High Performance Hypergraph Analytics of Domain Name System Relationships

Cliff A Joslyn
Pacific Northwest National Laboratory
Seattle, WA, USA
Cliff.Joslyn@pnnl.gov

Sinan Aksoy
Pacific Northwest National Laboratory
Richland, WA, USA
Sinan.Aksoy@pnnl.gov

Dustin Arendt
Pacific Northwest National Laboratory
Richland, WA, USA
Dustin.Arendt@pnnl.gov

Louis Jenkins
University of Rochester
Rochester, NY, USA
Louis.Jenkins@rochester.edu

Brenda Praggastis
Pacific Northwest National Laboratory
Seattle, WA, USA
Brenda.Praggastis@pnnl.gov

Emilie Purvine
Pacific Northwest National Laboratory
Seattle, WA, USA
Emilie.Purvine@pnnl.gov

Marcin Zalewski
Pacific Northwest National Laboratory
Seattle, WA, USA
Marcin.Zalewski@pnnl.gov

***Abstract*—We report on the use of novel mathematical methods in hypergraph analytics over a large quantity of DNS data. Hypergraphs generalize graphs, as used in network science, to better model complex multiway relations in cyber data. Specifically, casting DNS data from Georgia Tech's ActiveDNS repository as hypergraphs allows us to fully represent the interactions between *collections* of domains and IP addresses. To facilitate large-scale analytics, we fielded an analytical pipeline of two capabilities. HyperNetX (HNX) is a Python package for the exploration and visualization of hypergraphs, acting as a frontend. For the backend, the Chapel HyperGraph Library (CHGL) is a library for high performance hypergraph analytics written in the exascale programming language Chapel. CHGL was used to process gigascale DNS data, performing compute-intensive calculations for data reduction and segmentation. Identified portions are then sent to HNX for both exploratory analysis and knowledge discovery targeting known tactics, techniques, and procedures.**

***Index Terms*—Hypergraphs, DNS, high performance computing, Chapel.**

I. Introduction

Many problems in data analytics involve rich interactions amongst multiple entities, for which graph representations are commonly used. High order (high dimensional) interactions, which abound in cyber and social networks, can only be represented in graphs as highly inefficiently coded, “reified” labeled subgraphs. Lacking multi-dimensional relations, it is hard to address questions of “community interaction” in graphs: e.g., how is a collection of entities A connected to another collection B through chains of other communities?; where does a particular community stand in relation to other communities in its neighborhood?

Hypergraphs [2] are generalizations of graphs which allow edges to connect any number of vertices. Hypergraph methods are well known in discrete mathematics, and are closely related to important objects in data science such as bipartite graphs, set systems, partial orders, finite topologies, and especially graphs proper, which they directly generalize (every graph is a 2-uniform hyergraph). In HPC, hypergraph-partitioning methods help enable parallel matrix computations [7], and have applications in VLSI [12]. In the network science literature, researchers have devised several path and motif-based hypergraph data analytics (albeit fewer than their graph counterparts), such as in clustering coefficients [14] and centrality metrics [8].

Complex data commonly analyzed using network science methods, and especially including cyber data, often contain multi-way interactions. But while they thus present naturally as hypergraphs, still hypergraph treatments are very unusual compared to graph representations of the same data. This is due at least to the greater mathematical, conceptual, and computational complexity of hypergraph methods, since as data objects, hypergraphs scale as $O(2^n)$ in the number of vertices n, as opposed to $O(n^2)$ for graphs. In the face of this, complex data are typically collapsed or are simplified to graphs to ease analysis.

Our research group is dedicated to facing the challenge of the complexity of hypergraphs in order to gain the formal clarity and support for analysis of complex cyber data they provide. A substantial high-performance computing (HPC) component is thus necessary, despite hypergraph analytics not receiving much attention in the software engineering community at large, and the HPC community in particular. We thus pursue a two-fold approach to developing our methods:

1) We employ the **Ch**apel **H**yper**g**raph **L**ibrary (CHGL, https://github.com/pnnl/chgl) [11]), a library for hypergraph computation in the emerging Chapel programming

language [5], [6], for HPC hypergraph processing, large scale analysis, and data segmentation.

2) We explore single hypergraphs or collections of hypergraphs using **HyperNetX** (HNX, https://github.com/pnnl/HyperNetX), a Python library being developed by PNNL for exploratory data analytics and visualization of hypergraphs.

In our work, CHGL and HNX are two stages of an analytical pipeline: CHGL provides a highly abstract interface for implementation of HPC hypergraph algorithms over large data, identifying segments and subsets which can then be passed to HNX for more detailed analysis.

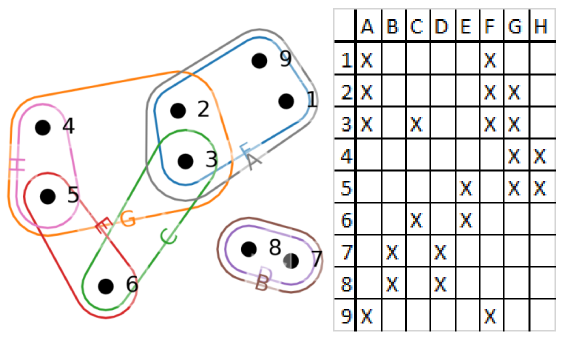
In this paper we first introduce the rudiments of hypergraph mathematics and hypernetwork science in the context of our CHGL and HNX capabilities. We then describe the DNS data set, selections of the ActiveDNS data sets from Georgia Tech University. We then describe CHGL, before going on to describe the results of our demonstration analyses. These include both basic global statistics like degree and edge size distributions, as well as exploratory and targeted discovery of small components. The exploratory discovery involves motif mining and computation of simple hypergraph metrics to discover outliers. On the other hand, targeted discovery is motivated by known bad activity. We built a blacklist of IP addresses and domains that follow a known pattern used by a global criminal operation as described in a FireEye Threat Research Blog entry [4].

II. Hypergraph Analytics

An undirected **hypergraph** is a pair $\mathcal{H} = \langle V, \mathcal{E} \rangle$ with V a finite, non-empty set of **vertices**, and $\mathcal{E}$ a non-empty multiset of **hyperedges** $e \in \mathcal{E}$ (or just "edges" when clear), where $\forall e \in \mathcal{E}, e \subseteq V$. Hypergraphs can be represented in many forms, two of which are shown in Fig. 1 for a small example $\mathcal{H}$ with $V = \{1, 2, \dots, 9\}$, representing $|V| = 9$ IP addresses.[1] On the left is an Euler diagram showing each of eight hyperedges $A, B, \dots, H$, representing domains, as a "lasso" around its vertices. On the right is a $V \times \mathcal{E}$ incidence matrix I, where a non-null $\langle v, e \rangle \in I$ cell indicates that $v \in e$ for some $v \in V, e \in \mathcal{E}$.
We call each hyperdge $e \in \mathcal{E}$ an s-edge where $s = |e|$. Thus all graphs are hypergraphs, in that all graph edges are 2-edges, for example $H = \{4, 5\}$, saying that the domain H has two IPs 4 and 5. But $F = \{1, 2, 3, 9\}$ is a 4-edge, with domain F having those four IPs. This is not representable in a graph. Where each column of the incidence matrix of a graph has exactly two cells, those of hypergraphs are unrestricted.

Our research group is pursuing hypergraph analytics as an analog to graph analytics [13]. While our development is consistent with others in the literature [8], [15], our notation and concepts are somewhat distinct. We say that two edges $e, f \in \mathcal{E}$ are s-**adjacent** if $|e \cap f| \geq s$ for $s \geq 1$. An s-star is a set of edges $\mathcal{F} \subseteq \mathcal{E}$ sharing exactly a common intersection $f \subseteq V$, with $|f| \geq s$, so that $\forall e_i, e_j \in \mathcal{F}$ we have $e_i \cap e_j = f$.

[1] $\mathcal{H}$ can also be represented as a bipartite graph on the disjoint union $V \sqcup \mathcal{E}$, with each component a distinct part.



	A	B	C	D	E	F	G	H
1	X					X		
2	X					X	X	
3	X		X			X	X	
4							X	X
5					X		X	X
6			X		X			
7		X		X				
8		X		X				
9	X					X		

Fig. 1: (Left) An Euler diagram of an example hypergraph $\mathcal{H}$. (Right) Its incidence matrix I.

An s-**path** is a sequence of edges $\langle e_0, e_1, \dots, e_n \rangle$ such that each e_{i-1}, e_i are s-adjacent for $1 \leq i \leq n$; and an s-**component** is a maximal collection of edges any pair of which is connected by an s-path. The s-**diameter** of an s-component is the length of its longest shortest s-path. Comparing again to graphs, graph paths are all 1-paths, and graph components all 1-components. Our example has two 1-components (shown obviously), but also four 2-components (listed edge-wise) $\{A, F, G, H\}, \{B, D\}, \{C\}$ and $\{E\}$. It's 3- and 4-components are each single edges of size larger than 3 or 4 (respectively), and it has no 5 or higher components.

Given a hypergraph $\mathcal{H}$, it is possible to construct smaller representations which capture important properties:

- Note that in our example, the edges $A = F$ and $B = D$, and the vertices $1 = 9$ and $7 = 8$, are equivalent, represented as duplicate columns and rows in I respectively. **Collapsing** is the process of combining these and replacing them with a representative, while also possibly maintaining a multiplicity count to be used for a weighting. The edges $\mathcal{E}$ are hereby transformed from a multiset to a set.
- Additionally, note that after collapsing, the smaller 1-component becomes an **isolated singleton**, effectively a collection of non-interacting vertices, or a diagonal block in I. These are especially common in DNS data. Pre-collapse, an isolated singleton would indicate the normal, *uninteresting* behavior in DNS where a single IP is associated with a single domain, and *vice versa*. But post-collapse, they indicate a collection of IPs and domains which are universally associated only with themselves, effectively forming a set of domain and IP aliases. In this work, these are counted and pruned, but in the future they could be attended to with respect to their multiplicities.
- Finally, note that $H \subset G$ is an **included** edge. Non-included edges are called **toplexes**, and not only is the collection of toplexes much smaller than $\mathcal{H}$, but it is sufficient to derive some hypergraph information, for example s-components.

Table I shows some important statistics for our example, first for the initial hypergraph, then after collapsing, and

	Initial	Collapsed	Non-Singleton Components
$\|V\|$	9	7	6
$\|E\|$	8	6	5
Aspect ratio	1.125	1.167	1.200
# Cells	23	14	13
Density	0.319	0.333	0.433

TABLE I: Basic hypergraph statistics for our example.

finally after removing isolated singletons from the collapsed hypergraph. For hypergraph data, a vastly high or low aspect ratio can indicate difficulty in analysis. Note that as reductions commence, the number of vertices, edges, and cells reduces, while density increases. Finally, Fig. 2 shows the distribution of node degree (# edges per node) and edge size.

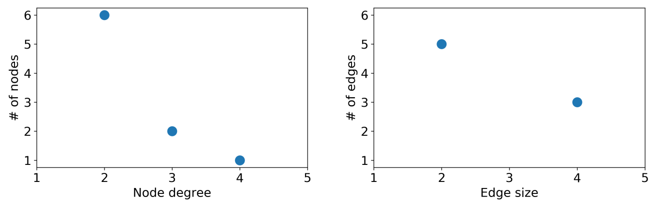


Fig. 2: (Left) Distribution of node degree (# edges per node) in our example. (Right) Distribution of edge size s.

In our pipeline the segmentation steps of collapsing, removing isolated singletons, and computing s-components are all performed using CHGL, as are node degree, edge size, and s-component size distributions. Subsequent exploration of the structures found within the components themselves, e.g., identification of stars and computation of diameters, are done via HNX. HNX builds on the popular library NetworkX [9], which offers metrics and algorithms for the analysis of graph data. Euler diagram visualizations that appear in this paper are provided directly by the HNX package.

III. Hypergraph Representations of DNS Data

The Domain Name System (DNS) provides a decentralized service to translate from the domain names that humans keep track of (e.g., `www.google.com`) to IP addresses that computers require to communicate. Perhaps somewhat counter-intuitively, DNS data present naturally as a hypergraph, in being a many-many relationship between domains an IPs. While typically this relationship is one-to-one, with each domain uniquely identifying a single IP address and *vice versa*, there are a number of circumstances which can violate this:

- Some domains have aliases so that multiple domains (e.g., misspellings) resolve to the same IP address.
- There are large hosting services in which one IP serves up multiple different websites.
- Some domains are used so frequently that they must be duplicated across hosts and therefore map to multiple IPs.
- IP addresses are randomly reassigned within some small IP block so the same domain may map to multiple IP addresses when queried over the course of a day.

In order to explore large volumes of DNS mappings we turned to ActiveDNS, a data set maintained by the Astrolavos Lab at Georgia Institute of Technology (https://activednsproject.org). The project submits daily DNS lookups for popular zones (e.g., com, net, org, biz, gov) and lists of domain names (e.g., Alexa Top 1M). The data is stored in Avro format (https://avro.apache.org) which provides structured records for each DNS lookup in a compressed binary file. Each record contains information including: query date, lookup input (often a domain name), data returned by a DNS server (often a list of IP addresses), and IP addresses of the DNS servers that answered the query.

Our group acquired data from the time period April 24–May 29, 2018, and in this paper we focus on the single day of April 26, 2018. This day consists of 1,200 Avro files with each file containing on average 900K records. There was some data cleaning necessary to remove records with empty lookup input or empty returned data. Additionally we removed any records in which the lookup input was an IP address or the returned data was a domain name. After cleaning, each file was reduced to approximately 180K records.

We structured these DNS data as a hypergraph on a vertex set V of IPs and edge set $\mathcal{E}$ of domains. Thus our hypergraphs $\mathcal{H}$ coded each domain as a collection of its IPs. We show results of our anlaysis below in Section V, including global statistics and the results of targeted exploration.

IV. Chapel Hypergraph Library (CHGL)

The Chapel HyperGraph Library (CHGL) [11] is a prototype exascale library written in Chapel [5], [6] that brings generation, representation, and computation of hypergraphs to the world of high performance computing (HPC). Thanks to Chapel, CHGL provides scalability in both shared memory and distributed memory contexts. Next, we discuss how hypergraphs are created (property maps), filtered (segmentation), and how metrics are computed in CHGL.

A. Property Map

In most cases, data underlying a hypergraph is more complex than CHGL's internal representation of vertices and hyperedges as consecutive integers. In such situations, a hash table that maps user-defined generic properties to the consecutive identifiers of vertices and hyperedges is used for translation. The properties are embedded in the internal representation of the hyperedges and vertices, allowing $O(1)$ bidirectional lookup as well as locality when iterating over the graph, shared-memory and distributed alike.

B. Segmentation

CHGL performs *segmentation*, or reduction, of the data in multiple highly-parallel phases. Segmentation reduces both the size of the graph to one that HNX can process in a reasonable amount of time and the computational workload on CHGL when computing metrics. Proper care is taken to ensure that references to the collapsed hyperedges and vertices are taken forward to the hyperedge or vertex that they collapsed into, and that all references to removed hyperedges and vertices are removed. This is performed in linear time and applies to both the graph and property map.

1) Collapsing Duplicates: To prune away redundant entities, which is generally useful for computation, hyperedges and nodes are placed into equivalence classes through the process of collapsing described in Section II. All but one arbitrarily chosen representative is removed from the graph. Determining the equivalence class of a vertex or hyperedge can be done by using a set or hash table, and can be performed in $O(|V|)$ or $O(|V| \log |V|)$ time, depending on the data structure used. In practice, the time complexity is often linear or quasilinear, but in the worst-case scenario when the hypergraph is fully connected, the time complexity is $O(|V|^2)$ or $O(|V|^2 \log |V|)$.

2) Removing Isolated Components: Isolated singletons, as described in Section II, tend to be uninteresting. After collapsing, these are pruned away in a straightforward manner.

3) Computing s-Components: We implemented computation of s-components using a parallel search method, where we iterate over edges in parallel, and every edge begins an independent search. The s-neighbors of an edge are marked with the component number originating from the initial edge. The component number is taken from a global atomic counter at the beginning of every parallel search. 1-Components are implemented by simply traversing the edges by following included vertices (edge to vertex to neighbor edge), but 2-components and higher require an implementation with set intersections to check the cardinalities of adjacencies. This implementation is well suited for a large number of small components because most components end up being searched by a single task. The best case scenario complexity of the parallel search algorithm is linear, and the worst is quadratic if the maximum number of component collisions occur. The average complexity in our case is close to linear since the DNS data has a large number of small components, and most components are handled by a single task.

C. Metrics

1) Vertex Degree and Edge Cardinality Distribution: Obtaining the vertex degree and edge cardinality distributions is simple and intuitive in CHGL, thanks to Chapel's high-level abstractions. This particular operation is short enough that it can be presented in full in Figure 3. We compute these both pre- and post-collapsing.

```
// Find largest degree of all vertices in the hypergraph
var N = max reduce [v in graph.getVertices()] graph.degree(v);
var degreeDist : [1..N] int;
forall v in graph.getVertices() with (+ reduce degreeDist) {
  degreeDist[graph.degree(v)] += 1;
}
```

Fig. 3: Obtaining the vertex degree distribution in CHGL.

2) s-Component Size Distribution: The s-component size distributions are computed, recording the number of nodes and edges in each s-component and how many s-components have each size. This allows us to understand how nodes and edges are distributed, e.g., is there one giant component and a few small components or are component sizes more uniformly distributed.

D. Blacklisted IP Address and DNS Name Search

To demonstrate CHGL's versatility, we use the property map to search for blacklisted DNS names and IP addresses. If there is a match we record the s-component that it is in from the cached s-components. This is provided as output to be analyzed, visualized, and observed in HNX.

V. Results

A. Loading and Compute Time

Execution times of the stages of the CHGL DNS processing pipeline are shown in Figure 4. s-Component computation dominates the execution time for 128 or more files. The s-components are reused when computing the s-component size distributions and in computing the s-component of the blacklisted IP addresses and DNS names, leading to them taking significantly less time. Collapsing duplicates and removing isolated components scale linearly, as is expected for their time complexity. The hypergraph is constructed in about the same amount of time it takes to collapse it, showing that processing DNS data is mostly compute-bound.

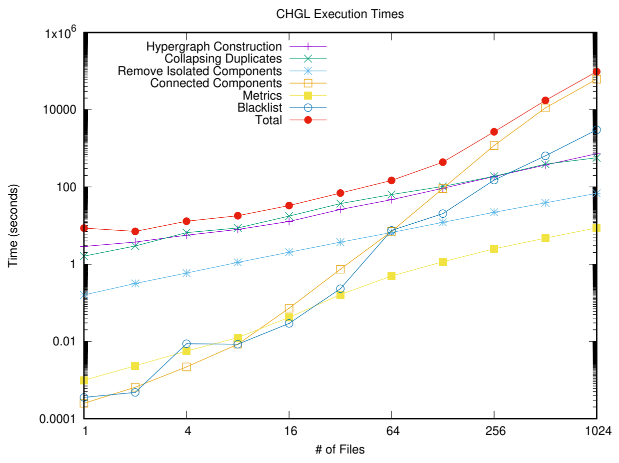


Fig. 4: Execution times (log-log scale).

B. Effectiveness of Reduction

The purpose of segmentation is to reduce the size of the graph while also maintaining the data that is of interest. Figure 5 shows the compression as a result of performing segmentation. Collapsing of duplicate edges results in the most compression, reducing the graph from 55% at one file to over 90% at 1024 files, which can be expected to improve further when more data is processed. Removing isolated components results in less compression as data size increases, likely due to the premature marking of components as isolated prior to having all of the data. Perhaps with larger amounts of data, there will be a convergence to a stable number of isolated components in the entirety of the DNS network. Note that there are very few duplicate IP addresses on smaller samples, but that may change as more data is processed; nonetheless, collapsing duplicate vertices may be unnecessary and can possibly save some time.

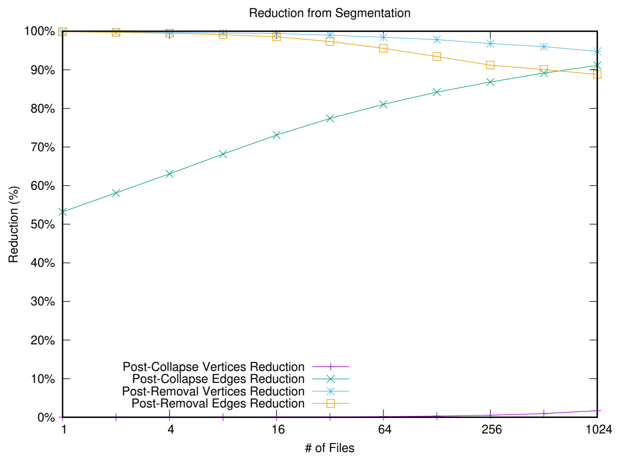


Fig. 5: Effectiveness of reduction from segmentation.

	Initial	Collapsed	Non-Singleton Components
$\|V\|$	10.6M	10.3M	557K
$\|E\|$	131.2M	11.0M	1.2M
Aspect ratio	0.081	0.941	0.460
# Cells	157.4M	25.7M	15.9M
Density	1.14 E-7	2.26 E-7	2.35 E-5

TABLE II: Basic hypergraph statistics for ActiveDNS data for April 26, 2018.

C. Basic hypergraph statistics

Above we reported on scaling of loading and compute time using CHGL on varying numbers of ActiveDNS files, from 1 to 1,024. Here we report on analysis of the hypergraph built from one full day, April 26, 2018, comprising 1,200 files. See Table II for basic count statistics.

The node degree and edge size distributions are shown in Figures 7a and 7c. Except for the small increase around $x = 10^2$ the node degree distribution looks like a power law or heavy tailed distribution typical in real-world graphs [1]. The degree distribution has a general decreasing tendency from $x = 1$ to $x = 70$, it increases by roughly 1,000 through $x = 80$, and then returns to the downward trend. We do not know why this occurs, but it is possible that it could be an artifact of DNS server configuration practices. Edge size distribution also seems to be heavy-tailed although somewhat more noisy for low edge sizes than the degree distribution.

D. Hypergraph collapsing

See the second column in Table II for the simple count statistics of the collapsed hypergraph. Notice that collapsing resulted in a much more square incidence matrix since only 2% of nodes were collapsed while 92% of edges were collapsed. The number of cells in the collapsed hypergraph incidence matrix is now reduced to 16% of the full hypergraph.

The distributions of node and edge duplicate counts are shown in Figure 6. Notice that the distribution of duplicate edge counts has a similar shape as the node degree distribution of the original hypergraph with a slight increase around $x = 10^2$. After seeing this it is possible that the nodes which had degree around 70-80, where this increase occurs, were actually in many duplicate edges which are now collapsed. The node degree distribution for the collapsed hypergraph found in Figure 7b further supports this hypothesis since the increase around $x = 10^2$ in the node degree distribution is absent.

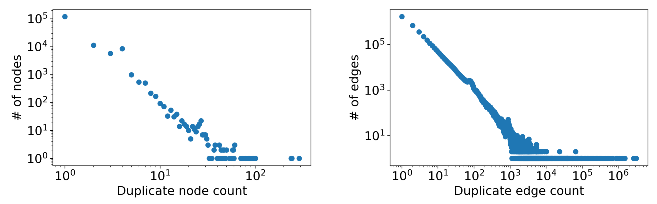


Fig. 6: Distribution of duplicate node counts (top) and edge counts (bottom).

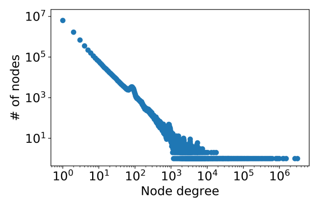
The edge size distribution post collapse is shown in Figure 7d. This distribution is very similar to that of the original hypergraph, although it appears less noisy up through approximately $x = 20$. This is not surprising since there were not many duplicate nodes removed, so edges that remained likely stayed close to their original size.

After collapsing duplicate nodes and edges we removed all 9,784,763 isolated singleton edges, or 89% of all remaining edges. The only differences between the collapsed hypergraph and the hypergraph after removal of isolated singleton components is the number of degree 1 nodes and the number of size 1 edges. Therefore, we omit the final node degree and edge size distributions since they are identical to the post-collapse distributions except for the points at $x = 1$.

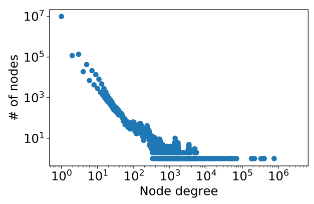
Comparing the pre-collapse (left), post-collapse (right), and post-removal distributions (not pictured) in Figure 7, we observe that hypergraph collapsing and removal significantly alters the shape of degree and edge size distributions. In addition to the qualitative differences apparent from the plots, these differences can also be quantified using the Kolmogorov-Smirnov (KS) distance metric, a normalized statistic between 0 and 1 in which larger values indicate greater degree distribution dissimilarity. In the case of the degree distributions (top row), KS distance suggests the pre-collapsing hypergraph differs significantly from the post-collapse and post-removal degree distributions, with KS values of 0.36 and 0.34, respectively. In the case of the edge-size distributions (bottom row), the most pronounced difference is between the pre-collapsing and post-removal edge size distribution, with a KS value of 0.60. Here, the large KS distance reflects the dramatic changes at the head of the distribution, where the number of 1-edges decreases from 118 million to 369 thousand.

E. s-Components

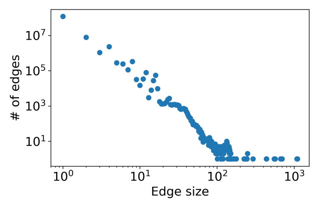
Our next step toward finding interesting subgraphs within the single day of ActiveDNS data was to compute s-components. CHGL computed s-components of the hypergraph post-collapse and post-removal of isolated singletons for $s = 1, 2, 3$. Before exploring these components themselves we



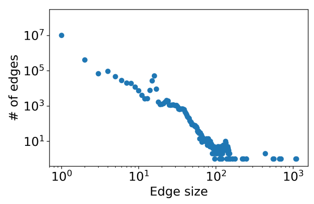
(a) Initial node degree distribution



(b) Post-collapse degree dist.



(c) Initial edge size distribution



(d) Post-collapse edge size dist.

Fig. 7: Node degree and edge size distributions, on a log-log scale, for April 26, 2018 DNS hypergraph. The x and y axes are the same across both node plots and across both edge plots to illustrate the changes through the collapsing procedure.

report the distribution of component sizes (both node and edge counts) which are found in Figure 8. As s increases the shapes of these distributions do not change much but they do tend to skew more toward smaller components and the distribution flattens slightly. This is required since every s-component is contained within some s'-component for $s' < s$: as s increases components can only decrease in size. These distributions also show that while there are some very large s-components the majority are very small. Additionally, we see that the notion of a "giant component" is much more prevalent in the set of 1-components than for $s = 2$ or 3. Indeed, as s increases the largest component breaks up and the jump between the largest component and second largest becomes smaller.

F. Exploration of segments using HNX

Once the hypergraphs were segmented into s-components by CHGL we proceeded to do exploratory analysis using HNX. In particular, we looked for:

- Occurrences of 1-stars within the 1-components, and
- s-components with maximum s-diameter for $s = 2, 3$.

Recall that a 1-star is a small hypergraph in which all edges pairwise intersect in one node, and that one node is the same across all pairwise intersections. The simplest 1-star has all edges of size 2, see Figure 10 for an example of this case. In our DNS use case a star is a collection of domains which all share exactly one IP address but each also have their own separate IP address(es). These are consistent with the behavior of content delivery networks (CDN), geographically distributed networks of servers with the goal of quickly and reliably serving up content to a variety of users, which could explain the existence of stars with a diverse set of IP addresses since a consideration for IP assignment is geographic location. Star motifs are also consistent with DNS sinkholes and domain hosting services.

We searched the 1-components for 1-stars and looked for size outliers. The distribution of number of edges per star is shown in Figure 9. We can see that there is one notable outlier, a star with 701 edges and 642 toplexes. The domain names within this star appear to be mostly randomly generated and from the .com and .net zones (e.g., `twlwta.com`, `comgslklpa.net`) and the common IP address within all domains is 17.17.17.17. A WHOIS search finds that this IP address is within the network range of Apple, Inc. The other 642 IPs present in this star come from 640 distinct of /16 ranges. This is consistent with DNS sinkhole behavior where traffic to a variety of (potentially malicious) domains is redirected to a benign location [3]. And, in fact, current (i.e., not on April 26) DNS searches for a sample of domains within this star have a Start of Authority (SOA) record with "sinkhole root@sinkhole" as the name and contact for the server.

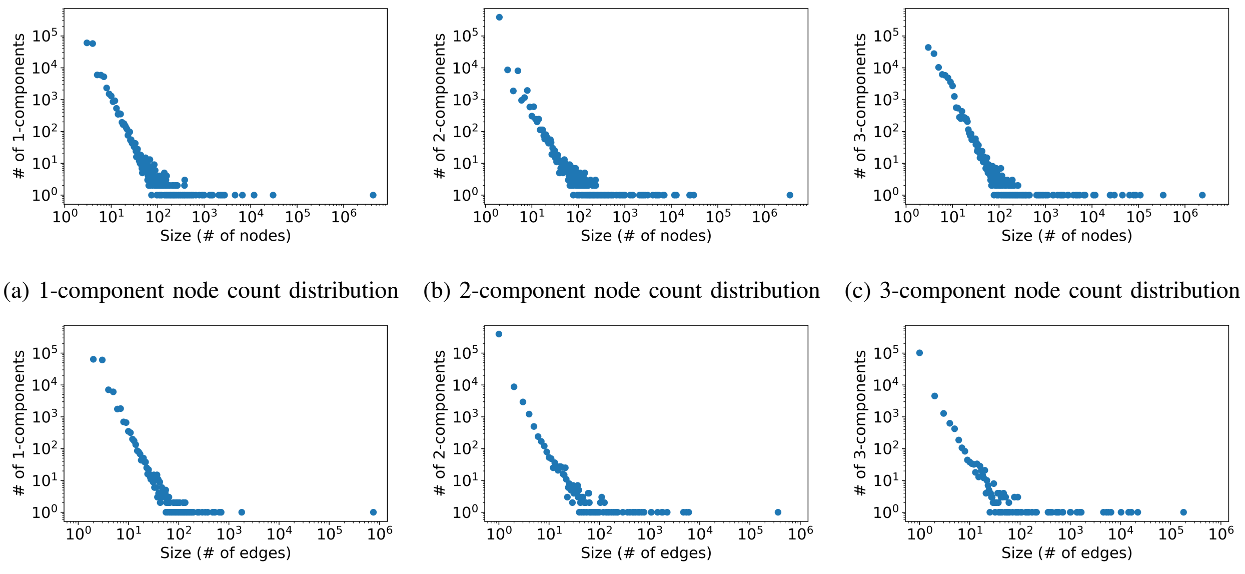
Unlike this largest star which had IP addresses in many different ranges, smaller stars such as the one shown in Figure 10 tend to have all IPs and domains within the same, or a relatively small set of, ranges and organizations. In this small example the central IP address is from Google Cloud whereas the leaves are from Microsoft Corporation. All five domains are registered through the hosting site GoDaddy.com.

To discover interesting 2-components (resp. 3-components) we calculated 2-diameters (resp. 3-diameters) of each of the components and look more closely at those with maximal diameter. In the case of the 2-components the maximum 2-diameter is 6 and there is only one 2-component with that 2-diameter, shown in Figure 11. The IP addresses in this component all belong to the IP range 103.86.122.0/24 and the domains are registered to GMO INTERNET, INC. Moreover, current DNS queries for most of these domains now resolve to IPs in the range 103.86.123.0/24 and have a time to live of only 120 seconds. This pattern of quickly changing of IP address is consistent with the *fast flux* DNS technique which can be used by botnets to hide malicious content delivery sites and make networks of malware more difficult to discover [10].

The large diameter 3-components tell different stories. The maximum 3-diameter is 3 and there are four 3-components with this 3-diameter. One has only one toplex with six sub-edges. Two others are fairly simple and, like the large 2-diameter 2-component, are somewhat chain-like tracing out a long path. The fourth is quite large with 70 nodes, 189 edges, and all IPs belonging to an IP range from Amazon Technologies Inc.

Thus far our exploratory analysis of stars and large diameter components results in some observations of typical patterns of DNS use with interesting behavioral characteristics. It is our hope that this type of exploration, when performed by network defenders on their own systems, might result in detection of abnormal patterns of behavior to augment their current threat hunting capabilities.

The final exploration we performed was on our blacklist of domains. Investigations by FireEye [4] recorded tactics, tech-



(a) 1-component node count distribution (b) 2-component node count distribution (c) 3-component node count distribution

(d) 1-component edge count distribution (e) 2-component edge count distribution (f) 3-component edge count distribution

Fig. 8: Node and edge count distributions, on a log-log scale, for s-components within simplified April 26, 2018 DNS hypergraph. The x and y axes are the same across all three node count plots and across all three edge count plots to illustrate the changes as s increases.

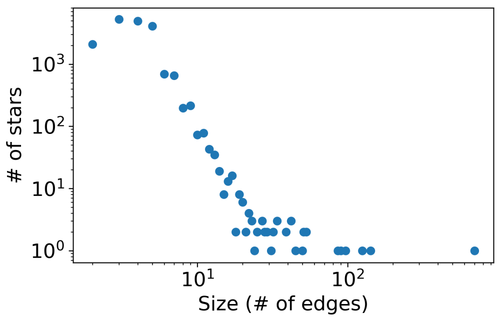


Fig. 9: Distribution of star sizes (# of edges).

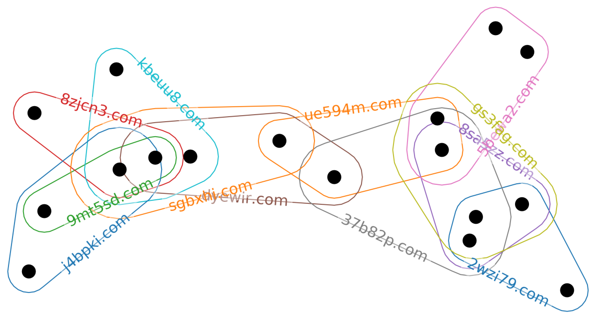


Fig. 11: The 2-component with largest 2-diameter.

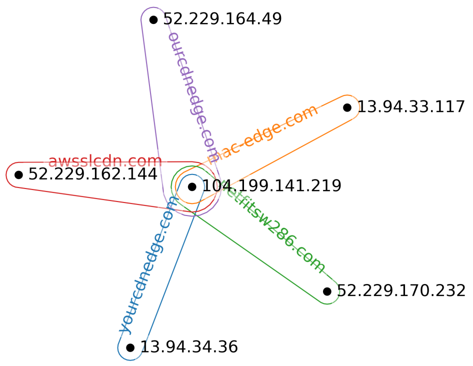


Fig. 10: A small star seen in the ActiveDNS data.

niques, and procedures (TTP) used by the FIN7 organization for various stages of the attack lifecycle. One in particular, for maintaining presence on a system, was to use command and control domains registered with “odd format and top-level domains.” The format they identified was four or five letters followed by one of the following extensions: pw, us, club, info, site, top, e.g., `ttjic.top`. We used CHGL to do a regex search within the April 26 data set for any domains that fit this pattern and found 2,088 matching domains. For each found domain we recorded the s-components which contain it, for $s \in \{1, 2, 3\}$. Many of the found domains are in the largest s-component and likely are not connected to one another within that component. But, in at least one case we found a set of ten domains that follow this regex pattern and are all contained within the same 2-component with 16 edges and 3-component with 13 edges. The 3-component, shown in Figure 12, is nearly a star. There is no common intersection among all domains although there are two central IPs and each domain contains at least one of these two IPs. All domains in this component

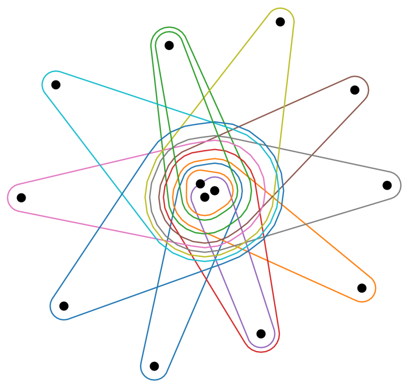


Fig. 12: A 3-component with 13 edges containing 10 blacklisted domains.

are registered by "Chengdu west dimension digital." Although it is not possible to discern whether or not these domains are part of the FIN7 network, we illustrate that this type of targeted analysis could be used to discover how known TTP signatures may be present within a data set.

VI. Conclusions and Future Work

While our research group has been developing hypergraph methods and mathematics over a moderate period, this paper reflects the first application of CHGL to cyber data, and the first use of HNX, which is newly released.

The current approach is limited in a number of ways. First, ActiveDNS records data from DNS lookups on a daily basis (or perhaps multiple times per day), but it does not do continual monitoring. This discrete sampling may mean that the pipeline misses patterns that would normally be seen in a more continuous approach. Additionally, the current analysis is for a single day, and extending to multiple days in the current architecture will exacerbate issues with memory bounds. This might be mitigated using a theory of dynamic hypergraphs (much like that of dynamic graphs) to understand the time-evolution of DNS or similar data.

Additionally, certain DNS relationships are ignored, such as recursive DNS records where one domain resolves not to an IP address but to another domain name. This would require more complicated mathematics than just hypergraphs, likely cell complexes or partial orders, which we have started to consider in our research but not yet in our analysis. We also ignore other pieces of metadata like the authority IP addresses (those servers which answered the DNS request).

Additional future work includes:

- We are extending our prior theoretical work [13] to a full consideration of the mathematical foundations of hypergraphs for data science, including spectral approaches and consideration of multiplicity weightings.
- A range of hypernetwork methods generalizing network science centrality, connectivity, clustering coefficients, etc. are available and under development for application.
- Also central to our approach is the consideration of hypergraphs as multidimensional objects, and thus inherently available for topological applications, including homology measurement for identification of loops and potential gaps in the underlying data.
- CHGL is also under active development to include topology, homology measures, a proper graph library, and a distributed data model.
- Finally, application and data analysis continues, including DNS, additional cyber data beyond DNS, and additional application domains including computational biology and social hypernetworks.

VII. Acknowledgements

This work was partially funded under the High Performance Data Analytics (HPDA) program at the Department of Energy's Pacific Northwest National Laboratory. Pacific Northwest National Laboratory is operated by Battelle Memorial Institute under Contract DE-ACO6-76RL01830.

Special thanks to William Nickless for helpful conversations surrounding the DNS analysis and interpretation.

References

[1] Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific american*, 288(5):60–69, 2003.

[2] Claude Berge and Edward Minieka. *Graphs and Hypergraphs*. North-Holland, 1973.

[3] Guy Bruneau. DNS Sinkhole. https://www.sans.org/reading-room/whitepapers/dns/dns-sinkhole-33523.

[4] N Carr, K Goody, S Miller, and B Vengerik. On the Hunt for FIN7: Pursuing an Enigmatic and Evasive Global Criminal Operation. https://www.fireeye.com/blog/threat-research/2018/08/fin7-pursuing-an-enigmatic-and-evasive-global-criminal-operation.html.

[5] B.L. Chamberlain, D. Callahan, and H.P. Zima. Parallel Programmability and the Chapel Language. *Int. J. High Perform. Comput. Appl.*, 21(3):291–312, August 2007.

[6] Bradford L. Chamberlain, Elliot Ronaghan, Ben Albrecht, Lydia Duncan, Michael Ferguson, Ben Harshbarger, David Iten, David Keaton, Vassily Litvinov, Preston Sahabu, and Greg Titus. Chapel comes of age: Making scalable programming productive. *Cray Users Group*, 2018.

[7] K.D. Devine, E.G. Boman, R.T. Heaphy, R.H. Bisseling, and U.V. Catalyurek. Parallel hypergraph partitioning for scientific computing. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*. IEEE, 2006.

[8] Ernesto Estrada and Juan A. Rodríguez-Velázquez. Subgraph centrality and clustering in complex hyper-networks. *Physica A: Statistical Mechanics and its Applications*, 364:581–594, may 2006.

[9] AA Hagberg, DA Schult, and PJ Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[10] jamie.riden. How Fast-Flux Service Networks Work. http://www.honeynet.org/node/132. Accessed: 2018-11-26.

[11] Louis P Jenkins, Tanver Bhuiyan, Sarah Harun, Christopher Lightsey, Sinan Aksoy, Tim Stavenger, Marcin Zalewski, Hugh Medal, and Cliff Joslyn. Chapel hypergraph library (chgl). In *2018 IEEE High Performance Extreme Computing Conf. (HPEC 2018)*, 2018.

[12] George Karypis and Vipin Kumar. Multilevel k-way hypergraph partitioning. *VLSI Design*, 11(3):285–300, jan 2000.

[13] Emilie Purvine, Sinan Aksoy, Cliff Joslyn, Kathleen Nowak, Brenda Praggastis, and Michael Robinson. A topological approach to representational data models. In S. Yamamoto and H. Mori, editors, *Human Interface and the Management of Information. Interaction, Visualization, and Analytics (LNCS, volume 10904)*, pages 90–109, 2018.

[14] Garry Robins and Malcolm Alexander. Small worlds among interlocking directors: Network structure and distance in bipartite graphs. *Computational & Mathematical Organization Theory*, 10(1):69–94, may 2004.

[15] J. Wang and T.T. Lee. Paths and cycles of hypergraphs. *Science in China Series A: Mathematics*, 42(1):1–12, 1999.