Revisiting AS-Level Graph Reduction

Erik C. Rye
US Naval Academy
Email: rye@usna.edu

Justin P. Rohrer
Naval Postgraduate School
Email: jprohrer@nps.edu

Robert Beverly
Naval Postgraduate School
Email: rbeverly@nps.edu

***Abstract*—The topological structure of the Internet – the interconnection of routers and autonomous systems (ASes) – is large and complex. Frequently it is necessary to evaluate network protocols and applications on "Internet-like" graphs in order understand their security, resilience, and performance properties. A fundamental obstacle to emulation and simulation is creating realistic Internet-like topologies of reduced order. We reexamine existing AS graph reduction algorithms and find that they struggle to capture graph theoretic properties of modern topologies and topologies obtained from different sources. We develop a new AS graph reduction method that performs well across time periods and data sets.**

I. Introduction

The Internet graph is large and complex, even when represented at a coarse Autonomous System (AS) level [27], [17]. Today, there are approximately ∼40,000 ASes with nuanced policy dictating their interconnection and data forwarding [10]. Research on the macroscopic behavior of the Internet is therefore frequently incomplete due to the practical challenges in obtaining vantage points, ground-truth, and validation. An alternative is to fully describe and model the Internet's AS-level graph. While prior work has e.g., successfully simulated the AS-level Internet topology to better understand complex routing decisions [9], SBGP, and RPKI [8], simulation does not reveal crucial implementation-specific behaviors.

Advances in virtualization and computing power enable an attractive middle ground between real-world experimentation and simulation: *emulation*. For instance, several platforms virtualize commercial hardware and permit actual network router and switch operating systems to be run in emulation [5], [14], [20]. Rather than the time and resource-intensive task of building physical network topologies, emulated testbeds can rapidly create (and re-create) complex networks.

Emulation has been used by operators to test configurations [15], while researchers have employed emulation to investigate route-leak detection, BGP convergence and stable paths [11], network latency [7], and CDN-ISP collaboration [21]. Unfortunately, the largest emulations reported in the literature use approximately 1,000 routers [15], whereas the AS-level topology is an order of magnitude larger.

We re-examine the problem of accurately *reducing* the size of a known Internet AS-level graph, while preserving its graph theoretic properties. In contrast to topology generators, e.g., [18], [26], [13], graph sampling removes edges and vertices, a technique that has been shown to outperform generative approaches [16]. A typical input to a graph reduction algorithm is the complete AS-level topology, for instance as inferred via CAIDA or RouteViews [12], [19]. We make no claims as to the representativeness of these input topologies, and indeed prior work has shown them to be incomplete [4].

Our starting point is the algorithms of Krishnamurthy et al. [16], who focus specifically on AS graph reduction. We re-examine the quality of the reduced graphs from these existing algorithms (evaluated on topologies now over a decade old) on the modern Internet topology. Further, we investigate graph reduction performance on input topologies obtained from active probing (CAIDA's ITDK) [12] versus passive BGP (Routeviews) [19]. On these data, we find important deviations in the fidelity of the reduced graphs resulting from these prior approaches. Based on these findings, we introduce two new graph reduction algorithms specifically designed for Internet topology reduction: KDD and KKD. We show that KDD and KKD better preserve a wider variety of graph properties of interest for AS topologies than existing techniques.

II. Background

Graph generation is well-studied. Constructive models such as Barabási-Albert, Waxman, Generalized Linear Preference, and Interactive Growth ([25], [2], [28], [6]) have been implemented ([18], [13], [26]) to generate topologies that mimic graph properties of the observed Internet. However, prior analysis suggests that *reductive* approaches outperform constructive models [16]. As such, we focus on graph reduction: sampling large networks in order to obtain smaller graphs that retain representative properties of the original graph or match those of a graph of similar number of vertices, or order.

Significant prior work has also examined graph reduction; a more comprehensive discussion is contained in [23]. Cem et al. reduce an AS graph from 2007 using Induced Random Vertex, Metropolized Random Walk and Random Edge sampling [3]. Whereas Cem examines degree distribution, clustering coefficient, and path length of a reduced graph compared to a single input graph, we compare against an unsampled topology of the same order from a previous snapshot in time. In addition, our work examines the graph spectra, hop-plot and sorted degree distributions. Vaquero et al. [24] introduce a Breadth-First Search (BFS) method for reducing the graph of routers within a single backbone service provider and evaluate end-to-end delay between various pairs in the reduced topology to those in real physical networks. By contrast, in this paper, we examine far larger AS-level topologies, and are concerned primarily with accurately representing graph theoretical characteristics within the reduced graphs.

TABLE I
SUMMARY OF GRAPH SAMPLING METHODS EXPLORED IN THIS PAPER.

Reduction Method	***Description***
CRE (Contraction of a Random Edge)	Select a random edge with uniform probability; adjacent nodes are contracted into one.
CRVE (Contraction of a Random Vertex-Edge)	Select a random vertex, then a random incident edge. Adjacent nodes contracted.
DRE (Deletion of a Random Edge)	A random edge is selected with uniform probability, then deleted from the graph. If disconnections occur, keep largest connected subgraph.
DRV (Deletion of a Random Vertex)	A vertex is selected with uniform probability, then deleted from the graph. If disconnections occur, keep largest connected subgraph.
DRVE (Deletion of a Random Vertex-Edge)	A vertex is selected randomly, then an edge incident to that vertex is selected randomly. This edge is then deleted. If disconnections occur, keep largest connected subgraph.
DHYB (Deletion Hybrid)	DHYB-X chooses between deletion of a random vertex/edge (DRVE) with probability X and deletion of a random edge (DRE) with probability (1-X)
EBFS (Exploration by Breadth-First Search)	A vertex in the graph randomly selected as the root node. Breadth-First Search algorithm is run until desired number of nodes visited; these nodes used to induce subgraph of original.
EDFS (Exploration by Depth-First Search)	A vertex in the graph randomly selected as the root node. Depth-First Search algorithm is run until desired number of nodes visited; these nodes used to induce subgraph of original.
KDD (k-core decomposition – DRVE – DRE)	Obtain k-core of the initial graph for largest k value such that the k-core contains more than n vertices. Remove vertices by DRVE from this k-core until n vertices remain, then delete random edges via DRE until m edges are present. See §IV-1.
KKD (k-core decomposition – k-deletion – DRE)	Obtain k-core of the initial graph for largest k value such that the k-core contains more than n vertices. Remove vertices of degree k from this k-core until n vertices remain, then delete random edges via DRE until m edges are present. See §IV-2.

Krishnamurthy performed some of the earliest work on reducing the entire Internet AS graph to a size suitable for emulation [16] rather than sampling networks in general, or sampling a particular portion of the Internet. In §V we re-examine this work by applying the Krishnamurthy's algorithms to modern AS topologies obtained from different sources.

III. METHODOLOGY

Our methodology is four-fold: i) implement the reduction techniques of [16]; ii) obtain topologies from different time periods and sources; iii) re-evaluate the reduction techniques in [16]; and iv) develop improved reduction algorithms.

Krishnamurthy [16] explores the first eight graph reduction methods summarized in in Table I. These can broadly be classified into three different categories: contraction, deletion, and exploration. Although we made contact with an author of [16], we were unable to obtain their reduction implementation. We therefore re-implemented their reduction algorithms and validated against the same input topologies as in the original work. As shown in §V, our results using the same data are similar to their findings, but not identical.

We consider AS-level topology data from two sources over two timeframes (Table II). RV1 and RV2 are snapshots of AS-level graphs constructed from the Border Gateway Protocol (BGP) paths passively observed by Routeviews [19]. RV1 is the identical dataset as used in [16]. CAIDA1 and CAIDA2 also consist of AS-level Internet-wide graphs, but are constructed using active traceroute probes [12]. By examining these varied data, we seek to test whether prior work overfits to particular time periods or data sources. While neither of these topologies is complete or necessarily correct [4], they represent the state-of-the-art publicly available topologies.

We define a *source-period* as a timeframe over which a chronologically later (and, hence, larger) topology is reduced to a smaller graph. Such a reduction permits evaluation of the representativeness of the reduced graph by comparing its properties to the smaller, earlier topology (as opposed to artificially comparing against properties of the input topology). We study four source-periods - RouteViews 2001-1998, Center for Applied Internet Data Analysis (CAIDA) 2001-1998, RouteViews 2014-1998, and CAIDA 2014-1998. We refer to the *initial instance* as the chronologically most recent topology for each source-period (May 2001 or December 2014). We define the *reduction endpoint* as the number of nodes in the earliest topology for each source-period (in our work, these are instances from January 1998).

In order to recreate and validate the work of [16], we examine RV1 and reduce the AS graph from 7 May 2001 at 1800 to a graph on the order of the 24 January 1998 instance. To determine whether the reduction mechanisms remain valid, we reduce an initial instance of RV2 from 1 December 2014 to a graph with order of the snapshot from 1 January 1998 (a span of almost 17 years and a node reduction of 93%).

We note that the ability of the reduction methods to maintain the desired graph theoretic properties is often proportional to the degree of reduction. While we start with an initial graph and reduce to an endpoint graph, the algorithms are iterative and produce intermediate graphs. Thus, we create chronologically intermediate graphs in order to compare our reduction methods to the actual evolution of snapshots observed in our datasets (which mirror the evolution of the Internet). In general, we have selected our intermediate graphs six months apart for the reductions from 2014 to 1998, and in four month intervals for the reductions from 2001 to 1998.

We evaluate a reduced graph against an unsampled snapshot of the same order. For example, if we reduce the RV2 graph from December, 2014 to a new graph with 22,640 nodes (a 51% reduction), we find a historic Routeviews graph in our dataset of the same order 22,640 (in this example 1 June, 2006). For our first metric, we have chosen the sets of intermediate graphs to compare against as described above; for the final three metrics, we compare only against the earliest

TABLE II
DATASETS EXAMINED

Dataset	*Source*	*Construction*	*Timeframe*	*Initial Order*	*Initial Size*	*End Order*	*End Size*	*% Reduction*
RV1	Routeviews	Observed AS_PATH	01/1998 - 05/2001	10,966	22,536	3,291	5,784	72.5
RV2	Routeviews	Observed AS_PATH	01/1998 - 12/2014	49,185	107,517	3,211	5,611	93.5
CAIDA1	CAIDA ITDK	Traceroute	01/1998 - 05/2001	11,045	24,484	3,233	5,773	70.7
CAIDA2	CAIDA ITDK	Traceroute	01/1998 - 12/2014	46,177	177,391	3,233	5,773	93.0

historical Internet instance from January 1998, the reduction endpoint. In this fashion, we always compare the reduced graph to a topological snapshot when it was of the order we are targeting. The graph metrics we use are as follows:

1) Average degree.
2) Spectral analysis of the normalized adjacency matrices. We (as in [16]) plot the 100 largest eigenvalues of these matrices. The magnitude of these eigenvalues provides an approximation of the degree of clustering in the 100 largest clusters within the graph.
3) Hop-plot, or cumulative distribution of vertex-pairs versus distance between the pairs in hops.
4) Sorted degree-distribution.

In order to judge which graph reduction method is most successful in creating a graph matching the properties of earlier topology snapshots, we use the *Mean Absolute Error (MAE)* between the reduction method data point and the corresponding data point of the target topology: $MAE = \frac{1}{n}\sum_{i}|m_i - I_i|$ where n is the total number of points compared, m_i is the reduction method data point, and I_i is the target graph's value. MAE measures the cumulative distance between the data of the target and the data obtained in our sampled graph, without accounting for the direction in which the error was realized (under or over). Further, because we find error values both less than and greater than 1, MAE does not under-emphasize the former or amplify the latter. Our results represent the average of 50 trials with different seeds.

IV. k-CORE REDUCTION

As we will show in §V, the best of the reduction methods, as determined by the fidelity to which they capture the endpoint's metrics, varies with the input AS graph considered. While Krishnamurthy et al. find that DHYB-0.8 performs best, we find that this value of 0.8 for a hybrid ratio overfits to the data – and that the reduction performance significantly depends on choosing a hybrid ratio appropriate for the input graph.

Motivated by the sensitivity of these algorithms to parameterization, we seek improved algorithms producing faithful reductions across our RV1, RV2, CAIDA1, and CAIDA2 (Table II) datasets. To this end, we draw from prior work finding k-cores of AS-level Internet graphs to be self-similar [1] to create new reduction algorithms exploiting this self-similarity.

Let $deg_H(v)$ be the degree of vertex v in subgraph H. For graph G with vertex set $V(G)$, the k-core of G is the maximal connected subgraph H of G such that: $deg_H(v) \geq k\ \forall v \in V(H)$. Thus, a graph k-core is a connected subgraph in which all nodes have degree at least k; if multiple such subgraphs exist, the k-core is the largest subgraph by number of nodes. For G with n nodes, the complexity of finding its k-core is $O(n)$, as it can be obtained by repeatedly deleting nodes of lowest degree (and their incident edges). We create and analyze two k-core-based methods: 1) k-core decomposition – DRVE – DRE (KDD); and 2) k-core decomposition – k-deletion – DRE (KKD).

1) KDD: KDD takes as input an initial AS graph, and a target number of both vertices and edges. Beginning with the input graph, we compute successive k-cores until the $(k+1)^{st}$-core contains fewer than the target number of vertices. We retain the smallest k-core (largest k) with more nodes than the target, and proceed to reduce this k-core using DRVE until reaching the desired vertex count. Last, we remove random edges if they do not partition the graph until we reach the target number of edges. The largest k for our purposes is $k = 2$ in both RV1 and CAIDA1; $k = 5$ for RV2 and $k = 8$ for CAIDA2.

The force behind the development of KDD stems from a desire to capture the most topologically important, highest-degree AS in the Internet graph, while removing less significant nodes based on degree. While degree is an imperfect indicator of graph structure and node importance, the highest-degree nodes most likely take part in the largest clusters within the graph, a desirable property to maintain. Further, high-degree ASes are more likely to have existed in earlier topology snapshots graphs than lower-degree nodes, against which we ultimately compare our reduced graphs.

2) KKD: As with KDD, we compute successive k-cores until the $(k+1)^{st}$-core contains fewer than the target number of vertices. We reduce the resulting k-core by randomly removing vertices with degree of k until reaching the target number of vertices. Last, we remove random edges if they do not partition the graph until we reach the target number of edges.

The motivation behind KKD is to understand the effect of different vertex removal methods after obtaining the smallest k-core. When the smallest k-core has been identified, the initial minimum degree of nodes within the graph is equal to k, though it may grow smaller as vertices of degree k are removed. While in our experiments removing nodes of degree exactly k is adequate to reach our target, this may not be possible in general.

V. RESULTS

A. Reexamination of Prior Work

We reexamine the results of [16] when reducing RV1. Our goal is to both validate these prior findings and our implementation of their algorithms (publicly available [22]). Our results largely mirror those of the prior work, with some minor discrepancies that we summarize here (full results and analysis available in [23].

We first examine average degree as compared to the target (earliest) AS topology instance. The deletion-hybrid (DHYB)-0.7 method produced an endpoint graph with the closest average degree, while [16] found DHYB-0.8 to be best. Although there is only a 0.247 average degree difference between DHYB-0.7 and 0.8, DHYB-0.7 is not mentioned in [16] while they find DHYB-0.5 and 0.6 to be secondary candidates. DHYB methods monotonically increase in average degree, necessarily including DHYB-0.7 among the best performers.

The hop-plot analysis finds a similar discrepancy. We find that DHYB-0.7 best matches the target topology hop-plot, while [16] finds DHYB-0.8 performing best (MAE of 0.00606), with DHYB-0.5 and 0.6 following. We find DHYB-0.7 has an MAE of 0.00261, while the second best method, KDD produces an MAE of 0.00442. KKD produces the most representative sampled graph when considering the degree distribution, with an MAE of 0.469. Of the reduction algorithms studied in [16], DHYB-0.7 again most closely matches the target degree distribution with a MAE of 0.604. Only spectral analysis confirms the exact conclusions of [16]. We find that DHYB-0.8 follows the Internet eigenvalue plot closest with a 0.00483 MAE. KDD is second best with an MAE of 0.0120.

Overall, our results vary slightly from those of the prior work when examining RV1. The differences are likely attributable to exploring more of the algorithm parameter space.

B. Modern Topologies

We next apply these methods to the more modern Routeviews data in RV2. In this source-period, the most recent instance is 1 December 2014 – an inferred AS-level graph of order 49,185 and edge-set cardinality 107,517, for an average degree of 4.37. The reduction endpoint is the order of the initial RV2 graph from 1 January 1998 (3,211 vertices and 5,611 edges, which is a node reduction of approximately 93.5%). We further consider intermediate instances of RV2 evenly distributed on the first of June and first of December.

Figure 1(a) shows average degree as a function of percent reduction for the various methods. The solid black line (labeled "Internet") corresponds to the metric values of a chronologically earlier and smaller instance taken unsampled from the same data source (RV or CAIDA). We thus evaluate the reduced graph against a true historic instance of the same size.

None of the non-DHYB methods approximate the target average degree curve well. DRVE is the best-performing non-DHYB method, but deviates from the average degree of the target after the 30% reduction mark and rapidly increases to the reduction endpoint. The DHYB reduction methods fare better, with DHYB-0.6 matching the target average degree most closely; the mean average degree of the 50 random trials is 3.56, which compares favorably to the 1 January 1998 Internet instance average degree of 3.49. DHYB-0.7 follows the target curve throughout much of the reduction. An inflection point is reached at about 70%, at which point DHYB-0.7 increases in average degree to around 4.4.

The spectral analysis of RV2 in Figure 2(a) reveals DHYB-0.6 performing best (MAE of 0.00707). Our k-core methods also perform well, with KDD obtaining an MAE of 0.00843 and KKD an MAE of 0.0134. Interestingly, DHYB-0.8, observed to perform best in [16], is seventh of the 18 algorithms with an MAE of 0.0762. DHYB-0.6 and KDD follow the target spectra well throughout all eigenvalues, but particularly throughout the lowest ordered (largest clusters). As seen in Figure 2(a) we also observe the tendency of DHYB reduction methods' normalized eigenvalues to monotonically increase according to their probability value.

The hop-plot results for RV2 given in Figure 2(b) produce a wide range of curves. Using contraction of a random edge (CRE), for example, results in a most-reduced graph in which approximately 85% of vertices are within 2 hops of each other. Conversely, in DRE, a comparable fraction of vertices are within 7 hops of each other. Several methods compare favorably to the target hop-plot at various hops; exploration by Depth-First Search (EDFS) most closely follows the target curve (MAE of 0.00366). DHYB-0.7 and 0.6 model the target fairly well with MAE of 0.00520 and 0.0102 respectively. At some hop values, we notice significant deviations from the target. For example, $hop = 3$ for DHYB-0.6 exhibits a difference of $\sim$10% of vertex pairs, indicating that the tail of the CDF is longer compared to target graph.

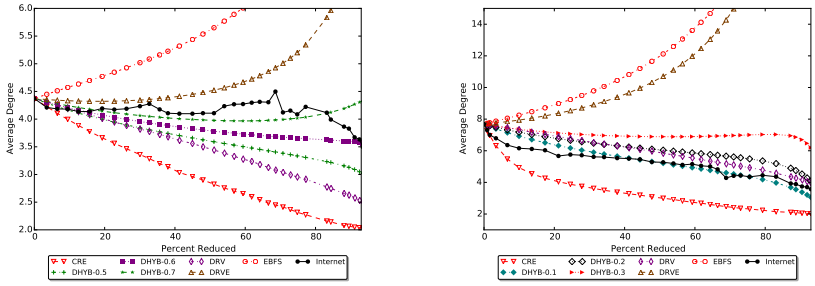
Finally, Figure 2(c) examines the degree distribution of the reduced graphs. CRE follows the target degree distribution poorly in both RV2 and RV1 as it tends to create one high degree vertex. Methods that approximate the highest-degree vertex tend to overestimate the degree throughout the remaining vertices, while the rest of methods follow the target well throughout the middle and low ranked vertices, but fail to approximate the highest-degree vertex by several hundred. In particular, contraction of a random vertex/edge (CRVE) has a mean highest-degree vertex of $\sim$631, compared to 640 for the target instance. However, CRVE overestimates the degree of the remaining vertices. KKD, KDD and DHYB-0.5, are the three best methods with MAE of $0.688, 0.706$, and 0.743.

C. Different Topologies

1) CAIDA 2001-1998: CAIDA1 covers the same source-period as RV1, thereby providing insight into the sensitivity of the results to the source input topology. Encouragingly, we find largely consistent results, with different parameterizations of the DHYB algorithm performing best.

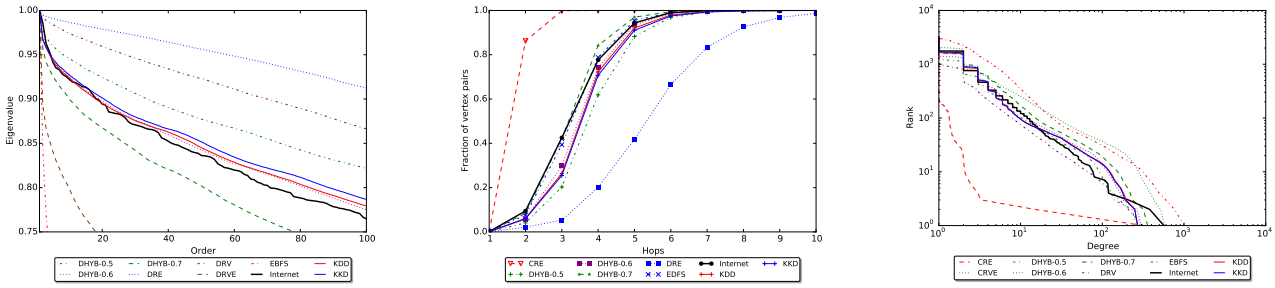
For average degree, we find DHYB-0.6 most closely matching the target graph, as compared to DHYB-0.7 for RV1. Spectral analysis of CAIDA1 reductions show DHYB-0.7 to perform best (MAE of 0.00367) as compared to DHYB-0.8 for RV1. Hop-plot analysis finds KDD and DHYB-0.7 to best fit the target (MAE of 0.00359 and 0.00405); DHYB-0.7 performed best in RV1. Finally, KKD most closely models the CAIDA1 target degree distribution (MAE of 0.486).

2) CAIDA 2014-1998: Figure 1(b) shows that the average degree of the initial instance is approximately eight, or twice that of the comparable Routeviews topology. This degree difference imparts surprising differences in the various algorithm



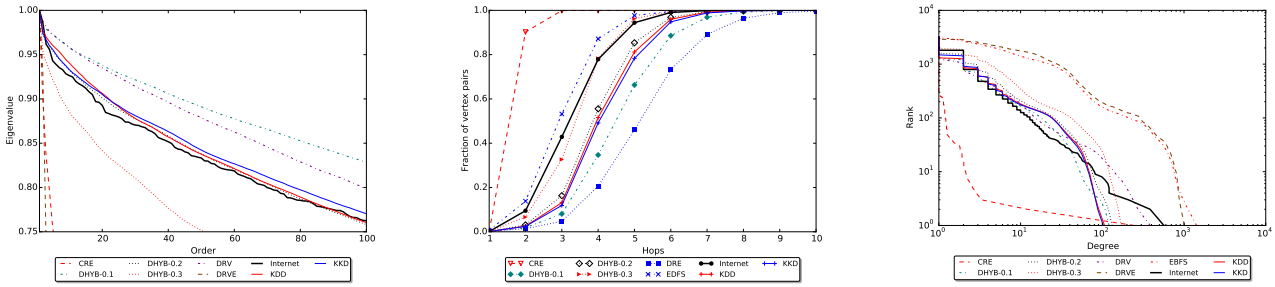
(a) Routeviews 2014-1998 (b) CAIDA 2014-1998

Fig. 1. Average degree



(a) Spectra (b) Hop-plot (c) Degree distribution

Fig. 2. RouteViews 2014-1998



(a) Spectra (b) Hop-plot (c) Degree distribution

Fig. 3. CAIDA 2014-1998

performance. DHYB-0.1 best captures average degree, followed closely by DHYB-0.2 and deletion of a random vertex (DRV). These reduction methods did not perform well with the previous RV1, RV2, and CAIDA1 datasets; the comparable RV2 source-period average degree was best approximated by DHYB-0.6, and no non-DHYB methods fared well. DHYB-0.1 produces an average degree of 0.509 below the earliest historical target, while DHYB-0.2 and DRV exhibit average degrees differences of 0.486 and 0.405 above.

Spectral analysis further highlights the difference among data source. As seen in Figure 3(a), DHYB-0.2 best reproduces the spectra on CAIDA2 (MAE of 0.00514), whereas the best for RV2 is DHYB-0.6, a 40% difference in hybrid probability. Further, the spectral analysis top performer in CAIDA1 is DHYB-0.7. KDD and KKD are the second and third best algorithms (MAE of 0.00650 and 0.0102).

The hop-plot metric, shown in Figure 3(b), similarly finds the DHYB techniques with low hybrid ratios performing best. DHYB-0.3 most closely matches the target (MAE of 0.00877). While DHYB-0.7 consistently performed well for RV1, RV2, and CAIDA1, here it ranks eleventh of the 18 reduction algorithms studied (MAE of 0.0519). The degree distribution plot for CAIDA2 is given in Figure 3(c). DHYB-0.1 best captures this metric, whereas DHYB-0.7 is the best of the non-k-core algorithms for RV1 and DHYB-0.5 is the best non-k-core reduction for RV2 and CAIDA1.

D. k-core

Among the metrics and datasets we consider, our k-core algorithms do not always produce the best fit to the target metric; however, they are regularly among the top three. Thus, while the other algorithms we consider are relatively sensitive to the input data, the k-core algorithms perform consistently well, suggesting that our k-core algorithms generalize well to Internet-like graphs. Table III summarizes our results.

We do not perform average degree comparison for KDD and KDD as they take a target edge number as input and reach the target average degree precisely. We observe that the

spectra of graphs from KDD outperform KKD; the spectra of KKD graphs are higher on average than those of both the target and KDD. This is likely due to the removal of only the k-degree vertices in the second step of KKD which preserves the largest, most connected clusters. KDD performs nearly as well as the best performing DHYB reduction methods, despite the fact that these have hybrid probabilities differing by 40%, indicating that KDD well maintains spectral properties of the target even through a high percentage of reduction.

Figures 2(c) and 3(c) show the degree histograms of KDD and KKD. In both RV2 and CAIDA2, these methods produce highest-degree vertices with degrees of several hundred less than the target, while we observe above-average performance from the 100 to 1 vertex degree range. KKD models the target better than KDD throughout much of the degree range; KDD tends to produce a slightly higher degree high-degree vertex, however. Against all 18 reduction methods examined, KKD is the best method to capture the degree histogram in three of the four source-periods, and is one of the top four for CAIDA2.

VI. Conclusions

We first re-examine and validate the graph reduction work of [16], and then examine the soundness of the results given modern topologies obtained from different sources. We find that a single probability parameter of the recommended algorithm (DHYB) from [16] can result in poor reductions depending on the input topology. Given this finding, we developed two novel k-core reduction algorithms to preserve the most central nodes and important graph structure. Across four different topologies our k-core reduction algorithms consistently perform well by accurately capturing spectral properties, hop-plot distribution, and degree distribution. Our results suggest that the performance of KDD and KKD are less sensitive to the input graph.

While our k-core decomposition methods show promise across varied sources and periods, there are several areas that warrant further research. In order to strengthen our findings that k-core reduction methods outperform other graph sampling algorithms, a rigorous examination of these methodologies should be undertaken to determine the mathematical underpinnings behind their metric-preserving properties. Because KDD and KKD require a target edge count, an enhancement to the algorithm could optimize over the number of edges to identify an edge count that best matches the target metrics. We leave as future work an evaluation of k-core sampling to reduce IP and router-level topologies, as well as non-Internet graphs and Internet-like graphs created constructively.

TABLE III
Summary of best methods per source-period considered

	RV1	**RV2**	**CAIDA1**	**CAIDA2**
Avg. Deg	DHYB-0.7	DHYB-0.6	DHYB-0.6	DHYB-0.1
Spectral	DHYB-0.8 KDD DHYB-0.7	DHYB-0.6 KDD KKD	DHYB-0.7 DHYB-0.8 DHYB-0.6	DHYB-0.2 KDD KKD
Hop Plot	DHYB-0.7 KDD DHYB-0.8	EDFS DHYB-0.7 DHYB-0.6	KDD DHYB-0.7 DHYB-0.6	DHYB-0.3 DRV EDFS
Deg. Dist.	KKD DHYB-0.7 DHYB-0.6	KKD KDD DHYB-0.5	KKD DHYB-0.5 DHYB-0.4	DHYB-0.1 DRE DRV

Acknowledgments

This work supported in part by NSF CNS-1213155 and DHS Cyber Security N66001-2250-58231. Views and conclusions are those of the authors and should not be interpreted as representing the official policies of the U.S. government.

References

[1] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. k-core Decomposition of Internet Graphs: Hierarchies, Self-Similarity and Measurement Biases. *arXiv preprint cs/0511007*, 2005.

[2] T. Bu and D. Towsley. On Distinguishing between Internet Power Law Topology Generators. In *INFOCOM*. IEEE, 2002.

[3] E. Cem, M. E. Tozal, and K. Sarac. Impact of Sampling Design in Estimation of Graph Characteristics. In *IPCCC*. IEEE, 2013.

[4] N. Chatzis, G. Smaragdakis, J. Böttger, T. Krenc, and A. Feldmann. On the Benefits of Using a Large IXP as an Internet Vantage Point. In *ACM IMC*, 2013.

[5] Christophe Fillot. Dynamips, July 2015. https://github.com/GNS3/dynamips/.

[6] C. X. A. Dimitropoulos, G. F. Riley, D. Krioukov, and R. Sundaram. Towards a topology generator modeling as relationships. 2005.

[7] M. Garetto and D. Towsley. Modeling, Simulation and Measurements of Queuing Delay Under Long-Tail Internet Traffic. *SIGMETRICS*, 2003.

[8] P. Gill, M. Schapira, and S. Goldberg. Let the Market Drive Deployment: A Strategy for Transitioning to BGP Security. In *SIGCOMM CCR*, 2011.

[9] P. Gill, M. Schapira, and S. Goldberg. Modeling on Quicksand: Dealing with the Scarcity of Ground Truth in Interdomain Routing Data. *SIGCOMM CCR*, 2012.

[10] V. Giotsas, M. Luckie, B. Huffaker, and K. Claffy. Inferring Complex AS Relationships. In *ACM IMC*, Nov. 2014.

[11] T. G. Griffin, F. B. Shepherd, and G. Wilfong. The Stable Paths Problem and Interdomain Routing. *IEEE TON*, 2002.

[12] Y. Hyun and k. claffy. The CAIDA Macroscopic Topology Data, 2014. http://www.caida.org/projects/macroscopic/.

[13] C. Jin, Q. Chen, and S. Jamin. Inet: Internet topology generator. 2000.

[14] Juniper Networks. Junosphere User Guide, 2015.

[15] S. Knight, H. Nguyen, O. Maennel, I. Phillips, N. Falkner, R. Bush, and M. Roughan. An Automated System for Emulated Network Experimentation. In *CoNEXT*. ACM, 2013.

[16] V. Krishnamurthy, M. Faloutsos, M. Chrobak, J.-H. Cui, L. Lao, and A. G. Percus. Sampling large internet topologies for simulation purposes. *Computer Networks*, 2007.

[17] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, k. c. claffy, and A. Vahdat. The Internet AS-level Topology: Three Data Sources and One Definitive Metric. *SIGCOMM CCR*, 2006.

[18] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *MASCOTS*. IEEE, 2001.

[19] D. Meyer. RouteViews, 2014. http://www.routeviews.org.

[20] J. Obstfeld, S. Knight, E. Kern, Q. S. Wang, T. Bryan, and D. Bourque. VIRL: the Virtual Internet Routing Lab. In *SIGCOMM*. ACM, 2014.

[21] I. Poese, B. Frank, S. Knight, N. Semmler, and G. Smaragdakis. PaDIS Emulator: An Emulator to Evaluate CDN-ISP Collaboration. In *ACM SIGCOMM*, 2012.

[22] E. Rye. Python graph reduction implementations, 2015. https://github.com/cmand/graphreduce.

[23] E. Rye and J. P. Rohrer. Graph Reduction for Emulated Network Experimentation. Technical Report CS-15-001, NPS, 2015.

[24] L. M. Vaquero, S. S. Lor, D. Audsin, P. Murray, and N. Wainwright. Sampling ISP Backbone Topologies. *Comm. Letters, IEEE*, 2012.

[25] S.-H. Yook, H. Jeong, and A.-L. Barabási. Modeling the Internet's Large-Scale Topology. *National Academy of Sciences*, 2002.

[26] E. Zegura, K. Calvert, and S. Bhattarcharjee. How to Model an Internetwork. In *INFOCOM*. IEEE, 1996.

[27] B. Zhang, R. Liu, D. Massey, and L. Zhang. Collecting the Internet AS-level Topology. *SIGCOMM CCR*, 2005.

[28] S. Zhou and R. J. Mondragon. Towards Modelling the Internet Topology: The Interactive Growth Model. *Teletraffic Sci. and Eng.*, 2003.