Unveiling Core Network-Wide Communication Patterns through Application Traffic Activity Graph Decomposition

Yu Jin, Esam Sharafuddin, Zhi-Li Zhang *
Department of Computer Science and Engineering
University of Minnesota
{yjin,shara,zhzhang}@cs.umn.edu

ABSTRACT

As Internet communications and applications become more complex, operating, managing and securing networks have become increasingly challenging tasks. There are urgent demands for more sophisticated techniques for understanding and analyzing the *behavioral characteristics* of network traffic. In this paper, we study the network traffic behaviors using *traffic activity graphs* (TAGs), which capture the interactions among hosts engaging in certain types of communications and their collective behavior. TAGs derived from real network traffic are large, sparse, yet seemingly complex and richly connected, therefore difficult to visualize and comprehend. In order to analyze and characterize these TAGs, we propose a novel *statistical traffic graph decomposition* technique based on *orthogonal nonnegative matrix tri-factorization* (tNMF) to decompose and extract the core host interaction patterns and other structural properties. Using the real network traffic traces, we demonstrate that our tNMF-based graph decomposition technique produces meaningful and *interpretable* results. It enables us to characterize and quantify the key structural properties of large and sparse TAGs associated with various applications, and study their formation and evolution.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations

General Terms

Measurement, Security

1. INTRODUCTION

Understanding and analyzing traffic characteristics are fundamental to the design, development and implementation of networks. The traditional emphasis of network traffic analysis has been on the statistical properties of traffic, leading to the important discoveries such as heavy-tails and long range dependence. As networking technologies continue to mature and evolve, and more sophisticated network applications are invented and deployed, operating, managing and securing networks have become increasingly challenging tasks, and require us to understand, analyze and model the *behavioral characteristics* of network traffic, such as communication patterns, interaction structures and trends of applications, users and other entities in the networks.

While traffic analysis for network security and management has been an active area of research, the majority of earlier work has focused on specific problems or aspects such as detecting heavy-hitters, identifying peer-to-peer (P2P) applications, and generating packet-level malware signatures (see, e.g., [1–3]) that are driven primarily by certain security application needs. There are relatively few studies which consider the traffic as a whole to extract general behavioral characteristics. Examples include individual (5-tuple) flow-level traffic clustering [4], aggregate PoP-level origin-destination (O-D) flow characterization and anomaly detection [5, 6], and host-level traffic behavior profiling using graph-theoretical [7] or information-theoretical [8] approaches.

In this paper, we study *network-wide communication patterns* among hosts that are engaging in certain types of communications or applications using *traffic activity graphs* (TAGs). In a TAG, nodes are IP addresses (hosts) and edges are observed flows that represent certain communications or interactions of interest among the IP addresses (hosts). Depending on the purpose of study, various criteria may be used to select flows of interest, and construct different TAGs that capture the relevant traffic activities among hosts under study. For example, using the NetFlow records collected at our campus border router, in this paper we model the communication patterns and interactions between hosts within our campus network and those outside hosts in the observed traffic using *bipartite* TAGs, where one set of nodes represent the inside hosts and another set of nodes represent the outside hosts, and edges between these two sets of hosts represent certain flows *selected based on ports that are associated with an application of interest*. We refer to these (bipartite) graphs as *application* TAGs. Examples of such TAGs include HTTP, Email, DNS, peer-to-peer (P2P), online chat and gaming applications.

In general, TAGs derived from real network data are large, sparse, seemingly complex and richly connected. For instance, when the number of nodes is large, nearly all of them contain so-called *giant connected components* (GCCs), which link together a majority of hosts (IP addresses) observed in the traffic, a phenomenon that has been observed in many social network studies. What is particularly interesting is the observation that TAGs associated with different applications exhibit *distinct* patterns and structures. These prop-

*The work is supported in part by the National Science Foundation grants CNS-0626808, CNS-0626812 and CRI 0709048.

SIGMETRICS/Performance'09, June 15–19, 2009, Seattle, WA, USA.

erties of application TAGs[1] are first observed and studied in [9], where the authors propose several graph-theoretical (average or distributional) metrics to help characterize and distinguish such graphs. While these metrics are useful in summarizing the overall statistical properties of the graphs, in general, they shed little light on how TAGs are formed and how they can be *meaningfully* interpreted.

In this paper we propose a novel *(statistical) graph decomposition* method based on *orthogonal nonnegative matrix tri-factorization* (tNMF) to analyze and extract the "core" host interaction patterns and other key structural properties of application TAGs. This technique is motivated by the observation that the matrix representations of application TAGs exhibit clear *block* structures, which suggest that they are composed of a number of *dense* sub-graphs representing "dominant" host groups (or "communities") with more intense interactions. In a sense, these dense subgraphs collectively form the "core" of the TAGs, capturing the most significant interactions among the dominant host groups. We formalize these observations and intuitions in the context of the proposed tNMF graph decomposition framework. More specifically, the tNMF method produces a *co-clustering* of the inside and outside hosts as well as a low-rank "core" matrix that represents the *overall interaction structure* among these groups and their *interaction intensities*. Each pair of inside and outside host groups with strong interactions corresponds to a *dense* (bipartite) subgraph in the original TAG and the bipartite (hyper)graph *induced* by the low-rank "core" matrix is referred to as the (core) *latent hypergraph* of the original TAG. In other words, the tNMF method *approximately decomposes* a TAG into a series of dense subgraph components and a (core) latent hypergraph representing inter-connection structures among the graph components.

Applying the tNMF method to various application TAGs (derived from our campus network datasets) such as HTTP, Email, DNS, P2P, online chat and gaming applications, we characterize and classify the typical structures of the resulting graph components and (core) latent hypergraphs. Through extensive experimental analyses, we demonstrate that the decomposition results not only capture the dominant structures in the original TAG, but also are amenable to meaningful interpretations. For instance, HTTP TAGs are largely formed by a series of star-like or mesh-structured dense graph components that are inter-connected primarily due to hosts appearing in multiple inside/outside host groups, but sometimes also through one inside/outside host group interacting with multiple outside/inside groups. The chat traffic graphs are formed by a series of much less dense subgraphs inter-connected by an overall star-like structure. In contrast, the P2P traffic graphs show more diverse structures, reflecting the diversity and complexity of P2P applications. Using these components and their structural properties, we also study the evolution of TAGs over time. Moreover, we also provide two examples to illustrate the potential utility of our tNMF method in practical network management tasks such as unknown application identification and suspicious/anomalous traffic activity (or application) detection. In summary, our tNMF-based framework provides an *easy-to-understand*, *interpretable* and *quantifiable* means to analyze and characterize key structural properties of large, sparse, complex and richly connected TAGs that are otherwise hard to visualize and comprehend.

The remainder of the paper is organized as follows. In Sec. 2, we present the overall characteristics of TAGs, in particular, their intrinsic block structures. We introduce our proposed tNMF graph decomposition method and address key practical issues in Sec. 3. In Sec. 4, we analyze, validate and interpret the decomposition results using real network traffic. We study the evolution of TAGs in Sec. 5, and illustrate the utility of the tNMF method in unknown application identification and anomaly detection in Sec. 6. Sec. 7 summarizes related work and Sec. 8 concludes the paper.

[1]In [9] where packet traces of relative short durations are used, these TAGs are referred to as *traffic dispersion graphs*.

2. TRAFFIC ACTIVITY GRAPHS

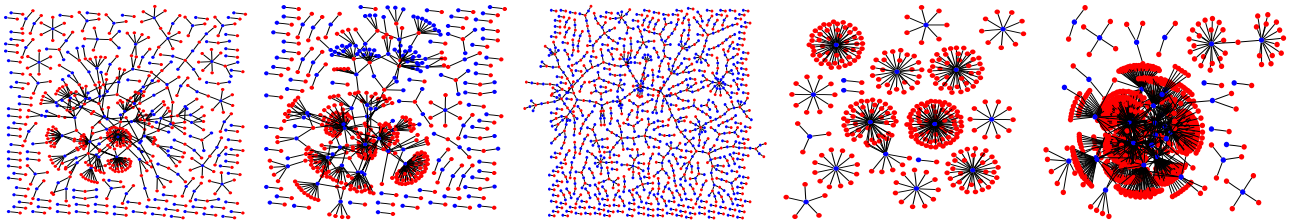
In this section, we introduce the *(bipartite) traffic activity graphs* (*TAGs*) defined in the context of the NetFlow data collected in our campus network and present some *visual* and *graph-theoretical* characteristics of such graphs. Further, using their matrix representations, we highlight the *block structures* inherent in the traffic activity graphs which motivate the statistical graph decomposition framework proposed in this paper.

Datasets. The primary datasets used in our study are non-sampled, Cisco NetFlow records from/to our campus network (with three class-B or /16 address blocks) to/from the rest of the Internet, collected at our campus border router over a month period. We also have access to several (tier-1) ISP datasets which contain *sampled* NetFlow records collected at various routers inside the ISP networks (One of the ISP datasets is used in Sec. 6 in our study of the Storm worm activities, as an example to illustrate the utility of our proposed tNMF decomposition method). Due to space limitation as well as for ease of exposition, we will introduce the notion of traffic activity graphs and present the proposed tNMF decomposition method in the context of our campus network datasets. Nonetheless, we remark that the proposed methodology and associated concepts are equally applicable to ISP datasets. Further, the overall observations and insights gained from our campus network datasets also hold for ISP datasets, although the specific results and their interpretations may vary.

2.1 Traffic Activity Graphs and Their Overall Characteristics

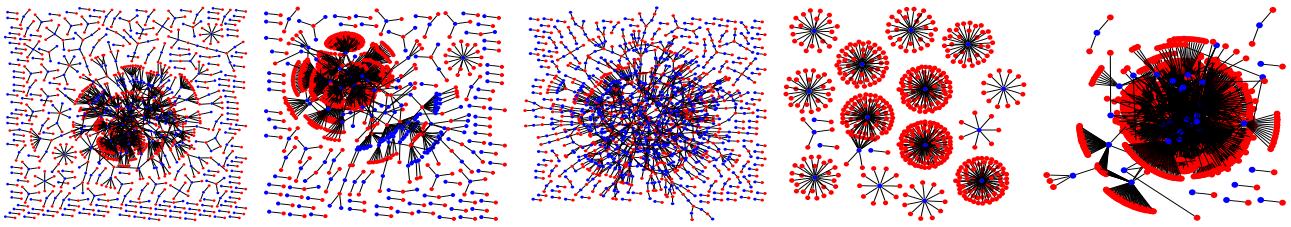
Using the campus network datasets, we introduce and define *application(-specific)* (or rather, *port-specific*) traffic activity graphs (TAGs) as follows. Given a set of *service ports* $\mathcal{P}$ associated with an application of interest (e.g., TCP ports 80 or 443 for Web applications[2]), let $\mathcal{F}$ be a collection of flows (observed during some time window) that use a port $p \in \mathcal{P}$ either in the source or destination port header field. The set of inside IP addresses (representing hosts inside our campus network, or inside hosts) and the set of outside IP addresses (outside hosts) which appear in $\mathcal{F}$ are denoted as $\mathcal{IH}$ and $\mathcal{OH}$, respectively (For the ISP datasets, we may refer to the set of subscribers as $\mathcal{IH}$ and the set of other Internet hosts as $\mathcal{OH}$. In addition, other application specific definition of $\mathcal{IH}$ and $\mathcal{OH}$ is also applicable, see Sec. 6). The ($\mathcal{P}$-specific) TAG $\mathcal{G} := \{\mathcal{V}, \mathcal{E}\}$ is a *bipartite* graph, with the vertex set $\mathcal{V}$ and edge set $\mathcal{E}$, where $\mathcal{V} = \mathcal{IH} \cup \mathcal{OH}$, and $e_{ij} \in \mathcal{E}$ if at least one flow from ih_i to oh_j exists in $\mathcal{F}$ (Depending on the purpose of analysis, a *weighted* version of such a graph can also be defined where the weight w_{ij} associated with the $e_{ij} \in \mathcal{E}$ represents, say, the number

[2]Depending on the applications and/or focus of the study, $\mathcal{P}$ may contain one or multiple ports. For instance, by considering TCP port 80 only, we focus on HTTP-only Web traffic, while including TCP port 443, we also include HTTPs traffic in the study. In some cases, ports in a given service port set $\mathcal{P}$ may be used by some "non-standard" applications other than the "well-known" application; for example, port 80 may be used by some P2P applications to penetrate firewalls. Since the majority of flows using port 80 are generated by Web applications/HTTP protocols, for convenience we refer to the TAGs derived from flows with ports 80 and 443 as "HTTP" TAGs. The same remark applies to other similarly named TAGs such as "Email" TAGs. Note also that unless otherwise stated in this paper, all ports refer to TCP ports.



(a) HTTP (b) Email (c) AOL Messenger (d) BitTorrent (e) DNS

Figure 1: Application TAGs with 1,000 flows: blue and red points denote inside and outside hosts, respectively



(a) HTTP (b) Email (c) AOL Messenger (d) BitTorrent (e) DNS

Figure 2: Application TAGs with 3,000 flows: blue and red points denote inside and outside hosts, respectively

of flows from ih_i to oh_j in $\mathcal{F}$). We remark here that in addition to application/port-specific TAGs defined above, other types of TAGs can also be defined, e.g., using other criteria for filtering and selecting flows from the (original) NetFlow datasets. Clearly what types of TAGs should be defined and used will depend on the purpose of study. For instance, as an application of the tNMF method to anomaly detection, in Sec. 6 we introduce two different types of TAGs to study the "anomalous" Storm worm activities.

Traffic activity graphs capture the *network-wide* communication and *interaction* patterns between inside and outside hosts of a network. They are primarily driven by the user activities or behaviors, moderated in part by the inherent "application structures" which determine how users or clients interact with other users or servers. Hence we would intuitively expect that different applications (e.g., client-server-based Web applications vs. P2P applications) may exhibit distinct graph structures or shapes. As first reported in [9], this is indeed the case. Using the Graphviz tool [10] (node radius = 0.1, edge weight = 2.0), in Fig.1 we present five representative application TAGs (derived from our campus network dataset that begins at 10:00am on 02/17/2006): HTTP (port 80 or 443), Email (port 25 or 993), AOL messenger (port 5190), BitTorrent (port 6881) and DNS (port 53, UDP). For clarity of graphing, here we only consider outgoing flows where the inside hosts are service requesters (accessing the specific service ports) and outside hosts are service providers (opening the service ports for access). The inside hosts (service requesters) are represented by *blue* dots, while the outside hosts are represented by *red* dots–hence the graphs are best viewed on a computer screen or a colored print-out. Fig. 1 shows the 5 example application TAGs using the *first 1000 flows*, while Fig. 2 using the *first 3000 flows*.

Clearly, these application TAGs display distinct shapes or structures. For example, HTTP and Email traffic graphs contain a number of more richly connected star-structures (centered either at an outside or inside host), while BitTorrent traffic graph contains a few (apparently isolated) dense star-structures centered at inside hosts only. In contrast, such structures disappear in the AOL messenger traffic graph, where all nodes are characterized with low degrees. Comparing the corresponding application tags in Fig. 1 and Fig. 2. we see that with more flows added to the graphs, the basic characteristics of these graphs appear to persist, with the core star-structures in the HTTP, Email, BitTorrent and DNS traffic graphs becoming denser. In all cases, some originally disconnected parts of the graphs start to connect and merge together–this phenomenon leads to the so-called *giant connected components* that we will discuss shortly.

Table 1 lists some key statistics[3] for a few selected application TAGs, each generated from a flowset of $|\mathcal{F}|$=10,000. More specifically, using the port(s) listed in the 2nd column of the table, we extract 10,000 *unique* flows containing the port(s) from the NetFlow dataset beginning at 10:00am on 02/17/2006 to generate the corresponding flowset $\mathcal{F}$ for each application TAG. The approximate duration spanned by the flows in each flowset is listed in the 3rd column[4]. The 4th ($|\mathcal{IH}| \times |\mathcal{OH}|$) column shows the number of nodes of the resulting bipartite traffic graphs derived from the flowsets. The *density* of the graphs, defined as $|\mathcal{E}|/(|\mathcal{IH}| \times |\mathcal{OH}|)$, is listed in the 5th column. We see that all of these graphs are extremely sparse. The next two columns list the average node degrees, $\bar{d}(ih)$ and $\bar{d}(oh)$, of the inside and outside hosts (IP addresses), respectively. A large $\bar{d}(ih)$ in general indicates the existence of popular (high-degree) inside hosts; P2P applications, such as BitTorrent, eMule and Gnutella, for example, have higher $\bar{d}(ih)$ values. In contrast, online chat applications, such as MSN Messenger and AOL Messenger, show no dominance of any host.

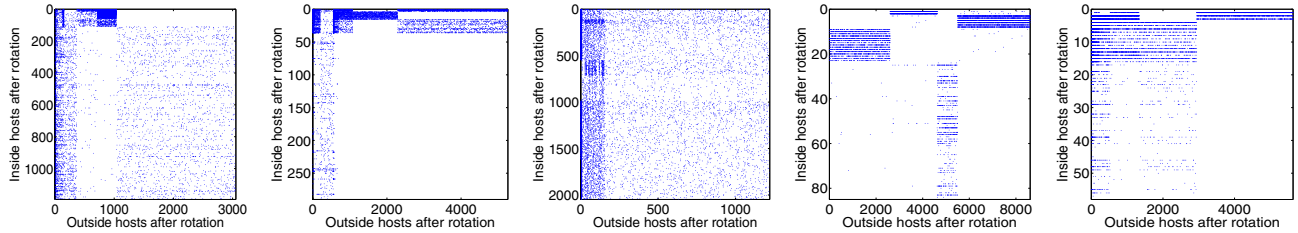
The most interesting characteristic is perhaps the existence of the so-called *giant connected component (GCC)* (i.e., the *largest connected component* in the TAG) that connects a majority of nodes in a TAG, when the number of flows (or the observation time interval) becomes sufficiently large. As illustrated in Figs. 1 and 2, when the number of flows is increased from 1000 to 3000, the (core) connected region in the HTTP, Email and AOL Messenger TAGs expands to connect more nodes and generally grows denser. From the 8th column in Table 1, for each application TAG with 10,000 flows (i.e., edges), the GCC connects 87.5% of all the nodes in the HTTP TAG, 94.1% in the Email TAG, 99.4% in the AOL Messenger TAG,

[3]Several other graph-theoretical metrics such as node degree distribution, joint degree distribution, depth, rich club connectivity and so forth are used in [9] to characterize and distinguish various TAGs. Due to space limitation, we exclude these statistics here.

[4]Note that since flows with ports 80 and 443 comprise the majority of the traffic, we have 10,000 flows within a time period of roughly 2.7 minutes. Likewise, for the Email we have 10,000 flows within a time period of roughly 58 minutes, whereas for other applications, up to 2 hours are needed to obtain 10,000 flows.

Table 1: Characteristics of different application TAGs using 10,000 unique flows

Type	Port(s)	Duration	$\|\mathcal{IH}\| \times \|\mathcal{OH}\|$	Density(10^{-3})	$d(ih)$	$d(oh)$	GCC(%)	GCC(inside×outside)	GCC(edges)
HTTP	80, 443	2.7mins	1193×3054	2.73	8.38	3.27	87.5	961×2756	9660
Email	25, 993	58mins	289×5262	6.58	34.60	1.9	94.1	111×5114	9790
AOL Msgr	5190	2.1hrs	2047×1221	4.00	4.89	8.19	99.4	2039×1209	9988
BitTorrent	6881	57mins	84×8610	13.83	103.76	1.01	89.6	35×7751	9013
DNS	53 (UDP)	2mins	57×5634	31.14	175.44	1.77	99.7	50×5626	9992
eMule	4662	81mins	33×9987	30.34	303.03	1	96.7	12×9690	9702
Gnutella	6346,6348	73mins	136×9760	7.53	73.53	1.02	94.5	55×9299	9538
MSN Msgr	1863	2.3hrs	1603×712	8.76	6.24	14.04	92.2	1562×572	9856



(a) HTTP (b) Email (c) AOL Messenger (d) BitTorrent (e) DNS

Figure 3: Block structures after rotating rows and columns of adjacency matrices using 10,000 flows

etc.. Even for the BitTorrent TAG which appears to be comprised of a few disconnected star structures (see Fig. 2), the largest connected component eventually connects 89.6% of all the nodes. The last two columns in Table 1 list the size (numbers of inside/outside hosts and edges) of the GCC for each application TAG.

2.2 Block Structures in TAGs

It is well known in random graph theory that for a fixed number of nodes, as the probability of (uniform) edge generation increases, a giant connected component emerges almost surely in such (uniformly generated) random graphs. On the other hand, the application TAGs (and their resulting GCCs) show high diversity and variability (e.g., as manifested by their degree distributions), suggesting that their formation is not purely random. In fact, these graphs show a strong cluster effect, or contain "latent structures" underlying the applications TAGs. We show the existence of such structures by using the matrix representation of the TAGs.

Given a bipartite TAG $\mathcal{G}$, we construct its adjacency matrix $A = [a_{ij}]$, where the rows and columns of A correspond to the hosts in $\mathcal{IH}$ and $\mathcal{OH}$, and $a_{ij} := 1$ if $e_{ij} \in \mathcal{E}$. While we know that A is a very sparse matrix (see Table 1), we permute the rows and columns of A to show that there exist "dense" blocks or sub-matrices in A. Fig. 3 presents the results for five example traffic graphs, where their corresponding adjacency matrices are displayed after we have selectively rotated their rows and columns. The block structures in the matrices are clearly visible, with certain areas far denser than others. The existence of dense vs. sparse blocks suggests that some groups of inside hosts tend to communicate or interact with certain groups of outside hosts, while rarely with other outside hosts. The block structures of A for different applications also show distinct patterns.

The existence of such latent structures is not surprising. Intuitively, they represent the underlying communication patterns or "social interactions" of users and services inside and outside our campus network. Such interactions lead to the formation of various "communities of interest" with distinct communication patterns or behaviors. For example, HTTP applications may provide different utilities, e.g., search engines, news services, blogs, photo or video sharing service, etc.. Requesters who are looking for a specific utility will connect to the providers of such a utility, and due to the role of search engines and "social influence," they are also more likely to connect to a few other popular providers. The service requesters and providers thus together form a distinct community in the HTTP traffic graph. The dense block structures shown in Fig. 3 also provide hints as to why and how the GCCs are formed. We see that for some inside groups, sometimes a subset of the group members communicate with more than one outside group (or a sub-group therein) and vice versa, resulting in various dense components to be connected with varying degrees of connectivity.

The block structures suggest that despite their sparsity, the application TAGs are composed largely of connected, dense sub-graphs that are not formed randomly, but represent certain latent host interaction patterns, or shared interests of various user communities. These observations and insights motivate us to identify and extract these "dense subgraphs" and the inside/outside host groups associated with them, so as to better understand and characterize network traffic graphs. In the remainder of the paper, we present a statistical graph decomposition technique based on *orthogonal nonnegative matrix tri-factorization*, referred to as *tNMF*, and discuss the experimental results.

3. GRAPH DECOMPOSITION USING TNMF

In this section we present a statistical graph decomposition technique based on *orthogonal nonnegative matrix tri-factorization* (*tNMF*), and apply it to extract dominant "graph structure components" in an application TAG. Each such component is a *dense* bi-partite subgraph consisting of a pair of inside host group and outside host group that are more strongly connected than any host not part of the inside/outside group. The collection of these subgraph components (together with their inter-connection) constitutes, in a sense, the "core" of the application TAG, capturing the dominant communication patterns or interaction structures between the inside and outside hosts.

3.1 The tNMF Method

Given an application TAG $\mathcal{G}$ representing the interaction patterns of m inside and n outside hosts, let $A_{m\times n}$ be the corresponding (binary) adjacency matrix A defined earlier. The problem of extracting the strongly connected subgraphs from $\mathcal{G}$, or equivalently the "dense" sub-matrices in A, can be formulated as a *co-clustering* problem where inside hosts and outside hosts are *jointly* clustered into k groups and l groups, respectively (in general $k, l \ll$

$\min(m,n)$). Each subgraph is now defined as a sub-matrix covered by a pair of inside/outside host groups, and the *dense* subgraphs can be identified from these $k \times l$ sub-matrices.

As illustrated in [11], this co-clustering problem can be formulated as an *orthogonal nonnegative matrix tri-factorization* (tNMF) problem: Given a nonnegative matrix $A_{m\times n}$, we factorize it (or more precisely, *approximately decompose* it) into three *low-rank nonnegative* matrices, $R_{m\times k}$, $H_{k\times l}$, and $C_{n\times l}$ so as to minimize the following objective function J subject to the orthogonality constraints on R and C:

$$\min_{R\geq 0, C\geq 0, H\geq 0} J(R,H,C) = ||A - RHC^T||_F^2 \quad s.t. R^TR = I \text{ and } C^TC = I, \tag{1}$$

where $||\cdot||_F$ is the Frobenius norm, and $k, l \ll \min(m,n)$. The NMF problem and its solution were first introduced in [12, 13] as an alternative approach to singular value decomposition (SVD) and other matrix decomposition methods (see Section 7 for a brief discussion on these and other related methods), and has been successfully applied in various machine learning applications [12, 14, 15]. The *orthogonal* constraints for R and C distinguish tNMF from other NMF algorithms and enable it to simultaneously cluster the rows and columns of a matrix [16].

The solution to the tNMF problem employs an iterative optimization procedure. We first initialize R, C and H to contain only positive random entries, and we then keep updating R, C and H using the following updating rules until the change in the relative square error (RSE), $RSE := ||A - RHC^T||_F^2/||A||_F^2$, falls below a predefined threshold θ, say, $\theta = 10^{-7}$,

$$\begin{aligned} R_{ip} &\leftarrow R_{ip}\frac{(ACH^T)_{ip}}{(RR^TACH^T)_{ip}}, \\ C_{jq} &\leftarrow C_{jq}\frac{(A^TRH)_{jq}}{(CC^TA^TRH)_{jq}}, \\ H_{pq} &\leftarrow H_{pq}\frac{(R^TAC)_{pq}}{(R^TRHC^TC)_{pq}}. \end{aligned} \tag{2}$$

It has been shown [11] that using the above updating rules, the RSE will monotonically decrease and converge to a local minimum. In Sec. 3.3 we will discuss several important practical issues in applying this tNMF method to decompose application TAGs.

3.2 Interpretation of tNMF Results

In the context of decomposing TAGs, we propose a novel interpretation of the tNMF decomposition results as follows. The orthogonal low-rank, nonnegative matrices R and C divide the rows and columns into k inside and l outside host groups, where $R_{\cdot p}$, $p = 1, \cdots, k$, and $C_{\cdot q}$, $q = 1, \cdots, l$, serve respectively as the "membership indicator" functions of the row groups and column groups. Since entries in R and C are nonnegative real numbers, this naturally gives rise to a *soft* co-clustering of the inside and outside hosts: R_{ip} (after row normalization) can be viewed as the "likelihood" of inside host i belonging to inside host group p, and C_{jq} the "likelihood" of outside host j belonging to outside host group q. R and C can also be used to construct a *hard* co-clustering where each inside/outside host is assigned to *at most one* inside/outside host group. For simplicity of exposition, this is the interpretation we will adopt in the remainder of this paper (although a soft clustering interpretation can also be used, which we leave as future work). In the following, we show how the hard clustering is constructed.

Using R and C, we define the inside/outside host group membership indicator matrices $\hat{R}$ and $\hat{C}$ as follows: $\hat{R}_{ip} = 1$ if $p = arg\max_j\{R_{ij} : R_{ij} > 0\}$, and 0 otherwise. In other words, we assign an inside host i to the inside host group p associated with the largest (nonzero) R_{ip} value. In particular, if all R_{ip}'s are zero, then host i is not assigned to any inside host group. In addition, when multiple groups are associated with the largest value, we randomly assign the host to one of these groups to resolve ties. The indicator matrix $\hat{C}$ is defined similarly. With $\hat{R}$ and $\hat{C}$ thus defined, we use IG_p to denote the pth inside host group, and OG_q the qth outside host group. Further, let $\mathcal{IG} := \{IG_1, \ldots, IG_k\}$ and $\mathcal{OG} := \{OG_1, \ldots, OG_l\}$ denote the collection of these inside and outside host groups.

We now introduce the *group density* matrix $\hat{H} = \{\hat{H}_{pq}\}$ where

$$\hat{H}_{pq} := \frac{(\hat{R}^TA\hat{C})_{pq}}{||\hat{R}_{\cdot p}||_1 \cdot ||\hat{C}_{\cdot q}||_1}, 1 \leq p \leq k, 1 \leq q \leq l,$$

and $||\cdot||_1$ is the L_1-norm. We see that $\hat{H}_{pq}$ is the density of the (bipartite) subgraph representing the interaction patterns between the members in IG_p and OG_q. A large $\hat{H}_{pq}$ value indicates a strongly connected bipartite subgraph, while a small or zero $\hat{H}_{pq}$ value suggests that only a few edges exist between some members of these two groups, or no edge at all. Using $\hat{H}$, we can identify and extract "dense" bi-partite subgraphs in the TAG $\mathcal{G}$. Formally, we say a bipartite subgraph $\mathcal{S}_{pq}$ in $\mathcal{G}$, where $\mathcal{S}_{pq} = \{[a_{ij}]|a_{ij} \in A, \hat{R}_{ip} = 1, \hat{C}_{jq} = 1, 1 \leq i \leq m, 1 \leq j \leq n\}$, is *dense* if $\hat{H}_{pq} \geq \delta$, for some appropriately chosen *density threshold* $\delta \in (0, 1]$, say, $\delta = 0.5$. These dense subgraphs in $\mathcal{G}$ thus represent dominant communication patterns among the inside/outside hosts with strong interaction structures. We refer to these dense subgraphs, $\mathcal{S}_{pq}$'s, of $\mathcal{G}$ as the *significant graph components*, or simply *graph components* of $\mathcal{G}$. We will analyze their structures and interpret their meanings in the context of various application TAGs in Sec. 4.

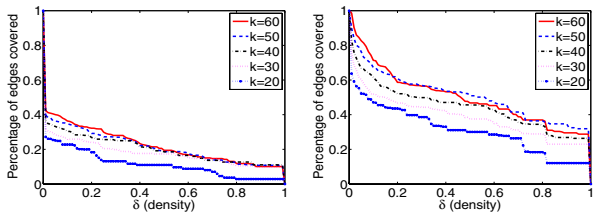
Furthermore, the group density matrix $\hat{H}$ induces a (weighted) bi-partite (hyper)graph $\mathcal{H} := \{\mathcal{IG} \cup \mathcal{OG}, \mathcal{E}_{\hat{H}}\}$, where the nodes represent the inside and outside groups, IG_p and OG_q, $1 \leq p \leq k$ and $1 \leq q \leq l$, and an edge $e_{pq} \in \mathcal{E}_{\hat{H}}$ if $\hat{H}_{pq} > 0$, and the weight associated with edge e_{pq} is exactly $\hat{H}_{pq}$. More generally, we can also define an unweighted (hyper)graph $\mathcal{H}_\delta$ where an edge $e_{pq} \in \mathcal{E}_{\hat{H}}$ *if and only if* $\hat{H}_{pq} > \delta$, i.e., if the density of the corresponding subgraph $\mathcal{S}_{pq}$ is at least δ. Hence, the induced hypergraph $\mathcal{H}$ represents the interaction patterns and their intensities between the various inside/outside host groups, and $\mathcal{H}_\delta$ captures the dominant interactions among the core host groups (or communities) of the inside/outside hosts. It is in this sense that we refer to the hypergraph $\mathcal{H}$ ($\mathcal{H}_\delta$) as the *(core) latent hypergraph* underlying (or generating) the original TAG $\mathcal{G}$. In particular, using $\mathcal{H}_\delta$ and the corresponding dense graph components $\mathcal{S}_{pq}$'s, we obtain an *approximate* "core" $\hat{\mathcal{G}}$ of the original $\mathcal{G}$ that captures the dominant interaction patterns among significant inside/outside host groups.

3.3 Practical Issues

We briefly discuss several key practical issues in applying the tNMF method to the (statistical) decomposition of application TAGs, and highlight the solutions we employ to: i) select the rank k and l and the density threshold δ, ii) improve the convergence rate and avoid local minima.

Selection of Rank and Density: Without loss of generality, we set $k = l$, which is an input parameter for the tNMF algorithm, and specifies an upper bound on the desired or expected groups formed by inside and outside hosts. As discussed earlier, the density threshold δ is a parameter for identifying dense subgraphs. Since the selection of appropriate k and δ depends on specific applications, in the context of TAG decomposition, our criteria for choosing k and δ are two-fold: to obtain *stable* graph components which contain

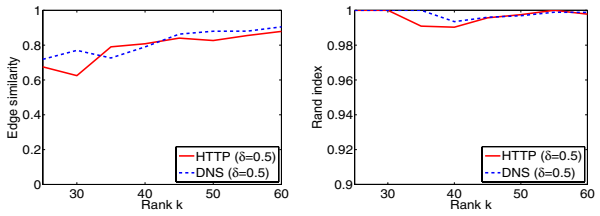
sufficient number of edges in the original TAG, i.e., with a good *edge coverage*.



(a) HTTP (b) DNS

Figure 4: Edge coverage for various k and δ.

We first study the edge coverage of extracted graph components by varying k and δ. Fig. 4(a)(b) show the edge coverage for HTTP and DNS graphs with different k and δ, respectively. In general, either increasing k or reducing δ will result in an increase of the edge coverage. However, we observe the number of covered edges converging when $\delta > 0.5$ and k exceeds 40 for HTTP and above 50 for DNS. This implies that when k exceeds a certain threshold, the "dense" subgraphs (with $\delta > 0.5$) become stable (similar observations are made in terms of other TAGs), hence we choose $\delta = 0.5$ in all the experiments for identifying significant subgraphs. We then apply a linear search of k's starting at 20 and with a small increment at a time. We note that different increments often lead to similar results since the dense subgraphs become persistent with a sufficiently large k (In case of $k \neq l$, we may search for the appropriate parameters by increasing k and l iteratively). To balance the accuracy and efficiency, we choose an increment of 5 in our experiments. We then use the two-way Kolmogorov-Smirnov goodness-of-fit test to compare edge coverage curves between consecutive k's. The null hypothesis is that two edge coverage curves are identical. We choose rank k if the KS test fails to reject the null hypothesis (i.e., the P-value is above 0.05). For example, we choose $k = 40$ for the HTTP TAG as in Fig. 4(a), since the P-value equals 0.0215 between $k = 35$ and $k = 40$, but the P-value becomes 0.3499 between $k = 40$ and $k = 45$. As another example, we choose $k = 45$ for the DNS graph (Fig. 4(b)) due to the P-value of 0.0001 between $k = 40$ and $k = 45$, and P-value of 0.0715 between $k = 45$ and $k = 50$.



(a) Edge similarity (b) Rand index

Figure 5: Similarity of graph components

So far, we have shown how to select k and δ to produce a good edge coverage of the original TAG. Now we investigate whether these edges lead to stable graph components. Let $\mathcal{E}_i$ and $\mathcal{E}_{i-1}$ be the extracted edge sets corresponding to k_i and k_{i-1}, respectively. We define the edge similarity as $(\mathcal{E}_i \cap \mathcal{E}_{i-1})/(\mathcal{E}_i \cup \mathcal{E}_{i-1})$. Fixing $\delta = 0.5$, we increase k by 5 at a time ($k_i = k_{i-1} + 5$). Fig. 5(a) shows the edge similarities (y-axis) associated with different k's (x-axis) for the HTTP and DNS graphs, respectively. We observe that the edge sets become more similar as k increases. For the chosen k ($k = 40$ for HTTP and $k = 45$ for DNS), the edge similarity is close to 85%. This again implies that as the k is sufficiently large, the extracted dense subgraphs become stable.

To evaluate the similarity of graph components formed by these extracted edge sets, we use Rand index as a measurement. Due to the high similarity of the edge sets, we compute the Rand index using the edges in $\mathcal{E}_i \cap \mathcal{E}_{i-1}$. Let $\mathcal{C}_i$ and $\mathcal{C}_{i-1}$ be the sets of components associated with edge set $\mathcal{E}_i$ and $\mathcal{E}_{i-1}$. The Rand index is defined as:

$$Rand(\mathcal{C}_i, \mathcal{C}_{i-1}) := (a + b)/\binom{n}{2}$$

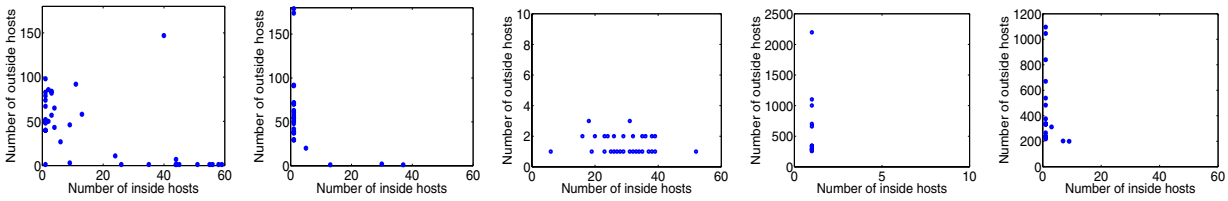
where a and b represent the number of edge pairs that are in the same or different cluster in $\mathcal{C}_i$ that are also in the same or different cluster in $\mathcal{C}_{i-1}$. The denominator is the total number of edge pairs and $n = |\mathcal{E}_i \cap \mathcal{E}_{i-1}|$. The Rand index ranges from 0 to 1, with higher value indicating more similar clustering results. Fig. 5(b) displays the Rand index corresponding to different k's for the HTTP and DNS graphs. The Rand index values are always close to 1, implying persistent dense clusters or subgraphs formed by these extracted edges for various k's.

Low Convergence Rate and Local Minima: Though the optimal solution of tNMF is unique, the random initialization of R, C and H in the basic tNMF optimization algorithm usually lead to both a low convergence rate and an unsatisfactory local minima solution. In addition, since the complexity of tNMF algorithm is $O(mnr)$, where m-by-n is the matrix size and r is the total number of iterations until convergence. Hence a low convergence rate results in a higher complexity of the algorithm. In this paper, we address this problem by employing a singular value decomposition (SVD) based initialization approach. The basic idea is to first apply the rank-k singular value decomposition (SVD) on A for spectral co-clustering [17]. We then project the rows onto the resulting k-dimensional subspace spanned by the top k (rows or columns) principal components, and perform k-mean clustering to obtain an initial clustering of inside/outside host clusters. We initialize R by perturbing the inside host cluster membership vectors to obtain an all-positive matrix, namely, setting $R := R + \epsilon$, where ϵ is a small positive constant to avoid zero entries. The initialization of C is done similarly. H is initialized with $R^{-1}AC^{-1^T}$, where R^{-1} and C^{-1} stand for the pseudo-inverse of R and C.

Through extensive experiment analysis, we find that our SVD-based initialization method not only improves the convergence rate significantly, but also enables our algorithm to find the "best" optimization solution. Using Email TAG as an example (with 100 experiments), the RSE using the SVD-based initialization is 0.34 ± 0.009, in comparison to 0.39 ± 0.027 using random matrix initialization. In addition, using the SVD-based initialization, the number of iterations to reach convergence is only 170.10 ± 71.09, while for the random matrix initialization is 323.57 ± 134.16. Hence, our SVD-based initialization method not only increases the approximation accuracy, but also enhances the speed of convergence.

4. RESULTS AND INTERPRETATIONS

We apply the tNMF method to various application TAGs derived from our NetFlow datasets to extract their core latent hypergraphs and associated significant graph components. In this section, we analyze the structures of these graph components and interpret their meanings. We also investigate how these graph components are connected to form the core latent hypergraphs. These decomposition results provide a meaningful and quantifiable way to understand, analyze and distinguish the structures of large traffic graphs that are otherwise hard to visualize and comprehend. In particular, it also sheds light on how the giant connected components (GCCs) of these graphs may be formed.



(a) HTTP (b) Email (c) AOL Messenger (d) BitTorrent (e) DNS

Figure 6: Sizes of significant graph components for five example application TAGs ($\delta = 0.5$).

Table 2: Graph components of various TAGs.

Type	$\|instar\|$	$\|outstar\|$	$\|bimesh\|$	Edge coverage
HTTP	10	9	15	19.2%
Email	23	2	2	23.5%
AOL Msgr.	0	27	17	17.7%
BitTorrent	12	0	0	77.9%
DNS	11	0	3	56.3%

4.1 Graph Component Structures

Applying the tNMF method to the application TAGs listed in Table 1, the sizes of the resulting (significant) graph components of each TAG are shown in Fig. 6. Due to space limitation, we only show the results of a representative TAG in each application category. In the figures, each point (x, y) represents a single graph component, where x is the number of inside hosts in each inside host group, and y is the number of outside hosts in each outside host group. The locations of points lead us to define three basic types of graph component structures. We refer to the graph components corresponding to the points on the line $x = 1$ (i.e., the inside host group containing only one inside host) as having an *in-star* structure, i.e., a star structure centered at an inside host. Similarly, we refer to the graph components corresponding to the points on the line $y = 1$ (i.e., the outside group containing only one outside host) as having an *out-star* structure, i.e., a star structure centered at an outside host.

Figure 7: A bi-mesh structure from the HTTP TAG.

The remaining points correspond to graph components which have at least two members in both its inside and outside host groups. We refer to them as having a *bi-mesh* structure, which represents fairly complex interactions or connectivity between the inside and outside hosts. An example of a bi-mesh structure is shown in Fig. 7. This bi-mesh structure consists of 24 inside hosts and 11 outside hosts, where inside hosts are represented by circles and outside hosts are denoted by squares. It contains more than 140 edges and most of the inside and outside hosts in their respective groups also have relatively high degrees, suggesting strong interactions between the members of these two inside/outside groups.

Table 2 summarizes the number of these three graph component structures for each of the TAGs. We see that different application TAGs exhibit great diversity in their graph component structures. For example, HTTP TAG contains a large number of bi-mesh structures and a few star structures. These bi-mesh structures may consist of hundreds of hosts as shown in the decomposition results of the HTTP TAG (Fig. 6(a)). In comparison, instant messaging application TAGs such as AOL messenger contain many out-star structures as well as a large number of relatively small bi-mesh structures, which usually consist of 2 outside hosts and 20 to 40 inside hosts. On the other hand, P2P application TAGs contain mostly in-star structures; the richness in connectivity of these TAGs seem to manifest in the "inter-connections" among the graph components, not within, unlike HTTP TAGs (see Section 4.3). The last column in Table 2 shows the percentage of edges covered by these extracted graph components in each TAG.

It is interesting to note that while TAGs are associated with vastly different applications (e.g., HTTP vs. P2P), most TAGs associated with the same or similar applications (e.g., various on-line chat applications) show very similar patterns. This observation suggests that the graph component structures capture the distinct characteristics of underlying application structures that determine how inside/outside hosts interact with each other.

4.2 Graph Component Interpretations

Using the IP addresses, their DNS names (if known) and other exogenous information sources (e.g., information on server Web sites), we have done extensive investigation to interpret and validate various graph components, namely, the inside/outside groups and their interactions. Due to space limitation, we only present a few examples. Recall that the TAGs in question are derived from outgoing flows originating from inside hosts of our campus network to outside hosts, where the inside hosts are "service requesters" while the outside hosts are "service providers." In other words, the port(s) used in identifying an "application" appear as destination ports in the flow records only.

HTTP. Due to the dominant volume of HTTP traffic and many activities associated with it, we observe a great variety of HTTP graph components representing different types of HTTP interactions. In HTTP TAGs, the in-stars and out-stars together account for 60% of all the components. The majority of the out-star structures are rooted at popular servers belonging to *Google*, *Yahoo*, etc., and the remaining are rooted at IP addresses belonging to CDN servers like *Akamai* or advertising sites like *DoubleClick*. Different from the out-star structures, the in-star structures tend to be rooted at IP addresses of NAT boxes, proxy servers and wireless access points.

In comparison to the star structures, the bi-mesh structures depict more sophisticated interactions between groups of service requesters and service providers. We are particularly interested in understanding the correlation of service providers that attract clients to access them simultaneously. Based on DNS names and other auxiliary information, we categorize various bi-mesh structures of HTTP TAGs derived from flow datasets at different times, and present their interpretations in Table 3. Because we rely heavily on external information such as DNS names, providing interpretation for all bi-mesh structures is not always achievable. In fact, we are able to explain 86.6% bi-mesh structures observed in an entire day. From Table 3, we conclude that the majority of bi-mesh structures in HTTP TAGs are formed due to three major reasons.

The first reason is the server farm effect (row 1), where servers belonging to a large Web site balance the workload by serving requests in turn or by only responding to requests for specific content. In this way, a client may establish connections with multiple server machines to complete one access to the Web site, and bi-mesh structures are formed by clients incidentally sharing several

Table 3: HTTP bi-mesh structure categorization.

Category description	Examples	Pct.
Server farms	Lycos, Yahoo, Google	13.0
Content delivery networks	LLNW, Akamai, SAVVIS, Level3	24.5
Advertising providers	DoubleClick, Advertising.com TribalFusion	14.8
News related	WashingtonPost, New York Times, Cnet	1.6
Media and photo sharing	ImageShack, tl4s2.com, casalmedia.com	4.9
Broadcasters of news and music	MusicMatch, Napster, BBC	3.3
Job-search related	monster.com, careersite.com	1.6
Online shopping related	Ebay, CostCo, Walmart	3.3
Social network related	FaceBook, MySpace	18.0
Blogs and review sharing	LiveJournal, xpc-mii.net	1.6
Unknown	-	13.4

servers. The second reason is correlated service providers. For example, Web sites often collaborate with CDN providers for fast content delivery (row 2), e.g., *Yahoo* with *Akamai*, and *Facebook* with *Limelight Networks*. As another example, Web sites correlate with advertisement delivery networks (row 3) like *DoubleClick* and *Advertising.com*. In both cases, when clients connect to a particular Web site, they will be redirected to hosts in CDN network for further service or they will retrieve ad contents from advertisement delivery networks automatically. Such server sharing behavior also leads to bi-mesh structures.

In both of the above cases, the formation of bi-mesh structures is determined by HTTP servers. However, the shared interest of clients is the third major cause of bi-mesh structures. For example, in Table 3 row 4 to row 10, we observe interest groups related to news, media and photo sharing, shopping and online social networks. This suggests that clients tend to access Web sites delivering similar types of content.

Email and DNS. We find that the Email TAG is decomposed mostly into in-stars corresponding to Email servers of the university or several "big" departments (e.g., CS, IT, Math, Medical School within our campus). The out-stars are mainly rooted at popular Email servers such as *Gmail*. In Fig. 2(b), one bi-mesh structure is caused by server relays for load balancing as in the HTTP case. The other interesting bi-mesh structure consists of inside Email servers belonging to some research labs and smaller departments, which talk to Email servers of a few academic institutions, and mail relays (some in Asia). We check them against DNS based blacklists [18], and find that two of the addresses are blacklisted and quite a few others belong to domains that no longer exist. Hence, we suspect that this bi-mesh may be formed due to Email spams. The DNS TAG looks similar to the Email graph, with a large number of in-stars and out-stars rooted at DNS servers. There are three bi-meshes, where the inside hosts consist of at least one DNS server along with a few Email servers (including our CS mail servers). These Email servers appear to be configured either to serve also as DNS servers, or perhaps more plausibly, to perform queries to outside DNS servers for reverse DNS look-ups to filter non-registered spam Email servers (as in the case of our CS Email servers).

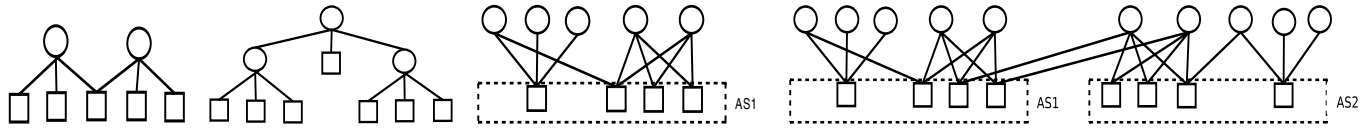
Instant Messaging and P2P Applications. The TAGs associated with Microsoft, Yahoo and AOL messengers have similar structures which are distinct from those of HTTP and Email. They are decomposed into mostly small-size bi-meshes with many cross-connections between them, indicating that members of inside groups communicate with members of multiple outside groups. All hosts in the outside groups are associated with the same (top two-level) domain name, meaning that these small-size bi-meshes are indeed the effect of server relays. In contrast, P2P applications such as BitTorrent, eMule and Gnutella contain a majority of in-star structures. A few of them are somewhat loosely connected, indicating that the inside hosts at which the in-stars are centered also happen to share a number of destination hosts.

4.3 Latent Hypergraphs and GCC Formation

The group density matrix $\hat{H}$ and the induced core latent hypergraph $\mathcal{H}$ capture the (dominant) interactions among various inside/outside host groups, and shed light on the formation of giant connected components of various application TAGs. Through detailed analysis of the latent hypergraphs (and the *reconstructed core graphs*) of various application TAGs using our campus flow datasets (as well as the ISP flow datasets), we find four (interconnection) structures that are most prevalent in the latent hypergraph structures. Most latent hypergraphs (and the resulting reconstructed core graphs) are formed predominantly using one or two of such structures. In Fig. 8(a-d), we provide a *schematic depiction* of these four typical structures that inter-connect two graph components. In the figure, we use circles and boxes to represent respectively inside hosts (service requesters or clients) and outside hosts (service providers or servers). An edge between two hosts indicates interactions between them (i.e., with observed flows between them). In the following, we provide some plausible interpretations of these four structures.

- *Randomly Connected Star Structure*, where a hypergraph is formed by various high degree in-stars rooted at inside hosts/clients randomly sharing leaf nodes (outside hosts/servers). P2P TAGs usually fall into this category.
- *Tree Structure*, where some star roots behave like clients to connect to other stars. This structure shows up typically in IRC, Email and DNS TAGs. We note that this structure does not appear in the hypergraphs of the application TAGs derived from our campus flow datasets, due to *limited visibility* (namely, we cannot observe the interactions among outside hosts); however, they do appear in application TAGs such as IRC, Email, and DNS TAGs obtained from the ISP flow datasets. For sake of completeness, we include this structure here.
- *Pool Structure*, where bi-meshs and out-stars are connected by a large number of low degree inside hosts/clients randomly communicating with the outside hosts/servers within these components. In addition, all the outside hosts/servers within these components share either the same (top two-level) domain name, or if their addresses are not DNS-resolvable, are associated with the same organization (based on the autonomous system number (ASN) using BGP routing data look-up). This seems to suggest that the pool structure is likely caused by server relays. Many application TAGs such as messengers and online games contain this type of structure.
- *Correlated Pool Structure*, where multiple pool structures are connected by multiple inside host/clients communicating with a number of outside hosts/servers in different pools. HTTP TAGs are a typical example of this category. For example, CDNs or on-line ad service networks form multiple pool structures, as they provide service to a large number of (sometimes unrelated) Web sites. Inside hosts/clients belonging to different groups that are accessing these Web sites will also access the corresponding CDNs or ad service networks, thus interconnecting multiple graph components.

Overall Summary. By decomposing various applications TAGs into significant graph components and core latent hypergraphs that are more amenable to analysis and interpretation, our tNMF-based



(a) Random connected star (b) Tree (c) Pool (d) Correlated pool

Figure 8: Schematic depiction of typical latent hypergraph formation.

TAG decomposition method provides a powerful and valuable tool to characterize, classify and understand the *structural* properties of large, sparse, yet richly connected TAGs that otherwise seem too complex to comprehend. Further, it reveals the underlying application structures that determine how users access services or interact with each other, and sheds light on the formation of such large and complex graphs. Based on our analysis of "canonical" graph components and latent hypergraph structures, we summarize that: i) For certain application TAGs such as Email, DNS and certain P2P applications, their main structures and rich connectivity can be decomposed into and captured by graph component structures with a diversity of "local" structures, some are simple (e.g., in-/out-star structures), others are complex (e.g., bi-mesh structures), which are inter-connected with relatively simple "global" structure, e.g., random star or tree structures; ii) However, for other applications such as online chat and game applications, the main structures and rich connectivity may be decomposed into and represented by the significant interactions (e.g., pool structures) of relatively simple graph component structures (e.g., mostly star-structures). For yet other applications (such as HTTP or Web), it is a combination of both components that contribute to their main structures and rich connectivity, for example, bi-mesh star structured graph components are inter-connected via correlated pool structures to form larger and more complex clusters, which may be then interconnected through random star structures. The structural complexity of HTTP or Web TAGs may not be too surprising in light that Web applications have evolved from the early simple client/server structure to today's "Web 2.0" driven by CDNs, search engines, on-line ad services, and Web services based on data centers and service-oriented architecture (SOA), forming a truly complex, dynamic and inter-weaving Web.

5. EVOLUTION OF TAGS

In the previous sections, we have studied the structural properties of various application TAGs as *static* objects: they are constructed by considering flows associated with an application of interest accumulated during a certain time window. Clearly traffic activities are dynamic; hence the resulting TAGs evolve over time. In this section we investigate the *temporal* properties of TAGs. Due to their prevalence and the rich structures, we use HTTP TAGs as an example to study the evolution of TAGs over time.

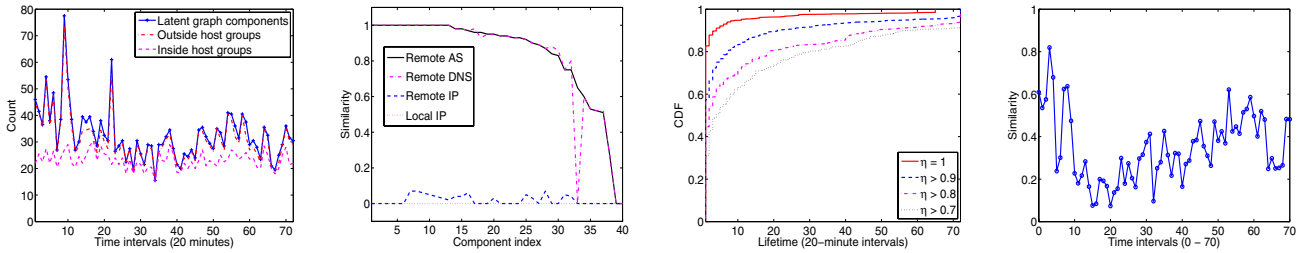
5.1 Metrics for Similarity Comparison

We take a one-day NetFlow dataset of our campus network and partition it into 20-minute intervals (72 in total). We construct a sequence of HTTP TAGs, $\mathcal{G}_t$, $1 \leq t \leq 72$, using the first 10,000 unique (outgoing) HTTP flows (with destination ports 80 or 443) observed during each time interval, and study their evolution over time. Intuitively, we would expect that for inside/outside host groups that represent dominant and frequent interaction patterns, while their individual members (in particular, inside hosts or clients) are likely to fluctuate and vary over time, the corresponding graph components as well as the core latent graph structures should stay fairly stable most of time. In order to compare the decomposition results (thereby the inside/outside host groups and their interaction structures) derived from HTTP TAGs over time, we first introduce several metrics.

Let $\mathcal{C}_s = \{C_i^s\}$ and $\mathcal{C}_t = \{C_j^t\}$ be the sets of significant graph components extracted from TAGs $\mathcal{G}_s$ and $\mathcal{G}_t$, $s < t$. Due to the dynamics of traffic activities and evolution of inherent "host community" structures (e.g., new members joining and old members leaving) as well as the artifact of the decomposition results, there is not necessarily a one-to-one correspondence between graph components in $\mathcal{C}_s$ and $\mathcal{C}_t$. For example, a graph component in $\mathcal{C}_s$ may be split or merged with other components in $\mathcal{C}_t$. In order to track the change of a particular graph component $C_i^s \in \mathcal{C}_s$ from time s to time t, we need to identify its (most likely) counterpart $C_j^t \in \mathcal{C}_t$. We adopt a simple "best-effort" matching algorithm as follows. Let $sim(C_i^s, C_j^t)$ denote an appropriately defined *similarity* metric for comparing components C_i^s and C_j^t at time intervals s and t. For each C_p^s, we say C_q^t is its counterpart (i.e., its "best match") if $q = arg\max_j sim(C_p^s, C_j^t)$ and $sim(C_p^s, C_q^t) \geq \eta$, where $\eta > 0$ is a pre-defined similarity threshold. If no such C_q^t is found, then C_p^s has no best match or counterpart at time interval t.

We introduce three similarity metrics to capture various relationships between two graph components (or their corresponding inside/outside host groups) over time. The *host-level similarity* (sim_h) is defined as the percentage of (inside or outside) hosts that C_i^s and C_j^t share in common, i.e., $sim_h(C_i^s, C_j^t) := |C_i^s \cap C_j^t| / |C_i^s \cup C_j^t|$. The *domain similarity* (sim_d) is defined as the percentage of hosts in two components that share the same domain name suffix (the top 3-level domain names for an address ending with country code and top 2-level domain names otherwise). Likewise, the *AS similarity* (sim_{as}) is defined as the percentage of hosts in two components that belong to the same AS. All three similarity metrics range between 0 and 1, with 1 indicating exactly identical components at the corresponding similarity level. Obviously, all inside hosts (local IPs) have the same domain name suffix and belong to the same AS owned by the university. Hence the last two similarity metrics are only useful in quantifying the similarity of outside host (remote IP) groups. However, the host-level similarity metric can be applied to both the inside and outside host groups associated with various graph components, yielding two similarity measures, one for the inside host (local IP) groups and one for the outside host (remote IP) groups.

Applying the tNMF decomposition to the HTTP TAGs $\mathcal{G}_t$, $1 \leq t \leq 72$, we obtain the graph components of each TAG. Fig. 9(a) displays the number of resulting graph components as well as the number of associated inside/outside host groups as a function of time, where $t = 0$ corresponds to the first 20 minutes in the 0th hour of the day. The figure shows that the number of graph components fluctuates and evolves over time. In particular, compared to business hours, the number of components during the wee hours of the morning tends to fluctuate more widely and are thus less stable. Using the similarity metrics defined above, we apply the best-effort matching algorithm to the graph components of two consecutive HTTP TAGs at 6:40pm and at 7:00pm, and find their best matches (for the purpose of exposition, we set $\eta = 0$). Fig. 9(b) shows the similarity scores of the graph components and their best matches, where the curves labeled "local IP" and "remote IP" are the host-



(a) Number of latent components at different time intervals

(b) Similarity of latent components at 6:40pm and 7:00pm

(c) Lifetime of latent components

(d) Average domain similarities at different times

Figure 9: Evolution of HTTP TAGs

level similarity scores of inside and outside host groups, respectively, while those labeled "remote DNS" and "remote AS" are the domain and AS similarity scores of outside host groups.

From Fig. 9(b) we see that the membership (generally clients) of inside host groups typically change frequently over time (all host similarities are equal to 0), which is not surprising. Outside hosts exhibit similar variability as that of inside hosts. However, because some of these outside hosts represent gateways (such as *Facebook* and *Myspace*), their similarity values are not as low as that of the inside hosts. Further, we notice that for most outside host groups, the new outside hosts tend to belong to the same domain or AS, indicating they provide the same/similar services or function in similar roles as before. Hence, for the HTTP TAGs derived from our campus network flow datasets, the outside host groups are good indicators for tracking the evolution of graph components over time. In addition, the two curves for the domain and AS similarity metrics are quite similar. We notice that for one of the domain name groups, the similarity values are close to zero while AS similarity is quite high ($sim_{as} = 0.65$). By investigating these domain names, we find that they indeed belong to the same AS, but are associated with different domain names, such as *questionmarket.com* and *Advertising.com*.

In the following we will use the domain similarity metric to study the temporal stability of graph components over time.

5.2 Temporal Stability of Graph Components

Given a graph component C_p^t observed during the time interval t, we say that it also appears at time s, $s \neq t$ and $1 \leq s \leq 72$, if its best match is C_q^s at time interval s such that $sim_d(C_p^t, C_q^s) \geq \eta$. (Note the domain similarity of two graph components is determined solely by the domain similarity of their associated outside host groups.) We define the *lifetime* of a graph component C as the number of time intervals that C appears in. With η ranging from 0.7 to 1, Fig. 9(c) plots the corresponding CDF of the lifetimes of various graph components observed during a one-day period. We see that even with $\eta = 1$, a few graph components appear in more than 65 time intervals, and with $\eta = 0.9$, about 10% of graph components appear in all 72 time intervals (whole day). Using $\eta = 0.9$, we say a graph component is *persistent* if it has a lifetime more than 6 hours (i.e., if it appears in at least 18 time intervals), otherwise it is referred to as *transient*.

Using the domain names of the outside host groups, we examine what constitutes the majority of persistent graph components. We find that a majority of them are associated with popular Web sites/services such as *Google*, *Yahoo*, *Facebook* as well as CDNs such as *LimeLight Network (LLNW)* and *Akamai*, where the outside host groups represent part of the server farms. Some of these persistent components contain also "correlated" servers/services, e.g., *Yahoo*, *DoubleClick* and *LLNW*. A few persistent components also represent groups of related Web sites such as *dictionary.com*, *thesaurus.com* and *lexico.com*, or *govideocodes.com* (video site) and *photobucket.com*, which appear to represent user interests. In other words, inside hosts that access one site are also likely to access the other sites in the (outside host) group. In contrast, most transient components seem to represent correlated Web sites, services and outside hosts that reflect user interests, some of which appear in multiple time intervals during the day, while others only appear in a short period of the day. In addition to some examples listed in Table 3, other examples include *cnet* (software news voice broadcast), *onvoy* (voice services) and *apple.com*; music services including *musicmatch*, *napster*, *moontaxi*, and *live365*; or travel-related services *grandex.com* and *weather.com*.

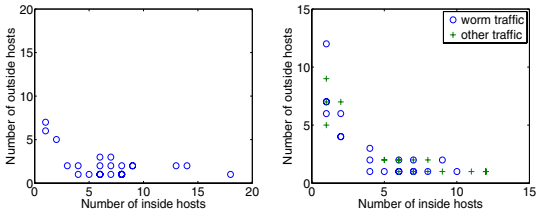
Furthermore, there is an implicit correlation between the "cohesiveness" of graph components and user interests and activities during different time periods of the day. In Fig. 9(d), we plot the average of the (domain) similarities between the graph components observed at time interval t with their "best matches" at $t+1$ ($\eta = 0$ is used for this purpose) as a function t. We see that between midnight and 2am or so ($t = 0$ to $t = 8$), the average similarities of the graph components are generally higher than other times. We find that an overwhelming majority of the graph components that appear during these periods, are associated with popular media sharing sites and other common Web services such as *Google*, *Yahoo*, *Microsoft* and *AOL*. Examining the inside hosts associated with these graph components reveal that most of them come from the residential hall subnets. Thus graph components during these time periods reflect activities and interests of residential hall students. During the business hours and evening ($t = 30$ to $t = 60$), many graph components are associated with common Web services, e.g., news, weather, etc., which appear to reflect dominant interests of users during those periods. On the other hand, during the wee hours of the morning, the traffic activities are much lower, and graph components appear to be more mixed: more (outside) service groups show up, each attracting roughly similar number of users (inside hosts), without any type of services dominating. The inside hosts associated with these graph components are also more diverse, including hosts in residential subnets, departmental machines, mail servers and other servers that appear to be running automated and scheduled processes, and the corresponding outside hosts vary from academic institutions to news sites and government agencies. Because traffic activities are less intense and more diverse, the graph components extracted by the tNMF method tend to be less cohesive, resulting in lower similarities among the graph components during two consecutive time periods. This also helps explain why we see a large number of graph components appearing during some of these time periods and they also tend to be less stable (see Figs. 9(a)).

6. APPLICATIONS

In previous sections, we have analyzed the typical structures of (significant) graph components and (core) latent hypergraphs pro-

duced by our tNMF-based TAG decomposition method, as well as how they evolve over time. In this section, we demonstrate how these analyses of graph components and hypergraph structures can be employed to identify, classify and understand "unknown" applications and their structures by using two examples.

The tNMF-based graph decomposition method not only helps us understand the structural properties of application TAGs associated with *known* applications (or service ports), but these structural properties can also be applied to facilitate "unknown" application identification as well as to analyze "anomalous" behaviors in known/unknown application TAGs. We briefly illustrate these two applications of the tNMF method via two examples. Due to space limitation, the full exploration of these topics and other applications of the tNMF method will be left to another paper.



(a) Port 4000 (b) Storm worm

Figure 10: Example applications using tNMF

Analysis of UDP Port 4000 Traffic. As an example of "unknown" application identification, we apply the tNMF method to the TAG formed by outgoing flows towards UDP port 4000 (i.e., as the destination port in the flows) within a certain time window in our campus flow datasets. Given limited information of application(s) running on port 4000, we decompose the TAG and analyze the structures of the resulting graph components and latent hypergraphs. Fig. 10(a) shows the size of extracted graph components, which contains 13 bi-meshs, 2 in-stars and 15 out-stars (some are overlapping in the figure). These components are connected to form an approximate pool structure. Further investigation shows these destination IP addresses belong to the same AS in China, indicating this TAG is likely associated with a messenger or game type of application. Googling these destination addresses reveals that they are associated with a messenger software (OICQ) that mainly uses UDP 4000 as the service port.

Analysis of Storm Worm Traffic. Storm worm is now a notorious and well-studied giant botnet in which bots communicate with each other through a P2P network (Overnet). It first appeared in late 2006 or early 2007. The ISP flow datasets were collected in early summer of 2007, during which the Storm worm is known to be highly active. As an example to illustrate the utility of the tNMF method, we apply it to analyze the structural properties of Storm worm TAGs and investigate how they may deviate from those of "normal" P2P applications. For this purpose, we have obtained a list of "known" bot addresses culled from the P2P queries to/from a Storm worm bot captured in a honeynet, and used this list to extract all flows in the ISP flow datasets that contain the IP addresses (either as source or destination) on the list. Note that unlike application TAGs discussed earlier, here, we ignore the port information when extracting the flowset. We construct two TAGs[5] from the resulting flow set: one TAG is constructed using flows containing both source and destination IP addresses on the list, thus it represents the communications among bots themselves (referred to as "worm traffic" in Fig. 10(b)); the other TAG is constructed using flows between bots (i.e., those IP addresses on the list) and "non-bot" hosts (IP addresses not on the list), thus it represents the communications between bots and non-bots (referred to as "other traffic" in Fig. 10(b)).

Applying the tNMF method to these two TAGs associated with the Storm worm, we obtain the graph components which are shown in Fig. 10(b). The "bot communication" TAG contains 8 bi-meshs out of 22 components ("o" points in the figure). The more common appearance of bi-mesh structures distinguishes it from other "normal" P2P networks where only randomly connected in-star structures are observed[6]. This indicates that the Storm worm bots tend to communicate more frequently with each other than peers in other "normal" P2P applications. We provide one plausible explanation for this structural difference or *anomaly*: the Storm worm botnet has a hierarchical structure (see [19]), where bots acquire commands from the botmaster through a set of supernodes. The role of the P2P communications between Storm worm bots is to query for the addresses of the supernodes. Hence the bi-meshes are likely due to the bots periodically sending queries to a few bots that store the addresses of the supernodes. In other words, the P2P communications in the Storm worm botnet are of certain mutual interest. This is in contrast to the host behaviors in most "normal" P2P applications, where users search for and share content in a "random" fashion. The "bots communicating with non-bots" TAG ("+" points in Fig. 10(b)) also contains a significant percentage of bi-mesh structures (7/20). Examining the DNS names and other relevant information (e.g., via *Google*) associated with non-bot IP addresses and ports used in the communication, we find many of the non-bots are (or function as) mail or http servers (perhaps functioning as distributed supernodes to provide proxies for communication between bots and the botmaster). The large number of bi-mesh structures reveals that Storm worm bots tend to exhibit somewhat correlated behaviors, either "collaborating" in accessing mail relays or http servers, or engaging in other "coordinated" malicious activities. Most of the communications with non-bot hosts seem to be Email spam activities.

7. RELATED WORK

Analysis of complex network graphs has recently received considerable attention in the literature, mostly due to the success of online social network applications. Many approaches have been proposed to help directly visualize complex graphs, e.g., [20] and [21]. These methods enable us to directly visualize and understand complex graphs; however, they do not provide us a way for characterizing and quantifying different graphs.

Various properties of complex network graphs have been studied. In particular, the community structures in network graphs attract the majority of research interest. Newman et al. [22] surveyed widely applied methods in physics and social sciences to extract community structures from complex graphs. Recently, a lot of work in computer science focuses on analyzing community structures from Web data [23] and social network data [24]. Our

[5]In the first (Storm worm communication) TAG, we construct a bipartite graph by putting source IP addresses on one side (rows in the adjaceny matrix) and destination IP addresses on the other side (columns in the adjaceny matrix). The resulting graph is not *strictly* bipartite, but nearly so, as there are only 10 (0.1%) IP addresses that appear as both source and destination in the flows. In the second (bots communicating with non-bots) TAG, a bipartite graph is constructed with bot IP addresses on one side and non-bot IP addresses on the other side.

[6]That bi-mesh structures are generally rare in "normal" P2P TAGs is likely due to the random peer selection method used by many P2P applications. Hence the probability of two P2P hosts sharing many peers repeatedly is typically very small in a short time window.

method differs from these approaches in that we focus on interpreting application-specific traffic activity graphs which are richly connected due to both user interests and service correlations.

One related work on host-level communities is [25], where historical communication patterns are used as "normal" profiles for preventing propagation of malwares. In contrast, we are not only interested in characterizing application specific communication patterns, but also in explaining the formation of these communities.

Our work is also related to other matrix factorization algorithms which can potentially provide graph partitioning or co-clustering results. For example, the SVD-based spectral graph partitioning algorithm, introduced in [17], has been proved to provide optimal bi-partitioning. However, the assumption of diagonal Σ matrix forces each inside host group to only communicate with one outside group, which does not describe the rich interactions among host groups. In addition, information-theoretic co-clustering methods perform simultaneous clustering over rows and columns of a specific matrix. For example, [26] obtains such co-clustering by minimizing the loss of mutual information between a low-rank approximation and the original matrix, while [27] achieves this goal by minimizing the codelength for the original matrix after rotating its rows and columns. However, these algorithms performs worse when the data is noisy, which is the case in our study.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we have studied the network traffic behaviors using traffic activity graphs (TAGs) associated with various types of communications. These TAGs are large, sparse, seemingly complex and richly connected, making them hard to visualize and comprehend as a whole. Based on the observation of prevalent block structures in such TAGs, we proposed the tNMF method for decomposing TAGs, and devised a systematic method for extracting the dominant substructures and characterizing their structural properties. We applied our method to various application TAGs derived from our campus NetFlow datasets such as HTTP, Email, DNS, various chat, P2P and online gaming applications. Through extensive experimental analyses, we demonstrated that the tNMF graph decomposition method provides an easy-to-understand, interpretable and quantifiable means to characterize and quantify the key structural properties of various TAGs as well as to study their formation and evolution. As examples to illustrate the utility of the proposed tNMF method, we also briefly touched on how they can be used for unknown application identification and anomalous traffic activity detection. These topics are part of our on-going research that will be reported in detail in another paper. Other future work includes designing community-preserving sampling algorithm and low complexity tNMF optimization algorithm to enhance the scalability of tNMF decomposition method.

9. REFERENCES

[1] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund. Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications. In *Proc. of ACM IMC*, 2004.

[2] T. Karagiannis, A. Broido, M. Faloutsos, and K. claffy. Transport layer identification of p2p traffic. In *Proc. of ACM IMC*, 2004.

[3] J. Newsome, B. Karp, and D. Song. Polygraph: automatically generating signatures for polymorphic worms. *Proc. of Security and Privacy, IEEE Symposium*, 2005.

[4] A. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *Proc. of ACM SIGMETRICS*, 2005.

[5] A. Lakhina, M. Crovella, and C. Diot. Characterization of network-wide anomalies in traffic flows. In *Proc. of ACM IMC*, 2004.

[6] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. Kolaczyk, and N. Taft. Structural analysis of network traffic flows. In *Proc. of ACM SIGMETRICS*, 2004.

[7] T. Karagiannis, K. Papagiannaki and M. Faloutsos. BLINC: Multilevel traffic classification in the dark. In *Proc. of ACM SIGCOMM*, August 2005.

[8] K. Xu, Z.-L. Zhang and S. Bhattacharyya. Profiling Internet backbone traffic: behavior models and applications. In *Proc. of ACM SIGCOMM*, August 2005.

[9] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese. Network monitoring using traffic dispersion graphs (tdgs). In *Proc. of ACM IMC*, 2007.

[10] Graphviz. http://www.graphviz.org/.

[11] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proc. of ACM KDD*, 2006.

[12] D. Lee and H. Seung. Learning the parts of objects. by non-negative matrix factorization. In *Nature*, 1999.

[13] D. Lee and H. Seung. Algorithms for non-negative matrix factorization. In *Proc. of NIPS*, 2000.

[14] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proc. of SIGIR*, 2003.

[15] H. Kim and H. Park. Extracting unrecognized gene relationships from the biomedical literature via matrix factorizations using a priori knowledge of gene relationships. In *Proc. of ACM TMBIO*, 2006.

[16] T. Li and C. Ding. The relationships among various nonnegative matrix factorization methods for clustering. In *Proc. of IEEE ICDM*, 2006.

[17] I. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proc. of KDD*, 2001.

[18] MX Toolbox Blacklists. http://www.mxtoolbox.com/blacklists.aspx.

[19] J. Stewart. Inside the storm: Protocols and encryption of the storm botnet. http://www.blackhat.com/presentations/bh-usa-08/Stewart/BH_US_08_Stewart_Protocols_of_the_Storm.pdf.

[20] Y. Jia and J. Hoberock and M. Garland and J. Hart. On the visualization of social and other scale-free networks. In *Proc. of IEEE InfoVis*, 2008.

[21] X. Yang and S. Asur and S. Parthasarathy and S. Mehta. A visual-analytic toolkit for dynamic interaction graphs. In *Proc. of ACM SIGKDD*, 2008.

[22] M. E. J. Newman. Detecting community structure in networks. In *Eur. Phys. J. B 38, 321-330*, 2004.

[23] Y. Dourisboure, F. Geraci, and M. Pellegrini. Extraction and classification of dense communities in the web. In *Proc. of WWW*, 2007.

[24] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Statistical properties of community structure in large social and information networks. In *Proc. of WWW*, 2008.

[25] P. McDaniel, S. Sen, O. Spatscheck, J. Van der Merwe, B. Aiello, and C. Kalmanek. Enterprise security: a community of interest based approach. In *Proc. of NDSS*, 2006.

[26] I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. In *Proc. of ACM KDD*, 2003.

[27] D. Chakrabarti, S. Papadimitriou, D. Modha, and C. Faloutsos. Fully automatic cross-associations. In *Proc. of ACM KDD*, 2004.