Unifying Unknown Nodes in the Internet Graph Using Semisupervised Spectral Clustering

Anat Almog
School of Electrical Eng.
Tel-Aviv University, Israel
anat@eng.tau.ac.il

Jacob Goldberger
School of Engineering.
Bar-Ilan University, Israel
goldbej@eng.biu.ac.il

Yuval Shavitt
School of Electrical Eng.
Tel-Aviv University, Israel
shavitt@eng.tau.ac.il

Abstract

Most research on Internet topology is based on active measurement methods. A major difficulty in using these tools is that one comes across many unresponsive routers. Different methods of dealing with these anonymous nodes to preserve the connectivity of the real graph have been suggested. One of the more practical approaches involves using a placeholder for each unknown, resulting in multiple copies of every such node. This significantly distorts and inflates the inferred topology. Our goal in this work is to unify groups of placeholders in the IP-level graph. We introduce a novel clustering algorithm based on semisupervised spectral embedding of all the nodes followed by clustering of the anonymous nodes in the projected space. Experimental results on real internet data are provided, that show good similarity to the true networks.

1. Introduction

The continual expansion of the Internet is daily proof of its growing impact. Research on the myriad aspects of internet has followed pace and is indeed crucial to any number of sectors. Concomitantly, research of various aspects of the Internet has proved to be essential. Internet maps are particularly of interest to many different communities, and as such, need to be as accurate and detailed as possible.

Internet maps are created at different levels of abstraction, the most common of which are: the autonomous system, address prefix, router, and IP. In this paper we focus on IP-level maps. Most IP-level Internet maps are created by aggregation of paths. The maps consist of nodes that are labeled by IP addresses, and edges that connect nodes which appear consecutively in some path. The paths are acquired through the use of active measurement methods — methods that actively send probe packets and use the reply packets to gather information about the replying routers, such as their IP address and the time delay of the probe packet [11, 3, 7]. The key obstacle is those routers that do not respond to probes, and hence remain anonymous: their IP address is unknown and so are the time delays of packets that reach them. Such anonymous routers can appear in many routes, but because their IP addresses are unknown, there is no way to tell whether they are separate anonymous routers or different instances of the same one. Thus multiple copies of each anonymous router appear in the aggregated map, causing an inflation of nodes, and a distortion of various characteristics of the true map. In the following text we call these copies *unknowns* (or unknown nodes), and the unresponsive routers, which they represent, are called *unknown roots*. A common and practical approach for dealing with unresponsive routers in an IP-level map that was suggested by Broido and Claffy [2], entails using a unique placeholder for each unknown. However, this approach perpetuates the inflation of the graph, as all of the unresponsive router copies are retained. Placeholder graphs contain many more nodes than IP-level graphs. The placeholder graphs have a number of nodes that is larger by an order of magnitude than pure-IP graphs (with no unknown nodes). Close to a third of their probed paths contain unresponsive nodes. When aggregating paths to form the topology, the over-representation becomes even worse, as multiple instances of known nodes from different paths are unified, while those of the unknowns cannot be merged. When analyzing DIMES [11] data from 10 arbitrarily chosen weeks, (from the end of 2007 and the beginning of 2008), we found that on average, 39.55% of all nodes were unknown, and 28.54% of the edges were adjacent to an unknown node. Another problem with using placeholders, is that although they preserve connectivity and hop count, they also change other properties of the graph, e.g., the degree distribution and clustering coefficients are altered significantly in the placeholder graph.

Several recent works have discussed with the inflation of the graph as a result of unknowns. Yao et al. [14] deal more with the theory of the problem. They formalize the problem of topology inference in the presence of anonymous routers, and after having formulated it as an optimization problem, show that both an exact and an approximate solution are intractable. They further suggest heuristics for minimizing the produced topology, but these have high computational complexity, and most of their experimentation was performed on very small networks. Jin et al. [6] suggest two methods of merging unknown nodes. The first uses ISOMAP to map nodes into a multidimensional space. In the projected space, it employs simple thresholding to basically merges any pair of unknown nodes that lie within a given threshold of each other. It also merges unknowns that share a neighbor and lie within a second threshold of each other. These thresholds are set according to experimental results on a known network. Nevertheless, these values may not be applicable to other network configurations. The second heuristic is a neighbor matching approach, that merges all nodes that share a neighbor and do not appear in the same traceroute path. In our experience, this is a very simplistic approach, and may result in inaccurate results, as the authors also pointed out. The most recent work that we know of is [5]. The authors rely on their knowledge of the domain to identify different structures that occur in graphs with unknown nodes. They propose a graph-based induction approach that detects these structures in the graph, and merges the unknowns in them to their corresponding unknown roots. This method uses only connectedness information and completely ignores the delays. It also relies on the completeness of the information, but measurements are frequently obtained from only a few vantage points, so that much information is missing.

In this paper we describe a method of recreating an IP-level graph, in which the unknown nodes are not overrepresented by unifying groups of unknowns, that are likely to stem from the same unknown root. We introduce a novel clustering algorithm based on semi-supervised spectral embedding followed by unsupervised clustering in the projected space. Applying our algorithm to placeholder maps yields IP-level maps that are much more meaningful and usable. By merging groups of unknowns, we arrive at a good approximation of the real graph, which can then be used to study the properties of the IP-level graph. The improved performance of the proposed method is demonstrated on large real internet data collected by the DIMES project [11].

The paper is organized as follows. In Section 2 we describe the IP distance matrix. In Section 3 we present our proposed clustering algorithm. Section 4 describes the Internet measurement data collection and pre-processing. Experimental results are shown in Section 5.

2. The IP Distance Matrix

The input to our clustering algorithm is a distance (time-delay) matrix. In this section we describe how active measurements are utilized to construct the matrix. Active measurement methods play an important role in studying the Internet topology. Specifically, the traceroute tool is widely used, for instance when building maps of the Internet through aggregation of routes. Traceroute works by sending a series of probes with increasing TTL (Time To Live) values. For every packet, including the traceroute probing packets, the TTL value is decreased by one by each router on the way to the destination. Once the TTL field reaches zero, the packet is discarded and an ICMP *TTL exceeded* packet is sent by the router to the packet source. Thus, sending a series of packets to a specific destination with increasing values of TTL should yield the IP addresses of all routers in the path between the origin and the destination, and the time it took each packet to traverse between the origin and the responding router and back. Aggregation of several paths results in a map, a graph in which the nodes are IP addresses, and the distances between them are defined based on the delays. It is this graph that we want to represent as a distance matrix.

While sending ICMP messages is common practice, many routers are configured to discard packets without responding with an ICMP *TTL exceeded* message. If removed from the emerging path, these unresponsive routers create gaps in an otherwise connected route. Thus, other methods have been suggested for dealing with non-responses. The placeholder method [2] entails assigning a placeholder to each gap in a route, that is, for each non-response. For example, if no response was received from a router between routers A and B, its placeholder node will be labeled A-1?1-B. A placeholder is unique per immediate neighbor pair. As most routers are part of several routes, they may have different pairs of immediate neighbors when measuring different paths. The result is that most unresponsive routers are represented by multiple copies in the placeholder graph.

The next step towards obtaining a distance matrix is to define a distance measure on the edges of the placeholder graph. The distance metric in many cases, as in ours, is based on time delays from the measurement origin. We define $delay(A, B)$ as the measured round-trip time of the path $A \rightsquigarrow B$, (the measured time for a packet to traverse from A to B and back). The unary form $delay(A)$ is sometimes used to denote the delay between the source of the measurement and A. We construct an edge in the graph for every pair of nodes A and B that are adjacent in some measured path. If a path was measured in which a response was received from A followed by a response from B, then the edge (A, B) is added to the graph. The time difference between these two responses can serve as an estimation of $delay(A, B)$, or the distance between A and B, and is set to be the value of the edge that connects them. Note that this value is only an estimation of the edge's true delay. As mentioned in [5], using round-trip path delays in order to estimate one-way edge delays can be difficult. In [4] the authors present a method of calculating a good estimation efficiently. However, our experimental results empirically show that the difference in round-trip delays serves as a good enough estimation for our purposes. Thus, a distance matrix D can be defined, where

$$D_{ij} = \begin{cases} \min\{delay(i,j), delay(j,i)\}, & \text{edge } (i,j) \text{ or } (j,i) \text{ exists;} \\ \infty, & \text{otherwise.} \end{cases} \tag{1}$$

and

$$\text{for any edge}(j,i) : delay(j,i) = delay(i) - delay(j) \tag{2}$$

In other words, for any two nodes i, j that are neighbors in some measured route, the distance between them is defined as the delay estimation. Nodes that are not connected are signified to be so by a distance value of ∞. Note that we regard the distances as symmetrical in this work, even though the actual delays are not.

Distances of unknown nodes need to be defined differently. These nodes are an exception as they each have two neighbors, but the distance between them and their neighbors is unknown. Nevertheless, this distance has to be defined, as it will be needed in the embedding process. The unknown distances are actually estimated when building the map from measurements in the following manner. Suppose path $S,\ \dots,\ A,\ A\text{-?-}B,\ B$ is measured. Then

$$D_{A,\ A\text{-?-}B} = D_{A\text{-?-}B,\ B} = \frac{(delay(S,B) - delay(S,A))}{2} \tag{3}$$

This distance matrix is the input to our algorithm — from it we build the affinity matrix that is the input of the embedding.

3. Semi-Supervised Spectral Clustering

The problem we are dealing with can be described as a clustering problem. However, it does not fit any of the three categories: unsupervised, supervised, or semi-supervised, usually used to differentiate learning tasks. Most of the nodes in our problem have labels (IP addresses), and the edges adjacent to them have a known value (delay). The rest of the nodes are unknown: they are missing a label, and the values of their adjacent edges are missing as well. Using all the information, we need to cluster only the unknown nodes. Since the problem calls for clustering based both on labeled and unlabeled data, it seems to fit the semi-supervised category. However, it differs in two important ways from common semi-supervised tasks: first, we use all the information prior to clustering, but the clustering itself works only on the unknown (unlabeled) nodes. In typical semi-supervised tasks, the algorithm would have to cluster labeled data correctly as well, whereas here it clusters only the unlabeled data. Second, in most semi-supervised problems there is no difference in the distance calculation of labeled and unlabeled nodes. In our problem, since the edges adjacent to the unlabeled nodes have unknown values, their values need to be estimated, while other edges have a given value. These differences, and others, make this problem unique in the way it uses labeled and unlabeled data.

We used the *spectral clustering* method described by Ng, Jordan, and Weiss [8], and modified it to fit the unique requirements of our clustering task. The original method is a variant of *kernel PCA* followed by *k-means*; for details see [8]. In the following sections we describe the specific issues and modifications to this algorithm that arise from the uniqueness of our problem.

3.1. Spectral Embedding

There are several works that have suggested embedding the Internet AS and address prefix graphs in metric spaces [9, 12, 13]. These levels of abstraction, unlike the IP level, do not suffer from the problems we set to solve in this paper.

The nodes of the Internet graph reside in a non-metric space, due to the fact that the triangle inequality does not hold with distances defined as the delays. For this reason, clustering the data per se is not a viable option. Our intuition was that embedding the data into a multi-dimensional metric space would preserve the affinity relations of the data — in other words, that Euclidian distances of the embedded data in the new space would be meaningful, and would be an indication of the true distance between nodes in the origin space. This indeed proved to be the case, as we will show later on in the results.

The first part of our algorithm, then, embeds the data into multi-dimensional space, using the spectral embedding method described in steps 1–4 of the algorithm in [8]. That is, it takes as input the distance matrix D described in equations (1)–(3), builds the required affinity matrix A from it, and uses it in the embedding process:

1. Form the affinity matrix A defined by

$$A_{ij} = \exp\left(\frac{-D_{ij}^2}{2\lambda^2}\right)$$

2. Define B to be the diagonal matrix whose (i,i)-element is the sum of A's i-th row, and construct the matrix L

$$L = B^{-\frac{1}{2}} A B^{-\frac{1}{2}}$$

3. Find the k largest eigenvectors of L and form the matrix X by stacking the eigenvectors in columns.

4. Form the matrix Y from X by renormalizing each of X's rows to have unit length.

Matrix Y is then the input to the clustering part (see section 3.2).

An important issue was **deciding which part of the data to embed**. We looked at three options: embedding only unknowns, embedding all nodes, or embedding only known nodes. As the only nodes we want to cluster are the unknowns, it seems logical that we only need to embed them. Obviously, though, most of the information would not have been utilized. Although we are only interested in a small group of nodes, it is clear that embedding the entire map will yield a much more accurate picture of all nodes, and specifically of the unknown nodes. Moreover, there is hardly any useful information about the unknown nodes themselves: the weight of the edges adjacent to them is also unknown; and their number, compared to all nodes, is very small. Everything we know about these unknowns is derived from their location in the graph, and hence from their neighbors.

This leads to the conclusion that all the nodes should be embedded. However this idea is also problematic, due to the missing edge values adjacent to the unknown nodes. Embedding the unknown nodes requires an estimation of these edge values, and there is no good way to do this. However, as we found in our experimentation, estimation of these edges is good enough, or at least, there are not enough of these unknown edges to damage the layout of the graph in the embedded space.

The third option is to embed only the known nodes, which form the majority of the nodes in the graph, and place the unknown nodes manually in the embedded space. The only real information we have about an unknown node is its neighbor pair (an unknown always has exactly two neighbors). This is easy to maintain in the embedded graph by simply placing the unknown node in the embedded space between the nodes which were its neighbors in the origin space.

In the end, examination of all options led us to use the second option - estimate the missing edge values in the origin space, and embed all nodes, including unknowns. It is clear that the first option is the worst. It is less clear why the second option should be better than the third, even though it was experimentally found to be. Apparently, errors added when estimating the missing edge values in the origin space are negligible most of the times, and the extra topological information is very helpful. This is similar to other semi-supervised tasks, where using the unlabeled data during the training stage improves the resulting classifier.

3.2. Clustering

The second part of our algorithm runs a simple *k-means* algorithm over the embedded data, as in steps 5–6 of the algorithm in [8]. The main difference is that we only cluster a small portion of the embedded data - only the unknown nodes. The known nodes (and edges), which are most of the graph, are just used in the embedding process, as there is no need to label them.
The original algorithm requires k, the number of clusters, as a parameter. In the general case of an Internet graph, as in most real-world clustering problems, this number is also unknown. Thus, to be able to run the algorithm on real data, we added a part that determines the correct number of clusters.

The size of the graphs on which we perform k-means is usually quite small, since we cluster only the unknowns. Therefore it is possible to search over different values of k, and choose the best one. Also, as our original graphs are also not very large, we assume that the number of unknown roots does not exceed 5, and can limit our search to k between 1 and 5. This search range can easily be expanded, and without a heavy time penalty. It is natural to use the distortion of the k-means algorithm as a goodness measure. However, distortion decreases monotonously with the number of clusters - the more clusters, the better. In actual clustering problems, and specifically in ours, this is not the case. While minimizing the distortion, one still needs to avoid over-fitting. The solution is to find the number of clusters after which the distortion decreases less swiftly. That is, find the "knee" (the inflexion point) in the distortion vs. number of clusters graph, (see Fig. 9). We define the "knee" of the graph using the slope of the curve: the point where the slope first decreases to less than 45°. Based on this definition, the algorithm successfully chooses the correct number of clusters most of the times. What is even more interesting, is that in some of the cases in which the algorithm chooses the **wrong** number of clusters, the resulting clustering is actually **better** than the one in which k is known. That is, the algorithm manages to select the number of clusters that are separable. We will elaborate on this in section 5.

3.3. Algorithm

As input we are given a set of nodes, most of which are labeled, (the known nodes), and the rest of which are unlabeled, (the unknown nodes). We are also given an unnecessary full distance matrix that contains inter-distances of the labeled nodes, and estimations of distances between the unlabeled nodes and their neighbors. Creation of the distance matrix is described in equations (1)–(3). Our goal is to cluster the unlabeled nodes.

1. Embed **all** nodes, (both labeled and unlabeled nodes), into a metric space, by using the *spectral embedding* algorithm described in section 3.1.
2. In the new projected space, cluster only the **unlabeled** nodes using *k-means*.
3. Merge all nodes from each cluster to a single unknown root.

4. Data Description and Preprocessing

Our data were taken from the DIMES[11] database. This DB is based on *traceroute* measurements performed routinely by hundreds of agents. The raw data obtained from these measurements is aggregated to yield topologies at different levels - the IP, router, and autonomous system levels. We query tables of the IP-level topology for nodes and edges. The value of an edge is taken to be the minimal that exists for it in the DB. We aim to apply our algorithm to data of complete networks, but obviously not of the entire Internet map as a whole. Note that for our tests, rather than acquiring a complete network from the database, we build maps of smaller subsets the network (the number of nodes is on the order of hundreds and up to a few thousands). This same process of dividing a map into several smaller maps and working on each separately could also be used when working with complete network maps. This is possible since unknown nodes stemming from a single root should all be in a fairly tight neighborhood. Also, running our algorithm on maps of this size is fast, thus avoiding the complexity issue mentioned in [5].

Data in the DIMES DB, which are real aggregated data, naturally contain many unknown nodes. Since these are actual unknown nodes, there is no known labeling for them, rendering them useless for our tests. We could cluster them, but we would not be able to verify the resulting clusterings. Therefore, we build graphs which contain no unknowns at first, and simulate unresponsive nodes in order to get graphs which also contain unknowns.

For our tests we needed a large number of maps, each of which is a tiny part of the internet map represented in the DIMES DB data. There are different ways to slice a random piece of a graph, and perhaps other methods could be examined in future work. We did the following: first we randomly choose a node (an IP) that met the criteria we defined for a simulated unknown root. We added the entire ball of a certain radius around this node to the map. The number of unknown roots in the graph was also a parameter. If more than one unknown root was required, the remainder were chosen out of the nodes of the graph. We added a ball of the same radius around each new unknown root to the map , so that there would be no difference in the depth of the environment around each unknown root, which could lead to bias. After building a graph with the required number of unknown roots, we split each root into unknowns. This was done by treating each root as an unresponsive router. When there was an unresponsive router in a trace, we placed a node in the gap and named it according to its neighbors in the path of the trace. Doing the same in an existing graph means replacing each unknown root (unresponsive router) by nodes matching each possible pair of incoming/outgoing neighbors. See Fig. 1 for an illustration of how an unknown root is split into unknowns. See Fig. 8 for an example of a network before and after splitting unknown roots.

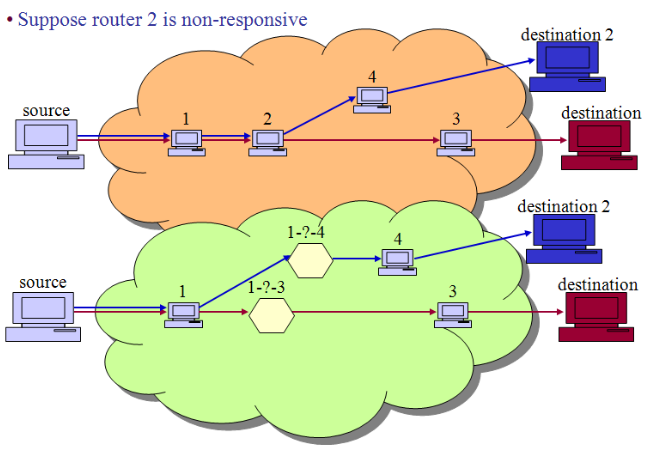

Figure 1. Illustration of how an unresponsive router is split into unknown nodes.

Criteria for potential unknown roots There are several criteria for selecting nodes to be unknown roots, some of which simplify the graphs, while others are there to ensure there is enough information in the graph.
The first criterion is the distance from other unknown roots. A node will not be chosen to be an unknown root if it shares two or more neighbors with an unknown root. Nodes that share two or more neighbors are usually very close to each other, and it is difficult to distinguish between them even manually. After splitting them into unknowns, these nodes will have at least one common unknown node. This makes the task of cleanly separating the unknowns impossible.
The second criterion is the degree of the node. In order to have enough information for the clustering, each unknown root must be split into enough unknowns. As described above, the number of unknowns is determined by the number of incoming and outgoing neighbors of the root. Therefore we require a minimum degree from potential roots. Likewise, they cannot be leaves, we need to have at least one incoming and one outgoing neighbor. We also limited the maximum degree of potential roots. This was done as a

convenience, and is not a real limitation. In fact, it is very likely that roots with a higher degree would have been unified even more successfully.

Preprocessing The graph is represented as a distance matrix. The distances are taken to be symmetrical, even though in Internet graphs this is often not the case. However, this is a requirement for the embedding and clustering processes, and empirically does not affect the results. Thus the distance between two nodes is taken to be the minimum known distance between them, as described in equation (1).

We tested two methods of embedding: embedding all nodes of the graph, and embedding only the known nodes. When embedding all nodes, we used the estimated value of the unlabeled edges (edges adjacent to unknown nodes), that is calculated as described in equation (3). When embedding only known nodes, we removed from the distance matrix the unlabeled edges. Also, for each unknown node, we added an edge connecting its two neighbors, with a delay corresponding to the sum of our estimation of the two original edges (those that were removed).

Another preprocessing step was developed for negative edges, as the distance metric cannot have a negative value. Negative delays with a large magnitude are classified as errors, and the relevant edges are discarded. If the magnitude was small, we assumed it was a small drift due to noise, and the relevant edges were given the value 0. After this step, the graph can become disconnected. We checked it for isolated nodes[1], and removed any that were present. We also removed self loops, if present, since these are clearly errors in the context of Internet graphs as well.

The last preprocessing step was trimming the graph. That is, we removed all the leaves of the graph, because they do not contribute to the embedding process.

5. Experimental Results

In the previous section we described how we created the graphs which are the input to our algorithm. In this section, we describe the results of different experiments we performed using datasets which contained graphs of different configurations.

5.1. Which Nodes to Embed

We found experimentally that embedding all nodes gave better results than embedding only the known nodes. This coincides with the transductive learning paradigm that unlabeled data can contribute additional information in the training phase. In our case both unlabeled and labeled nodes were used to get a better view of the manifold that contains the internet routers.

A standard non-parametric measure of clustering quality is the Rand index [10]. Several variants of the Rand index were recently suggested for measuring clustering quality. For the sake of completeness, the variant of the Rand matching score that we used is as follows: Let C_1 stand for the true clustering of the unknown nodes and C_2 stand for the clustering in question of those points. Then:

$$\text{Rand Score} := \frac{N_{0,0} + N_{1,1}}{N_{0,0} + N_{1,1} + N_{1,0} + N_{0,1}}$$

where

- $N_{0,0}$ is the number of pairs of points that belong to different clusters in both C_1 and C_2.
- $N_{1,1}$ is the number of pairs of points that belong to the same cluster both in C_1 and C_2.
- $N_{0,1}$ is the number of pairs of points which belong to different clusters in C_1, but belong to the same cluster in C_2.
- $N_{1,0}$ is the number of pairs of points which belong to the same cluster in C_1, but belong to different clusters in C_2.

The following (Fig. 2) are results of a run over 49 graphs with two unknown roots, with $\lambda = 50$ and different dimensions. The figure shows a comparison between the different embedding approaches based on the Rand score. Clearly, embedding also the unknown nodes significantly improves the clustering results. Similar results were achieved using other datasets and different parameter values.

5.2. Parameter Tuning

Most of our experiments were performed with maps from which real unknowns were filtered, and in which unknown roots were simulated. These maps allowed us to examine the effect of different values of the parameters on the results. There are two parameters in the spectral embedding algorithm; namely, the scale parameter λ and the dimensionality of the projected space. We next present a sensitivity analysis. We tested our algorithm over a shuffled set of graphs - graphs with different numbers of unknown roots. The set contained 22 graphs with two unknown roots, 22 with three, and 22 with four. It is quite clear when looking at the results that when using dimensions as high as 13, and actually even with dimension 7, the λ value does not make a big difference, if any and stays in a reasonable range (see Figure 3). We hence used dimension$= 13$ and $\lambda = 100$ as the best parameter values.

[1]An isolated node is a single node that is not connected to the rest of the graph.

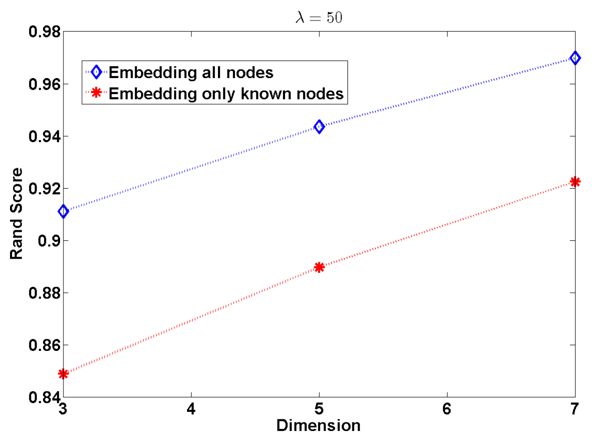

Figure 2. Comparison between the two methods for embedding unknowns in a run over 49 graphs with two unknown roots.

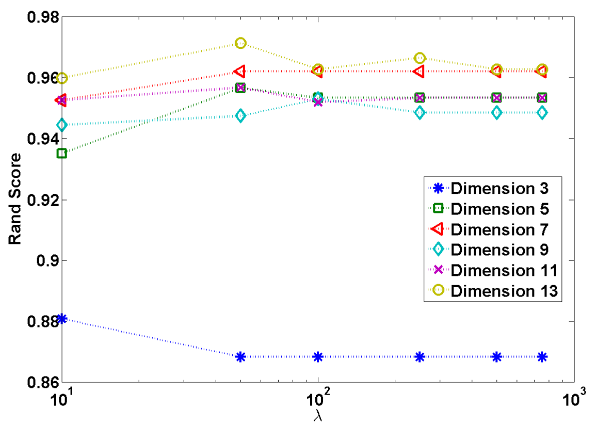

Figure 4. Average Rand score of graphs with two unknown roots per λ and dimension.

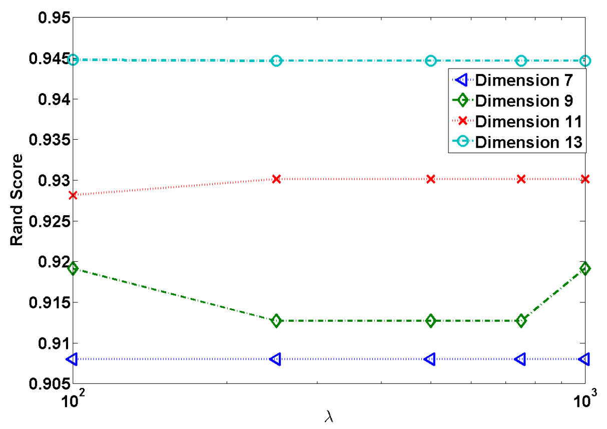

Figure 3. Average Rand score of graphs with 2,3 and 4 unknown roots per λ and dimension.

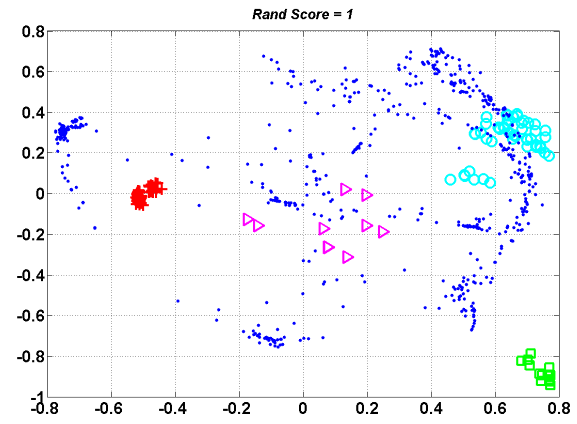

Figure 5. Clustering of a graph with four unknown roots in the projected space. Marker colors signify the real clusters: green, cyan, red, and purple. Marker shapes signify the algorithm clusterings: triangle, circle, plus sign, square. The blue dots are the known nodes in the projected space.

We further tested our algorithm on larger datasets, which yielded similar results. Following are examples of the results of two such large homogenous datasets — the first (Fig. 4) contains 92 graphs with two unknown roots in each, and the second contains 128 graphs with three unknown roots in each (Fig. 7).

Fig. 5 demonstrates the embedding and clustering of nodes in a graph with four unknown roots. The embedding used the best parameter values.

5.3. Finding the number of clusters

We ran the algorithm with the best parameter values over the mixed set of graphs (graphs with different k values). We ran it once with a known k, and once the algorithm selected k automatically. See Fig. 6 for the scatter graphs of the graph scores when k is unknown.

The scores were not very far apart, even in cases where the algorithm made a mistake. It is likely that the reason is that some graphs were not clustered correctly even when the real number of clusters was known, and a better, (though not perfect), clustering is possible when the value of k is not coerced. This may be the case when there are clusters that are not separable in the space with the number of dimensions used. Knowing the value of k, the algorithm may

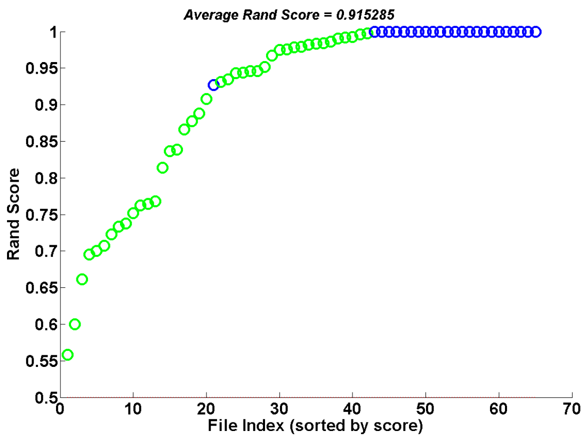

Figure 6. Results of data graphs (with 2,3 and 4 unknown roots) using the best parameter values. The value of k is *unknown*. Green circles are graphs for which the algorithm chose a wrong number of clusters.

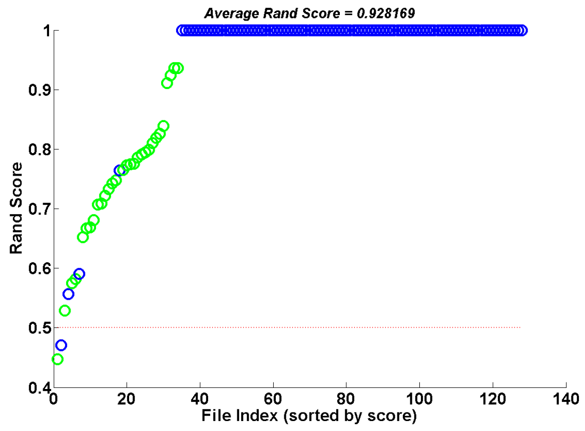

Figure 7. Results of data graphs with three unknown roots using the best parameter values. The value of k is *unknown*. Green circles are graphs for which the algorithm chose a wrong number of clusters.

separate some other set of points that should not necessarily be separated.

5.4. A Complete Example

The figures below show the stages on a single map. The first stage, as seen in Fig. 8, is building a map and choosing unknown roots. After choosing the roots, we split them into unknowns, according to the different paths they are a part of. Following the preprocessing steps, we run our algorithm on the map. We embed all points and cluster them using k-means. The algorithm is capable of choosing k automatically, using the distortion graph seen in Fig. 9. Fig. 10 shows the unknowns in the embedded space, and their clustering.

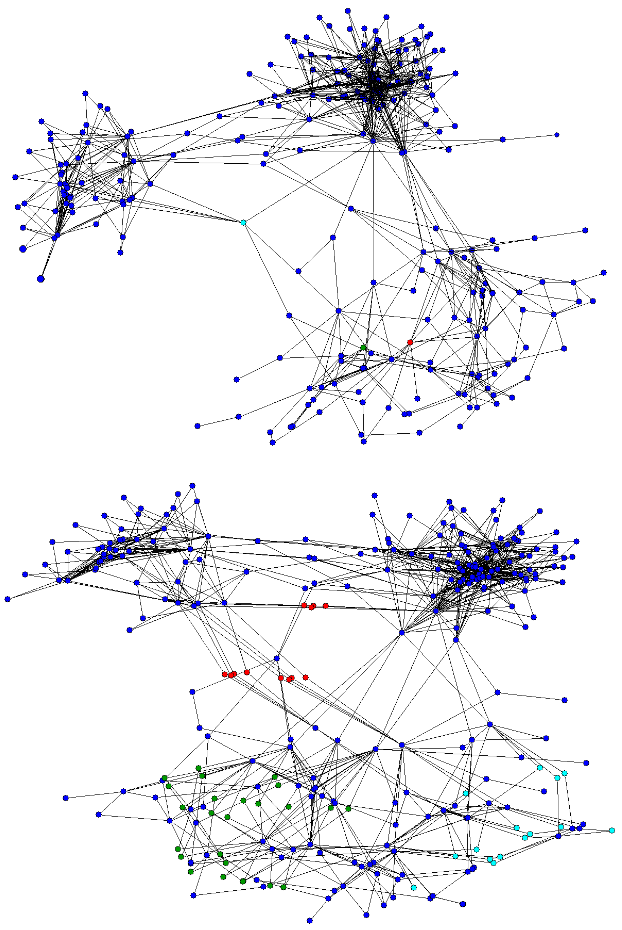

Figure 8. A map with three unknown roots: the red, green, and cyan nodes in the top figure are split into unknowns in the bottom figure. These figures were created using *Pajek*[1],

6. Conclusions

We presented a new method for dealing with unresponsive routers in the IP-level Internet graph. By adapting a spectral clustering algorithm to the unique semi-supervised nature of our task, we achieved surprisingly good results. This approach is an excellent step towards resolution of the real data. Below we highlight a number of extensions for future work.

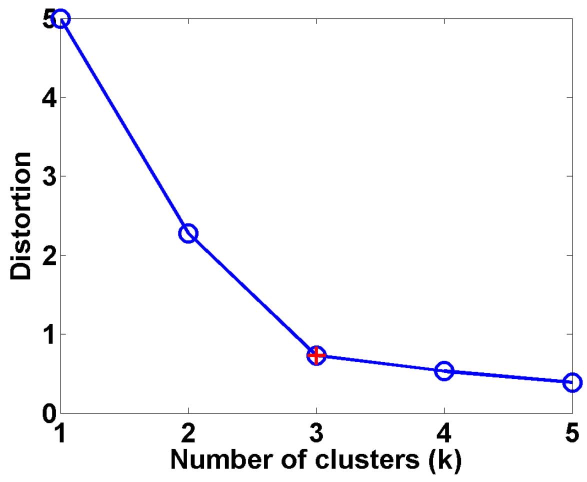

Figure 9. The distortion graph of the unknowns from Fig. 8 when running k-means with different values of k. k was correctly chosen to be three.

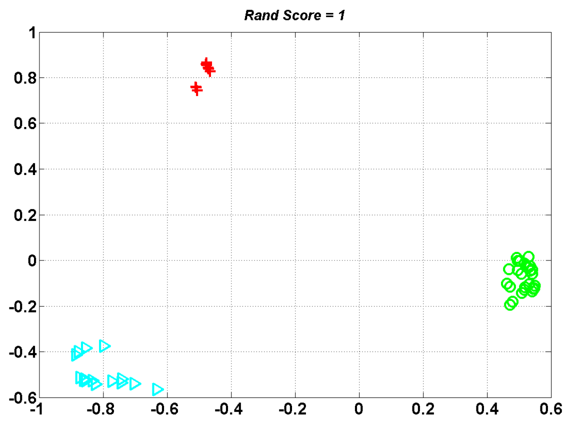

Figure 10. The clustering of the unknowns from Fig. 8 with $k = 3$. Marker colors signify the real clusters: green, cyan, and red. Marker shapes signify the algorithm clusterings: triangle, circle, plus sign.

Running on data with real unknown nodes and unifying them to the unknown roots they resulted from, was the purpose of this work. Obviously, though, not much can be said about the goodness of the solution, when only unlabeled data are used. For this reason, the process of creating labeled maps which contain unknowns, and which represent the real data, is crucial. Several assumptions were made in this process, which should be relaxed in order to reach the general case. On the other hand, this could lead to some difficult problems. For instance, unknown roots with low degrees may not be easy to unify, as they would be represented by a small, perhaps overly small, number of unknowns. The problem of several close unknown roots is also an interesting and challenging one. In many of these cases, it is not even clear to a human evaluator what the correct answer is.

Another open question regarding the generalization of the solution, has to do with the way a section of the graph is cut from the complete Internet topology. The algorithm works on a local environment. In the simulated data, we just crawl from some random node, and after we have a graph we choose more unknown roots from it. This guarantees that all unknown nodes belonging to the roots we are trying to find will be included in our graph, (the roots are included in the original graph, and the unknowns are split from them). In real unlabeled data, there is no way to determine which unknown was derived from which root, so it is not so straightforward to make sure the graph indeed contains all the unknown nodes that need to be unified.

An important question in regard to generalization is the question of finding k. This problem of finding the correct number of clusters is by no means unique - it has been dealt with using various approaches in many different works. We applied a very simple approach, that was relatively successful. However, there is room for improvement of this feature of our algorithm, which could be achieved by considering problem-specific requirements. For instance, we prefer the algorithm to produce more clusters than the true value rather than fewer. Unifying two unknown nodes that did not emerge from the same root may create an arbitrary change in the graph, e.g., may shorten the distance between two nodes significantly. On the other hand, if several instances of the same root node are unified into two nodes instead of into one, we have nevertheless improved the current situation, and proceeded towards a better presentation of the network.

References

[1] Pajek - Program for Large Network Analysis. http://pajek.imfm.si.

[2] A. Broido and K. Claffy. Internet topology: connectivity of IP graphs. In *SPIE International symposium on Convergence of IT and Communication '01*, Denver, CO, USA, Aug. 2001.

[3] CAIDA Macroscopic Topology Project Team. Caida archipelago, next generation active measurement infrastructure.

[4] D. Feldman and Y. Shavitt. An Optimal Median Calculation Algorithm for Estimating Internet Link Delays from Active Measurements. In *IEEE E2EMON*, May 2007.

[5] M. H. Gunes and K. Sarac. Resolving anonymous routers in internet topology measurement studies. In *IEEE INFOCOM*, Apr. 2008.

[6] X. Jin, W.-P. K. Yiu, S.-H. G. Chan, and Y. Wang. Network topology inference based on end-to-end measurements. *IEEE Journal on Selected Areas in Communications*, 24(12):2182–2195, Dec. 2006.

[7] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. E. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *OSDI*, pages 367–380, 2006.

[8] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.

[9] T. Ng and H. Zhang. Predicting internet network distance with coordinates based approaches. In *Infocom*, June 2002.

[10] W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(366):846–850, 1971.

[11] Y. Shavitt and E. Shir. DIMES: Let the internet measure itself. *ACM SIGCOMM Computer Communications Review*, 35(5):71–74, 2005.

[12] Y. Shavitt and T. Tankel. Big-Bang simulation for embedding network distances in Euclidean space. *IEEE/ACM Transactions on Networking*, 12(6):993–1006, Dec. 2004. An earlier version appeared in Infocom 2003.

[13] Y. Shavitt and T. Tankel. On internet embedding in hyperbolic spaces for overlay construction and distance estimation. *IEEE/ACM Transactions on Networking*, 16(1), Feb. 2008. An earlier version appeared in Infocom 2004.

[14] B. Yao, R. Viswanathan, F. Chang, and D. Waddington. Topology Inference in the Presence of Anonymous Routers. In *IEEE INFOCOM*, Mar. 2003.