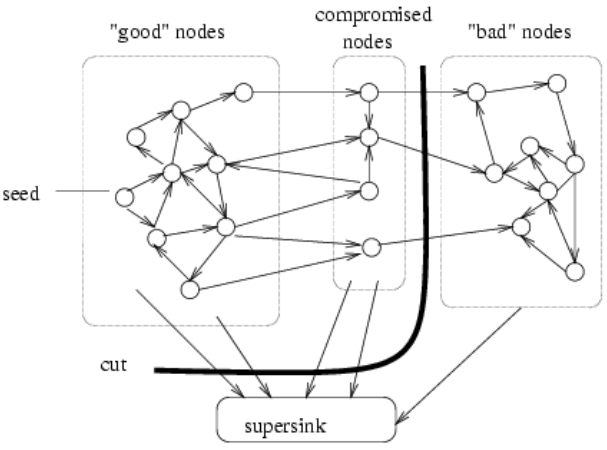



Figure 1: Advogato Attack Model

some trust capacity from its neighbors $k-1$ hops away from the source.

For the Advogato network, the seed nodes were chosen when the site was first created, and represented users that Levien had previously known and trusted. However, in the case of real world graphs, there is no obvious choice of seeds. The second difficulty lies in the capacity function $c_x$ across each hop $h$ away from the seed. The capacities were manually tuned for the Advogato dataset using an unknown procedure, and the problem of obtaining capacities is left underspecified by the paper. We address the question of how to initialize these parameters in our procedures.

### 2.3 Generative Models of Trust Networks

In order to verify and generalize our procedure, we would like to obtain graphs that are representative of real world graphs such as the Epinions or Slashdot reputation networks, while being similar in scale to the Advogato network. Such large graphs are not practical computationally speaking, so we seek smaller versions. To do so, we follow closely a generative model Dandekar (2009) designed to mimic graphical properties of trust networks. To review the work, let's first consider how we might go about modeling a trust network and start with the simplifying assumption that the edges are signed and undirected. A positive edge means trust, and a negative edge means distrust. Trust networks roughly follow preferential attachment Barabasi and Albert (1999). However, modeling the graph this way does not satisfy other properties of these networks, namely the number of triads. Therefore, Dandekar (2009) modeled trust networks through a time evolving process (a copying model R. Kumar and Upfal (2000)) governed by a several rules.

At each time step, an individual, $i$, may connect to the graph. This individual then finds some other individual (node) $j$ to interact with (probability $p$ of connection). If $i$ ends up deciding to trust $j$, then $i$ will trust the individuals that $j$ trusts with probability $r_p$. The individual $j$ may also distrust others, but $i$ need not distrust all of them. In this model, there is a 0.50 chance that $i$ will trust someone that $j$ does not trust. On the other hand, if $i$ doesn't end up trusting $j$, then $i$ won't form an opinion of those that $j$ trusts. For those that $j$ distrusts, $i$ will distrust them with probability $r_n$. This outlines the basic intuition behind the Dandekar generative model and aligns closesly with how we might imagine connections being formed in the real world.

**Algorithm 1** Trust-Copy Model $(n, p, r_p, r_n)$

```
for i = 1 to n do
  Pick a node j uniformly at random from {1, ..., i − 1}
  Create edge (i, j)
  Label the edge + with prob. p, and – with prob. (1 − p)
  for all neighbors k of j do
    if (i, j) is + then
      Create edge (i, k) with prob. r_p.
      if (j, k) is + then
        Label edge (i, k) +.
      else
        Label edge (i, k) + or – with prob. 1/2.
      end if
    else
      if (j, k) is – then
        Create edge (i, k) with prob. r_n.
        Label edge (i, k) –.
      end if
    end if
  end for
end for
```

Dendakar algorithm pseudocode

## 3 Dataset

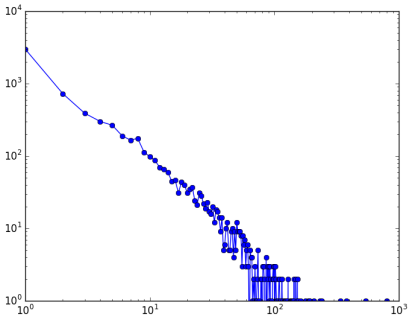
Here, we summarize the datasets used in our description, their network properties, and our preprocessing steps to transform them into representations suitable for comparison.

| | Advogato | Dandekar Graph |
|---|---|---|
| Nodes | 7425 | 7425 |
| Edges | 56461 | 54203 |
| Degree Dist. | Power Law | Power Law |
| Diameter | 9.0 | 20.0 |
| Clust. Coef. | 0.126 | 0.125 |

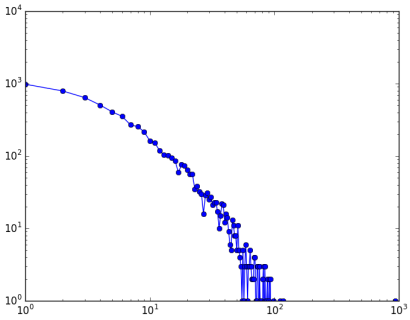
### 3.1 Advogato

The Advogato dataset, obtained from Trustlet.org, represents a snapshot in time of the trust graph model for the site Advogato.com (Ralph Levien), an online community site dedicated to free software development. On Advogato.com users can certify each other on 4 different levels: Observer, Apprentice, Journeyer, and Master. These roles are assigned to control the degree of responsibilities on the site, and thus imply the degree to which this user is trusted. However, since our project focuses on trust as a binary concept, we do not take into account user roles, and simply treat each user as a node. An edge $(s, t)$ indicates that $s$ has declared trust in $t$. There are no negatively weighted edges to downvote bad users. Instead, users that deemed to be bad have their certificates removed, causing their in-bound edges in the trust graph to be removed. This stops trust flow to the node, and effectively removes the user from the trusted community.

With regards to network properties, the graph



(a) Advogato Deg. Dist



(b) Dandekar Deg. Dist

Figure 2: Degree distribution

follows a preferential attachment model in its degree distribution (see Fig 2), and is roughly similar in clustering coefficient.

### 3.2 Dandekar Model

Based on the Dandekar algorithm described in the previous section, we generate a graph with the same number nodes and approximately the same number of edges as Advagato. As mentioned previously, this is a singed undirected graph where as Advagato is only a directed graph. We argue, however, that Dandekar still is an appropriate model for generating this graph.

In any real life like networks, there is always the implicit notion of distrust. Even if there isn't a "distrust" button, a user finds out who is not trustworthy by communicating to other users. Hence, a notion of negative edges or distrust has to be considered in a generative model of trust graphs, and we say that Advagato represents a trust graph with the negative edges removed. To further our claim, we modified Dandekar to a simple procedure of excluding negative edges. We found that it was actually impossible to produce a graph similar to Advagato (in terms of network properties) with this only positive edge model.

Our construction process is then to run the Dandekar algorithm and generate a graph with positive edges equal to the number of edges in Dandekar. Our parameters are $p = .65, r_p = .75, r_n = .4$. We then discard all negative edges in our graph. It should be noted that we seek a graph with 80% positive edges and 20% negative edges. This originates from the fact that other real world trust graphs (such as Epinions and Slashdot) have similar properties Dandekar (2009).

Our final step is to convert our undirected graph into a directed graph. Our method is as follows: for each undirected edge $(s, t)$, we convert it to a directed edge $(s, t)$ or $(t, s)$ or convert it to two bi-directed edges. Each of these three choices has an associated probability. We fix these probabilities so that we end up with about half of the edges being bi-directed and the remaining just directed in one way.

To verify that our graph is similar to the Advagato graph, we consider several network properties as seen in the table. In particular, we compute the degree distribution (Figure 2), clustering coefficient and network diameter. We see that all are very similar except for the network diameter. We note that the meaning of triads is different in this context as these graphs only have positive directed edges, thus we rely on the clustering coefficient. More practically speaking, we couldn't find the number of triads computationally speaking as it was too slow to run.

## 4 Methods

Below, we present the steps of GUACAMOLE, our novel evaluative procedure, with discussion on how each step was implemented. We also explain the mathematical reasoning and / or interpretation of our devised steps such as seed initialization and attack models.

At a high level, our procedure defines attack resistance as a classification task: given an initial set of trusted nodes found by the trust metric, how well is the graph able to distinguish a trusted node from a compromised node after an attack? To do so, consider a set of trusted nodes $V_{trust}$ found by the Advogato Trust Metric before an attack, and $V_{found}$ after the attack. We would like to create a score that represents the difference of these two sets. Two types of errors can arise:

#### 4.0.1 False Negative

Trusted nodes are deemed as untrustworthy. In practice, false negatives often result in unavailable services to trusted users.

#### 4.0.2 False Positive

Compromised nodes are deemed as trustworthy. False positives almost always lead to security breaches as untrusted users gain access to restricted service or sensitive information.

In real world networks, false positive results compromise networks to a greater degree, and must be avoided. Thus, we use au-ROC to measure the performance of our scoring metric. In our context, the area under the ROC-curve represents the probability that our algorithm ranks a trusted node higher than a compromised node.

**Algorithm 4.1** GUACAMOLE Algorithm

1: $V_{seed} \leftarrow SelectSeed(V)$
2: $C \leftarrow AssignCapacities(V, C_{max})$
3: $G_{expanded} \leftarrow ExpandedGraph$
4: $V_{true} \leftarrow AdvogatoMaxFlow(G_{expanded})$
5: $G_{attacked} \leftarrow RunAttackModel(G_{expanded})$
6: $V_{predicted} \leftarrow AdvogatoMaxFlow(G_{attacked})$
7: **return** $|V_{predicted} - V_{true}|/|V_{true}|$

Below, we elaborate on the details of each step.

## 4.1 Computing Advogato Set of Trusted Nodes

As mentioned in the previous section, there are two hidden difficulties with implementing the algorithm: the choice of seed nodes, and the total capacity distributed at each level. For the Advogato network, the seed nodes were created at the creation of the site, and represented users that Levien personally knew. However, in the case of real world graphs, such seeds could lead to biased results. Consider the situation where all the seeds appear in a small connected component. In this situation, Advogato trust metrics fail to assign trust to the rest of the network. An effective strategy of choosing seeds in Advogato and similar networks is selecting nodes of highest in-degrees because they are most trusted by the majority. This heuristically also improves the probability of choosing a set of seeds that connects to the entirety of the network. Often times, networks also contain distrust edges in addition to the trust edges. This motivates us to design a more generalized seed selection strategy. With the distrust edges, we can more accurately assess the "public opinion" on a particular node: a node is trusted only if it has many trust edges and few distrust edges. In the meantime, we would still prefer nodes with large number of trust edges than those with few trust edges according to our heuristics mentioned above. Therefore, we decide to select seeds with the highest "trust score", which is defined as:

$$Score = (\frac{P}{N})(\sqrt{P})$$

In this formula, $P$ is the number of inbound trust edges (positive in-degree) and $N$ is the number of inbound distrust edges (negative in-degree). Our formula punishes nodes with many distrust edges using the $\frac{P}{N}$ term and favors our heuristics through the $\sqrt{P}$ term.

## 4.2 Assigning Capacities

The second difficulty in the GUACAMOLE procedure is defining the trust capacities, denoted as $C$. $C$ can be specified as a list where the $i$th element determines the trust capacity of nodes of distance $i$ away from the seed. The Advogato algorithm sets $C$ to be $[800, 200, 200, 50, 12, 4, 2, 1]$ but it does not disclose its way of choosing $C$ (Advogato.com). GUACAMOLE uses the same $C$ to conform to Advogato's standards.
We have also implemented another mechanisms of choosing $C$ as described by Golbeck (2010). Golbeck's algorithm first chooses the initial trust capacity for seeds to be 800, same as the Advogato algorithm, and then determines the trust capacity at distance $l$ using the capacity at distance $l-1$. Specifically, we choose the capacities incrementally using this formula:

$$C_l = \frac{C_{l-1}}{\text{Out-Degree}(N_{l-1})/|N_{l-1}|}$$

where $N_l$ is the set of nodes of distance $l$ from the seed.
We discovered that both Advogato's choice of capacity and Golbeck's capacity lead to similar results, so we use the former for our experiments for its simplicity.

## 4.3 Max Flow to Obtain a Set of Trusted Nodes

To implement the trust metric, we follow the algorithm outlined in section of Advogato and implemented it. That is, we first construct this modified graph and run max flow on it. $x\_$ nodes that have

flow to super sink are labeled as trusted. A slight modification must be made in the case that we have multiple seed nodes. Normally, the Advogato algorithm would assume a single seed node. To make the modification, we create a new source node and attach it to all the nodes our seeds pointed to. We then remove all the seed nodes. Finally, we run max flow and say that the nodes that have a flow to the supersink are trusted.

### 4.4 Attack Model

Our attack model simulates three different types of attacks on the Advogato and generated trust graphs:

- **Delete** Node are removed from the network. Attackers have limited influence, and can only bring down communication to and from a node. Given a node $v$, we remove the node as well as all its neighboring edges from the graph.
- **Targeted** Several nodes are compromised. Attackers control a proportion $\alpha$ of nodes in the network and link to attackers. By redirecting trust edges, attackers are redirecting flow to other compromised nodes in an effort to increase the chances that they will be part of the final set of trusted nodes. Given a node $v$ and a set $V_{compromised}$, we remove all edges pointing to / from $v$ and create edges from $v$ to each node in $V_{compromised}$.
- **Reverse** Compromised nodes pose as good citizens. Again attackers control a proportion $\alpha$ of nodes, but this time link to both good and bad nodes. Linking to good nodes increases the chances that these nodes will be perceived as good. Given a node $v$, we remove all edges pointing to / from $v$ and create edges pointing from $v$ to randomly chosen nodes while maintaining the original in and out degrees of $v$.

For each attack type, we vary the $\alpha$ and $\beta$ parameters described in 4.4.2, and record the ROC curve for each trial; each trial is also run 100 times, and the results averaged across those runs in order to obtain a result with roughly 0.95 confidence interval.

#### 4.4.1 Choice of Nodes

We aimed to choose nodes on the augmented graph $G_{flow}$ in a way that corresponds to a real world attack scenario. In a real network, the nodes that are "trusted" and have a high degree of connectivity tend to be much harder to compromise, while the nodes that are on the edge of the network and less "trusted" have a higher likelihood of being compromised. To mimic these properties, we select nodes to compromise inversely proportional to the distance from the seed node $s$. probability $P(v) = 1/d(v)$ where $d(v)$ denotes the number of hops between d and the source node in breadth-first search. The following function is used to assign capacities for each tier of nodes at distances $1, 2, ...k$ from the source nodes.

#### 4.4.2 Parameters

We list the parameters of our attack model below along with our selected range of values to test:

- $\alpha$ Number of nodes compromised. This is a relatively small number compared to the number of nodes in the graph. We varied this parameter across the values $\alpha = 1, 0.01N, 0.5N, 0.1N, 0.25N$.
- $\beta$ Attack model. This can take on 3 possible values, one for each attack type listed above.
- $\gamma$ Proportion of edges compromised, as a fraction of the total number of nodes in the graph. We vary this across the ranges $\gamma = 0, 0.05, 0.10, ...1$; our results are in the subsampling section.

## 5 Results and Evaluation

### 5.1 Attack Simulation

These ROC curves show an expected pattern: as the number of nodes compromised increases, the trust metric is increasingly unable to distinguish between a good and bad node regardless of attack method. This shows that our attack model, when compromising beyond 0.05 of the total nodes in the network, makes it difficult for the trust metric to distinguish between trusted and untrusted nodes in the trust graphs.

When we compare attack methods, we observe that deletion affects classification the least, as the decline in the shape of the ROC curve is steady. Curves for reverse and targeted attack types decline drastically toward the $y = x$ random line when the number of compromised node increases compared to deletion. This makes sense since deletion only removes nodes from the network, whereas reverse and targeted attack models influence the edges of the graph, effectively redirecting trust flow. Thus, we can infer that the redirection of edges have a larger effect on the trust metric than the removal of nodes and edges. We pursue this line of analysis

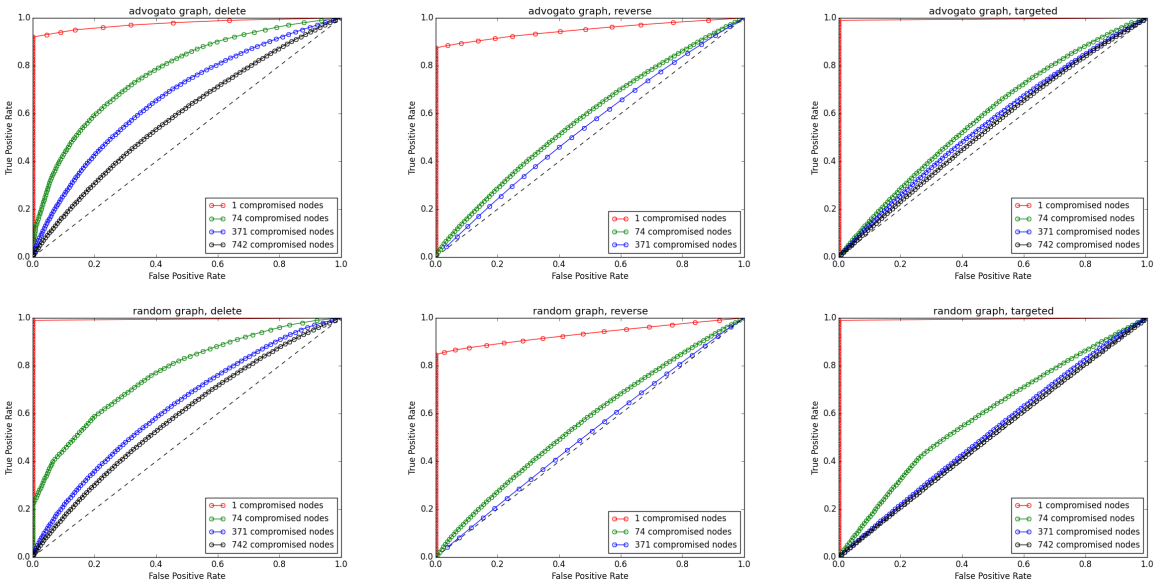



Figure 3: ROC curves for several attack scenarios on the Advogato and Dandekar graphs. Colors correspond to the $\alpha$ parameter, the number of nodes compromised

| $\alpha$ | $\beta$ | auROC |
|---|---|---|
| 1 | Delete | 0.978 |
| 1 | Reverse | 0.949 |
| 1 | Target | 0.995 |
| $0.01\|V\|$ | Delete | 0.773 |
| $0.01\|V\|$ | Reverse | 0.579 |
| $0.01\|V\|$ | Target | 0.579 |
| $0.05\|V\|$ | Delete | 0.671 |
| $0.05\|V\|$ | Reverse | 0.540 |
| $0.05\|V\|$ | Target | 0.556 |
| $0.1\|V\|$ | Delete | 0.591 |
| $0.1\|V\|$ | Target | 0.537 |

Table 1: au-ROC scores for Advogato

| $\alpha$ | $\beta$ | auROC |
|---|---|---|
| 1 | Delete | 0.995 |
| 1 | Reverse | 0.933 |
| 1 | Target | 0.995 |
| $0.01\|V\|$ | Delete | 0.771 |
| $0.01\|V\|$ | Reverse | 0.564 |
| $0.01\|V\|$ | Target | 0.594 |
| $0.05\|V\|$ | Delete | 0.631 |
| $0.05\|V\|$ | Reverse | 0.514 |
| $0.05\|V\|$ | Target | 0.523 |
| $0.1\|V\|$ | Delete | 0.590 |
| $0.1\|V\|$ | Target | 0.509 |

Table 2: au-ROC scores for Dandekar

further in 5.3.

Next, we compare the graphs themselves, looking at classification performance in the au-ROC tables. From the table, we can clearly see that the Advogato network and the Dandekar generated graph share similar attack resistance properties. Both perform similarly for all 3 attack models, despite differences in their network diameter. This lead to an investigation of the scale invariance of the trust metric, which we investigate in the next section, with the following motivation: given two graphs of different scales, could we still make a fair comparison between their attack resistance?

## 5.2 Scale-Invariance

Based on our simulation results, a question to investigate was whether trust score was sensitive to the scale of the input graph. If the score is independent, we can in theory compute trust scores for arbitrarily large graphs by sampling a subset of nodes and computing the score for the downsized graph. Additionally, this graph could be computed with our generative model such that it still satisfied the same properties of the original graph.

To test this idea, we used the Dandekar generative model with the same probability settings as before but with higher and lower node counts. We observe that the model produces roughly the same graphs in terms of node to edge ratio, clustering coefficient, and diameter to our previously gener-

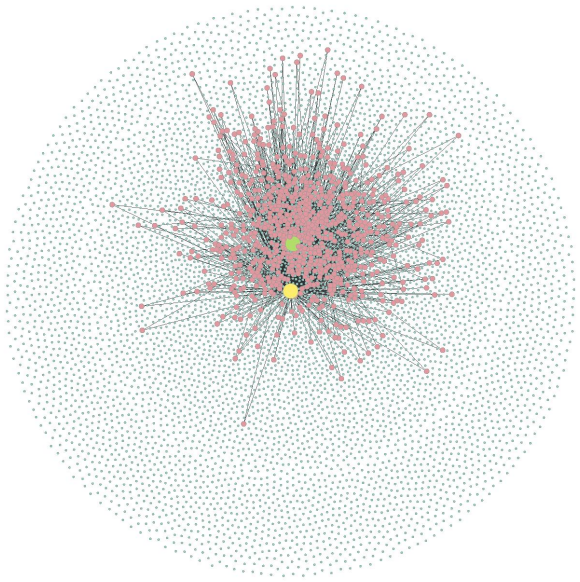


Figure 4: Visualization of the trusted set for the Advogato Graph pre-attack. Parameters are $\alpha = 1$, $\beta = 0.1$. Red nodes indicate trusted nodes. Green represents the seed and yellow the target.

ated graphs. In the interest of time, we fixed our parameters for ROC calculations to be $\beta = 0.1$, $\alpha = DEL$.

We then ran our attack model and computed ROC curves. We observe that, surprisingly, scale does seem to affect the ROC curves; for larger graphs, the curve tends to be flatter, given the same percentage of compromised nodes. We speculate that this could be due to the selection of compromised nodes, but this issue merits further investigation. Our current theory is that the correctness of the advagato trust metric scales linearly with the number of edges modified instead of nodes. However as we linearly increase the number of compromised nodes, the number of edges increases quadratically.

### 5.3 Sub-sampling

Another area we looked into involved looking at the effect of choice of edges on a graph's attack resistance. Since this is a broad area of investigation, for this project, we look at the effects of sub-sampling on the true positive rate of our attack model. Given a subset of edges with which trust flow is constrained, how accurately can we detect trusted nodes? To understand Advogato algorithm's behavior with sub-sampling, we made a hypothesis: sub-sampling has a larger influence on nodes with low in-degree than those with high in-degree. This is because nodes with low in-degree are more likely to become isolated to neighbors who are trusted, and further risk becoming completely isolated. At the same time, nodes with low degree are more likely to be the untrusted nodes since flow is less likely to reach them in the max-flow step. Hence, sub-sampling at a moderate ratio does not influence the correctness to a large degree.

However, as we sub-sample larger amounts of edges, the graph dissolves into smaller, isolated SCCs. This means that trust would only flow to those nodes living within the same SCC as the one the seeds belong to. This is supported by our sub-sampling simulation results: we observe a large drop in the true positive rate when we sub-sampling more than 0.60 of the edges.

To verify our hypothesis, we plotted the size of the largest SCC on the same x-axis as our sub-sampling graphs, and looked for a similar drop in the size of the largest SCC. From the figure, we observe that there was indeed a similar drop in largest SCC size at a sub-sampling of 0.60 of the total edges. We can conclusion that in both Advogato network and our random model, the Advogato algorithm still provides high true positive rates (¿0.85) even if we randomly discard half of the edges.

Interestingly, this also suggests that sub-sampling of edges is a potentially viable method to accelerate the algorithm to calculate the Advogato Trust Metric at the cost of a tolerable amount of inaccuracy. The bottleneck of Advogato is the max-flow algorithm, which has complexity $O(|E| * f)$, where $E$ is the set of edges in the graph and $f$ is the maximum capacity throughout the graph. Thus, by sub-sampling the number of edges, we may still be able to compute an approximate attack resistance.

## 6 Conclusion

In this report, we proposed a novel procedure for evaluating and comparing attack resistance between two graphs. We ran a simulation to evaluate the procedure on a real world trust graph and a generated trust graph. We observed from our results that the graphs performed similarly regardless of attack type and parameters.

Next, we investigated a couple of directions based on our simulation results. First, we looked into scale invariance, and observed that attack resistance was not scale-invariant. Second, we looked into sub-sampling the number of edges, and found that we could remove some percentage of the edges while maintaining a relatively good approximation of the attack resistance. This gave us a potential way of using down-sampling as a technique to speed

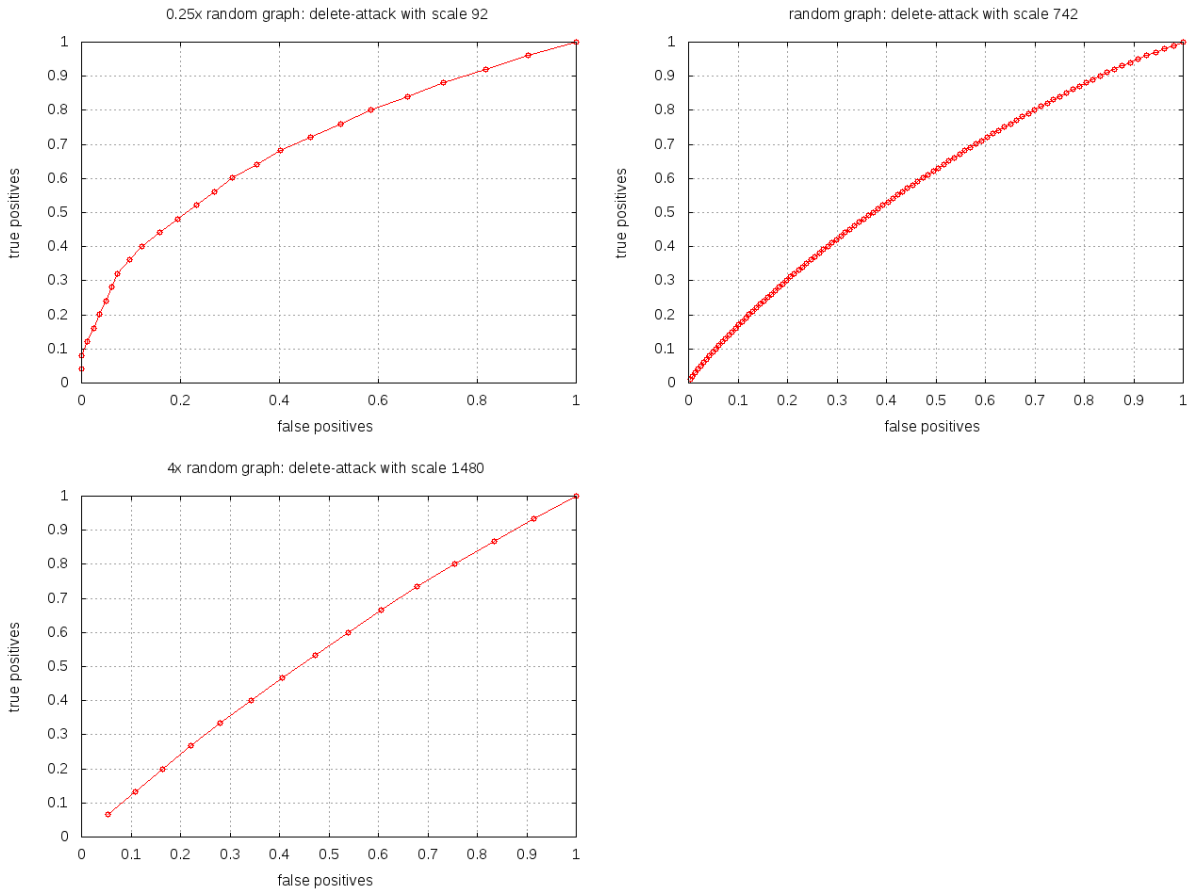



Figure 5: ROC curves for various network scales. For these experiments, we fixed our parameters $\alpha = 0.1$ and $\beta = DEL$

up the max flow algorithm and approximate relative attack resistance between graphs that have different numbers of edges.

There were many limitations to our project. First, our evaluative procedure is only tested on two types of real world trust graphs that are similar. Although there is work to support that these models are representative of real world trust graphs, it would strengthen our argument to test the procedure on a wide variety of trust graphs. Second, given more time and computational power, we would have further investigated the lines of analysis started in 5.3 and 5.2. Future works could look into either one of these as extensions to the current procedure. Finally, one interesting extension would be to take the GUACAMOLE procedure and modify it to use the Eigentrust metric; this would prove useful in the majority of contexts where the Eigentrust metric is computationally cheaper to calculate than the Advogato Trust Metric.

# References

A.-L. Barabasi and R. Albert. Emergence of scaling in random networks, 1999.

P. Dandekar. Analysis and generative model for trust networks, 2009.

J. Golbeck. *Computing with Social Trust*. Springer Publishing Company, Incorporated, 1st edition, dec 2010.

S. Kamvar, M. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks.

R. Levien. Advogato's trust metric. URL `http://www.advogato.org/trust-metric.html`.

R. Levien. Attack resistant trust metrics. Draft of Raph Levien's PhD thesis in compact formatting, 2004.

S. R. D. S. A. T. R. Kumar, P. Raghavan and E. Upfal. Stochastic models for the web graph. *In*

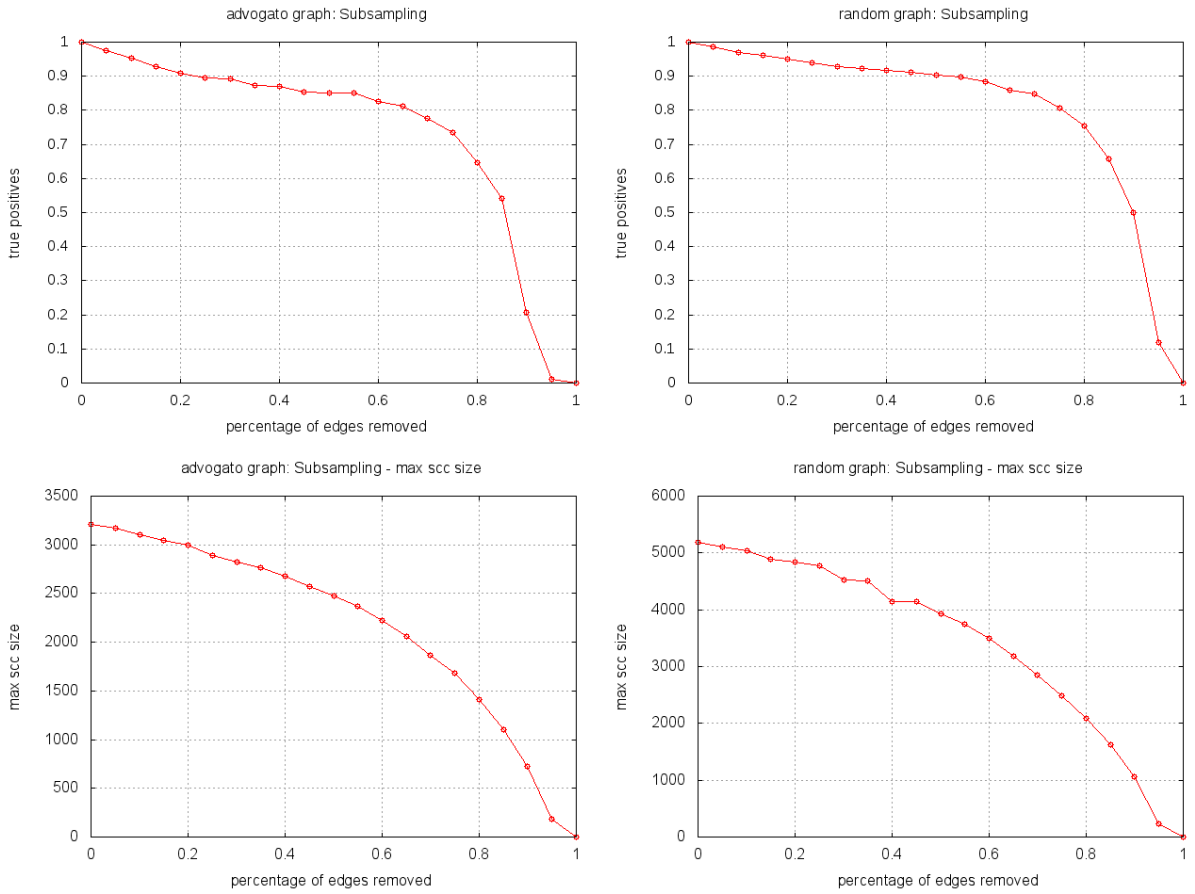



Figure 6: (Top Row) Effect of edge sub-sampling on the true positive rate. (Bottom row) Effect of edge sub-sampling on max SCC size. X-axes are aligned

*FOCS '00: Proceedings of the 41st Annual Symposium on the Foundations of Computer Science*, 2000.

M. L. Xinxin Fan, Ling Liu and Z. Su. Eigentrust++: Attack resilient trust management. *8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing.*