Tools for graph mining

Yiping Zhan

yzhan@cs.cmu.edu

12th June 2003

Abstract

Large real-world graphs often show interesting properties, such as power-law degree distributions and very small diameters. Discovering such patterns and regularities has a wide range of potential applications. It could help us with detecting outliers or abnormal subnetworks (such as terrorist networks or illegal money-laundering rings), maximizing efficiency of disease controlling, marketing, forecasting and simulations, to name a few.

A graph generating model, the "recursive matrix" (R-MAT) model is introduced and a method (AutoMAT) is shown for automatically estimating the input parameters for R-MAT in order for it to match a given real-world graph. Using these parameters, the resulting R-MAT graphs are shown to match many properties of real graphs. Also, a set of plots (A-plots) are introduced as original ways for viewing large graphs. Their applications in finding interesting patterns and outliers in real, large graphs are demonstrated.

Contents

1 Introduction

Graphs, networks and their surprising regularities have been attracting significant interest recently. The applications are diverse, and the discoveries are striking: The World Wide Web follows surprising power-laws, a "bow-tie" structure [9], while still has a surprisingly small diameter [3]. Similar startling discoveries have also been made for power laws in the Internet topology [17], for Peer-to-Peer (gnutella/Kazaa) overlay graphs [30], and for who-trusts-whom in the `epinions.com` network [29].

Finding patterns, laws and regularities in large real networks has numerous applications: link analysis, for criminology and law enforcement [13]; analysis of virus propagation patterns, on both social/email as well as physical-contact networks [35]; analysis of network performance, on the router connection networks; and maximizing effects of marketing efforts on social networks.

In fact, any table with two categorical columns, that is, any many-to-many relationship in database terminology [27] readily corresponds to a graph: a table of customers buying products yields a bipartite graph; similarly for actors playing in movies (such as `www.imdb.com`); scientific authors citing other authors leads to a directed graph; and so does a table with `epinions.com` users trusting others.

The long term motivation behind this work is *how do real graphs look like?* What patterns do they follow, in addition to (or replacement of) the power laws, the "six degrees of separation", the "bow-tie", or "jellyfish" shapes [33]. There will never be a final answer to that question, but at least we want to find as many such "laws" as we can.

Such "laws" are important for several reasons: They can help us spot outliers, that is, "abnormal" graphs or subgraphs (criminal rings, faulty Internet router configurations. etc.). They help us generate realistic graphs and discard unrealistic generators, for simulations, e.g., of Internet protocols, immunization policies etc., when real graphs are difficult, impossible, or illegal to collect. They may help us do extrapolations: given a real, evolving graph, we expect it to have $x\%$ more nodes next year; how will it then look like, assuming that the "laws" are still obeyed? Finally, they can check the realism of a small sample sub-graph: sampling is very useful, since most graph algorithms are super-linear on the node count, and thus prohibitive for large graphs.

Discovering and listing such laws is an important first step for graph mining. Ideally, we would like a generating model with the following properties: (a) It would have only a few parameters. (b) It would generate graphs that obey the above "laws", and it would match the properties of real graphs (degree exponents, diameters, etc.) with the appropriate values of its parameters. And, (c) It would generate the graphs quickly.

This work tries to answer two important questions for graph mining:

- How to generate realistic graphs?
- How to check properties/spot patterns of real and/or generated graphs?

We first introduce the R-MAT model for generating graphs and the AutoMAT method for estimating the input parameters of R-MAT for it to match a real graph. With a series of tests we check the properties of real graphs and their R-MAT counterparts. We also show plots/properties for real world graphs and introduce the A-plots as a convenient way for viewing large undirected graphs and spotting certain properties from such graphs.

The rest of this paper is organized as follows: Section 2 surveys the existing graph mining tools, mainly graph laws and generators. Section 3 describes our new proposed ideas for mining graphs. We describe the properties we use to check R-MAT graphs in Section 4. Section 5 gives the experimental results for two real graphs and their R-MAT counterparts as well as our observations in A-plots and the explanations. Section 6 gives the conclusions and in Section 7 we discuss some future research directions.

2 Survey

First, let us establish some terminology. An undirected graph $G = (V, E)$, or simply a graph, is a set V of nodes, and a set E of undirected edges between them. For example, the network of Internet routers and their physical links is an undirected graph. The definition of a directed graph is obvious. For example, the network of who-trusts-whom in the `epinions.com` database [29] forms a directed graph. The above graphs will also be referred to as self-graphs, to emphasize their difference from the bipartite graphs, like, for example, the graph of the movie-actor database (`www.imdb.com`). A bipartite graph has two sets of nodes, V_1 ("actors") and V_2 ("movies"), with edges between them (which actor played in which movie). All of the above types of graphs are not weighted. In a weighted graph, each edge has an associated weight. For example, consider the Internet routers, the physical links between them have different bandwidths.

The adjacency matrix A of an undirected graph is defined as $a_{ij} = 1$ if there is an edge from node i to node j and $a_{ij} = 0$ otherwise. If a graph is weighted, a_{ij} is the weight of the corresponding edge. The adjacency matrix of an undirected graph is symmetric. All real-world large graphs have very sparse adjacency matrices.

Next we survey related work. First we summarize important patterns that have been discovered in real graphs, then we discuss existing graph generators and their limitations.

2.1 Patterns and "Laws"

The patterns seem to form the following groups:

Power laws: Skewed distributions, and laws of the form $y = x^a$, appear very often. Such a law comes out as a line of slope a, when we plot y versus x in log-log scale. In the Internet topology graph, the degree distribution follows such a power-law [17]. That is, the count of nodes with degree k (C_k), versus the degree k, is a line in log-log scale. The biggest eigenvalues of the adjacency matrix of the Internet graph also follows a power law.

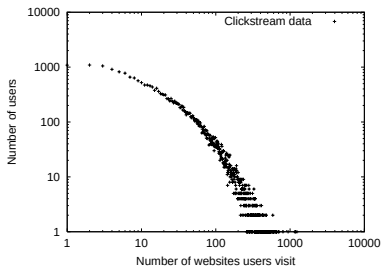


Figure 1: The Clickstream dataset: A degree distribution which does not obey a power law.

The World Wide Web graph also obeys power laws [20]: the in-degree and out-degree distributions both follow power-laws, as well as the number of the so-called "bipartite cores". Redner shows that the citation graph follows a power law with exponent 3 [28]. Recently, deviations from the power laws have been observed.

Deviations/generalizations from power laws, and "DGX": For the WWW graph, Pennock et al. find that, although the Web shows a power-law degree distribution on the whole, the distribution is "unimodal" for individual categories such as, say, news sites [26]. Specifically, it is curving, like the one in Figure 1. This figure gives the degree distribution of the Clickstream dataset (who-visits-what bipartite graph - see Section 5 for details). Bi et al. show that a truncated, discretized lognormal ("DGX") includes the power laws as a special case, while it matches well unimodal distributions, like the one from the Clickstream dataset above [6].

Diameter - small world phenomena: Most real graphs have surprisingly small diameters: Milgram discovered the famous "six degrees of separation" in a who-knows-whom graph [23]. Albert et al. estimate the diameter of the Web to be 19 [3]. Tauro et al. introduce the "jellyfish" model for Internet topology, where they observe that the "effective" diameter of the Internet is very small [33]. Reittu et al. [1] prove that, under certain assumptions, the diameter of power law graphs is practically constant, growing like $O(\log \log N)$.

Measures: There is a huge list of measures in the literature of computer networks, social networks and graph theory. To name a few, for a node, we have the clustering coefficient, "prestige", and "importance" [11]; for a whole network, we can compute the "expansion", "resilience", "distortion" [32], and the characteristic path length [10]; for each edge, the "stress". Later on we describe the measures we used for our experiments and our reasons for choosing them.

2.2 Graph generators

The earliest, and most famous graph generating model is the Erdos-Renyi model [15]: A node picks another node at random, and gets connected to it. This model exhibits fascinating phase-transition properties, but it provably

[1] `http://www.ercim.org/publication/Ercim_News/enw50/reittu.html`

violates the power laws above.

Recent graph generators can be grouped in two classes: *degree-based*, and *procedural* generators.

Degree-based generators are less powerful for pattern discovery. Given a degree distribution, typically one that follows a power-law, they try to construct a graph that matches this degree distribution [1] [2] [21] [25]. These generators give no insights about the graph, and they do not even try to match other criteria, like the diameter, eigenvalues, etc., of a graph.

Procedural generators: This is the class that our proposed R-MAT method belongs to. The typical representative here is the Barabasi-Albert (BA) method [3] with the "*preferential attachment*" idea: Keep adding nodes; new nodes prefer to connect to nodes with high degrees. Surprisingly, this simple mechanism leads to power-law tails in the degree distribution. Procedural generators try to find such simple mechanisms, to generate graphs that match a property of the real graphs and, typically, the power law degree distribution. However, the original BA method leads to a degree power law with the slope fixed at 3, and has a relatively large diameter $O(\log N)$ [8]. Modifications and alternatives include the "rewiring" idea [4], the copying mechanism [20], the modifications [10], and the "winners don't take all" model [26]. The BRITE graph generator [22] uses components from both the Barabasi-Albert model and the Waxman model [36]. The latter belongs to the subclass of generators that include geometry, assuming that the nodes are points (e.g., on the plane); wiring decisions take into account the geographical distances. The Heuristically Optimized Tolerance (H.O.T.) method is another generator considering geometry [16]. However, the Waxman method gives graphs that do not fit power laws well, and H.O.T. is studied only for trees.

In general, all of the above generators fail to meet one or more of the following goals: (a) The generator should be procedural. (b) There should be a way to estimate its parameters. (c) It should be able to generate all types of graphs (directed/undirected, bipartite, weighted). (d) It should satisfy more criteria (like diameter, eigenvalue plots) in addition to the degree distribution.

3 Proposed Ideas

The contributions outlined here are threefold: (1) The R-MAT (Recursive MATrix) graph generating model is introduced. (2) We present a tool (AutoMAT) that does automatic parameter-fitting for R-MAT And, (3) We introduce the "A-plots" for visualizing large graphs, specifically, their adjacency matrices. These are described below. Some symbols used in this section and Sections 4 and 5 are summarized in Table 1.

Symbol(s)	Definition
a, b, c, d	Probability of dropping an edge into partitions in the R-MAT model. $a + b + c + d = 1$
$S1, S2$	For a self R-MAT graph, $S1 = S2$. The total number of nodes is 2^{S1}. For a bipartite R-MAT graph, the total number of nodes are 2^{S1} and 2^{S2} for the two types of nodes. These node numbers include those "hidden" nodes that have an outdegree (indegree) of zero. The dimension of the starting adjacency matrix in the R-MAT model is $2^{S1} \times 2^{S2}$.
E	Number of unique edges in an R-MAT graph.
$\hat{C}_{in,i}, \hat{C}_{out,i}$	Counts of nodes with an indegree (outdegree) of i for an R-MAT graph.
$C_{in,i}, C_{out,i}$	Counts of nodes with an indegree (outdegree) of i for a real-world graph that AutoMAT tries to match.
d_{in}, d_{out}	The biggest indegree (outdegree) whose corresponding count AutoMAT uses for fitting R-MAT input parameters.
$N1, N2$	For a real-world self graph, $N1 = N2$, which is the number of nodes in the graph. For a real-world bipartite graph, $N1$ and $N2$ are the numbers of the two types of nodes.
λ_1	The first eigenvalue of an adjacency matrix of a graph.
$\mathbf{u}_1, \mathbf{v}_1$	The first left- and right-eigenvectors of an adjacency matrix.
$v_{1,i}$	The i'th element of $\mathbf{v}_1$.
I_i	The network value of node i in an undirected graph. Same as $v_{1,i}$ for an undirected graph.

Table 1: Table of symbols

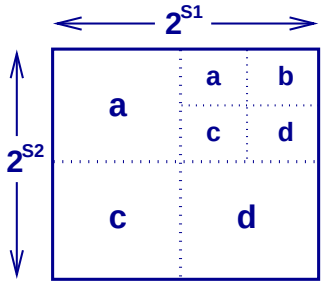


Figure 2: The R-MAT model: The $2^{S1} \times 2^{S2}$ adjacency matrix is subdivided into four equal partitions. Edges are dropped into the matrix one at a time, choosing partitions with probabilities (a, b, c, d) as shown. The process continues recursively.

3.1 The R-MAT generator

It is necessary for a realistic graph generator to generate graphs with skewed indegree distributions, skewed outdegree distributions and small diameters. We would also want a simple model to achieve those goals. These are the thoughts we have when designing and testing the R-MAT graph generator.

The R-MAT model generates a graph by operating on its adjacency matrix in a recursive manner. It is based on the idea of self-similar multi-fractals, where we put in the same skewed probability distribution over all scales in the graph. 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 skewed probability distribution. The probabilities are represented by the ordered set (a, b, c, d) with $a + b + c + d = 1$, as is shown in Figure 2. Let E denote the total number of edges we want to generate in the adjacency matrix, then there are E non-zero elements in A. Starting 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 and d, respectively (some details about using a, b, c and d values can be found in Appendix Section A); this process goes on for each partition of the adjacency matrix until the partition becomes a 1×1 matrix. Duplicate edges are eliminated.

The above procedure works straightforwardly for generating directed self graphs. For bipartite graphs, the adjacency matrix is not square, but we can apply the R-MAT model to rectangular adjacency matrices easily with rectangular partitions. When the length or height of the submatrix becomes 1, further partitions are only in the dimension that is bigger than 1.

Why would we expect R-MAT to work well? Intuitively, the R-MAT algorithm gives skewed degree distributions, with multi-fractal patterns in its adjacency matrix. The recursive nature of the generating algorithm could lend itself readily in simulating the community-within-community pattern which happens in real-world social

networks.

3.2 Automatic R-MAT parameter estimation (AutoMAT)

We just argued qualitatively that R-MAT has the potential to match some properties of real graphs, but how can we test R-MAT graphs against real ones? Given a real graph, we need a method for finding out the input parameters of R-MAT so that the R-MAT graph matches the real dataset. AutoMAT is a tool for doing this.

There are five input parameters for R-MAT: $a, b, c, S1, S2$, and E (Table 1). The goal for parameter estimation for the R-MAT graph is to find a way to figure out the input parameters such that the R-MAT graph that is generated matches a given real graph in as many aspects as possible. However, we need to choose priorities when it comes to the criteria we use for parameter fitting. The degree distribution of a real graph is one of the first things that researchers want to match for any graph generator. It is also one of the least computationally expensive criteria to check. Therefore, for estimating the R-MAT input parameters, we match exactly the number of edges of a real graph and by optimizing $a, b, c, S1$ and $S2$, we try to obtain the best match for the degree distributions. It turns out that upon matching the degree distributions, R-MAT allows an extra degree of freedom which we use in order to match the Scree plot (see Section 4 about the Scree plot).

Cost function: Currently R-MAT generates directed self graphs and bipartite graphs. There are two degree distributions (indegree and outdegree) for each graph. Our goal is to minimize the weighted sum of square differences in the degree distribution plots in log-log scale between the R-MAT generated graph and the real graph. The upper portions of the distribution plots have less fluctuations in the logarithm scale and these parts are used for fittings of R-MAT input parameters. For the sum of square differences, the value that corresponds to a degree of i is weighted by $1/i$, which is the derivative of $\log i$, to account for the skewed density of data points on the degree axis in logarithm scale. Furthermore, the sum of square differences in the two distribution plots are weighted according to the counts of the one-degree nodes. The Nelder-Mead downhill simplex minimization algorithm [34] is used to find the optimal R-MAT parameters that minimize the cost function F (Equation 1). In Equation 1, $\hat{C}_{in,i}$ and $C_{in,i}$ are the counts of nodes with an indegree of i for the R-MAT graph and the real graph, respectively. In practice, d_{in} is chosen so that the number of nodes with an indegree greater than d_{in} is often less than 10. Same logic applies for the part of equation that corresponds to the outdegrees.

$$F(a,b,c,S1,S2,E) = \frac{1}{(\log C_{in,1})^2}\sum_{i=1}^{d_{in}}[\frac{1}{i}(\log \hat{C}_{in,i} - \log C_{in,i})^2] + \frac{1}{(\log C_{out,1})^2}\sum_{i=1}^{d_{out}}[\frac{1}{i}(\log \hat{C}_{out,i} - \log C_{out,i})^2] \tag{1}$$

Algorithm: Estimation of R-MAT input parameters is done in steps. First E is fixed at the number of edges of the given real graph. and we try to determine the optimal values for $S1$ and $S2$. Since $S1$ and $S2$ have

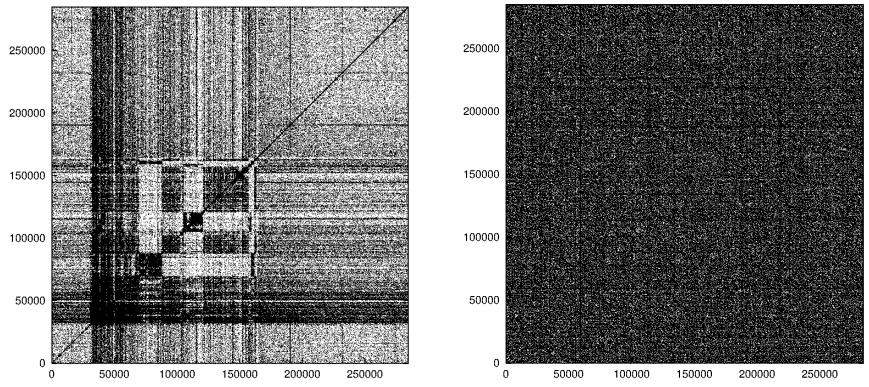
1) Fix *E* to the desired number of edges.
2) Try all reasonable *S1* and *S2* combinations near to *ceiling*($\log_2 N1$) and *ceiling*($\log_2 N2$) and minimize *F* by changing *a*, *b* and *c*. Get the optimal *S1* and *S2* values.
3) Fix *S1* and *S2* as well as *E*. For many different values of *a*, minimize *F* by changing *b* and *c*. Find the range of *a* values that gives optimal fit to the degree distributions.
4) Select the best Scree plot by changing the value of *a*.

Figure 3: The steps of AutoMAT

to be integers, and their values must satisfy 2^{S1} and 2^{S2} be the total number of nodes (including those with degrees of zero) in the R-MAT graph, good starting values for them are $ceiling(\log_2 N1)$ and $ceiling(\log_2 N2)$ ($N1$ and $N2$ are the numbers of nodes in the real graph). By doing an exhaustive search of all plausible $S1$ and $S2$ combinations, finding the minimized F value by varying a, b and c for each $S1$ and $S2$ pair, we get the optimal combination of $S1$ and $S2$. The next step is to fix $S1$ and $S2$ as well as E. For different a values, we use the Nelder-Mead simplex algorithm to fit the optimal b and c values. Since approximately (ignoring the effects of duplicate edge elimination) the two degree distributions are determined by $a + b$ and $a + c$, respectively. In practice, within a certain range of a values, the optimal b and c values are approximately linear functions of a. We get equally good fits for all such a, b and c combinations in terms of minimized F values. Now we can vary the value of a in order to pick a specific a, b and c combination to obtain the best fit for the first eigenvalues of the adjacency matrix. These steps are summarized in Figure 3.

3.3 A-plots

Is it possible to plot an adjacency matrix so that we can visually inspect a graph? If we naively plot the adjacency matrix we get plots like the ones shown in Figure 4. In Figure 4(a) the matrix is plotted without any sorting of the nodes. Though there seem to be some patterns in it, we can not really tell much from the plot with all the details of data collection and preprocessing mangled in there. Not surprisingly, if we randomize the nodes for plotting the adjacency matrix we just see a largely homogeneous block (Figure 4(b)). So if we want to get anything interesting out of plots of adjacency matrices we need to order the nodes in some way. It turns out that interesting patterns do show up when we order the nodes properly, say, by their network values. The "A-plots" (for "Adjacency matrix-related plots") consist 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 versus its rank of network value (D-RV plot, for Degree versus Rank



(a) (b)

Figure 4: Plots of the adjacency matrix of a router connection graph. (a) shows the adjacency matrix plotted without any sorting of the nodes. (b) shows the adjacency matrix with the nodes in random order.

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). Interesting patterns in these plots readily reveal certain properties of the graph. We propose these as valuable tools for an overall view of an undirected graph.

4 Criteria for Tests

We need to compare graphs generated by the R-MAT model against real-world graphs. The criteria for comparison and our reasons for choosing them are listed below:

(1) Degree plots: These are the plots for the count of nodes with in-degree/out-degree k ($C_{in,k}/C_{out,k}$), versus the degree k. That is, these are the plots of the probability density functions of the in-degree (out-degree) of nodes in graphs. This is the easiest criterion to justify: there is an overwhelming number of real power-law graphs [5]. In these graphs, the degree plot will come out as a straight line. Even for the graphs that deviate from power laws and show a tilt (see Figure 1), the log-log scale degree plot should come out as a truncated parabola, as "DGX" dictates. In either case, a good graph generator should match the degree plot of the target real graph.

(2) Scree plot: This is the plot of the first eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_i, \ldots$ of the adjacency matrix, versus their rank i, again in log-log scale. The reason for choosing this criterion is that often the eigenvalues follow a power law [17]. Moreover, the eigenvalues contain much information about the connectivity of the graph and the relative sizes of its connected or near-connected components. We choose to fill with zeros the diagonal of the adjacency matrix. Filling it with 1's is another reasonable choice, but the impact on the eigenvalues was very small.

(3) Network values of nodes: The elements $v_{1,i}$ in the first eigenvector $\mathbf{v}_1$ of the adjacency matrix correspond

to the "network values" of nodes in an undirected graph [29]. In a directed graph, the first left eigenvector $\mathbf{u}_1$ and the first right eigenvector $\mathbf{v}_1$ give the "hub" and "authority" values of nodes in a graph [12]. Thus, we do the SVD, estimate the first left- and first right-eigenvectors, sort their elements in decreasing order, and plot the values against their rank, in log-log scale.

There are lots of other potential criteria we can use to compare an R-MAT generated graph with a real-world one. Some of the choices are:

(1) "Hop-plot" and the effective diameter: The neighborhood function $F(h)$ [25] is the number of pairs of nodes that can reach each other within h hops or less. The "hop-plot" is the number of pairs $F(h)$, versus h. The "effective diameter" is defined as the number of hops in which 90% of the reachable pairs of nodes are reached [33]. The "small world phenomenon" states that real graphs have small diameters, which a successful graph generator should match. Moreover, the neighborhood function shows how well connected a graph is.

(2) Min-cut: A min-cut of a graph $G = (V, E)$ is a partition of the set of vertices V into two sets: V_1 and $V - V_1$ 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. The min-cut sizes of a graph and its subgraphs reflect the intrinsic dimensionality and other properties of the graph [31].

(3) Edge stress distribution: When we send one unit of traffic through the shortest path between every pair of nodes in a graph, the traffic that goes through a particular edge is its stress [18]. Stress shows the importance of an edge during the communication.

(4) Clustering coefficient: The clustering coefficient of a node is the ratio between the number of edges among its immediate neighbors and the maximum number of possible edges among those neighbors. Real-world social networks typically show much higher clustering coefficients than random graphs [5].

(5) Distribution of bipartite cores: A bipartite core C_{ij} has all possible connections between a group of i nodes and another group of j nodes. The distributions of bipartite cores of a graph of webpages were shown to follow certain power laws [20].

Our AutoMAT parameter fitting tool seeks to match the degree distributions and the Scree plot by changing the input parameters of R-MAT. In Section 5 we will show these fitting results as well as comparing the final R-MAT graphs with their real-world counterparts using the network value plots. Other criteria are also used for evaluating R-MAT graphs (not included in this work, see Appendix Section B for details).

5 Experiments

The questions we wish to answer through experiments are:

- **Q1:** How well does AutoMAT work? How do R-MAT generated graphs compare with real graphs for (a) directed, and (b) bipartite graphs?
- **Q2:** How can A-plots be used for analyzing large graphs?

The real-world datasets we used for the experiments are:

- *Epinions:* This is a graph of who-trusts-whom from `www.epinions.com`. It is a directed graph with $75,879$ nodes and $508,837$ edges.
- *Clickstream:* This is a bipartite graph obtained from an ISP which collects information about Internet users' browsing behavior. There are $23,396$ users, $199,308$ URLs and $952,190$ edges in this graph.
- *Router:* This is an undirected graph of network routers, obtained from `www.isi.edu/scan/mercator/maps.html`. It is an undirected graph with $284,805$ nodes and $898,492$ edges.

5.1 AutoMAT R-MAT graphs

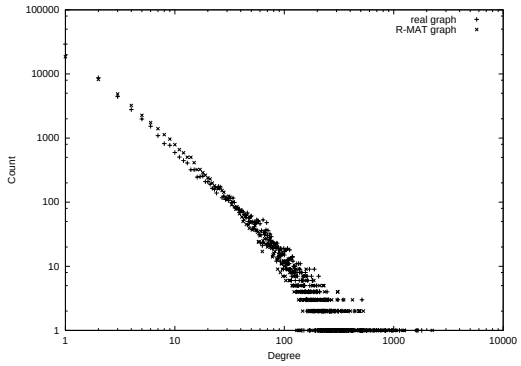
Epinions: We show the indegree, outdegree, eigenvalue and eigenvector plots for this dataset in Figure 5. The parameters used for R-MAT are produced by AutoMAT: $a = 0.550, b = 0.228, c = 0.212, S1 = 17, S2 = 17, E = 508837$.

Observations: The R-MAT graph shows excellent match to the real dataset in both indegree distribution and outdegree distribution. The match for the Scree plot is also pretty good. Since we do not try to match the network values of the R-MAT graph to the real dataset in AutoMAT, the fact that the network value plots also match very well is a pleasant surprise.

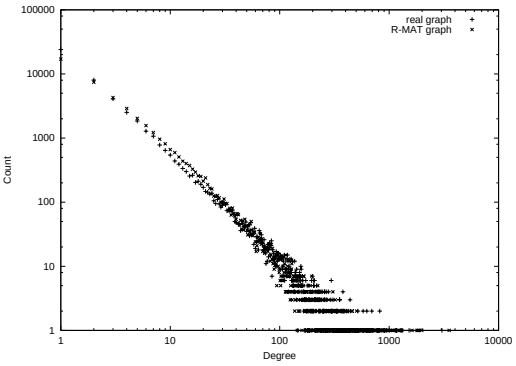
Clickstream: The results are shown in Figure 6. The parameters obtained using AutoMAT are: $a = 0.675, b = 0.00309, c = 0.181, S1 = 15, S2 = 23, E = 952190$.

Observations: We see that the outdegree of the real dataset follows a DGX-like distribution while the indegree follows the power law. Still R-MAT is able to match both degree distributions very well. Any graph generator that is unable to generate a DGX degree distribution as well as a power law degree distribution will fail in generating a graph like this Clickstream dataset. The R-MAT graph matches the real dataset in the Scree plot reasonably well. Again, the network values match those of the real graph very well.

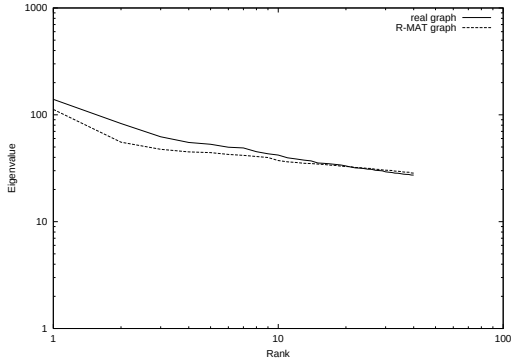
Thus, we see that the R-MAT generated graphs match the properties of real-world graphs very well for our tests. Some other tests that were performed on these R-MAT graphs against the real datasets are presented in Appendix Section B. It is impressive that these good matches are achieved with such a parsimonious model.



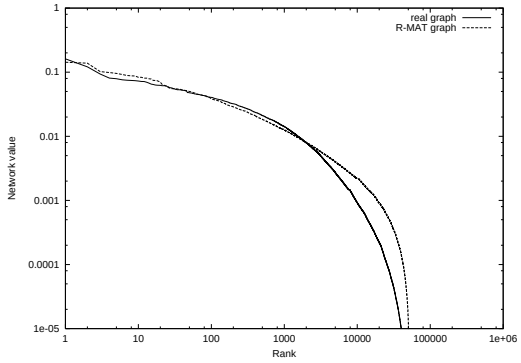
(a) Indegree distribution ($C_{in,k} \sim k$)



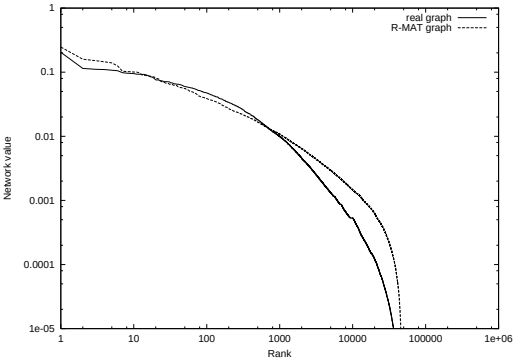
(b) Outdegree distribution ($C_{out,k} \sim k$)



(c) Scree plot ($\lambda_i \sim i$)

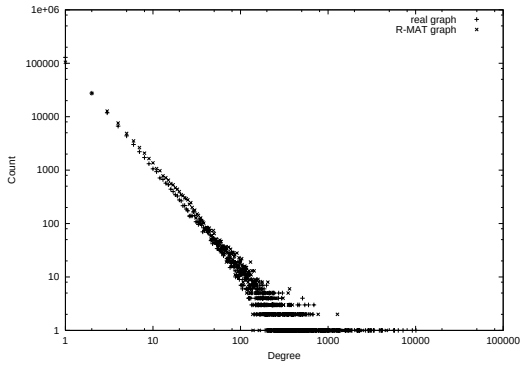


(d) Plot of the "hub" network values

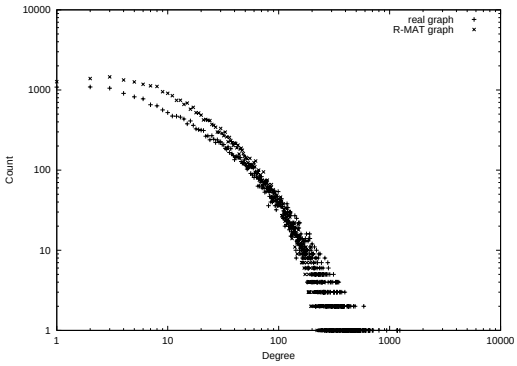


(e) Plot of the "authority" network values

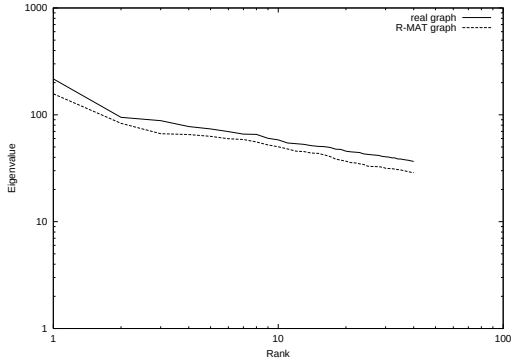
Figure 5: Epinions dataset: These are the indegree distribution, outdegree distribution, the Scree plot, and the network value plots. The crosses and dashed lines represent the R-MAT graph, while the pluses and solid lines represent the real graph.



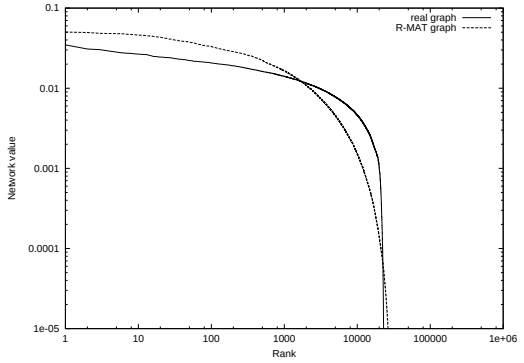
(a) Indegree distribution ($C_{in,k} \sim k$)



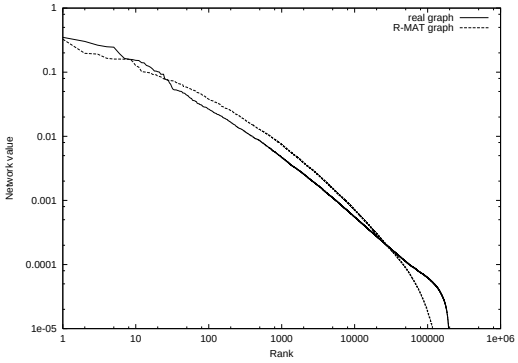
(b) Outdegree distribution ($C_{out,k} \sim k$)



(c) Scree plot ($\lambda_i \sim i$)

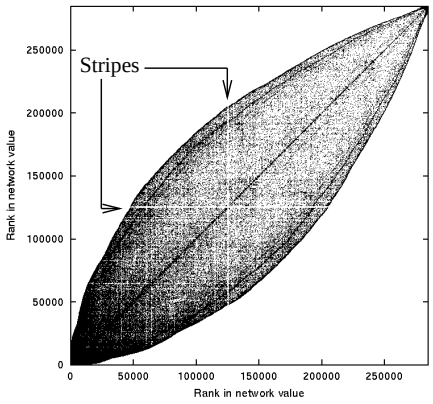


(d) Plot of the "hub" network values

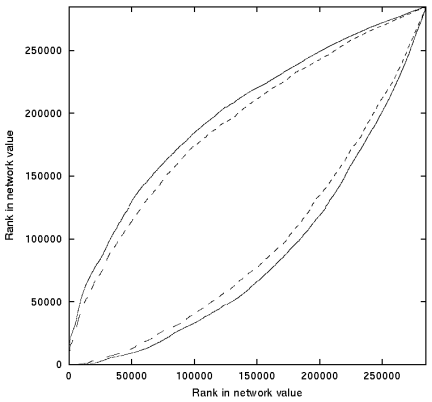


(e) Plot of the "authority" network values

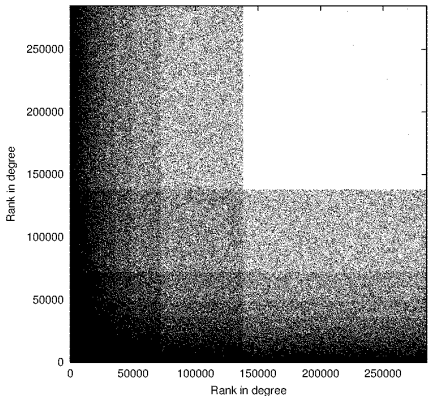
Figure 6: Clickstream dataset: These are the indegree distribution, outdegree distribution, the Scree plot, and the network value plots. The crosses and dashed lines represent the R-MAT graph, while the pluses and solid lines represent the real graph.



(a) RV-RV plot



(b) Estimated boundaries from Eqs 2,3



(c) RD-RD plot

Figure 7: A-plots for the Router graph: (a) shows the RV-RV plot. The "boundary lines" and the "secondary lines" in the RV-RV plot can be calculated from the first eigenvalue and eigenvector of the adjacency matrix, as are shown in (b) (solid lines are for the "boundary lines" and dashed lines are for the "secondary lines"). (c) shows the RD-RD plot.

5.2 A-plots

Figures 7 and 8 show A-plots for the Router dataset. Figure 7 shows the RV-RV and RD-RD plots, and Figure 8 shows the D-RV plot under different scalings. We can make the following observations:

Observation 1 ("Water-Drop") *The RV-RV plot has a clean and smooth oval-shaped boundary for the edges in the graph.*

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 (See Appendix Section C for details) of a matrix imply that

$$I_i = \frac{1}{\lambda_1} \times I_j \tag{2}$$

where λ_1 is the largest eigenvalue of the adjacency matrix of the graph. Therefore the boundary of edges in the RV-RV plot can be calculated from the first eigenvalue and eigenvector (See Appendix Section C for details). Figure 7(b) shows just this; the solid lines represent edges involving one-degree nodes. These are obviously the boundary lines 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" (in the sense of high network value) neighbor. Then, $I_i \geq \frac{1}{\lambda_1} I_j$. Therefore all edges are confined within the boundary in the RV-RV plot.

Observation 2 (Nested Water-Drops) *There is 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(See Appendix Section C for details):

$$I_i = \frac{2}{\lambda_1} \times I_j \tag{3}$$

The dashed lines in Figure 7(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 3 (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 4 (Isolated Components) *There are dots in the largely empty white square in the corner of the RD-RD plot.*

Explanation: These dots represent connections between one-degree nodes. All dots (edges) in this area correspond to two-node isolated components.

Observation 5 (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 7(a) and 8(a, b) respectively).

Observation 6 (High degree does not mean high Network Value) *Figure 8(c) shows several points in the D-RV plot having high degrees, but low network values (and thus low ranks). Thus, high degree does not imply high network importance.*

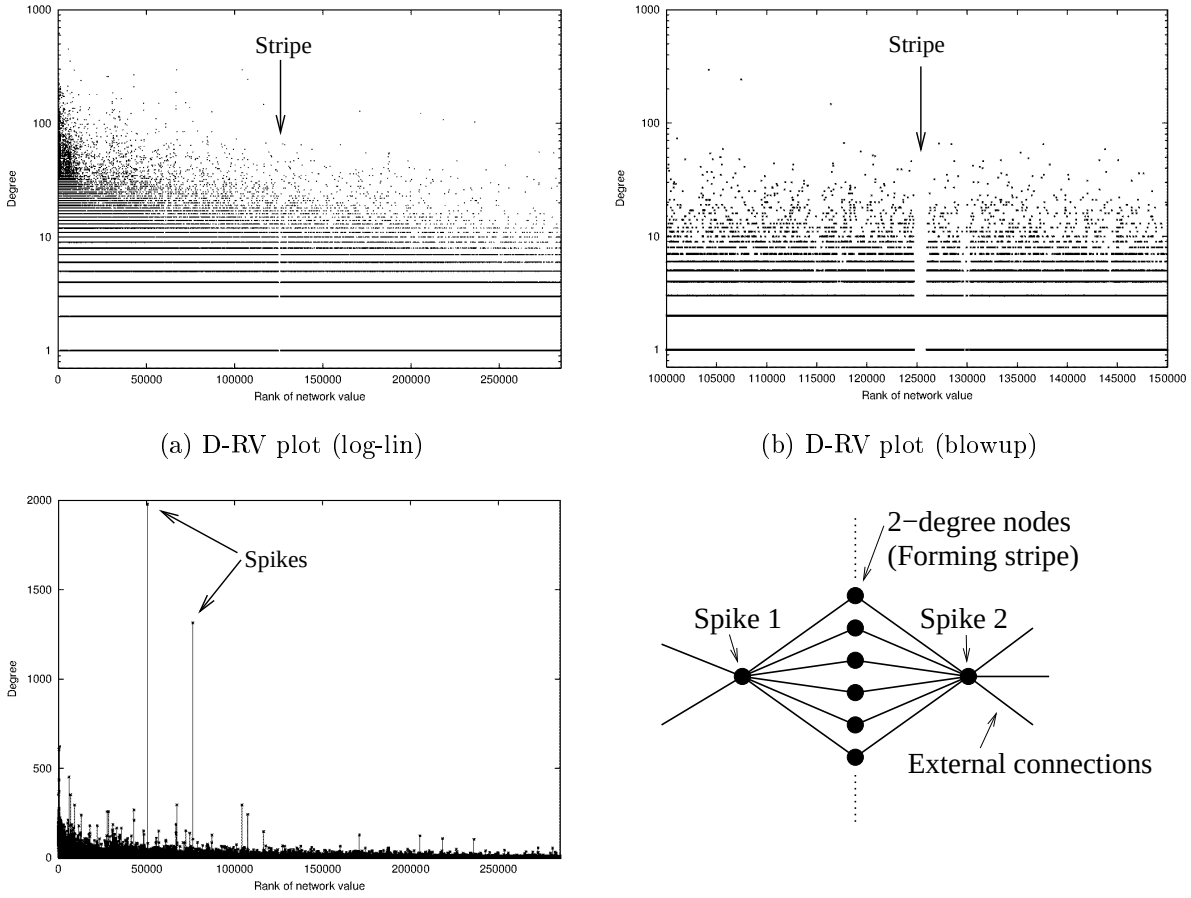
Explanation: The D-RV plot in Figure 8(c) shows that the two highest-degree nodes have relatively low network values. This is counter-intuitive - how could it possibly be the case, in a power-law graph? Is it a data collection error?

The answer is surprising, and actually it also explains the most prominent white stripe in Figure 8(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 8(d). 'Spike1' and 'Spike2', being away from the core of the network, have much lower network values than what their high degrees would promise. Their satellites (all the 2-degree nodes connected to them) have identical network values, which cause the most prominent white stripe in Figure 8(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 spotted this strange pattern, which would go undetected if we only used the traditional tools like the degree distribution plots.

6 Conclusions

Novel tools for mining large graphs are proposed. The major contributions of this work are:

- The R-MAT graph generator: Not only can R-MAT be tuned to match a given real-world graph in degree distributions (power law distributions as well as DGX distributions) and the Scree plot, the R-MAT graphs also match real graphs in network values.



(a) D-RV plot (log-lin)

(b) D-RV plot (blowup)

(c) D-RV plot (lin-lin)

(d) Spike-and-stripe links

Figure 8: A-plots for the Router dataset: (a) shows the D-RV plot, and (b) shows a blowup of a portion, clearly demonstrating the "white stripes" phenomenon. (c) shows the D-RV plot in the linear-linear scale; nodes with the highest degree do not have the highest network value. (d) shows the actual network configuration of routers involved in the stripe and spikes.

- AutoMAT parameter estimation method for the R-MAT graph generator: The method works well in finding the input parameters of the R-MAT graph generator so that the R-MAT graph matches a given real graph in degree distributions and the Scree plot.
- A-plots: These plots provide new viewpoints for inspecting large graphs. We noticed some striking patterns ("water-drops", white stripes, spikes, etc.) and we showed how to interpret them.

Thanks to the proposed tools, we were able to make several interesting observations. We showed how A-plots can be used to spot outliers, and make observations about the degree of nodes, by simply looking at the adjacency matrix in different ways. When its input parameters are estimated with AutoMAT, R-MAT can match a given real graph with respect to the degree distributions, the Scree plot, and by virtue of the model, the network values of the nodes and the diameter (Appendix Section B) of the graph as well. This is a great advantage of R-MAT over all previous graph generators, which mostly only seek to generate skewed degree distributions, if at all, with some simple mechanisms. R-MAT is the first graph generator that focuses on matching additional criteria. The R-MAT model is successful in generating directed graphs and bipartite graphs that match real-world ones.

7 Future Research Directions

Further work could extend the application of R-MAT for generating weighted graphs and undirected graphs. In fact, given the way R-MAT works, it can naturally generate weighted graphs. The number of edges dropped into any 1×1 cell in the adjacency matrix could be viewed as a weight for the graph (currently we are eliminating all the duplicate edges for generating unweighted graphs). The distribution of the weights of the edges thus generated will be skewed. This is the case for some weighted datasets we have checked. Therefore we believe R-MAT should be a promising model for weighted graphs. This aspect of R-MAT is not matched by any previous graph generating models. With some minor modifications, R-MAT could be used for generating undirected graphs, too. A simple option is to do the following: (a) throw away the half of the square adjacency matrix below the diagonal; and (b) mirror the upper half to it. This way we get a symmetric adjacency matrix for an undirected graph. Thus R-MAT may become a powerful graph generating model that handles self graphs as well as bipartite graphs, directed graphs as well as undirected graphs, and weighted graphs as well as unweighted graphs.

More criteria need to be tested to explore R-MAT further. Some of this work has already been done by other researchers (see Appendix Section B). Testing R-MAT graphs against real graphs with many criteria is useful at this point.

Theoretical analysis establishes that the eigenvector of the second smallest eigenvalue of an undirected graph's Laplacian matrix (defined as $L = diag(d) - A$, where d is the vector of degrees of nodes, A is the adjacency matrix of the graph) contains information about separating a graph into two approximately equal "communities"

with minimal number of cross-community edges between them (related to the min-cut of a graph) [14]. Similar conclusions go for other eigenvectors. If this method can be applied to mining large real-world graphs, we could have a new way of generating A-plots as well as getting the min-cut sizes for graphs. The A-plots made this way may give us better ways of getting a general idea of how the graph is connected.

References

[1] W. Aiello, F. Chung, and L. Lu, "A random graph model for massive graphs," *STOC*, 2000, pp. 171-180.

[2] W. Aiello, F. Chung, and L. Lu, "Random evolution in massive graphs," *FOCS*, 2001, pp. 510-519.

[3] R. Albert and A.-L. Barabasi, "Emergence of scaling in random networks," *Science*, 1999, pp. 509-512.

[4] R. Albert and A.-L. Barabasi, "Topology of evolving networks: local events and universality," *Physical Review Letters*, 2000, pp. 5234-5237.

[5] R. Albert and A.-L. Barabasi, "Statistical mechanics of complex networks," *Reviews of Modern Physics,74*, 2002, pp. 47.

[6] Z. Bi, C. Faloutsos, and F. Korn, "The DGX distribution for mining massive, skewed data," *KDD*, 2001, pp. 17-26.

[7] D. Blandford, G. E. Blelloch, and I. Kash, "Compact representations of separable graphs," *ACM/SIAM SODA*, 2003.

[8] B. Bollobas and O. Riordan, "The diameter of a scale-free random graph," *Combinatorica*, 2002.

[9] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, A. Tomkins and J. Wiener, "Graph structure in the web: experiments and models," *WWW Conf.*, 2000.

[10] T. Bu and D. Towsley, "On distinguishing between internet power law topology generators," *INFOCOM*, 2002.

[11] S. Chakrabarti, "Mining the web: Discovering knowledge from hypertext data.," Morgan Kaufmann, 2002.

[12] S. Chakrabarti, B. E. Dom, S. R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. Kleinberg, "Mining the link structure of the world wide web," *IEEE Computer, 32*, 1999, pp. 60-67.

[13] H. Chen, J. Schroeder, R. Hauck, L. Ridgeway, H. Atabakhsh, H. Gupta, C. Boarman, K. Rasmussen, and A. Clements, "Coplink connect: Information and knowledge management for law enforcement," *Decision Support Systems, 34*, 2002, pp. 271-285.

[14] I. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," *KDD*, 2001, pp. 269-274.

[15] P. Erdos and A. Renyi, "On the evolution of random graphs.," *Publication of the Mathematical Institute of the Hungarian Academy of Science, 5*, 1960, pp. 17-67.

[16] A. Fabrikant, E. Koutsoupias, and C. H. Papadimitriou, "Heuristically optimized trade-offs: A new paradigm for power laws in the internet (extended abstract)," *STOC*, 2002.

[17] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology.," *SIGCOMM*, 1999, pp. 251-262.

[18] C. Gkantsidis, M. Mihail, and E. Zegura, "Spectral analysis of internet topologies," *SIGCOMM*, 2003.

[19] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *Technical Report TR 95-035*, 1995.

[20] J. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins," "The web as a graph: Measurements, models and methods." *Proceedings of the International Conference on Combinatoric and Computing*, 1999.

[21] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, "Extracting large-scale knowledge bases from the web," *VLDB*, 1999.

[22] A. Medina, I. Matta, and J. Byers, "On the origin of power laws in internet topologies," *SIGCOMM*, 2000, pp. 18-34.

[23] S. Milgram, "The small world problem," *Psychology Today, 2*, 1967, pp. 60-67.

[24] C. R. Palmer, P. B. Gibbons, and C. Faloutsos, "Anf: A fast and scalable tool for data mining in massive graphs," *SIGKDD*, 2002.

[25] C. R. Palmer and J. G. Steffan, "Generating network topologies that obey power laws," *GLOBECOM*, 2000.

[26] D. M. Pennock, G. W. Flake, S. Lawrence, E. J. Glover, and C. L. Giles, "Winners don't take all: Characterizing the competition for links on the web ," *Proceedings of the National Academy of Sciences, 99*, 2002, pp. 5207-5211.

[27] R. Ramakrishnan and J. Gehrke, "Database management systems," McGraw-Hill. Third edition, 2002.

[28] S. Redner, "How popular is your paper? an empirical study of the citation distribution," *European Physical Journal B, 4*, 1998, pp. 131-134.

[29] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," *SIGKDD*, 2002, pp. 61-70.

[30] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design," *IEEE Internet Computing Journal, 6*, 2002.

[31] A. L. Rosenberg and L. S. Heath, "Graph separators, with applications," Kluwe Academic/Plenum Publishers, 2001.

[32] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network topologies, power laws, and hierarchy (Technical Report 01-746)," University of Southern California, 2001.

[33] S. L. Tauro, C. Palmer, G. Siganos, and M. Faloutsos, "A simple conceptual model for the internet topology," *Global Internet, San Antonio, Texas*, 2001.

[34] S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical recipes in C: the art of scientific computing," Cambridge Univ Press, 1993.

[35] C. Wang, J. C. Knight, and M. C. Elder, "On computer viral infection and the effect of immunization," *ACSAC*, 2000, pp. 246-256.

[36] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal of Selected Areas in Communications, 6*, 1988, pp. 1617-1622.

Appendix

A Details about Using a, b, c, d Values in R-MAT

Here we describe a subtle problem in R-MAT as well as our solution to it. For the recursive steps of R-MAT, if we always use the exact a, b, c and d values, the degree distributions of the resulted graph will show "wiggles". An example is shown in Figure 9. To eliminate the "wiggles", we vary the values of a, b, c and d around their original values randomly for each recursive step in R-MAT. There are different ways to do this in practice. However, we need to be careful about choosing the method, because the randomization we apply to vary a, b, c and d can result in different results in the final degree distributions. It is actually the case that if we apply the randomization in significant enough amplitudes to remove the "wiggles" then these random numbers have a very significant effect on the resulted degree distribution. Hence, we need to fix the random numbers involved for different a, b, c and d values if we want to apply any fitting methods to tune the input parameters of R-MAT in order to match a given real graph. We currently vary the a, b, c and d values by having $a_i = a(1+\epsilon_a), b_i = b(1+\epsilon_b), c_i = c(1+\epsilon_c), d_i = d(1+\epsilon_d)$ and use the normalized values of a_i, b_i, c_i and d_i for dropping edges in recursive step i. $\epsilon_a, \epsilon_b, \epsilon_c$ and ϵ_d are generated from a uniform distribution in $[-0.6, +0.6]$ for different steps and they are fixed for all R-MAT runs. It works well for removing the "wiggles" and also allows AutoMAT to do the fittings on the input parameters of R-MAT.

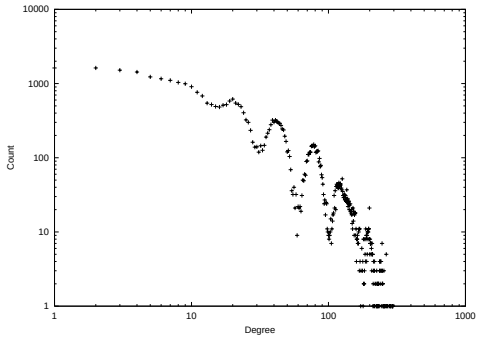


Figure 9: Using the exact values of a, b, c and d for dropping the edges at each recursive step leads to "wiggles" in the degree distributions of the R-MAT graph.

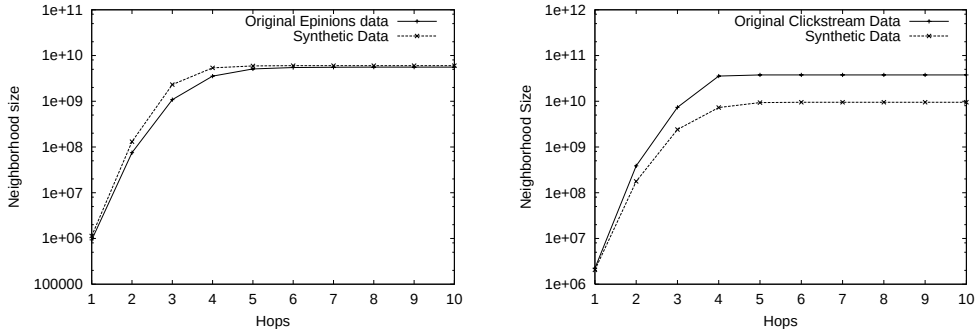
B Other Criteria for Testing R-MAT Graphs

The following is a brief summary of some work done exclusively by other researchers (Deepayan Chakrabarti et al. for the Hop-plots and stress distributions, and Daniel Blandford et al. for the min-cuts). We will present them here for completeness of the tests on R-MAT graphs. These experiments further support that R-MAT is a

realistic generator for large directed and bipartite graphs.

B.1 Hop-plot and effective diameter

Figure 10 compares the Hop-plots for the R-MAT graphs to the real graphs for the Epinions and the Clickstream datasets. To compare the effective diameter for directed graphs, the directed edges were treated as if they were undirected. We can see that the R-MAT graphs match the Hop-plots for real graphs very well, that they have small effective diameters (3-4) just like real graphs do.



(a) (b)

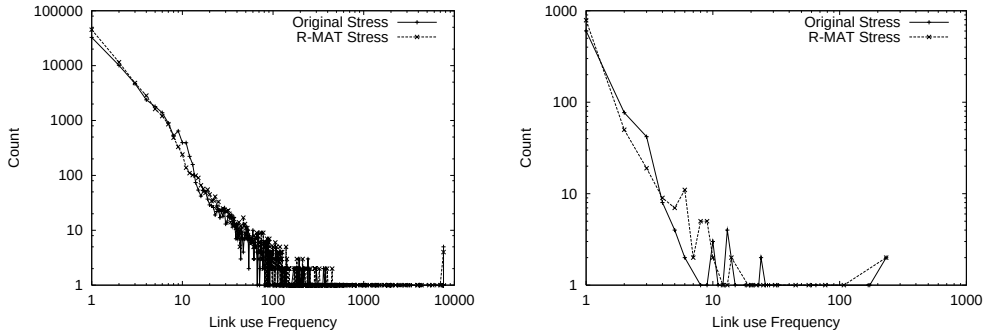
Figure 10: Hop-plots for the Epinions (a) and Clickstream (b) datasets. The dashed lines represent the R-MAT graphs, while the solid lines represent the real graphs.

B.2 Stress distribution

Figure 11 shows the stress distributions of the Epinions and Clickstream graphs, along with those of the graphs generated using AutoMAT. To make these plots, 100 starting nodes were randomly chosen for flooding the network. The process was stopped when each packet had reached 1000 nodes. It is observed in these plots as well as in plots for various other datasets that the count of nodes with a given stress value seems to follow a power-law for real graphs, except at the highest stress value. The nodes corresponding to these high-stress values might constitute the "core" of the network. We can see in Figure 11 that the R-MAT stress distributions match the real graphs very well.

B.3 Min-cut sizes

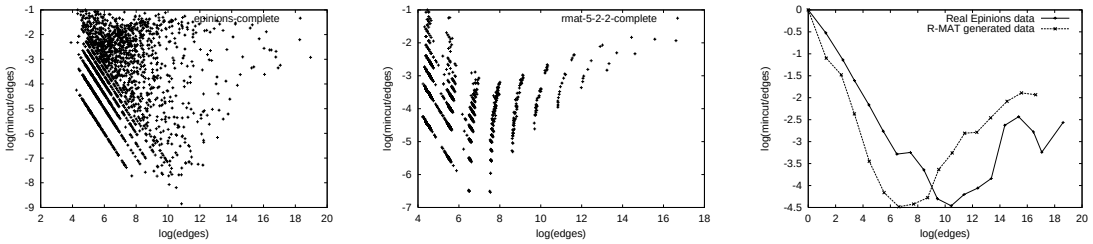
The min-cut sizes were plotted for the Epinions dataset and the corresponding AutoMAT generated graph (Figure 12). For each graph the Metis graph partitioning library [19] was used to generate a separator, as is described



(a) (b)

Figure 11: Stress distributions for the Epinions (a) and Clickstream (b) datasets. The dashed lines represent the R-MAT graphs, while the solid lines represent the real graphs.

by Blandford, etc. [7]. Figure 12 shows that the shapes of the min-cut plots are similar for the AutoMAT graph and the real Epinions graph.



(a) Epinions min-cuts (b) AutoMAT min-cuts (c) Averaged min-cuts

Figure 12: Min-cut plots for the Epinions dataset and the corresponding R-MAT graph.

C Details About Calculating Boundary lines in RV-RV Plots

For an undirected graph with n nodes and adjacency matrix A, let I_i denote the network value of node i. Let λ_1 and $\mathbf{v}_1$ be the first eigenvalue and the first eigenvector of A. We have:

Lemma 1: *If node i is connected to nodes $j1, j2, \ldots, jk$ ($k \geq 1$), then $I_i = \frac{1}{\lambda_1} \sum_{l=1}^{k} I_{jl}$.*

Proof 1: *Since λ_1 and $\mathbf{v}_1$ are the first eigenvalue and first eigenvector of the square matrix A. We have $A\mathbf{v}_1 = \lambda_1 \mathbf{v}_1$. The result of both sides is a $n \times 1$ vector. Note that $I_i = v_{1,i}$ and $A_{ij} = 1$ only when nodes i and j are connected. The i'th element of this $n \times 1$ vector as is calculated at the left side is $\sum_{l=1}^{k} I_{jl}$. At the right side, it is $\lambda_1 I_i$. Therefore $I_i = \frac{1}{\lambda_1} \sum_{l=1}^{k} I_{jl}$.*

As a result of **Lemma 1**, if node i is only connected with node j, we have Equation 2. If node i is connected to two nodes of equal network importance, we have Equation 3.

Assuming that we have sorted the elements of vector $\mathbf{v}_1$ in decreasing order, let f be the mapping from the rank of a node i to its network value $I_i = v_{1,i}$, i.e., $I_i = f(i)$. f is a discrete non-increasing function by this definition, however, to simplify the explanations, we are going to use it as if it is a continuous decreasing function below, just to give the intuition. If we know that node i is a one-degree node, then the corresponding edge for node i is going to be at position $(i, f^{-1}(\lambda_1 I_i))$ and $(f^{-1}(\lambda_1 I_i), i)$ in the RV-RV plot, since the network importance of the node that node i is connected to is $\lambda_1 I_i$. For calculating the "water-drop" boundaries we just find the positions of these edges for many different i's. Similarly, to calculate the secondary lines we calculate points $(i, f^{-1}(\frac{1}{2}\lambda_1 I_i))$ and $(f^{-1}(\frac{1}{2}\lambda_1 I_i), i)$.