Visualization of Large Networks with Min-cut Plots, A-plots and R-MAT [⋆,⋆⋆]

Deepayan Chakrabarti [a] Christos Faloutsos [b] Yiping Zhan [c]

[a] *Research Scientist, Yahoo! Research, 701 1st Ave, Sunnyvale, CA 94089.*

[b] *Professor, School of Computer Science, CMU, Pittsburgh, PA 15213.*

[c] *Work done while a Ph.D. candidate in the Dept. of Biological Sciences and School of Computer Science, CMU, Pittsburgh, PA 15213.*

Abstract

What does a 'normal' computer (or social) network look like? How can we spot 'abnormal' sub-networks in the Internet, or web graph? The answer to such questions is vital for outlier detection (terrorist networks, or illegal money-laundering rings), forecasting, and simulations ("how will a computer virus spread?").

The heart of the problem is finding the properties of real graphs that seem to persist over multiple disciplines. We list such patterns and "laws", including the "min-cut plots" discovered by us. This is the first part of our NetMine package: given any large graph, it provides visual feedback about these patterns; any significant deviations from the expected patterns can thus be immediately flagged by the user as abnormalities in the graph. The second part of NetMine is the A-plots tool for visualizing the adjacency matrix of the graph in innovative new ways, again to find outliers. Third, NetMine contains the R-MAT (Recursive MATrix) graph generator, which can successfully model many of the patterns found in real-world graphs and quickly generate realistic graphs, capturing the essence of each graph in only a few parameters. We present results on multiple, large real graphs, where we show the effectiveness of our approach.

Key words: Min-cut plots, A-plots, R-MAT, abnormal subgraphs, graph generator

⋆ Previous versions of this material were presented in SIAM Data Mining 2004 and in the SIAM DM 2004 Workshop on Link Analysis, Counter-terrorism and Privacy.

⋆⋆This material is based upon work supported by the National Science Foundation under Grants No. IIS-0083148, IIS-0113089, IIS-0209107 IIS-0205224 INT-0318547 SENSOR-0329549 EF-0331657IIS-0326322 CNS-0433540 by the Pennsylvania Infrastructure Technology Alliance (PITA) Grant No. 22-901-0001. Additional funding was provided by donations from Intel, and by a gift from Northrop-Grumman

1 Introduction

Graphs and networks have attracted significant interest recently, due to their ability to model many distinct real-world datasets under one intuitive framework. For example, the World Wide Web of webpages connected by hyperlinks, the Internet of computers connected by routers, social networks of individuals, protein interaction networks, and many other datasets can be easily expressed as a graph of *nodes* (webpages, computers, individuals, etc.) connected by *edges* (hyperlinks, routing links, etc.) The question we must ask is: *How can we visualize the information contained in such large graphs efficiently?*

Laying out the graph on a computer screen quickly becomes a hard problem as the graph size grows to millions (or even billions) of nodes of edges (such as the WWW graph). One solution is to look for *patterns* or motifs that appear in most real-world graphs, and indeed, many surprising regularities and "laws" have recently been discovered. The World Wide Web, the Internet topology and Peer-to-Peer networks follows surprising power-laws (Broder et al., 2000; Faloutsos et al., 1999; Barabási, 2002), exhibit strange "bow-tie" or "jellyfish" structures (Broder et al., 2000; Tauro et al., 2001), while still having a small diameter (Albert and Barabási, 2002). Finding patterns, laws and regularities in large real networks has numerous applications, from criminology and law enforcement (Chen et al., 2003) to analyzing virus propagation patterns (Pastor-Satorras and Vespignani, 2001) and understanding networks of regulatory genes and interacting proteins (Barabási, 2002) and so on. A method for visualizing common patterns can immediately tell the user how the given graphs matches "expectations," and whether there are any abnormalities that merit further attention.

To find better patterns (and, concurrently, better anomaly detection algorithms), we need to model real-world graphs well. In other words, we can develop a good graph *generator*, which can create synthetic yet "realistic" graphs. Then, given any graph as input, we can check to see if the generator can build a "similar" graph under any set of parameters. If not, then it might represent an anomaly in the input graph, such as a misconfigured router in the Internet graph.

Which patterns should we be looking for? How can we spot suspicious/erroneous subgraphs quickly? These are the questions that our NetMine system focuses on. The main contributions of this paper are that

- it proposes the "min-cut plots", an interesting pattern to check for while

Corporation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

analyzing a graph
- it proposes the "A-plots" as a tool for quickly finding suspicious subgraphs/nodes
- it scales very well with size of the graph for all its tasks, and thus is able to quickly handle graphs of hundreds of thousands of nodes
- it shows how to interpret these plots, and how we found surprising patterns and outliers on real graphs
- finally, it describes a recent, promising graph generator, "R-MAT", so that the user could generate similar, synthetic graphs.

The rest of this paper is organized as follows: Section 2 surveys the existing graph laws and generators. Section 3 presents our proposed methods and algorithms for mining large graphs. Section 4 gives the experimental results. We conclude in Section 5.

2 Background and Related Work

A *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is a set $\mathcal{V}$ of N nodes, and a set $\mathcal{E}$ of E edges between them. The adjacency matrix $\mathbf{A}$ of graph $\mathcal{G}$ is an $N * N$ matrix, with entry $a(i, j) = 1$ if the edge (i, j) exists, and 0 otherwise. The edges may be undirected (like the network of Internet routers and their physical links) or directed (like the network of who-trusts-whom in the *epinions.com* database (Richardson and Domingos, 2002)). *Bipartite graphs* have edges between two sets of nodes, like, for example, the graph of the movie-actor database (`www.imdb.com`). We split the discussion of related work into two parts: graph patterns, and graph generators.

2.1 Patterns and "Laws"

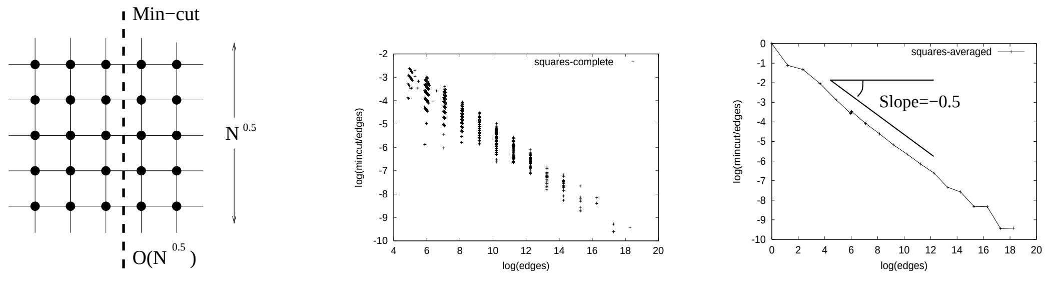
Skewed distributions, and power laws of the form $y = x^a$, appear very often. Power-laws have been observed for the degree distributions of the Internet, the WWW and the citation graph, the distribution of "bipartite cores" ($\approx$ communities), the eigenvalues of the adjacency matrix and others (Faloutsos et al., 1999; Kleinberg et al., 1999; Albert and Barabási, 2002). Recently, Pennock et al. (Pennock et al., 2002) observed deviations from power-laws for the Web graph, which are well-modeled by the truncated, discretized lognormal ("DGX") distribution of Bi et al. (Bi et al., 2001). Graphs also exhibit a strong "community" effect (Gibson et al., 1998; Kumar et al., 1999). Most real graphs like the Web and the Internet have surprisingly small diameters (Albert and Barabási, 2002; Tauro et al., 2001).Apart from these, there are many other measures such as clustering coefficient, expansion, resilience, prestige, influence, stress and so on (Chakrabarti, 2002; Gkantsidis et al., 2003; Tangmunarunkit et al., 2001; Palmer et al., 2002). Broder et al. (Broder et al., 2000) show that the WWW has a "bow-tie" structure, while Tauro et al. (Tauro et al., 2001) find that the Internet topology is organized as a set of concentric circles around a small core, like a "Jellyfish".

2.2 Graph Generators

The earliest graph generating model is by Erdős and Rényi. However it provably violates the power laws above. Recent graph generators can be grouped in two classes: *degree based* and *procedural*. Given a degree distribution (typically following a power-law), the degree-based ones try to find a graph that matches it (Albert and Barabási, 2002; Newman, 2003), but without giving any insights about the graph or trying to match other criteria (like small diameter, eigenvalues etc.). On the other hand, procedural generators (like our proposed R-MAT method) try to find simple mechanisms to generate graphs that match a property of the real graphs and, typically, the power law degree distribution. The typical representative here is the Barabasi-Albert (BA) method with the "*preferential attachment*" idea: keep adding nodes; new nodes prefer to connect to existing nodes with high degrees. Many modifications and alternatives to the basic idea have been proposed; some generators also include the geometrical layout of nodes in their models (Albert and Barabási, 2000, 2002; Newman, 2003; Pennock et al., 2002; Bu and Towsley, 2002). The BRITE generator (Medina et al., 2000) uses components from several of the above models.

In general, all of the above generators fail to meet one or more of the following goals: (a) the generator should be procedural (b) it should be able to generate all types of graphs (directed/undirected, bipartite, weighted) (c) it should match both power-law degree distributions and the "unimodal" distributions observed by Pennock et al. (Pennock et al., 2002) (d) it should satisfy more criteria (like diameter, eigenvalue plots), in addition to the degree distribution.

A related field is that of relational learning (Dzeroski and Lavrac, 2001); however, this focuses on finding structure at a more local level while our work focuses on the global level. Other topics of interest involving graphs include graph partitioning,frequent subgraph discovery,finding cycles in graphs,and many others. These address interesting problems, and we are investigating their use in our work.



(a) A grid and its min-cut (b) 2D grid min-cut plot (c) Averaged min-cut plot

Fig. 1. Plot (a) shows a portion of a regular 400x400 2D grid, and a possible min-cut. Plot (b) shows the full min-cut plot, and plot (c) shows the averaged plot. If the number of nodes is N, the length of each side is $\sqrt{N}$. Then the size of the min-cut is $O(\sqrt{N})$, which leads to a slope of -0.5, which is exactly what we observe.

3 Proposed Method

The contributions of NetMine are threefold: (1) we present the "min-cut plots", a new pattern for comparing a synthetically-generated graph to a real one; this is in addition to the list of patterns described in Section 2 above, (2) we present a novel tool called A-plots for interactive inspection of graphs and for finding erroneous/outlier nodes and subgraphs, and (3) we present the R-MAT graph generator, which can match almost all these patterns. Each of these provides us with a tool to visually probe the graph dataset; they let us infer, at a glance, some interesting properties of the graph while, at the same time, highlighting possible anomalies. In this section, we describe each of these parts; the observations we can make from visualizing them are better explained via experiments, described later in section 4.

3.1 "Min-cut plots":

Several criteria have been previously proposed to compare a synthetic graph to a real-world graph. These include degree distributions, hop-plots, scree-plots and others. NetMine includes all these, and adds "min-cut plots".

A min-cut of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a partition of the set of vertices $\mathcal{V}$ into two sets $\mathcal{V}_1$ and $\mathcal{V} - V_1$ such that both partitions are of approximately the same size, and the number of edges crossing partition boundaries is minimized. The number of such edges in the min-cut is called the min-cut size. Min-cut sizes of various classes of graphs has been studied extensively, and are known to have important effects on other properties of the graphs (Rosenberg and Heath, 2001). For example, Figure 1(a) shows a regular 2D grid graph, and one possible min-cut of the graph. We see that if the number of nodes is N, then the size of the min-cut (in this case) is $O(\sqrt{N})$.

The min-cut plot is built as follows: given a graph, its min-cut is found, and the set of edges crossing partition boundaries deleted. This divides the graph into two disjoint graphs; the min-cut algorithm is then applied recursively to each of these sub-graphs. This continues till the size of the graph reaches a small value (set to 20 in our case). Each application of the min-cut algorithm becomes a point in the min-cut plot. The graphs are drawn on a log-log scale. The x-axis is the number of edges in a given graph. The y-axis is the fraction of that graph's edges that were included in the edge-cut for that graph's separator.

Figure 1(b) shows the min-cut plot for the 2D grid graph. In plot (c), the value on the y-axis is averaged over all points having the same x-coordinate. The min-cut size is $O(\sqrt{N})$, so this plot should have a slope of -0.5, which is exactly what we observe. In section 4, we will see how the min-cut plots of real-world graphs look like, and how their similarities with figure 1 allow us to characterize them.

3.2 **A-plots:**

A simple way to find suspicious nodes/subgraphs in a large graph could be very useful in a variety of situations. However, the obvious approach of trying to visualize the graph does not work very well: visualization of large graphs is notoriously tough and time consuming, and is a research topic in its own right. Our next tool, called A-plots, is another way to visually hunt for anomalies. It consists of three types of plots for undirected graphs: (1) the plot of the adjacency matrix with nodes sorted in decreasing order by their network values (*RV-RV* plot, for Rank of network Value), (2) the plot of the degree of a node verses its rank of network value (*D-RV* plot, for Degree verses Rank of network Value), and (3) the plot of the adjacency matrix with nodes sorted in decreasing order by their degrees (*RD-RD* plot, for Rank of Degree). Together, these plots often reveal interesting patterns and properties of the graph.

Consider, for example, the *D-RV* plot for, say, the Internet router graph. In general, we expect nodes with high degree to be near the core of the graph (the Internet "backbone"), and thus also have high network value. Thus, before seeing the *D-RV* plot, we might expect it to decay smoothly, with low degree being correlated with low rank in network value. However, as we shall in Section 4, there are sudden spikes in this plot, which we can immediately pick out visually as outliers. Thus, visual inspection of the *D-RV* plot reveals anomalies which might have gone unnoticed otherwise. The reasons behind this particular anomaly will be discussed later in Section 4.

3.3 The R-MAT generator:

A good graph generator, with a proper choice of parameters, should be able to match a given real-world graph; the absence of a good match may indicate abnormalities in the given graph. Several previous graph generators have been described in Section 2, but they all fail in one aspect or another. The goals a graph generator should achieve are that the generated graph should:

- (*g1*) match the degree distributions (power laws or not)
- (*g2*) exhibit a "community" structure
- (*g3*) have a small diameter, and match other criteria.

A good generator, matching these patterns, can be useful for many applications:

- *Detection of abnormal subgraphs/edges/nodes:* Abnormalities should deviate from the "normal" patterns, so a good graph generator will *not* be able to match an abnormal graph very well. Thus, given an input graph, we can check for anomalies using a three-step process: (1) find the best matching parameter set for our graph generator, (2) generate a synthetic graph using these parameters, and (3) visually compare the patterns of the input graph against the generated graph. Absence of a fit may point to anomalies.
- *Simulation studies:* Algorithms meant for large real-world graphs can be tested on synthetic graphs which "look like" the original graphs. This is particularly useful if collecting the real data is hard or costly.
- *Realism of samples:* Most graph algorithms are super-linear on the node count, and thus prohibitive for large graphs. We might want to build a small sample graph that is "similar" to a given large graph. In other words, this smaller graph needs to match the "patterns" of the large graph to be realistic.
- *Extrapolation of data:* Given a real, evolving graph, we expect it to have $x\%$ more nodes next year; how will it then look like, assuming that our "laws" are still obeyed? For example, in order to test the next-generation Internet protocol, we would like to simulate it on a graph that is "similar" to what the Internet will look like a few years into the future.

In the following, we will describe our R-MAT graph generator, show that it matches goals *g1-g3*, and develop a parameter-fitting method for it.

3.3.0.1 **<u>Main Idea:</u>** We provide a method which fits both unimodal and power-law graphs using very few parameters. Our method, called **R**ecursive **MAT**rix, or R-MAT for short, generates the graph by operating on its adjacency matrix in a recursive manner. Recursive generators have been proposed

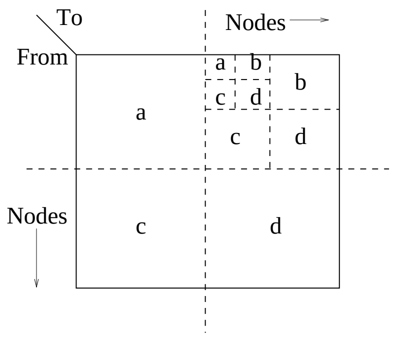


Fig. 2. The **R-MAT** model

in passing before (Palmer and Steffan, 2000), but the emphasis was on network issues.

3.3.0.2 Fast Algorithm to generate Directed Graphs: Recall that the adjacency matrix **A** of a graph of N nodes is an $N * N$ matrix, with entry $a(i,j) = 1$ if the edge (i,j) exists, and 0 otherwise. The basic idea behind **R-MAT** is to recursively subdivide the adjacency matrix into four equal-sized partitions, and distribute edges within these partitions with a unequal probabilities: starting off with an empty adjacency matrix, we "drop" edges into the matrix one at a time. Each edge chooses one of the four partitions with probabilities a, b, c, d respectively (see Figure 2). Of course, $a + b + c + d = 1$. The chosen partition is again subdivided into four smaller partitions, and the procedure is repeated until we reach a simple cell (=1×1 partition). This is the cell of the adjacency matrix occupied by the edge. The number of nodes in the R-MAT graph is set to 2^n; typically $n = \lceil \log_2 N \rceil$. There is a subtle point here: we may have *duplicate* edges (ie., edges which fall into the same cell in the adjacency matrix), but we only keep one of them. To smooth out fluctuations in the degree distributions, we add some noise to the (a, b, c, d) values at each stage of the recursion and then renormalize (so that $a + b + c + d = 1$). Table 1 shows the symbols used in the paper.

3.3.0.3 Discussion: Meeting the Goals: Intuitively, our technique is generating "communities" in the graph. Typically, $a \geq b$, $a \geq c$, $a \geq d$.

- The partitions a and d represent separate groups of nodes which correspond to communities (say, soccer and automobile enthusiasts).
- The partitions b and c are the *cross-links* between these two groups; edges there would denote friends with separate interests.
- The recursive nature of the partitions means that we automatically get sub-communities within existing communities (say, motorcycle and car enthusiasts within the automobile group).

Symbol	**Meaning**
N	Number of nodes in the real graph
2^n	Number of nodes in the R-MAT graph
E	Number of edges in the real graph, and in the R-MAT generated graph *after* duplicate elimination
(a, b, c, d)	Probabilities of an edge falling into partitions in the R-MAT model. $a + b + c + d = 1$.

Table 1
Table of symbols.

The third bullet results in "communities within communities" (goal *g2*). The skew in the distribution of edges between the partitions ($a \geq d$) leads to lognormals and the DGX distribution (goal *g1*). We shall show experimentally that R-MAT also generates graphs with small diameter and matching other criteria as well (goal *g3*).

3.3.0.4 Parameter fitting with AutoMAT-fast: The R-MAT model can be considered as a *binomial cascade* in two dimensions. We can calculate the expected number of nodes c_k with out-degree k:

$$c_k = \binom{E}{k} \sum_{i=0}^{n} \binom{n}{i} \left[p^{n-i}(1-p)^i\right]^k \left[1 - p^{n-i}(1-p)^i\right]^{E-k}$$

where 2^n is the number of nodes in the R-MAT graph and $p = a + b$. Fitting this to the observed outdegree distribution gives us the estimated values for $p = a + b$ (and similarly $q = a + c$ for the indegree distribution). Conjecturing that the $a : b$ and $a : c$ ratios are approximately $75 : 25$ (as seen in many real world scenarios), we can calculate the parameters (a, b, c, d).

3.3.0.5 Extending R-MAT to Undirected Graphs: An undirected graph must have a symmetric adjacency matrix. We achieve this by generating a directed graph with $b = c$ and then using a "clip-and-flip" on the resulting adjacency matrix. This involves throwing away the half of matrix above the main diagonal and copying the lower half to it. The effect of this is twofold: first, since $b = c$, the number of edges in the final undirected matrix is approximately equal to that in the directed graph; and second, this technique guarantees that the resulting matrix will be symmetric, and hence the corresponding graph will be undirected.

3.3.0.6 Extending R-MAT to Bipartite Graphs: For a bipartite graph, the length and height may be different, and the adjacency matrix will be a rectangle instead of a square. Here too, we set the length and width to be powers of 2, denoted by 2^{n_1} and 2^{n_2}. While dropping edges, we might form a partition with a length(height) of 1; in such a case, further partitions are just top-bottom(left-right) with the appropriate probabilities.

4 Experiments

The questions we wish to answer are:

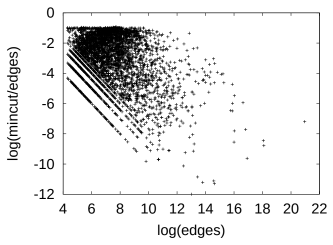
- **[Q1]** How do the min-cut plots look for real-world graphs? Does R-MAT match them? What can the min-cut plot tell us about the graph?
- **[Q2]** How can A-plots be used for analyzing large graphs?
- **[Q3]** How does R-MAT compare with existing generators for undirected graphs?
- **[Q4]** How does R-MAT compare with existing generators for directed graphs?
- **[Q5]** How does R-MAT compare with existing generators for bipartite graphs?

The datasets we use for our experiments are:

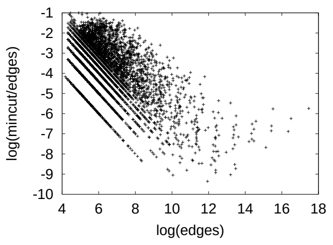
- *Epinions:* A directed graph of who-trusts-whom from epinions.com (Richardson and Domingos, 2002): $N = 75,879; E = 508,960$.
- *Epinions-U:* An undirected version of the *Epinions* graph: $N = 75,879; E = 811,602$.
- *Clickstream:* A bipartite graph of Internet users' browsing behavior (Montgomery and Faloutsos, 2001). An edge (u, p) denotes that user u accessed page p. It has $23,396$ users, $199,308$ pages and $952,580$ edges.
- *Lucent* is an undirected graph of network routers, obtained from `www.isi.edu/scan/mercator/maps.html`. $N = 112,969; E = 181,639$.
- *Router* is a larger graph (the SCAN+Lucent map) from the same URL, which subsumes the *Lucent* graph. $N = 284,805; E = 898,492$.
- *Google* is a graph of webpage connectivity from the Google (Google Programming Contest, 2002) programming contest. $N = 916,428; E = 5,105,039$.

4.1 [Q1] Min-cut Plots:

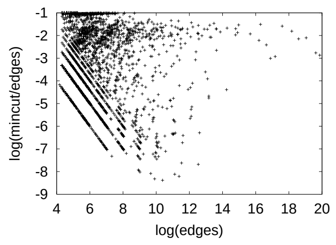
What do the min-cut plots for real-world graphs look like? Are they like min-cut plots for random graphs, or are they completely different? Figure 3 shows min-cut sizes of some real-world graphs. For each graph, we used the Metis graph partitioning library (Karypis and Kumar, 1995) to generate a separator,



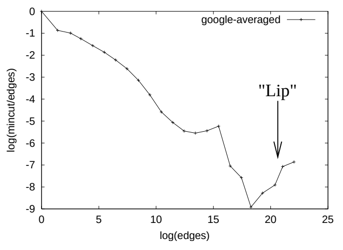
(a) "Google" min-cut plot



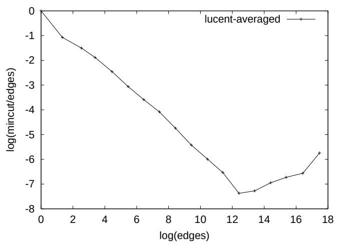
(b) "Lucent" min-cut plot



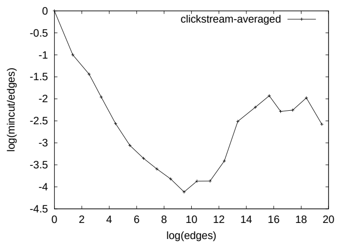
(c) "Clickstream" min-cut plot



(d) "Google" averaged



(e) "Lucent" averaged



(f) "Clickstream" averaged

Fig. 3. These are the min-cut plots for several datasets. We plot the ratio of min-cut-size to edges versus number of edges on a log-log scale. The first row shows the actual plots; in the second row, the cutsize-to-edge ratio is averaged over all points with the same number of edges.

as described by Blandford, Blelloch, and Kash (Blandford et al., 2003).

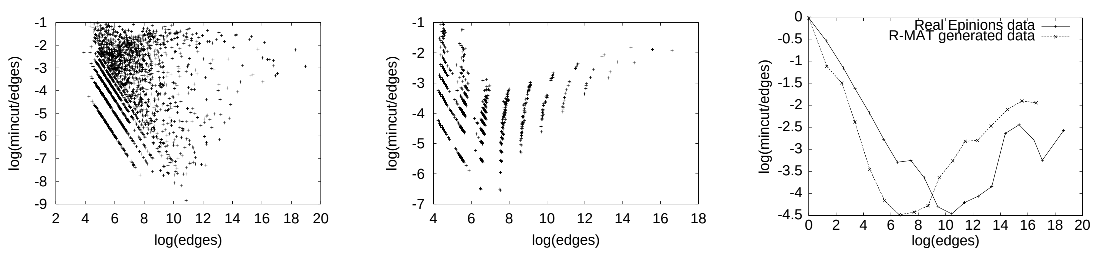
For random graphs, we expect about half the edges to be included in the cut. Hence, the min-cut plot of a random graph would be a straight horizontal line with a y-coordinate of about $\log(0.5) = -1$. A very separable graph (for example, a line graph) might have only one edge in the cut; such a graph with N edges would have a y-coordinate of $\log(1/N) = -\log(N)$, and its min-cut plot would thus be on the line $y = -x$.

As we can see from Figure 3, the plots for real-world graphs do not match either of these situations, meaning that *real-world graphs are quite far from either random graphs or simple line graphs.* Indeed, it appears as if the graph looks like a grid when the number of edges is low (as in figure 1), but with a distinguishing "lip" when the number of edges is very high.

Observation 1 (Noise) *We see that real-world graphs seem to have a lot of "noise" in their min-cut plots, as shown by the first row of Figure 3.*

Observation 2 ("Lip") *The ratio of min-cut size to number of edges decreases with increasing edges, except for graphs with large number of edges, where we observe a "lip" in the min-cut plot.*

Thus, the min-cut plot seems to have a linear slope over a wide range of the x-axis, but with a (as yet unexplained) "lip" at the end. Now, given any real-



(a) "Epinions" min-cuts (b) R-MAT min-cuts (c) Averaged min-cuts

Fig. 4. Here, we compare min-cut plots for the Epinions dataset and a dataset generated by R-MAT, using properly chosen parameters (in this case, a=0.5, b=0.2, c=0.2, d=0.1) We see from plot (c) that the shapes of the min-cut plots are similar.

world graph, we can visualize its min-cut plot, and any significant deviations from this general pattern might be due to anomalies in the graph.

The min-cut plot contains important information about the graph (Rosenberg and Heath, 2001). Hence, any synthetically generated graph meant to simulate a real-world graph should match the min-cut plot of the real-world graph. In Figure 4, we compare the mincut-plots for the Epinions graph with a graph generated R-MAT. As can be seen, the basic shape of the plot is the same in both cases, though the R-MAT plot appears to be shifted slightly from the original.

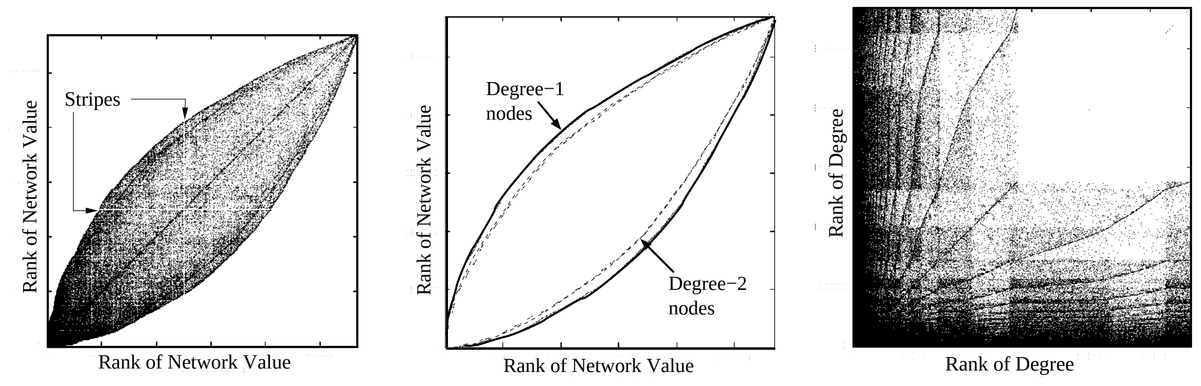
Observation 3 *The graphs generated by R-MAT appear to match the basic shape of the min-cut plot for several real-world graphs.*

Summary of min-cut plots: Given an input graph, the min-cut plot is a useful visual tool which helps us understand its structure. Most real-world graphs exhibit high noise, but on average, they look like a grid (in some dimension) for low values on the x-axis, and a "lip" for high values. Significant deviations from this pattern would indicate the presence of outliers at some scale in the graph.

4.2 [Q2] A-plots:

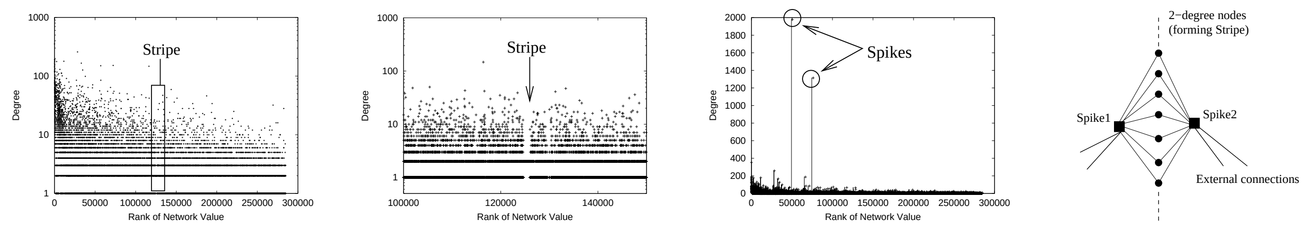
Figures 5 and 6 show A-plots for the Router dataset. Figure 5 shows the *RV-RV* and *RD-RD* plots, and Figure 6 shows the *D-RV* plot under different scalings. We can make the following observations:

Observation 4 ("Water-Drop") *The RV-RV plot has a clean and smooth oval-shaped boundary for the edges in the graph.*



(a) *RV-RV* plot (b) Estimated boundaries (c) *RD-RD* plot

Fig. 5. A-plots for the "Router" graph: Plot (a) shows the *RV-RV* plot, and a very interesting "Water-Drop" pattern is immediately apparent. The outermost "boundary stripes" are due to nodes of degree one (solid curve) and two (broken curve), whose positions can be calculated using Equations 1,2 as shown by plot (b). Plot (c) shows the *RD-RD* plot.



(a)*D-RV* plot (log-lin) (b)*D-RV* plot (blowup) (c)*D-RV* plot (lin-lin) (d) Spikes-and-stripes

Fig. 6. A-plots for the "Router" dataset: Plot (a) shows the *D-RV* plot, and plot (b) shows a blowup of a portion, clearly demonstrating the "white stripes" phenomenon. Plot (c) shows the *D-RV* plot in the linear-linear scale; nodes with the highest degree do not have the highest network value. Plot (d) shows the actual network configuration of routers involved in the stripe and spikes. An explanation is provided in Observation 9 in the text.

Explanation: The boundary of the edges is defined by the one-degree nodes in the graph. There are many such nodes because of the power law distribution of the degrees. Let I_i denote the network value of node i; if node i has a degree of one and node j is the only node it is connected to, the properties of spectral decomposition of a matrix imply that

$$I_i = 1/\lambda_1 * I_j \tag{1}$$

where λ_1 is the largest eigenvalue of the adjacency matrix of the graph (Zhan, 2003). Therefore the boundary of edges in the *RV-RV* plot can be calculated from the first eigenvalue and eigenvector. Figure 5(b) shows just this; the solid curve represents degree-one nodes. These are obviously the boundary curves for plot (a).

We also see that there is no edge at all outside of the boundary. Let node i have network value I_i, and have node j with network value I_j as its most "important" neighbor (in the sense of high network value). Then, $I_i \geq 1/\lambda_1 * I_j$. Therefore all edges are confined within the boundary in the RV-RV plot.

Observation 5 (Nested Water-Drops) *There are a pair of "secondary" lines within the boundary of the edges in the RV-RV plot.*

Explanation: These lines are the results of some two-degree nodes. When a node i has two degrees and the two nodes it is connected to have about the same network values (say, I_j), we can calculate where the involved edges will show up in the *RV-RV* plot similar to the one-degree case:

$$I_i = 2/\lambda_1 * I_j \tag{2}$$

The dashed lines in Figure 5(b) show the results, which match with the *RV-RV* plot. The presence of these "secondary" lines in the plot means that a significant number of the two-degree nodes in the graph are connected to two "similar" (similar as is defined by similar network values) nodes. The presence of the faint "tertiary" lines can be explained accordingly.

Observation 6 (Diagonal) *There is more or less a solid line that goes through the diagonal of the RV-RV plot even though the adjacency matrix does not include any self-edges.*

Explanation: This means a node is more likely to be connected with "similar" nodes.

Observation 7 (White Stripes) *There are white stripes (both vertical and horizontal) visible in both the RV-RV and the D-RV plots.*

Explanation: The stripes come from a large number of nodes that are connected to exactly the same nodes, usually just one or two. Since nodes that are connected the same way have exactly the same network values, they show up as a group and become visible in the *RV-RV* and *D-RV* plots (Figures 5(a) and 6(a, b) respectively).

Observation 8 (Isolated Components) *The largely empty white square in the corner of the RD-RD plot results from connections between one-degree nodes.*

Explanation: Any dots (edges) in this area correspond to two-node isolated components.

Observation 9 (Degree vs. Importance) *Figure 6(c) shows several points in the D-RV plot having high degree, but low network value (and thus low rank). Thus, high degree does not imply high "importance".*

Explanation: The *D-RV* plot in Figure 6(c) shows that the two highest-degree nodes actually have low 'network value'. This is counter-intuitive - how could it possibly be the case, in a power-law graph? Is it a data collection error?

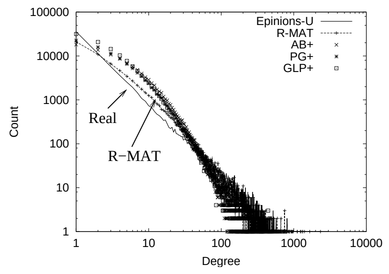
The answer is surprising, and actually also explains the white stripe in Figure 6(a,b): The two highest-degree nodes (labeled 'Spike1' and 'Spike2'), and a large number of two-degree nodes, form a subgraph like the one shown in Figure 6(d). 'Spike1' and 'Spike2', being away from the core of the network, have much lower network value than what their high degree would promise. Their satellites (= all the 2-degree nodes connected to them) have identical, relatively high network values, which cause the white strip in Figure 6(a,b). We are currently investigating with domain experts the reasons for such a weird sub-graph. However, our point is that the proposed *D-RV* and *RV-RV* plots exactly spotted this strange pattern, which would go undetected if we only used the traditional, or even recent tools, like degree-plots, scree-plots etc.

Summary of A-plots: The three types of A-plots described above follow very well-defined patterns, such as the "nested waterdrops" and diagonals. Anomalous behavior in such plots can be easily detected, as evidenced by the "spikes and stripes" anomaly detected in the real-world Router dataset. In a graph of about 10^5 nodes and 10^6 edges, this small abnormal subgraph would have gone completely unnoticed without this visualization tool.

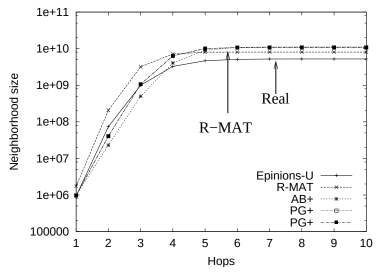
4.3 [*Q*3] *R-MAT on undirected graphs:*

Apart from degree distributions, we compare the models on their singular value vs. rank plot, first singular vectors (network values) vs. rank plots, "hop-plot" (number of reachable pairs vs. number of hops) and "effective diameter" (Palmer et al., 2002; Tauro et al., 2001), and stress distribution (Gkantsidis et al., 2003) (the stress of an edge is the number of shortest paths between node pairs that it is a part of). For undirected graphs, eigenvalues and singular values are equivalent; for bipartite graphs, eigenvalues may not exist.

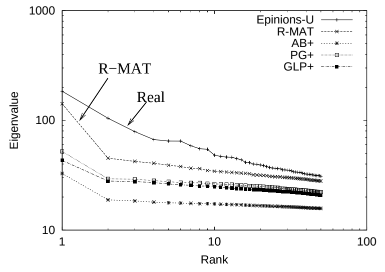
We compared R-MAT to the *AB* (Albert and Barabási, 2000), *GLP* (Bu and Towsley, 2002) and *PG* (Pennock et al., 2002) models, chosen for their popularity or recency. All of these are used to generate undirected graphs; they have not been used for directed or bipartite graphs. Thus, we can compare them with R-MAT only on *Epinions-U*. Also, we are unaware of any good parameter-fitting mechanisms for these generators, so for each generator, we exhaustively find the best parameter values. We use *AB+*, *PG+* and *GLP+* to stand for the original algorithms augmented by our parameter fitting.



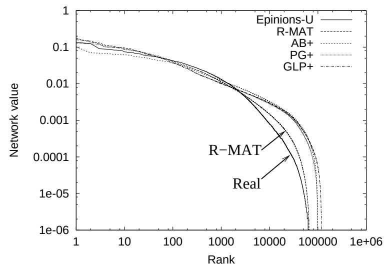
(a) Count vs Degree



(b) The Hop-plot



(c) Singular values



(d) First singular vec.

Fig. 7. *The Epinions-U Undirected Graph:* The R-MAT plots gives the best fit to the *Epinions-U* graph (solid line) among all the generators.

We show results in Figure 7 for the *Epinions-U* undirected graph. Notice that R-MAT is very close to *Epinions-U* in all cases, while the competitors are not. Recall that all the y-scales are logarithmic, so small differences actually represent large deviations. The "stress distribution" plot is similar, but is not shown due to lack of space.

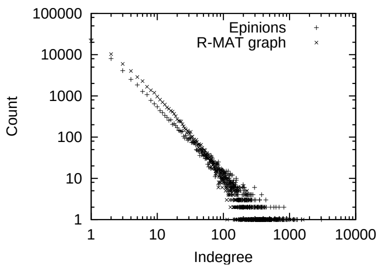
4.4 [Q4] R-MAT on Directed Graphs:

We can see from Figure 8 that the match between R-MAT and the *Epinions* dataset is very good. The effective diameter is 4 for both the real graph and for the R-MAT generated graph. The other models considered are not applicable.

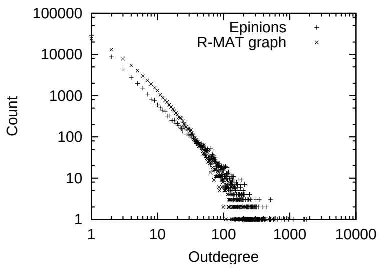
4.5 [Q3] R-MAT on Bipartite Graphs:

R-MAT matches the bipartite *Clickstream* dataset very well (Figure 9) *including* the "un-powerlaw-like" outdegree distribution. The other models are not applicable.

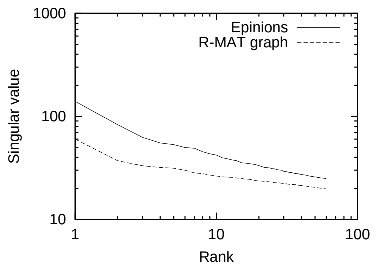
Summary of R-MAT: The experiments above demonstrate the ability of R-MAT to match the patterns of several real-world graphs. Thus, it is another



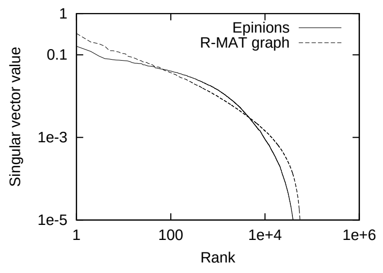
(a) Epinions Indegree



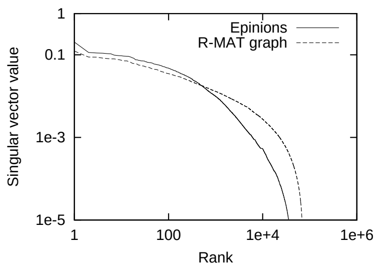
(b) Outdegree



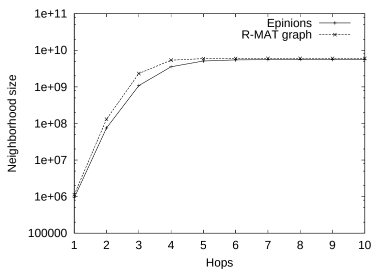
(c) Singular values



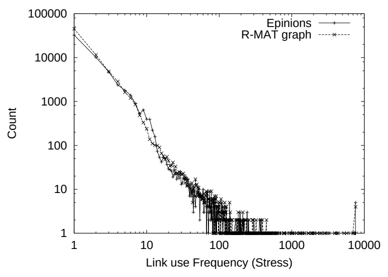
(d) First left singular vec.



(e) First right singular vec.



(f) Hop-plot



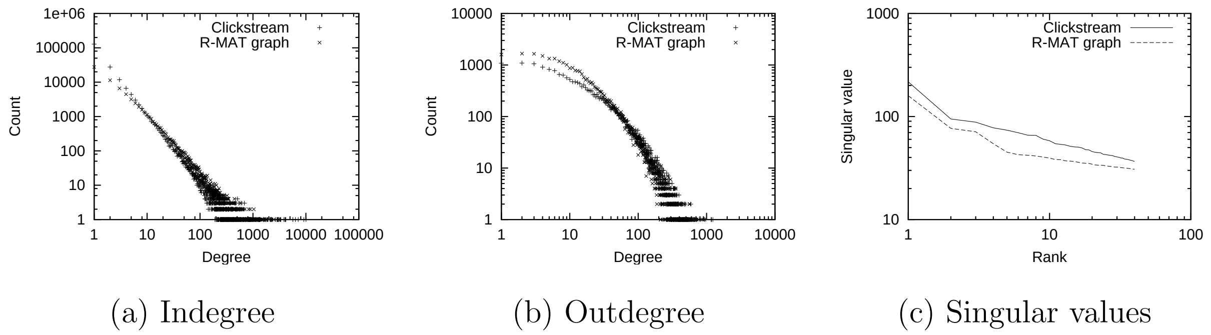
(g) Stress

Fig. 8. *The* Epinions *Directed Graph:* The *AB+, PG+* and *GLP+* methods **do not apply**. The crosses and dashed lines represent the R-MAT generated graphs, while the pluses and strong lines represent the real graph.

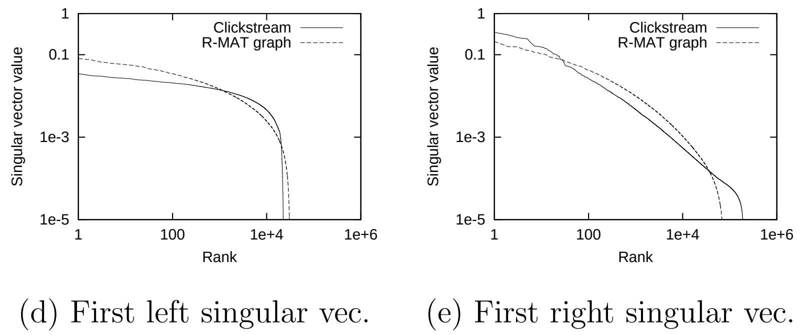
useful tool in anomaly detection on graphs; if we *cannot* find some parameter setting of R-MAT which matches some input graph, then the graph itself might have abnormalities.

4.6 Time and space requirements:

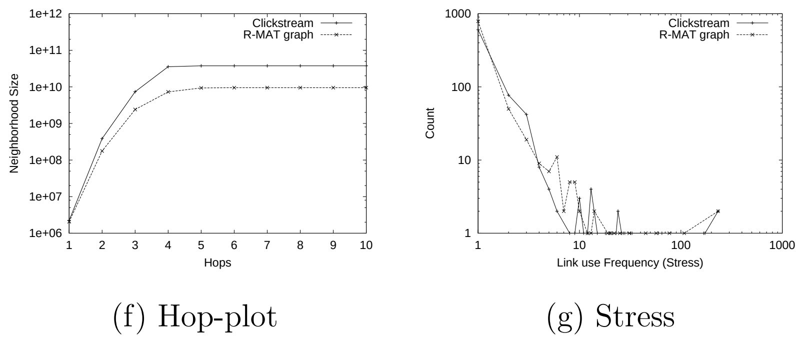
An important concern while developing NetMine was scalability to large datasets. Almost all of the graph operations mentioned in this paper can be computed quickly and efficiently: for instance, degree distribution computation requires only two sorting passes, and hopplots using a randomized method require only $O(N+E)$ time and $O(N)$ extra space (Palmer et al., 2002). The largest singular values and singular vectors can be computed by iterative methods which,



(a) Indegree (b) Outdegree (c) Singular values



(d) First left singular vec. (e) First right singular vec.



(f) Hop-plot (g) Stress

Fig. 9. *The* Clickstream *Bipartite Graph:* The *AB+, PG+* and *GLP+* methods **do not apply**. The crosses and dashed lines represent the R-MAT generated graphs, while the pluses and strong lines represent the real graph.

while having problem-dependent time and space complexity, are very fast and parallelizable in practice (Berry, 1992). A-plots can be generated using these plus an extra sorting pass over the data. R-MAT needs $O(E \log N)$ time and $O(E)$ space. The 2-way graph partitioning algorithm used by NetMine requires $O(N)$ time (Karypis and Kumar, 1995), implying a time complexity of $O(N \log N)$ for computing min-cuts plots.

5 Conclusions

Visualization is a key component in any data mining application, including graph mining. However, as against trying to visualizing the entire graph as is, we show how visualizing certain functions of the graph, such as degree distributions, min-cut plots, and A-plots, can be more useful. These help us visualize the same graph from different perspectives, and one can easily detect patterns in such plots that might not be apparent from just visualizing the entire graph data structure.

We propose several new tools for mining and visualizing the information in large graphs. Our emphasis is on scalable algorithms that can handle arbitrarily large graphs. When applied on real graphs, our new tools discovered patterns that were not visible with the known tools (like degree plots, hop-plots etc).

The contributions of this work are:

- **Min-cut plots:** They show the relative size of the minimum cut in a graph partition. For regular 2-d and 3-d grid-style networks (like Delaunay triangulations for finite element analysis), these plots have a slope that depends on the intrinsic dimensionality of the grid. However, for real graphs, these plots show significantly more 'noise', as well as a 'lip'. Thus, visualizing these plots can give significant information aout the graph; in fact, significant deviations from the common pattern might imply anomalies in the graph.
- **A-plots**: These plots provide new viewpoints for inspecting large graphs. We noticed some striking patterns ("water-drops", stripes, "lone" points), and we showed how to interpret them. In addition, we used these to detect actual anomalies in a large Router graph dataset.
- **R-MAT**: This simple, parsimonious graph model is able to match almost all the patterns of real-world graphs, and is more widely applicable and accurate than other existing graph generators. We also showed how the parameters of R-MAT can be efficiently set to mimic any given graph; this allows the user to quickly generate similar graphs which are synthetic yet *realistic.*

Moreover, we propose a list of natural tests which hold for a variety of real graphs: matching the power-law/DGX distribution for the in- and out-degree; the hop-plot and the diameter of the graph; the singular value distribution; the values of the first singular vector ("Google-score"); and the "stress" distribution over the edges of the graph. All of these are packaged into the NetMine package, freely available for download from `http://www.cs.cmu.edu/~deepay/mywww/software/NetMine-Basic-03-30-2004.tgz`, which provides a useful toolkit for analyzing and visualizing large graphs in a scalable manner, as shown by our experiments. We believe that these and other novel methods for visualizing the information hidden in a large graph would be invaluable tools for the data analyst, helping her understand and explore the information from many different perspectives to gain a clearer picture of the whole.

References

Albert, R., Barabási, A.-L., 2000. Topology of complex networks: local events and universality. Physical Review Letters 85 (24).

Albert, R., Barabási, A.-L., 2002. Statistical mechanics of complex networks. Reviews of Modern Physics.

Barabási, A.-L., May 2002. Linked: The New Science of Networks, 1st Edition. Perseus Publishing.

M. Berry, 1992. Large scale singular value computations. International Journal of Supercomputer Applications 6 (1), 13–49

Bi, Z., Faloutsos, C., Korn, F., 2001. The DGX distribution for mining massive, skewed data. In: KDD.

Blandford, D., Blelloch, G. E., Kash, I., 2003. Compact representations of separable graphs. In: SODA.

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J., 2000. Graph structure in the web: experiments and models. In: WWW.

Bu, T., Towsley, D., 2002. On distinguishing between Internet power law topology generators. In: INFOCOM.

Chakrabarti, S., 2002. Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufmann.

Chen, H., Schroeder, J., Hauck, R., Ridgeway, L., Atabaksh, H., Gupta, H., Boarman, C., Rasmussen, K., Clements, A., January 2003. Coplink Connect: Information and knowledge management for law enforcement. CACM 46 (1), 28–34.

Dzeroski, S., Lavrac, N. (Eds.), 2001. Relational Data Mining. Springer Verlag.

Faloutsos, M., Faloutsos, P., Faloutsos, C., 1999. On power-law relationships of the Internet topology. In: SIGCOMM. pp. 251–262.

Gibson, D., Kleinberg, J., Raghavan, P., 1998. Inferring web communities from link topology. In: ACM Conference on Hypertext and Hypermedia.

Gkantsidis, C., Mihail, M., Zegura, E., 2003. Spectral analysis of Internet topologies. In: INFOCOM.

Google Programming Contest, 2002. `http://www.google.com/programming-contest/`.

Karypis, G., Kumar, V., 1995. A fast and high quality multilevel scheme for partitioning irregular graphs. Tech. Rep. TR 95-035.
URL `www-users.cs.umn.edu/~karypis/publications/partitioning.html`

Kleinberg, J., Kumar, S. R., Raghavan, P., Rajagopalan, S., Tomkins, A., 1999. The web as a graph: Measurements, models and methods. In: Intl. Conf. on Combinatorics and Computing.

Kumar, S. R., Raghavan, P., Rajagopalan, S., Tomkins, A., 1999. Extracting large-scale knowledge bases from the web. In: VLDB. Edinburgh, Scotland.

Medina, A., Matta, I., Byers, J., 2000. On the origin of power laws in Internet topologies. In: SIGCOMM.

Montgomery, A. L., Faloutsos, C., 2001. Identifying Web browsing trends and patterns. IEEE Computer 34 (7).

Newman, M. E. J., 2003. The structure and function of complex networks. SIAM Review 45, 167–256.

Palmer, C. R., Gibbons, P. B., Faloutsos, C., 2002. ANF: A fast and scalable tool for data mining in massive graphs. In: SIGKDD. Edmonton, AB, Canada.

Palmer, C. R., Steffan, J. G., November 2000. Generating network topologies that obey power laws. In: GLOBECOM.

Pastor-Satorras, R., Vespignani, A., 2001. Epidemic spreading in scale-free networks. Physical Review Letters 86 (14).

Pennock, D. M., Flake, G. W., Lawrence, S., Glover, E. J., Giles, C. L., 2002. Winners don't take all: Characterizing the competition for links on the Web. Proceedings of the National Academy of Sciences 99 (8), 5207–5211.

Richardson, M., Domingos, P., 2002. Mining knowledge-sharing sites for viral marketing. In: SIGKDD. Edmonton, Canada, pp. 61–70.

Rosenberg, A. L., Heath, L. S., 2001. Graph Separators, with Applications. Kluwer Academic/Plenum Publishers.

Tangmunarunkit, H., Govindan, R., Jamin, S., Shenker, S., Willinger, W., 2001. Network topologies, power laws, and hierarchy. Tech. Rep. 01-746, U. Southern California.

Tauro, S. L., Palmer, C., Siganos, G., Faloutsos, M., 2001. A simple conceptual model for the Internet topology. In: Global Internet, San Antonio, Texas.

Zhan, Y., 2003. Tools for graph mining. Master's thesis, Carnegie Mellon University.