Structural Graph Indexing for Mining Complex Networks

Hakan Kardeş and Mehmet Hadi Güneş
Department of Computer Science and Engineering, University of Nevada, Reno
Email: {hkardes, mgunes}@cse.unr.edu

Abstract—**Systems such as proteins, chemical compounds, and the Internet are being modeled as complex networks to identify local and global characteristics of the system. In many instances, these graphs are very large in size presenting challenges in their analysis. Hence, graph indexing techniques are developed to enhance various graph mining algorithms. In this paper, we propose a new Structural Graph Indexing (SGI) technique that does not limit the number of nodes in indexing to provide an alternative tool for graph mining algorithms. As indexing feature, we use common graph structures, namely, star, complete bipartite, triangle and clique, that frequently appear in protein, chemical compound, and Internet graphs. Note that, SGI lists all substructures matching structure formulations and other graph structures can be identified and added to the SGI.**

Keywords: complex networks, graph indexing, graph mining, structural graph indexing

I. Introduction

Many systems can be modeled as a complex network to understand local and global characteristics of the system. Studying network models of systems provides a new direction towards understanding biological, chemical, technological or social systems in a better way. In many cases, systems under investigation are very large and the corresponding graphs have large number of nodes/edges requiring graph mining techniques to derive information from the graph. Several graph mining techniques have been developed to extract useful information from graph representation and analyze various features of complex networks [7]. In order to speed up graph queries, usually an index of the graph is derived according to some predefined index features.

Graph indexing is often utilized by graph search algorithms that look for a sub-graph within a graph database. For example, given a graph database G={g_1, g_2, ..., g_n} and a subgraph s, we are interested in identifying all graphs g_i that contain the subgraph s. This query is shown to be NP-complete [9] and becomes challenging as the size of graphs increase. For example, a typical graph of router-level Internet consists of millions of nodes making it impractical to perform many operations on the whole graph. In such cases, graph indexing allows operations to be more efficient.

In this paper, we propose a new structural indexing approach to provide an alternative tool for graph mining algorithms. For indexing, we specify a set of common graph structures such as star, complete bipartite, triangle and clique. These structures are ubiquitous in biological, chemical, technological, and social networks. In order to reduce computational complexities, we index these structures within the original graph in a consecutive manner. We first identify star structures, and then the complete-bipartite, triangle and clique structures from the preceding ones. Our approach, different from previous ones, does not limit the size of subgraph considered in indexing. However, it may be limited as maximum clique search are NP-complete [9].

In the remainder of the paper, we first detail structural graph indexing in Section II. Then, we present an evaluation of the approach on Internet topologies in Section III. Finally, we conclude in Section IV.

II. Structural Graph Indexing

In this section, we first present existing graph indexing methods and then present our structural graph indexing approach.

A. Related Work

The need for mining large graphs in an efficient manner increases as researchers look into new complex networks. Several studies have been carried out to make graph mining in an efficient manner using indexing techniques [11], [12], [14], [19], [22]. Many of the tools have constraints that limit the number of nodes/edges in index graphs or are not capable of operating on very large graphs. The graph indexing studies can be mainly categorized into two categories, namely, *path-based* and *structure-based* approaches.

Path-based graph indexing approaches use path expressions as indexing features such as GraphGrep [19] and Daylight [11]. GraphGrep enumerates all paths in the graph up to the length $maxL$. Then, it looks for each g_i whether it contains all paths up to $MaxL$ for a graph query q_i. A significant feature of path-based approaches is that paths can be manipulated easier than general graphs. However, as Yan et. al. indicated, path is a simple structure loosing structural information of a graph, and hence false positive ratio of path-based methods would be very high [22]. In addition, the number of paths in a graph database increases exponentially making path-based methods impractical for very large graphs.

Alternatively, structure-based graph indexing approaches identifies subgraphs to be indexed as in gIndex [22]. gIndex first searches for the frequent subgraphs in the graph, then indexes these frequent structures. An issue in this case is that frequent subgraph discovery increases complexity and exponential number of frequent fragments may exist under

```
Let $G = (V, E)$; $S \leftarrow \phi$;
for (each $node\ v \in V$)
    $S \leftarrow S \cup s_{(v,\phi)}$
for (each $edge\ e(a,b) \in E$)
    $s_{(a,ns)} \leftarrow s_{(a,(ns \cup \{b\}))}$
    $s_{(b,ns)} \leftarrow s_{(b,(ns \cup \{a\}))}$
for (each $s_{(v,ns)} \in S$)
    if $|ns| < 2$
    $S \leftarrow S - s_{(v,ns)}$
```

Fig. 2. Alg.1 - Star Structure Indexing

low frequency support. Therefore, in their study, they limit the number of nodes and index frequent structures up to 10 nodes.

In this paper, we propose an alternative structural indexing approach to search and process queries efficiently even in very large graphs. As indexing features, we use commonly observed graph structures: star, complete bipartite, triangle and clique. An important feature of these structures is that each one is comprised from the previous one where clique contains complete bipartite structures and complete bipartite contains star structures.

B. Structure Models

In structural indexing, we index predefined structures that are commonly observed in complex networks. In particular, we index star, complete bipartite, triangle and clique structures (shown in Figure 1) in a given graph $G = (V, E)$. An important difference of our approach from the previous studies is that we do not limit the size of subgraph considered in indexing. We index all maximal graphs that match the structure formulation. For instance, a maximal clique is a clique that cannot be extended by adding one more vertex from the graph. However, the substructure size in indexing may be limited when needed since maximal clique search is known to be NP-complete [9]. In order to reduce computational complexities, we index the structures within the original graph in a consecutive manner. That is, we first identify star structures, and then the complete-bipartite, triangle and clique structures from the preceding ones as detailed below.

Structure 1: Star ($K_{1,n}$)

We first index the star structure where a node has multiple neighbors as shown in Figure 1-(a) and (b). All star structures within a graph $G = (V, E)$ are represented as $s(v_i, ns_i)$ where $v_i \in V$ and ns_i is the set of all neighbors of v_i. We index maximal star structures for each node using the algorithm in Figure 2. The algorithm first builds a star structure $s_{(v,\phi)}$ for each node v without any neighbors. Then, for each edge $e(a, b)$, it appends neighbor sets of nodes a and b to the other one. Finally, the algorithm removes star structures $s_{(v,ns)}$ that have less than two neighbors.

Structure 2: Complete Bipartite ($K_{m,n}$)

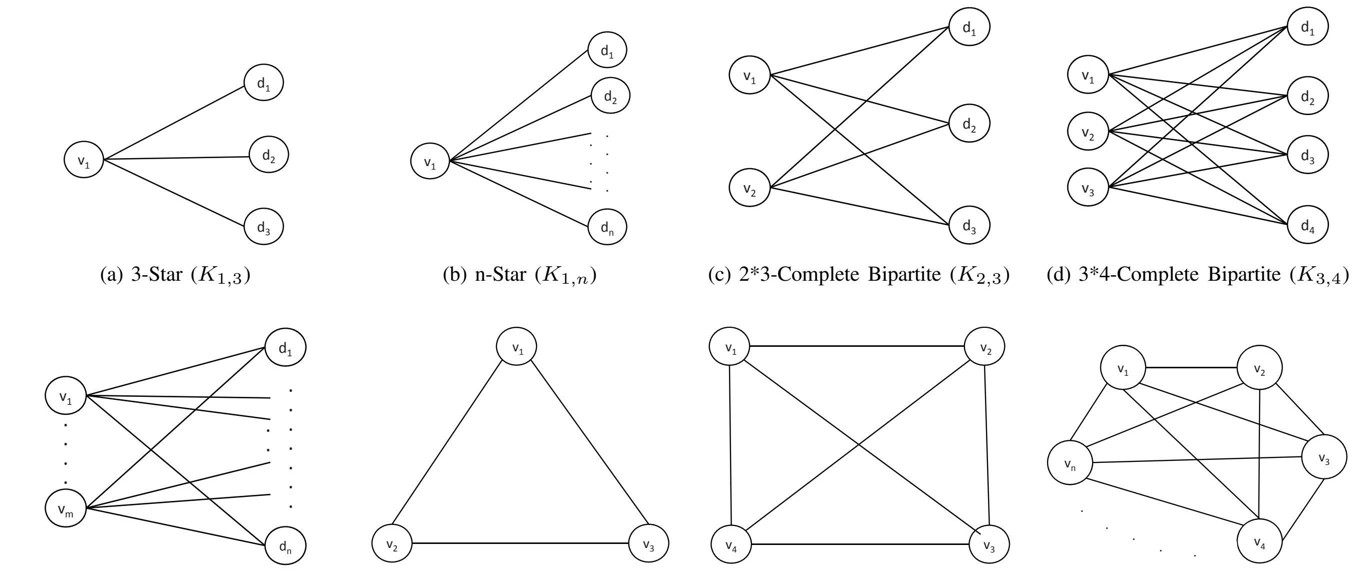
The second structure we index is complete bipartite, shown in Figure 1-(c), (d) and (e). A complete bipartite graph $G = (V_1 \cup V_2, E)$ is a bipartite graph such that V_1 and V_2 are two distinct sets and for any two vertices $v_i \in V_1$ and $v_j \in V_2$, then there is an edge between them (i.e., $\exists\ e^*_{(v_i,v_j)} \in E$). The complete bipartite graph with partitions of size $|V1| = m$ and $|V2| = n$ is denoted as $K_(m, n)$. Note that, star structure is a special case of a bipartite graph (not necessarily complete) where $m = 1$. Moreover, finding complete bipartite subgraph $K_{(m,n)}$ with maximal number of edges $m.n$ is an NP-complete problem [15].

```
INPUT: $S$ from Alg.1 in Figure 2
Let $G = (V, E)$; $K \leftarrow \phi$
for (each $s_{(a,ns)} \in S$)
    $L_{can} \leftarrow \phi$
    for (each $b_i \in ns$)
        $L_{can} \leftarrow L_{can} \cup ns^*$ where ($\exists\ s_{(b_i,ns^*)} \in S$)
    $L_{can} \leftarrow L_{can} - \{a\}$
    $R_{can} \leftarrow ns$

    for (each $v_i \in L_{can}$)
        $R_{new} \leftarrow R_{can} \cap ns_i^+$ where ($\exists\ s_{(v_i,ns_i^+)} \in S$)
        if ($|R_{new}| \geq 2$)
            $L_{new} \leftarrow \{a\} \cup \{v_i\}$
            for (each $v_j \in L_{can}$)
                if ($R_{new} \subset ns_j^\#$) where ($\exists\ s_{(v_j,ns_j^\#)} \in S$)
                    $L_{new} \leftarrow L_{new} \cup \{v_j\}$
            $K \leftarrow K \cup k_{(L_{new},R_{new})}$
```

Fig. 3. Alg.2 - Complete Bipartite Structure Indexing

We index all complete bipartite structures in the graph G using indexed star structures as in Figure 3. In the algorithm, for each star structure $s_{(a,ns)}$ where ns is the neighbor set of the node a, we identify the maximal complete bipartite involving the node a. For this purpose, we find second hop neighbors of a by iterating over the ns set and unifying them under L_{can} set that indicates candidates for the left side of the complete bipartite while the ns set is the candidate set for the right hand side. Then, we first find a $K_{2,n}$ and then grow it to $K_{m,n}$. In finding $K_{2,n}$, we iterate over each candidate node in the L_{can} and determine the neighbor intersection with a. If the intersection set is larger than two, then these nodes belong to the right hand side. In the second step, we grow the $K_{2,n}$ by finding all nodes in the left hand side (i.e., L_{can}) that has the right hand side nodes (i.e., R_{new}) as a neighbor.

Complete bipartite structure is ubiquitous in many complex networks. For example, [4] examines the structure of protein-protein interaction networks and showed that the graph of all protein-protein interactions is made up of complete bipartite structures containing two disjoint sets of nodes in which each node in one set is connected to every node in the other set. Additionally, bipartite graphs are very relevant in the Internet, as such subgraphs correspond to nodes that have identical sets of neighbors. Studying those nodes would be of particular interest to understand the structure of the Internet as these are duplicated structures in the graph [8].



(a) 3-Star ($K_{1,3}$) (b) n-Star ($K_{1,n}$) (c) 2*3-Complete Bipartite ($K_{2,3}$) (d) 3*4-Complete Bipartite ($K_{3,4}$)

(e) m*n-Complete Bipartite ($K_{m,n}$) (f) 3-Clique (K_3) (Triangle) (g) 4-Clique (K_4) (h) n-Clique (K_n)

Fig. 1. Structural Models

```
INPUT: $S$ from Alg.1 in Figure 2
Let $G = (V, E)$; $T \leftarrow \phi$
for (each $s_{(a,ns)} \in S$)
  for (each ($\{b, c\} \in ns$)
    if ($\exists\ e_{(b,c)}$)
      $T \leftarrow T \cup t_{(a,b,c)}$
```

Fig. 4. Alg.3 - Triangle Structure Indexing

Structure 3: Triangle (K_3)

Third, we index the triangle structure which is a clique structure of three nodes as shown in Figure 1-(f). We index all triangles in the graph by iterating over the star structures as in Figure 4. In the algorithm, for each star structure $s_{(a,ns)}$, we pick a pair of nodes b and c from the ns set and determine whether they have an edge $e_{(b,c)} \in E$.

Structure 4: Clique (K_n)

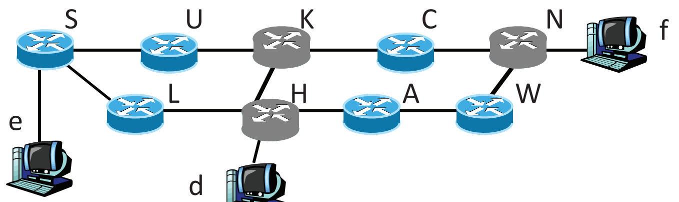
Finally, we index clique structures shown in Figure 1-(g) and (h). A clique in graph $G = (V, E)$ is a subset of the vertex set (i.e., $C \subseteq V$) such that there are edges between all node pairs (i.e., $\forall(c_i, c_j) \in C, \exists e_{(c_i,c_j)} \in E$, when $i \neq j$).

We index all maximal clique structures (that has more than three nodes) in the graph using complete bipartite structures as in Figure 5. We first get the set of nodes from each complete bipartite $k_{(m,n)}$ and look for cliques that are formed by those nodes. Note that, any clique larger than three nodes in the graph G will be indexed as multiple bipartite structures. Hence, we do not need to consider all nodes in the graph when indexing maximal clique structures. The clique search algorithm works recursively on each node from the $k_{(m,n)}$ as the pivot node in the $L1$ set and considers other nodes as candidate nodes in the $L2$ set. The function, moves each node from the $L2$ set to the $L1$ set if it is connected to all nodes in the $L1$ and then recursively tries to grow the structure with remaining nodes as candidates. When there are no more candidates to consider in $L2$ set then a clique has been identified. Note that, this algorithm is not optimal and better solutions for finding all cliques are proposed in [5], [17].

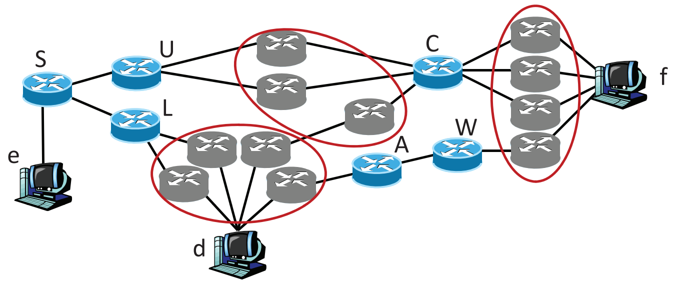
```
INPUT: $K$ from Alg.2 in Figure 3
Let $G = (V, E)$; $C \leftarrow \phi$
for (each $k_{(m,n)} \in K$ )
  for (each $a \in k_{(m,n)}$)
  findCliques($\{a\}, k_{(m,n)} - \{a\}$)

FUNCTION findCliques(L1, L2)
  if ($|L2| = 0$) and ($|L1| > 3$)
    $C \leftarrow C \cup c_{(L1)}$
  else
    for (each $b \in L2$)
      if ($\exists e_{(b,v)}\ \forall v \in L1$)
        $L^* \leftarrow (L2 \cap ns_i)$ where ($s_{(b,ns_i)} \in S$)
        findCliques($L1 \cup b, L^*$)
```

Fig. 5. Alg.4: Clique Structure Indexing

This structure has been observed in many fields. For example, in computational biology many problems can be solved by finding maximal or all cliques within the graph. Similarly, [18] models protein structure prediction as a problem of finding cliques in a graph whose vertices represent positions of subunits of the protein; [6] finds a hierarchical partition of an electronic circuit into smaller subunits using cliques; and [16] uses cliques to describe chemicals in a chemical database that have a high degree of similarity with a target structure.



(a) Sampled Network



(b) Resulting Network

Fig. 6. Topology Collection

III. Evaluation on Internet Topologies

In this section, we use router level Internet topologies to analyze the structural graph indexing approach.

A. Preliminaries

The need for accurate Internet topology samples has increased over the last decade. Sample maps provide an understanding of the global topology helps developing protocols and applications that can adapt to the underlying network. Furthermore, router-level Internet maps are useful to analyze the topological characteristics of the Internet and to design topology generators that can produce Internet-like synthetic network topologies to be used in various simulation studies. The confidentiality of Internet Service Provider topologies necessitates the research community to use other ways to collect Internet topologies. Several research groups and institutions have developed various tools and methodologies [1], [13], [3], [20], [21] to collect the required topology information from the Internet. These measurement studies utilize the Internet debugging tool, traceroute, or its variants to collect a large number of path traces from a set of vantage points.

Nonetheless, constructing a router level Internet Topology map using traceroute is not a straightforward process. Some routers don't respond to the traceroute probes and are called anonymous routers. They are denoted by a '*', instead of an IP address, in a traceroute output. An anonymous router may appear in several path traces. Each occurrence of '*' needs to be treated as a potentially different router. Therefore, there is a need to identify anonymous nodes that are caused by the same router to be able to accurately construct the underlying network.

For instance, Figure 6 presents the effect of anonymous routers on a sampled network. In the topology in Figure 6-(a), we assume that routers H, K and N are unresponsive to measurement probes and are called anonymous routers. When we run traceroute queries between vantage points, represented as d, e, and f in Figure 6-(a), collected path traces are as follows

```
d - * - L - S - e
d - * - A - W - * - f
e - S - L - * - d
e - S - U - * - C - f
f - * - C - * - * - d
f - * - C - * - U - S - e
```

The collected path traces, will generate a topology as shown in Figure 6-(b). The resulting topology in Figure 6-(b) is considerably different than the underlying topology in Figure 6-(a) indicating the significance of anonymous router resolution task to obtain more accurate sample topologies.

Even a small number of anonymous routers may considerable alter observed topology from the sampled network [10]. Based on the number of anonymous routers in the topology and the way of topology collection, there may be huge number of '*' in the collected set of path traces. For example, daily Internet topologies collected by iPlane infrastructure have more than 12M anonymous nodes along with 400K known interfaces [2].

In [10], we determine different cases under which anonymous nodes appear. We identified that anonymous routers cause parallel, clique-like, bipartite-like, and star structures in observed topology. We assume that IP alias resolution process is completed on the data set beforehand. Then, we propose a graph based induction method to handle anonymous routers within collected path traces. The structural graph indexing proposed in this paper can facilitate the graph based induction approach in resolving anonymous routers.

B. SGI on Anonymous Router Resolution

In this section, we illustrate how our method can be applied to the router level Internet Topology to resolve anonymous routers.

Graph Transformation

In our experiments, we use iPlane datasets [13]. In these datasets, there are two types of nodes, namely known (i.e., the ones with an IP address) and unknown (i.e., anonymous). When there is an anonymous router, it will appear as a '*' in the traceroute output. If multiple path traces pass through an anonymous router between routers with known IP address, there will be multiple parallel *-substrings between these known nodes. As an example, in Figure 6-a, traceroute queries from e to f will return path traces including *-substrings as

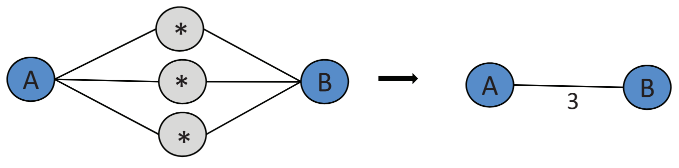
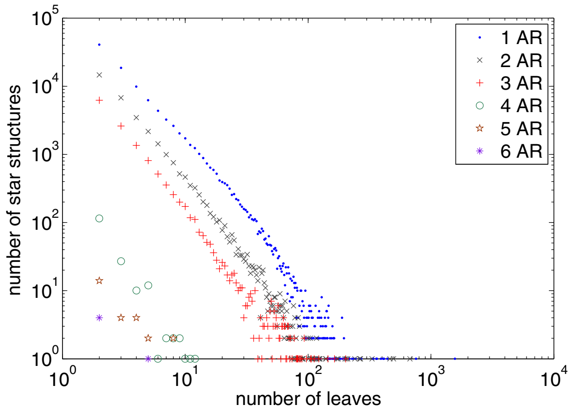


Fig. 7. Sample Transformation

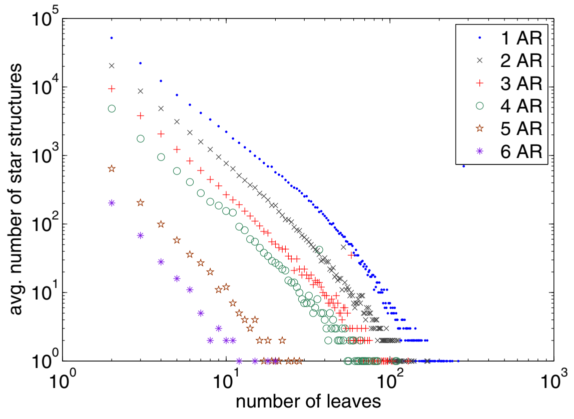
$(U, *_1, C)$ and $(U, *_2, C)$ respectively. This may then result in two parallel *-substrings between U and C in the resulting topology map as shown in Figure 6-b. In the example, there is a *-substring that includes only one anonymous router. A similar pattern can be observed for *-substrings of larger lengths in a typical data.

In order to resolve this type of anonymous routers, we need to detect the same *-substrings (i.e., same length *-substrings with the same known nodes at the end points) [10]. In our method, we ignore all nodes which don't have any anonymous neighbor, since they don't have any effect for the resolution process. While reading the traces from iPlane database, we identify the anonymous routers which are in between two known nodes. We read all *-substrings from the database and construct a new graph $\bar{G} = (\bar{V}, \bar{E})$. We represent each *-substring as an edge e(a,b,l) where a is the first known node, b is the second known node and l is the label of the edge representing the number of anonymous nodes between a and b as in Figure 7. We add each $e(a, b, l)$ only once to our new graph $\bar{G}$. We add a and b to $\bar{V}$. This process can be called as initial pruning(IP). After this step, we sequentially index star, complete bipartite, triangle and clique structures in graph $\bar{G}$ with SGI algorithm.

Structure Statistics

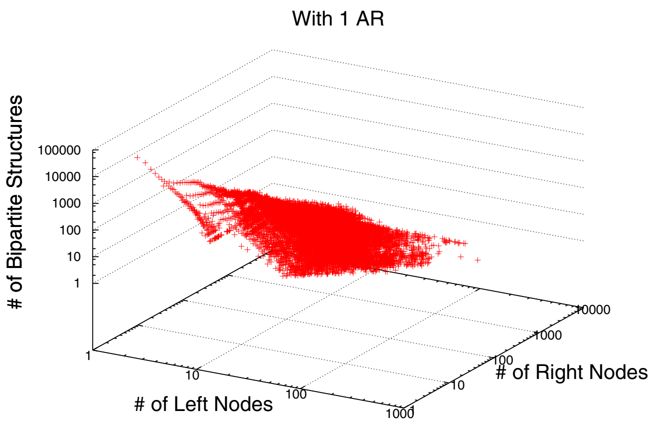


(a) 2006

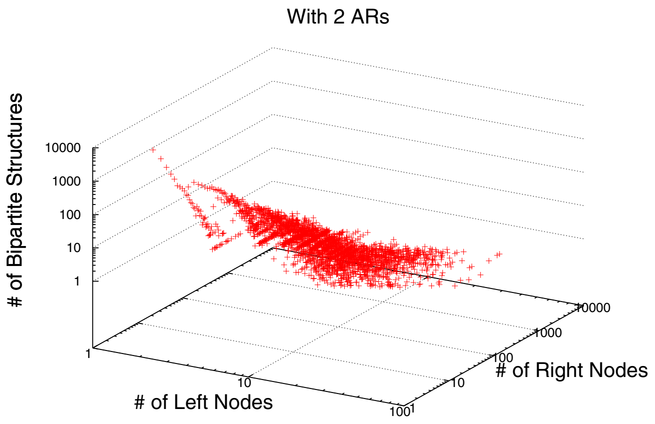


(b) Average

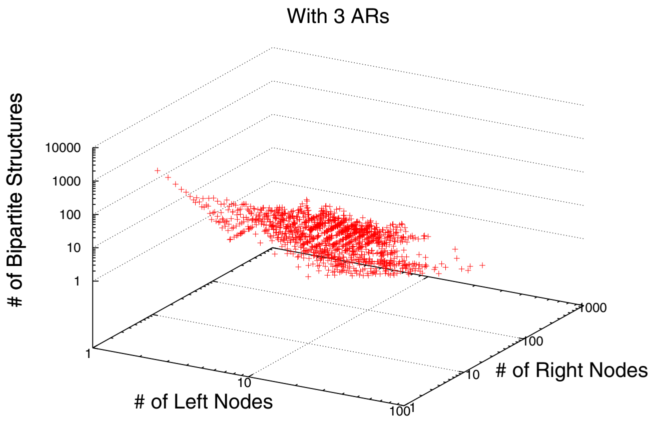
Fig. 8. Star Structures



(a) 2006 (1 anonymous node)



(b) 2006 (2 anonymous nodes)

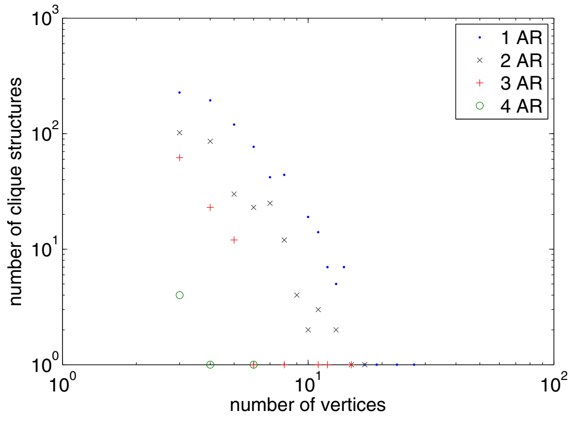


(b) 2006 (3 anonymous nodes)

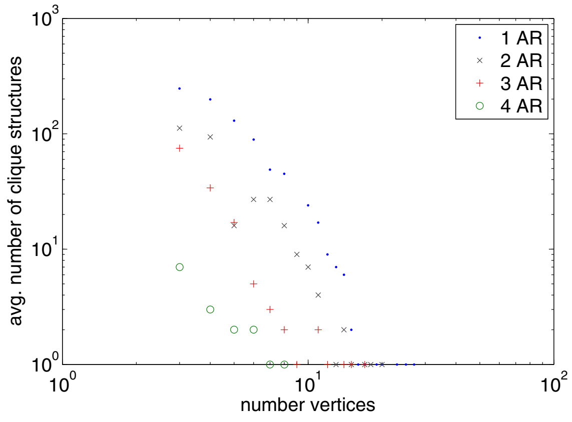
Fig. 9. Complete Bipartite Structures

We present the indexing results for a dataset collected in 2006 and the average indexing results of 6 different datasets collected between 2006-2009 in Figures 8, 9 and 10. For the dataset collected in 2006, we have 7,043,618 anonymous nodes and 172,532 known nodes.

When the length of *-substring increases, the number of



(a) 2006



(b) Average

Fig. 10. Clique Structures

structures found within the topology decreases. We present only the number of structures with less than 7 anonymous nodes as for higher lengths the number of structures found is almost zero. According to our experiments, while the number of structures with 1 anonymous node has not changed considerably in consecutive years, there has been a significant increase in the number of structures with 3 and more anonymous routers.

Resolution Results

We have 7,043,618 anonymous nodes in initial data. We first resolve the anonymous nodes between two known nodes. Then, we use SGI algorithm to resolve anonymous nodes. Starting from the maximum clique, we resolve all anonymous nodes within the clique and triangle structures. Then, we resolve anonymous nodes within the complete bipartite and star structures. The number of resolved anonymous nodes at each step given in Table I. Using SGI, we resolve more than 98 percent of anonymous nodes.

IV. Conclusion

In this paper, we presented the Structural Graph Indexing (SGI) for efficiently mining complex networks. As indexing feature, we utilize graph structures such as star, complete bipartite, triangle and clique that frequently appear in protein, chemical compound, and Internet graphs. SGI lists all substructures matching structure formulations and other graph structures can be identified and added to the SGI. Different from previous approaches, SGI does not limit the number of nodes in the indexing structure and provides an alternative tool for graph mining algorithms. In evaluation of SGI, we performed experiments on genuine Internet topology datasets to resolve anonymous routers.

	#Anony. nodes	#Resolved
IP	7,043,618	6,769,486
Clique	274,132	628
Bipartite	273,504	92,121
Star	181,383	78,214
Final	103,169	6,940,449

TABLE I
SGI for Resolving Anonymous Routers in iPlane Data-set

References

[1] *Archipelago Measurement Infrastructure*. http://www.caida.org/projects/ark.

[2] *iPlane*. http://iplane.cs.washington.edu/.

[3] *iToM: Internet Topology Mapper*. http://itom.utdallas.edu.

[4] N. M. A. Thomas, R. Cannings and C. Cannings. On the structure of protein-protein interaction networks. *Biochem. Soc. Trans.*, 31:1491 – 1496, 2003.

[5] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, 1973.

[6] J. Cong and M. Smith. A parallel bottom-up clustering algorithm with applications to circuit partitioning in vlsi design. In *DAC '93: Proceedings of the 30th international Design Automation Conference*, pages 755–760, New York, NY, USA, 1993. ACM.

[7] D. Cook and L. Holder. *Mining graph data*. John Wiley & Sons, 2006.

[8] D. Fay, H. Haddadi, A. Thomason, A. W. Moore, R. Mortier, A. Jamakovic, S. Uhlig, and M. Rio. Ieee/acm transactions on networking 1 weighted spectral distribution for internet topology analysis: Theory and applications.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[10] M. Gunes and K. Sarac. Resolving anonymous routers in Internet topology measurement studies. In *IEEE INFOCOM*, Phoenix, AZ, April 2007.

[11] C. A. James, D. Weininger, and J. Delany. Daylight theory manual daylight version 4.82. daylight chemical information systems, 2003.

[12] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Exploiting local similarity for indexing paths in graph-structured data. In *In ICDE*, pages 129–140, 2002.

[13] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *OSDI*, November 2006.

[14] T. Milo and D. Suciu. Index structures for path expressions, 1997.

[15] R. Peeters. The maximum edge biclique problem is np-complete. *Discrete Applied Mathematics*, 131(3):651 – 654, 2003.

[16] N. Rhodes, P. Willett, A. Calvet, J. B. Dunbar, and C. Humblet. Clip: similarity searching of 3d databases using clique detection. *J Chem Inf Comput Sci.*, 43(2):443 – 448, 2003.

[17] J. M. Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7(3):425 – 440, 1986.

[18] R. Samudrala and J. Moult. A graph-theoretic algorithm for comparative modeling of protein structure. *Journal of Molecular Biology*, 279(1):287 – 302, 1998.

[19] D. Shasha, J. T. L. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *In Symposium on Principles of Database Systems*, pages 39–52, 2002.

[20] Y. Shavitt and E. Shir. DIMES: Let the Internet measure itself. *SIGCOMM Comput. Commun. Rev.*, 35(5):71–74, 2005.

[21] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies using rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, February 2004.

[22] X. Yan, P. S. Yu, and J. Han. Graph indexing: A frequent structure-based approach, 2004.