Partitioning the Internet

Matthias Wachs
TU München
wachs@in.tum.de

Christian Grothoff
TU München
grothoff@in.tum.de

Ramakrishna Thurimella
University of Denver
ramki@cs.du.edu

Abstract

This paper presents experimental results for calculating both node- and edge separators on Autonomous System graphs generated from BGP routing information. The separator of a network graph describes a range of interesting properties as it captures components that are critical to overall connectivity. These components play special roles in terms of routing and deserve special attention from those in-charge of network security and resilience.

We present empirical evidence showing that the Autonomous System Graph (AS Graph) is hard to separate and large portions always remain connected even in the case of a significant number of concurrent Byzantine failures of Autonomous Systems or connections between Autonomous Systems.

1. Introduction

Internet connectivity can be surprisingly brittle with respect to failures at physical locations. There are several examples where failures at a single physical location have had a significant impact on connectivity for a whole region or country [8, 15, 18].

This work is about algorithms to help answer the question of how resilient the Internet is against Byzantine failure of a small set of providers (nodes) or peering links between providers (edges).

To answer this question we analyzed the topology of the Internet's backbone by generating graphs from Internet measurement data and used a new heuristic to find the smallest possible set of networks or connections to be removed to split the Internet. This paper presents empirical evidence that the Internet's backbone infrastructure is surprisingly well connected. The modern Internet consists of Autonomous Systems (ASes) which are linked together using BGP. A high-level approximation for the Internet topology is thus the AS Graph, which models the peering relationships between ASes.

We use a graph-theoretic approach by finding sets of nodes (networks) and edges (connections), the so called separators, to be removed from the network graph. A *separator* is a set of edges or nodes that partitions a graph into sizable connected components. We will use the term separator to mean both edge separator and node separator; it will be clear from the context which type of separator is used.

The primary goal of this work is to characterize the size of the separator that would have to be removed to fragment the AS graph and thereby the Internet. Determining separators for network topologies is useful in various respects, in particular:

- Elements of the separator represent critical infrastructure since the removal of them would fragment the network and significantly reduce its value. Networks with large separators have higher resilience.

- Elements of the separator are particularly useful for the placement of traffic monitoring (or even manipulation) equipment since a large fraction of long-distance communications must cross those links.

Any separator has to balance two goals: both the size of the separator as well as the size of the largest resulting component should be small. We will refer to the size of the largest remaining connected component in relation to the overall size of the graph as χ. For this work, we will assume that a value for χ is given and that the goal is to find a small χ-separator.

Using the AS graph creates the problem of obtaining a sufficiently accurate approximation of the actual graph. We address this problem by using two methods to obtain approximations of the AS graph. We then demonstrate that the differences between these two approximations are small (in terms of the size of the separator in relation to the size of the AS graph). We also consider the problem that not all ASes are of equal relevance. By mapping page access statistics from Alexa to ASes, we obtain a weighted AS graph which we then try to separate as well.

2. Background and Related Work

Given a graph with n nodes of total weight W, a χ-separator is a subset of nodes or edges whose deletion partitions a connected graph into connected components where the largest component has no more than χW nodes, for some fixed $1/2 \leq \chi < 1$. In other words, χ determines how *balanced* a separator is. In a connected graph of n nodes and m edges, one can exhaustively delete every separator of size k, examine the connected components for balance (e.g. size of the largest connected component as a fraction of the original graph). This gives us an $O((m+n)\binom{n}{k})$ time algorithm.

For fixed k, this is a polynomial-time algorithm; otherwise, this algorithm has $\Omega(n^k)$ complexity. It should be noted that even for constant values of $k > 3$, this brute-force algorithm is impractical. Furthermore, if there is a polynomial time algorithm for finding a χ separator of size k, then that algorithm can be used to find a k-clique in a given graph [14]. In light of this, one can consider approximation algorithms that relax the problem constraints: increase the size of the separator for a given χ, or tolerate a slightly less balanced separator in favor of a smaller separator.

Historically, Kernighan and Lin recognized the importance of the graph partitioning problem and its various applications as far back as in 1970 [12]. In their landmark paper, they also proposed a heuristic which has served as a benchmark for other graph partitioning heuristics ever since.

Another common class of partitioning heuristics are derived from the technique of graph coarsening: the graph to be partitioned is made smaller by collapsing nodes and edges, the resulting smaller graph is partitioned, and the separator in this smaller graph is extended to the original graph during the uncoarsening step [4, 9]. Though it was apparent that these early works were promising, it remained unclear whether these heuristics would consistently produce good quality separators for graphs arising in a wide range of application domains until Karypis and Kumar [11] explored the partitioning problem systematically for various versions of the problem. They proposed refinements to the coarsening heuristic and implemented them. Their implementation, which they distribute under the name Metis, is available for free. A remarkable feature of Metis is its fast execution time and its applicability to graphs arising out of varied domains. We compare our results against those produced by Metis.

Kleinberg [13] defined (ε,k)-failures and proposed an algorithm to monitor the connectivity of a network against such failures. These failures are events in which an adversary deletes up to k nodes or edges, after which there are two sets of nodes A and B, each at least an ε fraction of the network, that are disconnected from one another. A set of nodes is an (ε,k)-detection set D if, for any (ε,k)-failure of the network, some two nodes in D are no longer able to communicate. Finding detection sets, though appears superficially related to the problem we consider in this paper, is simpler than finding good separators.

Other related works include finding approximate separators by Feige and Mahdian [7], polynomial time approximation schemes for finding most balanced minimum separators that separate two designated vertices s and t [3], and finding weak points in networks [2] using semidefinite programming. The last paper is a study on the impact on node and edge failures on connectivity that is quite similar to the one presented in this paper. The main difference is that they start with a network where nodes are categorized into clients and servers and where any server is able to provide the required service. They present algorithms to calculate lower and upper bounds on the minimum connectivity (after failures). The results presented in [2] work for graphs that are typically orders of magnitude smaller than those presented in this paper.

3. Calculating Separators

This section describes the heuristics we used to find node and edge separators in the network graphs. We initially attempted to find separators using Metis [11]; however, Metis did not produce good cuts for larger values of χ and for weighted graphs the resulting separators were quite large. We believe this is mainly because Metis is not optimized to create χ-separators with $\chi > \frac{3}{4}$. The separators calculated using our heuristics are sometimes significantly smaller than those computed by Metis, justifying the development of our own heuristics.

3.1. Finding Edge Separators

We found that minor variations of the well-known Kernighan-Lin heuristic for finding edge separators [12] give us the best results for the network graphs considered in this paper. Define $S(A,B)$, the separator for a given vertex partition (A,B), to be the set of edges which have one end point in A and the other in B. We then intend to minimize the size of the separator, $|S(A,B)|$. The KL heuristic tries to achieve this by starting with an arbitrary partition and applying successive vertex swaps until no amount of swapping helps, i.e. when the heuristic reaches a local minimum. A feasible starting solution is created using a breadth-first traversal from a random starting node. To describe the algorithm, we need the following definition. When a pair

of vertices $a \in A, b \in B$ is swapped, we say it results in $gain(a,b) = |S(A,B)| - |S(A \cup \{b\} - \{a\}, B \cup \{a\} - \{b\})|$. Note that the gain is negative if swapping a with b increases the size of the separator.

When we are seeking to partition the input graph into *unequal* parts (i.e. $\chi > \frac{1}{2}$), we employ the trick suggested by Kernighan-Lin [12] of adding an appropriate number isolated "dummy" nodes to the graph.

3.2. Edge Separators for Weighted Graphs

Let us now consider graphs with node weights where weights model the relative importance of different nodes. Clearly for this version of the problem, dummy nodes cannot be used effectively. Let W_A represent $\Sigma_{v \in A} w_v$ where w_v is the weight of vertex v. For brevity the total weight of the graph W_V is denoted simply as W. We will refer to $\chi \cdot W$ as the *threshold*. A χ partition, $\chi > \frac{1}{2}$, of the node set V is a partition into two sets A and B where both W_A and W_B are below the threshold. Note that a χ partition always exists as long as the weight of every individual node is under the threshold.

Algorithm: Weighted Edge Separator

Input: $G(V,E)$ with node weights, a χ partition A and B.

Output: Updated χ partition A, B with potentially fewer edges between A and B

```
1  Coarsen G by merging u with its neighbor v if
   degree(u) is 1 and w_u + w_v < t;
2  Unmark all nodes;
3  max_gain ← 0 ;
4  cumulative_gain ← 0;
5  while unmarked nodes exist in A and B do
6      g_A ← gain from best feasible move from A ;
7      g_B ← gain from best feasible move from B ;
8      g_AB ← gain from best feasible swap ;
       // only unmarked nodes are
          feasible
9      g ← max(g_A, g_B, g_AB) ;
10     cumulative_gain ← cumulative_gain + g;
11     switch g do
12         case g_A
13             perform best feasible move from A ;
14             mark node;
15         endsw
16         case g_B
17             perform best feasible move from B ;
18             mark node;
19         endsw
20         case g_AB
21             perform best feasible swap ;
22             mark nodes;
23         endsw
24     endsw
25     if cumulative_gain > max_gain then
26         max_gain ← cumulative_gain ;
27         checkpoint steps taken so far ;
28     end
29 end
30 undo the steps taken after the last checkpoint ;
31 return A, B;
```

For weighted graphs, we make three significant changes to Kernighan-Lin. First, we use a simplistic variant of graph partitioning [11] where we iteratively merge all nodes of degree one with their neighbor as long as the weight of the resulting node is under the threshold. Especially for some of our larger and sparser graphs, this results in a significant reduction in the problem size without impacting the quality of the solution.

Second, in order to deal with weighted nodes, we introduce the notion of a *feasible* swap; a swap is feasible if after the swap both sides are below the threshold. Only feasible swaps are considered in our innermost loop of Kernighan-Lin.

Our third change is that in addition to swaps, we consider (feasible) *moves*, where a single node is moved from one side to the other side (and both sides are below the threshold after the swap).

The algorithm is summarized in Algorithm Weighted Edge Separator. The initial χ partition to our algorithm is found as in the unweighted case. One key implementation technique that we employed is ordering the nodes in the decreasing order of gain and searching on this ordered lists. This gives us the ability to *abort* the search the moment we discover that the gain at the current indices is less than what we already know is possible from the previous considerations. From this point on, none of the remaining possibilities will offer a better gain. Since searching takes place repeatedly in the innermost loop of the heuristic, this simple observation results in significant savings in the run time of the algorithm.

3.3. Finding Node separators

Next we consider *node* separators. The discussion given in this section applies to both weighted and unweighted graphs. As in the previous section, for weighted graphs we only consider *feasible* moves or swaps.

A χ-node separator C is a subset of the node set V such that the weight of every connected component in the subgraph induced by $V - C$ is under the threshold. The size of the separator is $|C|$ and it is this size we seek to minimize. It is important to note that we only con-

Algorithm: Weighted Node Separator

Input: $G(V,E)$ with node weights, a χ node partition C derived from a good quality edge separator.

Output: Updated χ partition A,B with potentially fewer edges between A and B

```
1  current_best ← cost of C;
2  for h_max ← 2 to 30 do
3      for heat ← h_max down to 1 do
4          c_a ← Best node separator cost from
5                  20 iterations of Push(C, heat);
6          c_b ← Best node separator cost from
7                  20 iterations of Pull(C, heat);
           // Pull is the same as
              Push, only with A and B
              exchanged.
8          if current_best ≥ min(c_a, c_b) then
9              current_best ← min(c_a, c_b);
10             C ← node separator corresponding
11                 to min(c_a, c_b);
12         end
13     end
14     if no change in C in the last 5 iterations then
15         return C;
16     end
17 end
18 return C;
```

sider the cardinality of C and not the sum of the weights of the nodes in C. Note that if we have edge separator of size k, then we can easily construct a node separator of size *at most* k by picking, arbitrarily, either u or v for each edge (u,v) from the edge separator.

Once we have an initial node separator, we employ simulated annealing to improve the quality of the node separator (Algorithm Weighted Node Separator). For a node u in the cut C, let $N_A(u)$ and $N_B(u)$ represent the neighbors of u in A and B respectively. In each iteration, we consider two operations for a vertex u from the cut: push u and pull u. Push u removes u out C and places it in A. In addition, this operation moves $N_B(u)$ from B to C in order to maintain the invariant that C is a node cut. Pull u, on the other hand, removes u out C and places it in B. Also $N_A(u)$ is moved from A to C in order to maintain the invariant that C is a node cut. Clearly, push (resp. pull) u reduces the size C by one if $N_B(u) = \emptyset$ (resp. $N_A(u) = \emptyset$). If $N_B(u) \neq \emptyset$, a push u operation increases the node cut size by $N_B(u) - 1$. The cost of a move, either a pull or a push, is the net change in the number of nodes of C. As before, these moves are performed only if they are feasible, i.e. after the move both A and B are below the threshold. With simulated

Algorithm: Push(C, heat)

Input: Weighted graph $G = (V,E)$, node separator C, two partitions A and B of $V - C$, and a positive integer *heat*

Output: Updated A,B,C after possibly moving some nodes u from C to A and some neighbors u from B to C.

```
1  current_cost ← cost of C;
2  foreach u ∈ C do
3      Let N_B denote {v | v is a neighbor of u in B};
4      if weight of (A ∪ {u}) ≤ χ · W then
5          new_cost ← cost of (C ∪ N_B − {u}) ;
6          δ ← new_cost − current_cost ;
7          r ← random number between 0 and δ;
           // r can be negative if
              δ < 0
8          if r ≤ heat then
9              move u from C to A ;
10             move N_B from B to C ;
11             current_cost ← new_cost ;
12         end
13     end
14 end
```

annealing, lower-cost moves are performed with higher probability, with the general chance of success being determined by the current amount of heat. The problem with this approach is that if we were to use simulated annealing in the usual fashion — slowly cooling down from 'infinite' heat, the initial moves would destroy the good properties possessed by our initial node cut which was derived from a good edge cut.

4. Graph Generation

For our analysis, we generated graphs representing the Internet from actual measurement data. All measurement data we used represent the state of the Internet in late 2010 or early 2011. Section 4.1 describes how we used BGP routing information gathered by the Route Views project [1] to generate a graph representing peering relationships between Autonomous Systems (ASes). Naturally, the resulting graph is only a rough approximation of the actual relationships since the data provide only a local view of the routing topology and do not model policy restrictions that may be present. In Section 4.2 we try to compensate for this by combining the BGP generated AS graph with an AS graph based on IP-level forward-path measurement topology information provided by CAIDA [2].

Finally, we wanted to differentiate between net-

[1] http://www.routeviews.org/

[2] http://www.caida.org/

Table 1: Characterization of the AS graphs generated using Route View's BGP snapshot from December 30th 2010.

Monitor Name	LINX	SYDNEY	WIDE
Prefixes	8,414,813	1,543,223	680,980
Nodes	35,872	36,543	36,315
Edges	75,170	67,485	52,885

works that provide "important" services and networks that are not widely used. For example, AS 56357 is currently an AS for research consisting only of a single router — clearly this network is hardly significant to the Internet as a whole. Section 4.4 describes how we used HTTP access statistics as one possible method for evaluating network relevance and assigning weights to the nodes in the graphs.

4.1. Construction of AS Graphs from BGP Routing Information

We build a first set of AS graphs using the AS routing information collected by the Route Views project. The Route Views project collects BGP information by operating ten BGP routers in different locations. The router's routing tables and the received BGP updates are written to disk and made publicly available on a daily basis.

We use the path information from BGP for our graph generation by adding occurring ASes to our graph as nodes. For consecutive ASes in the AS Path we add an edge to the graph.

For this work we initially selected three different BGP datasets from three different BGP routers to investigate which impact the number of announced IP prefixes in the routing tables on the resulting graph has: The LINX dataset (London, UK) contained over 8 million announced prefixes, the average size SYDNEY dataset (Sydney, Australia) contains 1.5 million prefixes and the very small WIDE dataset only about 680,000 prefixes. The resulting graphs have almost the same number of nodes with around 36,000 nodes. The number of edges differs, depending if the router is located in a well-connected area of the core Internet or if it is located in a peripheral area. When we merge the resulting graphs, we obtain a graph with 36,697 nodes and 79,464 edges. Based on this modest increase in the graph size from merging multiple points of view, we expect that using a few more vantage points would not add a significant number of additional nodes or edges. Still the resulting AS graph is incomplete; alternative methods, such as the one discussed in the next section, can be used to discover some of the missing edges, such as those that typically are not widely advertised via BGP.

Table 2: Characterization of the AS graphs generated using CAIDA's Routed AS Links dataset

Dataset	Cycle 1249	Cycle 1248	Cycle 1250
AS Sets	63	54	63
MOAS	1,456	1,455	1,263
Nodes	19,376	19,416	19,343
Edges	43,654	44,371	42,273

Table 1 provides some further statistics on the resulting AS graphs.

4.2. Construction of AS Graphs from Traceroutes

In addition to the graphs from section 4.1, we generated AS graphs based on IP-level forward path measurements conducted by CAIDA. CAIDA performs regular path measurements to every routed /24 IP network. Based on this measurements CAIDA provides the "IPv4 Routed /24 AS Links Dataset" [10] where they mapped the contained IPs to ASes using Route Views BGP data and extracted the connections between the ASes. This dataset contains pairs of connected ASes and additional information about the processing process. In particular, the data distinguishes between direct and indirect links. Links are marked as indirect if one or more hops on the IP-path cannot be mapped to an AS.

Table 2 characterizes the AS graphs generated using the "IPv4 Routed /24 AS Links Dataset". Combining the AS graphs from the three datasets results in an AS graph with 22,271 nodes and 57,867 edges.

4.3. Merge of BGP and Traceroute graphs

To obtain a best possible graph representation of the Internet, we merged all the graphs mentioned before into one big graph. So we can prevent the drawbacks from both graphs and get a graph with a non-local, non-directed view on the Internet.

We merged the BGP-based graphs and the CAIDA AS links based graphs in an resulting graph containing 36,715 nodes and 99,852 edges.

4.4. Weight Generation

We also wanted to be able to assign a weight to each AS reflecting its importance, where importance is based on the popularity of the service the AS provides to end-users. Ideally, the importance rating would reflect the impact of separating the AS from the rest of the network. As a simple approximation of actual im-

portance, we used the popularity of hosted websites as an indicator for the importance of networks. As a basis for the popularity of websites we used a ranked list of websites provided by Alexa[3]. To use this list to assign weights based on the total number of page views to ASes, we had to link DNS names to AS numbers.

First, we mapped DNS names to IP addresses, which can then be mapped to its AS. A simple DNS lookup up for each hostname is not sufficient as this would not consider DNS caching, DNS-based load balancing and ultimately only provides a local view of name resolution from the view of our system. Instead, we obtained a global view of DNS resolution using 89 nodes on PlanetLab from all over the world. Each of the nodes was used to perform DNS lookups for the complete list of the top 100,000 websites. To bypass web server load balancing and DNS caching, three sequential lookups of the whole list were performed.

The resulting dataset consists of 27,801,818 IP addresses; 18,066 IP addresses from invalid private IP blocks were filtered and 1,269,239 lookups failed. After filtering invalid and duplicate addresses, 159,247 unique IP addresses were found. These IP addresses were then mapped to AS numbers using CAIDA's "Routeviews Prefix to AS mappings Dataset", a dataset available from CAIDA providing an IP prefix to AS number mapping based on BGP dumps.

5. Experimental Results

We used the heuristics presented in Section 3 to calculate separators for $\chi \in [\frac{1}{2}, 1)$. Since the heuristics are randomized, running the heuristic several times typically yields significantly different results. The plots in this section all show the results for five runs (not averaged, each run is represented by a single dot) where the result for each run is the best result obtained during 10 iterations of the original heuristic. On an Intel i7 920, our Java implementation used about 1 GB of memory and took typically around a minute to execute a single iteration of either heuristic.

We provide all experimental results for the AS graph generated by combining the data from BGP routing tables (Section 4.1) with the traceroute graph. The combined graph has 36,715 nodes and 99,852 edges.

5.1. Unweighted AS Graphs

First, we determined the size of the separators for the AS graph without weights. A χ-separator in this case simply ensures that the largest remaining connected component would contain at most a fraction of χ of the number of ASes. Figure 1 shows the size of the edge separators found by our heuristic (Section 3.1) in relation to χ. The relationship between the size of the edge separator and χ is linear, suggesting that separating the graph by edge removal requires incrementally cutting of ASes at the corners of the network.

[3] http://www.alexa.com/

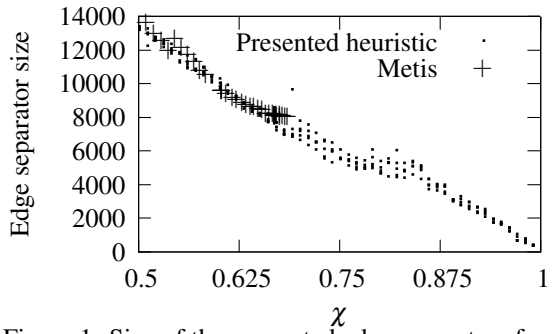

Figure 1: Size of the computed edge separators for different values of χ for the unweighted AS graph.

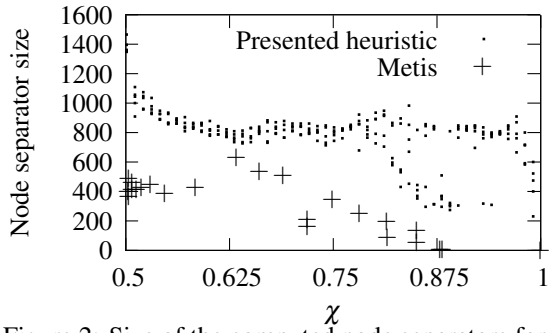

Figure 2: Size of the computed node separators for different values of χ for the unweighted AS graph.

When compared to Metis, the main difference is that our heuristic is able to produce good separators for larger values of χ.

Figure 2 shows the size of the node separators in relation to χ. As expected, the size of the node separators is significantly smaller. What is surprising is that the heuristic does not seem to always work consistently well for values of χ close to 1; after all, the size of any separator should be monotonically decreasing as χ increases.

For unweighed node separators, Metis produces significantly better results compared to the heuristic presented earlier. However, the results are somewhat less predictable as well.

5.2. Weighted AS Graphs

Figure 3 shows the size of the edge separator in relation to χ for weighted graphs. As expected, it is monotonically decreasing. However, the sharp drop from over 15,000 edges to around 500 edges at only 54% is quite surprising. The reasons for this are likely

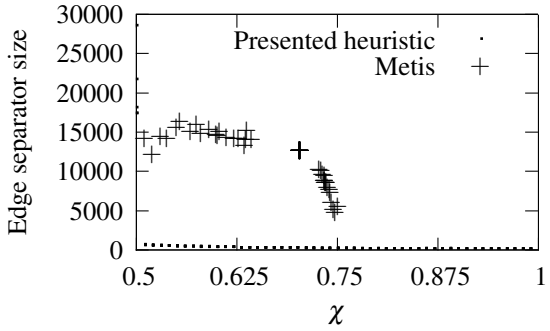

Figure 3: Size of the computed edge separators for different values of χ for the weighted AS graph.

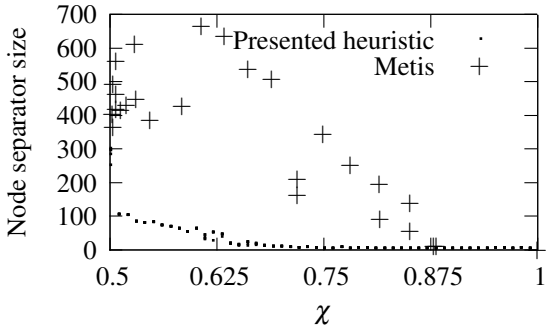

Figure 4: Size of the computed node separators for different values of χ for the weighted AS graph.

two-fold. First, achieving near-perfect balance for the weighted graph is harder because it essentially requires solving two knapsack problems, leaving little room for minimizing the size of the separator. Second, the extreme weight distribution of the ASes permits rather small separators (compared to the unweighted graph) because isolating a few ASes can have a huge impact on the size of the largest connected component.

For node separators, this second effect is even stronger. Figure 4 shows that removing a handful of ASes could be in theory quite devastating for reachability. These ASes are typically a combination of high-traffic ASes (such as Google) and ASes that are key for connectivity (such as Level 3). The first effect is less strong here, because the ASes that are part of the node separator are not part of any connected component, making it much easier to achieve a separation where the remaining connected components have less than χ weight. As a result, for node separators, χ close to $\frac{1}{2}$ is not as restrictive.

For the weighted AS graphs, the size of the edge separator is again significantly larger for the AS graph combining BGP and traceroute data (Figure 3). What is surprising here is that the size of the edge separator typically increases by 100%, not just by the 37–64% range one might predict from the increase in the AS graph size. We suspect that this is because the ASes present in the BGP graph but absent in the traceroute data typically have the minimum weight, making their separation insignificant. So the algorithm is forced to separate the same high-weight ASes, just this time with a significant number of additional edges.

Compared to Metis, the edge separators for the weighted graphs are significantly smaller using our heuristic. Metis seems to be unable to take advantage of the significant weight differences between the nodes. Again, Metis also does not produce separators for values of $\chi > \frac{3}{4}$.

For node separators, this effect is less pronounced (Figure 4). This is likely because targeting ASes with high weights or high degree is equally effective in both cases. As a result, the increase in the size of the node separator for the combined AS graph is typically modest, with the exception of χ values close to 0.5.

For node separators on the weighted graph, Metis performs again significantly worse than the presented heuristic; furthermore, Metis's results are widely dispersed and lack a clear trend towards smaller separators for increasing values of χ.

6. Discussion

This research followed the common approach of combining traceroute and BGP data to obtain a reasonable approximation of the AS graph. Similar to existing studies that attempt to characterize AS graphs [16], the resulting AS graphs are not complete [5, 6, 17]. The AS graphs could theoretically be augmented to be more accurate, for example using additional or simply more accurate measurements such as those described in [1, 19]. In an AS graph critical infrastructure is often not represented according to her significance for connectivity. Internet Exchange Points (IXPs) might be treated as sets of highly-connected nodes whereas a single underseacable might be represented as multiple logical edges which are much harder to separate. So applying our heuristic to a physical map of the Internet could bring interesting insights how to disconnect a certain region and partition the Internet.

Based on the observed differences in terms of separator sizes between the experimental results using only traceroute data and the results for the combined graph, we conclude that our heuristic for determining separators would remain a good method for calculating separators, and we suspect that the changes in the size of the separators would not be too dramatic.

We believe that the calculated separators are applicable to the various applications stated in the introduction:

- The elements of the separators do represent criti-

cal infrastructure; while additional edges in the AS graph may make them less critical and a smaller separator would point to even more critical systems, the separators do point to organizations and relationships that are key to global connectivity. The weighted AS graphs are useful in this application domain.

- The locations calculated would be useful for the placement of traffic monitoring equipment; additional edges might allow some traffic to escapes surveillance (but many applications, such as early warning systems, do not require completeness). Even smaller separators may reduce the cost; nevertheless, the heuristic represents an advance over existing algorithms.

7. Conclusion

This work characterized small edge and node separators in AS graphs and provided extensive data on the size of χ-separators for various values of χ. Analyzing the AS graphs, we found that allowing only a slight imbalance between the resulting connected components can significantly reduce the size of the separator. However, it would take the failure of 20–30 ASes to make make about 40% of HTTP accesses fail for the AS graphs used in this study (Figure 4). Furthermore, this number represents only a lower bound, as the observable AS graph is naturally a subset of the actual AS graph. As Byzantine failure of more than 20–30 ASes seems unlikely, separating the Internet on the AS-level is thus unlikely to work; future work in this area will thus have to include additional information about BGP routing policies or geographical knowledge about physical connections to detect critical infrastructure.

References

[1] Brice Augustin, Balachander Krishnamurthy, and Walter Willinger. Ixps: mapped? In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 336–349, New York, NY, USA, 2009. ACM.

[2] George Dean Bissias, Brian Neil Levine, and Ramesh K. Sitaraman. Assessing the vulnerability of replicated network services. In *Proceedings of the 6th International COnference*, Co-NEXT '10, pages 24:1–24:12, New York, NY, USA, 2010. ACM.

[3] Paul S. Bonsma. Most balanced minimum cuts. *Discrete Applied Mathematics*, 158(4):261–276, 2010.

[4] Thang Nguyen Bui and Curt Jones. A heuristic for reducing fill-in in sparse matrix factorization. In *PPSC*, pages 445–452, 1993.

[5] Qian Chen, Hyunseok Chang, Ramesh Govindan, Sugih Jamin, Scott Shenker, and Walter Willinger. The origin of power-laws in internet topologies revisited. In *INFOCOM*, 2002.

[6] Rami Cohen and Danny Raz. The internet dark matter - on the missing links in the as connectivity map. In *INFOCOM*, 2006.

[7] Uriel Feige and Mohammad Mahdian. Finding small balanced separators. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pages 375–384, New York, NY, USA, 2006. ACM.

[8] M. Hachman. Sabotage Suspected in Silicon Valley Cable Cut. `http://www.pcmag.com/article/print/239065`, April 2009.

[9] Bruce Hendrickson and Robert W. Leland. A multi-level algorithm for partitioning graphs. In *SC*, 1995.

[10] Young Hyun, Bradley Huffaker, Dan Andersen, Emile Aben, Matthew Luckie, kc claffy, and Colleen Shannon. The ipv4 routed /24 as links dataset - 12/30/2010,12/29/2010. `http://www.caida.org/data/active/ipv4_routed_topology_aslinks_dataset.xml`.

[11] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20:359–392, December 1998.

[12] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell system technical journal*, 49(1):291–307, 1970.

[13] Jon Kleinberg, Mark Sandler, and Aleksandrs Slivkins. Network failure detection and graph connectivity. *SIAM J. Comput.*, 38:1330–1346, August 2008.

[14] Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351:394–406, February 2006.

[15] Brid-Aine Parnell. Epic net outage in Africa as FOUR undersea cables chopped: Ship blunders allegedly to blame. `http://www.theregister.co.uk/2012/02/28/east_africa_undersea_cables_cut/`, February 2012.

[16] Amir H. Rasti, Nazanin Magharei, Reza Rejaie, and Walter Willinger. Eyeball ases: from geography to connectivity. In *Internet Measurement Conference*, pages 192–198, 2010.

[17] Matthew Roughan, Simon Jonathan Tuke, and Olaf Maennel. Bigfoot, sasquatch, the yeti and other missing links: what we don't know about the as graph. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, IMC '08, pages 325–330, New York, NY, USA, 2008. ACM.

[18] R. Sauder. Connecting The Many Undersea Cut Cable Dots. `http://www.cyberspaceorbit.com/ConnectingTheDots.htm`, February 2008.

[19] Fabien Viger, Brice Augustin, Xavier Cuvellier, Clémence Magnien, Matthieu Latapy, Timur Friedman, and Renata Teixeira. Detection, understanding, and prevention of traceroute measurement artifacts. *Comput. Netw.*, 52:998–1018, April 2008.