# Tools for Large Graph Mining

by

Deepayan Chakrabarti

Submitted to the Center for Automated Learning and Discovery
in partial fulfillment of the requirements for the degree of

## Doctor of Philosophy

at

## Carnegie Mellon University

June, 2005

Thesis Committee:
    Christos Faloutsos (`christos@cs.cmu.edu`)
    Guy Blelloch (`guyb@cs.cmu.edu`)
    Christopher Olston (`olston@cs.cmu.edu`)
    Jon Kleinberg (`kleinber@cs.cornell.edu`, External member)

**Abstract**

Graphs show up in a surprisingly diverse set of disciplines, ranging from computer networks to sociology, biology, ecology and many more. How do such "normal" graphs look like? How can we spot abnormal subgraphs within them? Which nodes/edges are "suspicious?" How does a virus spread over a graph? Answering these questions is vital for outlier detection (such as terrorist cells, money laundering rings), forecasting, simulations (how well will a new protocol work on a realistic computer network?), immunization campaigns and many other applications.

We attempt to answer these questions in two parts. First, we answer questions targeted at *applications*: what patterns/properties of a graph are important for solving specific problems? Here, we investigate the propagation behavior of a computer virus over a network, and find a simple formula for the *epidemic threshold* (beyond which any viral outbreak might become an epidemic). We find an "information survival threshold" which determines whether, in a sensor or P2P network with failing nodes and links, a piece of information will survive or not. We also develop a scalable, parameter-free method for finding *groups* of "similar" nodes in a graph, corresponding to homogeneous regions (or *CrossAssociations*) in the binary adjacency matrix of the graph. This can help navigate the structure of the graph, and find un-obvious patterns.

In the second part of our work, we investigate recurring patterns in real-world graphs, to gain a deeper understanding of their structure. This leads to the development of the R-MAT model of graph generation for creating synthetic but "realistic" graphs, which match many of the patterns found in real-world graphs, including power-law and lognormal degree distributions, small diameter and "community" effects.

# 1 Introduction

Informally, a graph is a set of nodes, and a set of edges connecting some node pairs. In database terminology, the nodes represent individual entities, while the edges represent relationships between these entities. This formulation is very general and intuitive, which accounts for the wide variety of real-world datasets which can be easily expressed as graphs. Some examples include:

- *Computer Networks:* The Internet topology (at both the Router and the Autonomous System (AS) levels) is a graph, with edges connecting pairs of routers/AS. This is a self-graph, which can be both weighted or unweighted.

- *Ecology:* Food webs are self-graphs with each node representing a species, and the species at one endpoint of an edge eats the species at the other endpoint.

- *Biology:* Protein interaction networks link two proteins if both are necessary for some biological process to occur.

- *Sociology:* Individuals are the nodes in a social network representing ties (with *labels* such as friendship, business relationship, trust, etc.) between people.

In fact, any information relating different entities (an $M : N$ relationship in database terminology) can be thought of as a graph. This accounts for the abundance of graphs in so many diverse topics of interest, most of them large and sparse.

There is, however, a dichotomy in graph mining applications: we can answer specific queries on a particular graph, or we can ask questions pertaining to real-world graphs in general. Examples of the former would include questions such as: "Find natural partitions of nodes in a given graph," or "Determine outlier edges in this graph," and so on. For the latter, we ask questions such as: "What graph patterns or laws hold for (almost) all real-world graphs," or "How do graphs evolve over time, in general" and so on. The separation between the two is, however, not strict, and there are applications requiring tools from both sides. For example, in order to answer *"how quickly will viruses spread on the Internet five years in the future,"* we must have models for how the Internet will grow, how to generate a synthetic yet realistic graph of that size, and how to estimate the spread of viral infections on graphs.

In my thesis, I explore issues from both sides of this dichotomy. Since graphs are so general, these problems have been studied in several different communities, including computer science, physics, mathematics, physics and sociology. Often, this has led to independent rediscovery of the same concepts in different communities. In my work, I have attempted to combine these viewpoints and then improve upon them.

The specific problems I investigated in my research are as follows. The first three sections investigate applications of graph mining on *specific* graphs. In section 2, we analyze the problem of viral propagation in networks: *"Will a viral outbreak on a computer network spread to epidemic proportions, or will it quickly die out?"* We investigate the dependence of viral propagation on the network topology, and derive a simple and accurate *epidemic threshold* that determines if a viral outbreak will die out quickly, or survive for long in the network. In section 3, we study information survival in sensor networks: Consider a piece of information being spread within a sensor or P2P network with failing links and nodes. *What conditions on network properties determine if the information will survive in the network for long, or die out quickly?* In section 4, we answer the question: *How can we automatically find natural node groups in a large graph?* Our emphasis here is on a completely automatic and scalable system: the user only needs to feed in the graph dataset, and our Cross-associations algorithm determines both the number of clusters and their memberships. In addition, we present automatic methods for detecting outlier edges and for computing "inter-cluster distances."

Next, in section 5, we discuss issues regarding real-world graphs in general: *How can we quickly generate a synthetic yet realistic graph? How can we spot fake graphs and outliers?* We discuss several common graph patterns, and then present our R-MAT graph generator, which can match almost all these patterns using a very simple 3-parameter model. Finally, section 6 presents the conclusions of this thesis.

# 2    Epidemic thresholds in viral propagation

*"Will a viral outbreak on a computer network spread to epidemic proportions, or will it quickly die out?"*

3

The importance of this question in computer security applications is obvious. Our contributions, as detailed in [20], are in answering the following questions:

- *How does a virus spread?* Specifically, we want an analytical model of viral propagation, that is applicable for *any* network topology.

- *When does the virus die out, and when does it become endemic?* Conceptually, a tightly connected graph offers more possibilities for the virus to spread and survive in the network than a sparser graph. Thus, the same virus might be expected to die out in one graph and become an epidemic in another. What features of the graph control this behavior? We find a simple closed-form expression for the "epidemic threshold" below which the virus dies out, but above which it can become an epidemic.

- *Below the threshold, how quickly will the virus die out?* A logarithmic decay of the virus might still be too slow to have practical impact.

We will first present our mathematical model of viral propagation in Section 2.1, and then derive the epidemic threshold condition in Section 2.2. Finally, we experimentally demonstrate the accuracy of our model in Section 2.3.

## 2.1   Model of Viral Propagation

We use the SIS model of viral infection on undirected graphs [3, 20]. Here, each node can be in one of two states: healthy but suceptible (S) to infection, or infected (I). For ease of exposition, we assume very small discrete timesteps of size $\Delta t \rightarrow 0$. Within a $\Delta t$ time interval, an infected node $i$ tries to infect its neighbors with probability $\beta$. At the same time, $i$ may be cured (and thus, susceptible again) with probability $\delta$.

The full Markov Chain for this model is exponential in size, and intractable for large $N$. Hence, we use the "independence" assumption, that is, the states of the neighbors of any given node are independent. Thus, we replace the problem with Equation 1 (our "non-linear dynamical system" discussed below), with only $N$ variables instead of $2^N$ for the full Markov chain. This makes the problem tractable, and we can find closed-form solutions. Note that the independence assumption places no constraints on network topology; also, the "independence assumption" is empirically very close to the full Markov Chain.

Let the probability that a node $i$ is infected at time $t$ by $p_i(t)$. A node $i$ is healthy at time $t$ if it did not receive infections from its neighbors at $t$ *and* $i$ was uninfected at time-step $t-1$, or was infected but was cured at $t$. Denoting the probability of a node $i$ being infected at time $t$ by $p_i(t)$:

$$1 - p_i(t) \;=\; (1 - p_i(t-1)) \cdot \zeta_i(t) + \delta \cdot p_i(t-1) \cdot \zeta_i(t) \quad i = 1 \ldots N \qquad (1)$$

where $\zeta_i(t)$ is the probability that a node $i$ will not receive infections from its neighbors in the next time-step, and by the independence assumption,

$$
\begin{aligned}
\zeta_i(t) \;&=\; \prod_{j:neighbor\ of\ i} \left( p_j(t-1)(1-\beta) + (1 - p_j(t-1)) \right) \\
&=\; \prod_{j:neighbor\ of\ i} (1 - \beta * p_j(t-1)) \qquad\qquad (2)
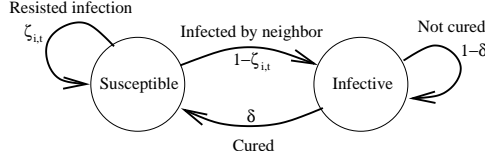\end{aligned}
$$

4

Figure 1: *The SIS model, as seen from a single node:* Each node, in each time step $t$, is either Susceptible (S) or Infective (I). A susceptible node $i$ is currently healthy, but can be infected (with probability $1 - \zeta_{i,t}$) on receiving the virus from a neighbor. An infective node can be cured with probability $\delta$; it then goes back to being susceptible. Note that $\zeta_{i,t}$ depends on the both the virus birth rate $\beta$ and the network topology around node $i$.

Equation 1 represents our *NLDS* (Non-Linear Dynamical System). Figure 1 shows the transition diagram.

## 2.2 The Epidemic Threshold

Using the *NLDS* equation, we can determine the *epidemic threshold* $\tau_{NLDS}$ which determines whether a viral outbreak dies out or becomes an epidemic under *NLDS*. Specifically, $\tau_{NLDS}$ is a value such that (for *NLDS*)

$$\beta/\delta < \tau_{NLDS} \quad \Rightarrow \quad \text{infection dies out over time, } p_i(t) \to 0 \text{ as } t \to \infty \; \forall i$$
$$\beta/\delta > \tau_{NLDS} \quad \Rightarrow \quad \text{infection survives and becomes an epidemic}$$

Surprisingly, $\tau_{NLDS}$ depends only on one number: the largest eigenvalue of the graph.

**Theorem 1 (Epidemic Threshold).** *In* NLDS*, the epidemic threshold $\tau_{NLDS}$ for an undirected graph is*

$$\boxed{\tau_{NLDS} = \frac{1}{\lambda_{1,A}}} \tag{3}$$

*where $\lambda_{1,A}$ is the largest eigenvalue of the adjacency matrix* **A** *of the network.*

**Definition 1 (Score).** *Score $s = \frac{\beta}{\delta} \cdot \lambda_{1,A}$.*

Theorem 1 provides the conditions under which an infection dies out ($s < 1$) or survives ($s \geq 1$) in our dynamical system. We can ask another question: if the system is below the epidemic threshold, how *quickly* will an infection die out?

**Theorem 2 (Exponential Decay).** *When an epidemic is diminishing (therefore $\beta/\delta < \frac{1}{\lambda_{1,A}}$ and $s < 1$), the probability of infection decays at least exponentially over time.*

## 2.3 Experiments

We show simulation results on two datasets: Star-10K with $10,000$ nodes organised as a star graph, and the Oregon graph of network connections between Autonomous Systems (http://topology.eecs.umich.edu/data.html) with $11,461$ nodes and $32,730$ edges. All
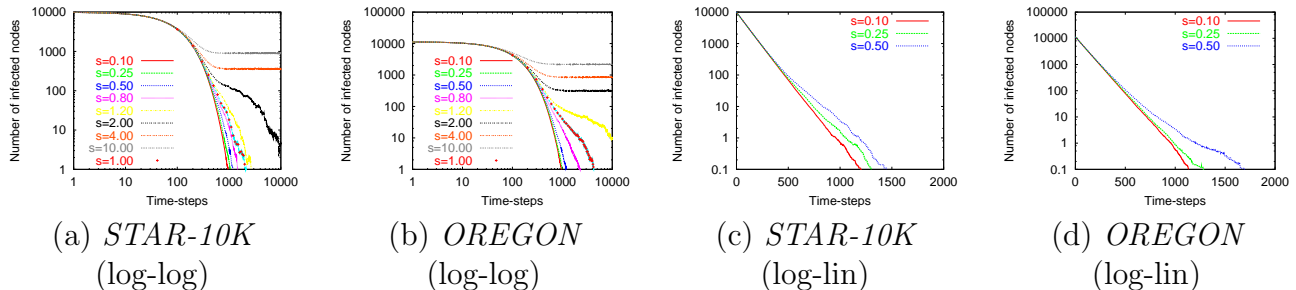
Figure 2: *Accuracy of* $\tau_{NLDS}$: The number of infected nodes is plotted versus time for various values of the score $s$. (a,b) Plots are shown on log-log scales, with the $s = 1$ case shown with the dotted line. When $s < 1$, the infection dies out quickly, while when $s > 1$, it survives in the graph. (c,d) Plots are shown in log-linear scales when $s < 1$ (below threshold). The plots are linear, implying exponential decay.

nodes are initially infected, and we plot the number of infected nodes over time. Figures 2(a,b) show that when $s < 1$, the infection dies out, but it survives when $s > 1$, agreeing with Theorem 1. Also, figures 2(c,d) show that the decay is exponential when $s < 1$, showing the correctness of Theorem 2.

Thus, Theorems 1 and 2 allow us to distinguish between viral epidemic and extinction.

# 3    Information survival in sensor and P2P networks

*"Consider a piece of information being spread within a sensor or P2P network with failing links and nodes. What conditions on network properties determine if the information will survive in the network for long, or die out quickly?*

Sensor and Peer-to-peer (P2P) networks have recently been employed in a wide range of applications, including oceanography, infrastructure monitoring and parking space tracking [13]. We look at the problem of survivability of information in a sensor or P2P network under node and link failures. For example, consider a sensor network where the communication between nodes is subject to loss (link failures), and sensors may fail (node failures). In such networks, we may want to maintain some static piece of information, or "datum", which, for the sake of exposition, we refer to as "Smith's salary". If only one sensor node keeps Smith's salary, that node will probably fail sometime and the information will be lost. To counter this, nodes that have the datum can broadcast it to other nodes, spreading it through the network and increasing its chances of survival in the event of node or link failures. *Under what conditions can we expect Smith's salary to survive in the sensor network?* The problem is similar to that for viral propagation, and we again use the non-linear dynamical systems approach.

## 3.1    Model for Information Propagation

As in section 2, we have a sensor/P2P/social network of $N$ nodes (sensors or computers or people) and $E$ directed links between them. Our analysis assumes very small discrete timesteps of size $\Delta t$, where $\Delta t \rightarrow 0$. Within a $\Delta t$ time interval, each node $i$ has a probability $r_i$ of trying to broadcast its information every timestep, and each link $i \rightarrow j$ has a probability
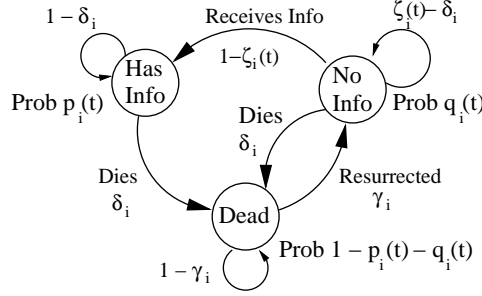
Figure 3: *Transitions for each node:* This shows the three states for each node, and the probabilities of transitions between states.

$\beta_{ij}$ of being "up", and thus correctly propagating the information to node $j$. Each node $i$ also has a node failure probability $\delta_i > 0$ (e.g., due to battery failure in sensors). Every dead node $j$ has a rate $\gamma_j$ of returning to the "up" state, but without any information in its memory (e.g., due to the periodic replacement of dead batteries).

Again, we will convert this into a dynamical system, and answer questions in that system. Let the probability of node $i$ being in the "Has Info" and "No Info" states at time-step $t$ be $p_i(t)$ and $q_i(t)$ respectively. The equations of the dynamical system turn out to be:

$$
\begin{aligned}
p_i(t) &= p_i(t-1)\,(1-\delta_i) \\
&\quad + q_i(t-1)\,(1-\zeta_i(t)) & (4) \\
q_i(t) &= q_i(t-1)\,(\zeta_i(t)-\delta_i) \\
&\quad + (1-p_i(t-1)-q_i(t-1))\,\gamma_i & (5) \\
\zeta_i(t) &= \Pi_{j=1}^{N}\,(1-r_j\beta_{ji}p_j(t-1)) & (6)
\end{aligned}
$$

From now on, we will only work on this dynamical system. Specifically, we want to find the condition for fast extinction under this system, where the expected number of "carriers" of information die off exponentially quickly over time.

## 3.2   Information Survival Threshold

Define $\mathbf{S}$ to be the $N \times N$ *system matrix*:

$$
\mathbf{S}_{ij} = \begin{cases} 1-\delta_i & \text{if } i=j \\ r_j\beta_{ji}\dfrac{\gamma_i}{\gamma_i+\delta_i} & \text{otherwise} \end{cases} \tag{7}
$$

Let $|\lambda_{1,\mathbf{S}}|$ be the magnitude of the largest eigenvalue (in magnitude). Let $\hat{C}(t)$ to be the expected number of carriers at time $t$ according to this dynamical system; $\hat{C}(t) = \sum_{i=1}^{N} p_i(t)$.

**Theorem 3 (Condition for fast extinction).** *Define $s = |\lambda_{1,\mathbf{S}}|$ to be the "survivability score" for the system. If*

$$\boxed{s = |\lambda_{1,\mathbf{S}}| < 1}$$

7

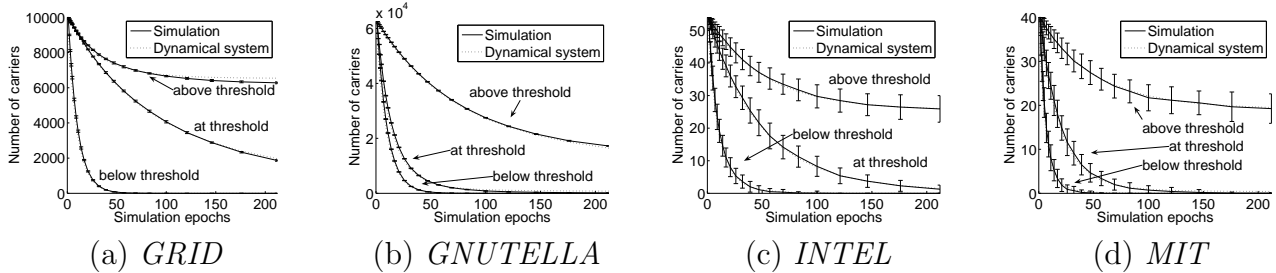| (a) GRID | (b) GNUTELLA | (c) INTEL | (d) MIT |

Figure 4: *Number of carriers versus time (simulation epochs):* Each plot shows the evolution of the dynamical system (dotted lines) and the simulation (solid lines). There are two observations: (1) The dynamical system (dotted lines) is very close to the simulations (solid lines), demonstrating the accuracy of Equations 4-5. (2) Also, the number of carriers dies out very quickly below the threshold, while the information "survives" above the threshold.

then we have fast extinction in the dynamical system, that is, $\hat{C}(t)$ decays exponentially quickly over time.

**Definition 2 (Threshold).** *We will use the term "below threshold" when $s < 1$, "above threshold" when $s > 1$, and "at the threshold" for $s = 1$.*

**Corollary 1 (Homogeneous case).** *If $\delta_i = \delta, r_i = r, \gamma_i = \gamma$ for all $i$, and $\mathbf{B} = [\beta_{ij}]$ is a symmetric binary matrix (links are undirected, and are always up or always down), then the condition for fast extinction is:*

$$\boxed{\frac{\gamma r}{\delta(\gamma+\delta)}\lambda_{1,\mathbf{B}} < 1}$$

**Corollary 2.** *Our model subsumes the SIS model of viral infection as a special case.*

## 3.3 Experiments

We ran experiments on (a) a $100 \times 100$ grid, (b) one snapshot of the Gnutella graph [19], (c) a 54-node sensor network from Intel [15], and (d) a 40-node network from MIT [14]. As we can see from figure 4, the information survives above the threshold, but dies off quickly below the threshold, matching Theorem 3.

# 4 Automatically grouping correlated nodes using Cross-Associations

*"How can we automatically find natural node groups in a large graph?"*

Clustering is one of the most common methods of data analysis, and it has many applications in graph mining. For example, given a list of customers and the products they buy, we might want to find customer "groups," and the product "groups" they are most interested in. The same can be done for documents versus words, bank accounts versus transactions,

websites connecting to other websites: in short, any problem where entities are connected by relationships.

We focus only on *unweighted* graphs, which can be represented as adjacency matrices with 0/1 entries; we use the terms "graph" (with nodes) and "matrix" (with rows and columns) interchangeably. A good graph clustering algorithm should have the following properties: **(P1)** It should automatically figure out the number of clusters, **(P2)** It should cluster rows and columns simultaneously, **(P3)** It should be scalable, **(P4)** It should allow incremental updates, and **(P5)** It should apply to both self-graphs and bipartite graphs. For bipartite graphs, we have row and column clusters, representing node groups in the "from" and "to" parts of the bipartite graph. For self-graphs, we might have an additional condition that the row and column clusters be identical (i.e., only "node" groups, instead of "from" and "to" groups). This algorithm should have the following three goals: **(G1)** Find clusters, **(G2)** Find outliers, and **(G3)** Compute inter-cluster distances.

In Section 4.1, we formulate our data description model, and a corresponding *cost function* for each possible clustering. Based on this, we describe our *automatic, parameter-free* algorithm for finding good clusters in Section 4.2. We then use these clusters to find outliers and compute inter-cluster distances in Section 4.3. Experimental results are provided in Section 4.4.

## 4.1 Data Description Model

Intuitively, we seek to group rows and edges (i.e., nodes) so that the adjacency matrix is divided into rectangular/square blocks as "similar" or "homogeneous" as possible. The homogeneity would imply that the graph nodes in that (row or column) group are all "close" to each other, and the density of each region would represent the strength of connections between groups. These regions of varying density, which we call *cross-associations*, would succinctly summarize the underlying structure of associations between nodes.

To compress the matrix, we would prefer to have only a few blocks, each of them being very homogeneous. However, having more clusters lets us create more homogeneous blocks (at the extreme, having 1 cluster per node gives $M \cdot N$ perfectly homogeneous blocks of size $1 \times 1$). Thus, the best compression scheme must achieve a tradeoff between these two factors, and this tradeoff point indicates the best values for $k$ and $\ell$.

We accomplish this by a novel application of the overall MDL philosophy, where the compression costs are based on the number of bits required to transmit both the "summary" of the row/column groups, as well as each block given the groups. Thus, *the user does not need to set any parameters*; our algorithm chooses them so as to minimize these costs. We discuss the exact cost function below.

The compression cost has two parts: the *description cost* needed to describe the groups, and the *code cost* to encode the matrix given the groups. The description cost includes the cost for transmitting the number of clusters ($\log^\star k + \log^\star \ell$ for $k$ row groups and $\ell$ column groups), the sizes of each group, and the number of "ones" in each block of the matrix, apart from some terms which are the same for all clusterings. The code cost is the sum of costs to encode each block, which is just the size of the block multiplied by its *entropy*. The mathematical details are in the full thesis document.
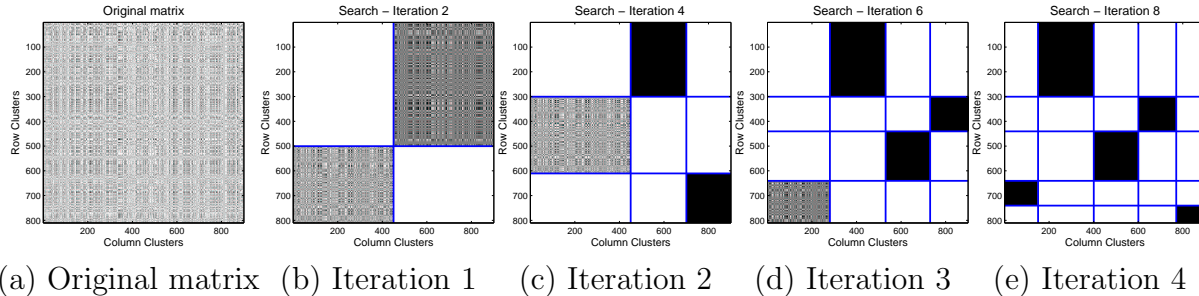
| (a) Original matrix | (b) Iteration 1 | (c) Iteration 2 | (d) Iteration 3 | (e) Iteration 4 |

Figure 5: *Snapshots of algorithm execution:* Starting with the matrix in plot (a), the clustering is successively refined and the number of clusters increased till we reach the correct clustering in plot (e).

## 4.2   (G1) Graph Clustering Algorithm

We can now assign a cost to each possible clustering; given two distinct clusterings, we can use the cost function to choose between the two. How can we use this to *find* a good clustering? In other words, how do we pick the best numbers of clusters $k^*$ and $\ell^*$, and the memberships of these clusters?

 We propose an iterative two-step scheme to answer this question:

1. SHUFFLE (inner loop): For a given $k$ and $\ell$, find a good arrangement (i.e., cross-association). We do this using a greedy approach: for each row in the matrix, we shuffle it to the row group which minimizes the code cost, and we do the same for all columns, and iterate till no further cost benefits occur.

2. SPLIT (outer loop): Efficiently search for the best $k$ and $\ell$ ($k, \ell = 1, 2, \ldots$). This is accomplished by picking out the most inhomogeneous row/column group, and splitting it in two. The split criterion is the following: for each row/column in this group, we send it to the new split group if its removal from the current group makes the current group more homogeneous (measured via entropy).

The mathematical details are in the full thesis document, and in [8, 7]. The entire algorithm is linear on the number of edges in the graph. We can also show that our algorithm matches all our desiderata (**(P1)**-**(P5)**). Figure 5 shows an example run of this algorithm.

## 4.3   Finding outlier edges and inter-cluster distances

Having found the underlying structure of a graph in the form of node groups (goal **(G1)**), we can utilize this information to further mine the data. Specifically, we want to detect outlier edges (goal **(G2)**) and compute inter-group "distances" (goal **(G3)**). Again, we use our information-theoretic approach to solve all these problems efficiently.

### 4.3.1   (G2) Outlier edges

Which edges between nodes are abnormal/suspicious? Intuitively, an outlier shows some deviation from normality, and so it should hurt attempts to compress data. Thus, an edge

whose removal significantly reduces the total encoding cost is an outlier. Our algorithm is: find the block where removal of an edge leads to the maximum immediate reduction in cost (that is, no iterations of SHUFFLE and SPLIT are performed). All edges within that block contribute equally to the cost, and so all of them are considered outliers.

$$\text{``Outlierness'' of edge } (u, v) := T(\mathbf{A}'; k, \ell, \Phi, \Psi) - T(\mathbf{A}; k, \ell, \Phi, \Psi) \quad (8)$$

where $\mathbf{A}'$ is $\mathbf{A}$ without the edge $(u, v)$. This can be used to *rank* the edges in terms of their "outlierness".

### 4.3.2 (G3) Computing inter-group "distances"

How "close" are two node groups to each other? Following our information theory footing, we propose the following criterion: If two groups are "close", then combining the two into one group should not lead to a big increase in encoding cost. Based on this, we define "distance" between two groups as the relative increase in encoding cost if the two were merged into one:

$$Dist(i, j) := \frac{Cost(merged) - Cost(i) - Cost(j)}{Cost(i) + Cost(j)} \quad (9)$$

where only the nodes in groups $i$ and $j$ are used in computing costs. We experimented with other measures (such as the absolute increase in cost) but Equation 9 gave the best results.

To computing outliers and distances between groups, only the statistics of the final clustering need to be used (i.e., the block sizes $n_{i,j}$ and their densities $P_{i,j}$). Thus, both can be performed efficiently for large graphs.

## 4.4 Experiments

We demonstrate results on two real-world datasets. The first is *CLASSIC*, which is a bipartite graph of Usenet documents from Cornell's SMART collection, and the words present in them (see [9]). The documents belong to three distinct groups: MEDLINE (medicine), CISI (information retrieval), and CRANFIELD (aerodynamics). The second is *GRANTS*, a set of NSF proposal documents from several disciplines (physics, bio-informatics, etc.), versus the words in their abstracts. The results are shown in Figure 6.

CLASSIC: We see that the cross-associates are in agreement with the known document classes (left axis annotations). We also annotated some of the column groups with their most frequent words. Cross-associates belonging to the same document (row) group clearly follow similar patterns with respect to the word (column) groups. For example, the MEDLINE row groups are most strongly related to the first and second column groups, both of which are related to medicine. ("insipidus," "alveolar," "prognosis" in the first column group; "blood," "disease," "cell," etc, in the second).

Besides being in agreement with the known document classes, the cross-associates *reveal further structure.* For example, the first word group consists of more "technical" medical

insipidus, alveolar, aortic, death, prognosis, intravenous

blood, disease, clinical, cell, tissue, patient

shape, nasa, leading, assumed, thin

manifolds, operators, harmonic, operator, topological

undergraduate, education, national, projects

providing, studying, records, developments, students, rules, community

abstract, notation, works construct, bibliographies

paint, examination, fall, raise, leave, based

encoding, characters, bind, nucleus, recombination

coupling, deposition, plasma, separation, beam

meetings, organizations, session, participating

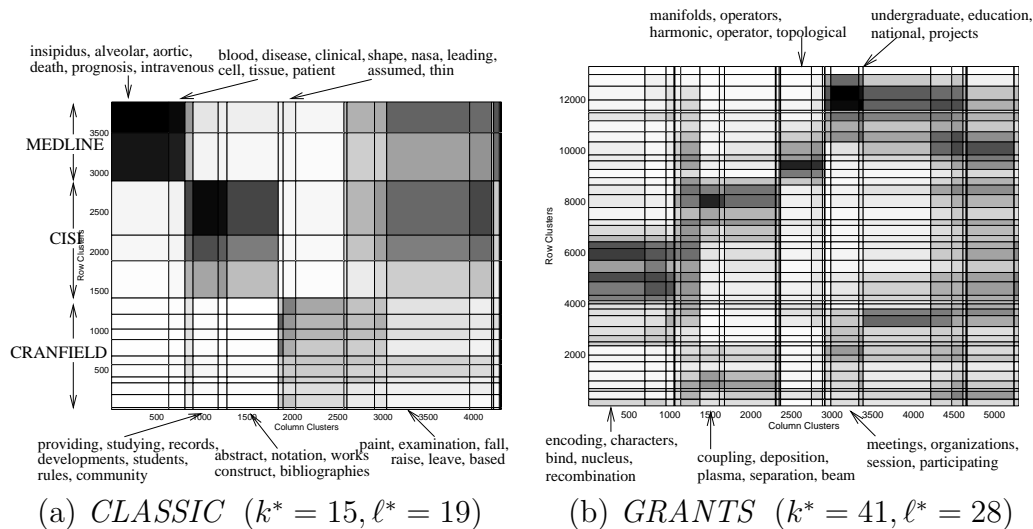(a) *CLASSIC* $(k^* = 15, \ell^* = 19)$  (b) *GRANTS* $(k^* = 41, \ell^* = 28)$

Figure 6: *Cross-associations for CLASSIC and GRANTS*: Due to the dataset sizes, we show the Cross-associations via shading; darker shades correspond denser blocks (more ones). We also show the most frequently occurring words for several of the word (column) groups; these clearly belong to different categories of words.

terms, while second group consists of "everyday" terms, or terms that are used in medicine often, but not exclusively[1]. Thus, the second word group is more likely to show up in other document groups (and indeed it does, although not immediately apparent in the figure), which is why our algorithm separates the two.

GRANTS: Again, meaningful clusters are extracted. Figure 6(b) shows the most common terms in several of the column clusters. They show that the groups found make intuitive sense: we detect clusters related to biology ("encoding," "recombination," etc), to physics ("coupling," "deposition", "plasma," etc), to material sciences, and to several other well-known topics.

# 5   The R-MAT graph generator

*"How can we quickly generate a synthetic yet realistic graph? How can we spot fake graphs and outliers?"*

While we answered application-specific questions in the previous sections, the focus of this section is on real-world graphs in general. What do real graphs look like? What patterns or "laws" do they obey? This is intimately linked to the problem of designing a good graph generator: a realistic generator is one which matches exactly these graph "laws." These patterns and generators are important for many applications, such as the detection of abnormal subgraphs/edges/nodes, simulation studies when collecting the real-world graph is hard or costly, checking the realism of graph samples, and so on.

---

[1]This observation is also true for nearly all of the (approximately) 600 and 100 words belonging to each group, not only the most frequent ones shown here.

There are several desiderata from a graph generator. **(P1)** *Realism*: It should only generate graphs that obey all (or at least several) of the above "laws", and it would match the properties of real graphs (degree exponents, diameters etc., that we shall discuss later) with the appropriate values of its parameters. **(P2)** *Procedural generation*: Instead of creating graphs to specifically match some patterns, the generator should offer some *process* of graph generation, which automatically leads to the said patterns. This is necessary to gain insight into the process of graph generation in the real world: if a process cannot not generate synthetic graphs with the required patterns, it is probably not the underlying process (or at least the sole process) for graph creation in the real world. **(P3)** *Parsimony*: It should have a few only parameters. **(P4)** *Fast parameter-fitting*: Given any real-world graph, the model parameters should easily tunable by some published algorithms to generate graph similar to the input graph. **(P5)** *Generation speed*: It should generate the graphs quickly, ideally, linearly on the number of nodes and edges. **(P6)** *Extensibility*: The same method should be able to generate directed, undirected, and bipartite graphs, both weighted or unweighted.

This is exactly the main part of this work. We propose the *Recursive Matrix* (R-MAT) model, which naturally generates power-law (or "DGX" [5] ) degree distributions. We show that it naturally leads to small-world graphs and also matches several other common graph patterns; it is recursive (=self-similar), and it has only a small number of parameters.

The rest of this section is organized as follows: Section 5.1 provides a brief survey of existing graph laws and generators. Section 5.2 presents the idea behind our R-MAT method. Finally, section 5.3 gives the experimental results, where we show that R-MAT successfully mimics large real graphs.

## 5.1   Related Work

We will discuss the related work in two parts: graph patterns, and graph generators.

**Graph Patterns and "Laws":**   The main graph patterns appear to be: power laws, small diameters and community effects. Power laws often show up in the in- and out-degree distributions of graphs, as well as in their eigenvalue-versus-rank plots (also known as scree plots) [12, 2]. Recently, deviations from power laws have also been observed [18]. Another pattern is the "hop-plot," which measures the average growth of the neighborhood size with the number of hops, and allows us to compute the "effective diameter" within which 95% of the reachable node pairs are connected by a path. The "effective diameter" is typically very small, even for large graphs. Other patterns include the distribution of components in the first eigenvector (the "network value" of nodes), "stress" distributions (the number of minimum shortest paths each edge lies on), and so on.

**Graph Generators:**   The earliest model was the Random Graph model [10]. It has very interesting phase transition properties but does not match power laws or show any community structure. More recently, preferential attachment generators have been proposed [4]: these provide a *process* for graph generation that automatically leads to power laws, but few other patterns are explored. Some generators develop power laws as a result of resource optimization under constraints [11], and others also include the effects on geography on graph
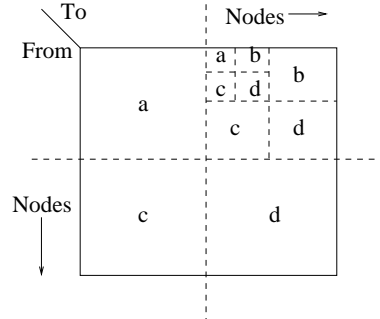
Figure 7: *The R-MAT model:* The adjacency matrix is broken into four equal-sized partitions, and one of those four is chosen according to a (possibly non-uniform) probability distribution. This partition is then split recursively till we reach a single cell, where an edge is placed. Multiple such edge placements are used to generate the full synthetic graph.

generation [21, 16], but again more patterns need to be studied. More references are available in the thesis document.

## 5.2   The R-MAT methodology

Most of the current graph generators focus on only one graph pattern – typically the degree distribution – and give low importance to all the others. What we would like is a tradeoff between parsimony (property **(P3)**), realism (property **(P1)**), and efficiency (properties **(P4)** and **(P5)**). Our R-MAT generator attempts to address all of these concerns.

The R-MAT generator creates directed graphs with $2^n$ nodes and $E$ edges, where both values are provided by the user. We start with an empty adjacency matrix, and divide it into four equal-sized partitions. One of the four partitions is chosen with probabilities $a, b, c, d$ respectively ($a + b + c + d = 1$), as in Figure 7. The chosen partition is again subdivided into four smaller partitions, and the procedure is repeated until we reach a simple cell ($=1 \times 1$ partition). The nodes (that is, row and column) corresponding to this cell are linked by an edge in the graph. This process is repeated $E$ times to generate the full graph. There is a subtle point here: we may have *duplicate* edges (i.e., edges which fall into the same cell in the adjacency matrix), but we only keep one of them when generating an unweighted graph. To smooth out fluctuations in the degree distributions, some noise is added to the $(a, b, c, d)$ values at each stage of the recursion, followed by renormalization (so that $a + b + c + d = 1$). Typically, $a \geq b$, $a \geq c$, $a \geq d$. Next, we show how this simple construction can match many real-world patterns, while requiring only 3 parameters.

## 5.3   Experiments

Figure 8 shows results on a *CLICKSTREAM* bipartite graph of the browsing behavior of Internet users [17]. An edge $(u, p)$ denotes that user $u$ accessed page $p$. There are $23,396$ users, $199,308$ pages and $952,580$ edges. R-MAT matches all of the patterns very well, while several other popular/recent models (AB [1], PG [18] and GLP [6]) do not even apply. The
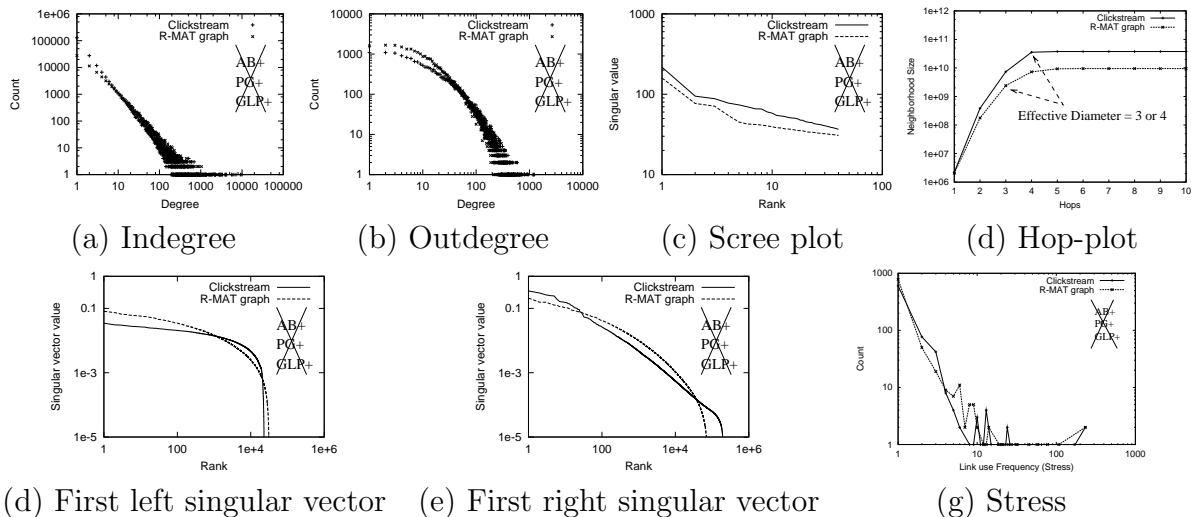
14

(a) Indegree     (b) Outdegree     (c) Scree plot     (d) Hop-plot



(d) First left singular vector    (e) First right singular vector     (g) Stress

Figure 8: *Results on 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.

thesis document shows more examples on directed and undirected graphs.

# 6 Conclusions

Graphs are ubiquitous; they show up in fields as diverse as ecological studies, sociology, computer networking and many others. There is a dichotomy in graph mining applications: we can answer specific queries on a particular graph, or we can ask questions pertaining to real-world graphs in general. I have explored issues from both sides of this dichotomy.

- *How does a virus propagate over the given graph? When does a viral outbreak die out?* We proposed the *NLDS* model and discovered the relationship between the epidemic threshold and the largest eigenvalue of the graph.
- *Under what conditions does a piece of information survive in a given sensor network with failing nodes and links?* Once again, a non-linear dynamical system allows us to answer this question.
- *How can we automatically cluster nodes in a given graph?* Our contribution is the development of an MDL-based framework that allows us to quickly and automatically estimate both the *number* of clusters in the data, and their memberships.
- *What patterns and "laws" hold for most real-world graphs? How can we* generate *synthetic yet "realistic" graphs?* We proposed the R-MAT graph generator, which can match many of the patterns in real-world graphs, while requiring only 3 parameters.

To conclude, we developed tools for mining graph datasets under a variety of circumstances, each of which is important in its own right; combined together, their usefulness is increased even more. "How susceptible will the Internet be to viral infections if its grows by $x\%$ nodes and $y\%$ edges?" Our work can shed some light.

# References

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

[2] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 2002.

[3] N. T. J. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Hafner Press, 2nd edition, 1975.

[4] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, pages 509–512, 1999.

[5] Z. Bi, C. Faloutsos, and F. Korn. The DGX distribution for mining massive, skewed data. In *KDD*, pages 17–26, 2001.

[6] T. Bu and D. Towsley. On distinguishing between Internet power law topology generators. In *INFOCOM*, 2002.

[7] D. Chakrabarti. AutoPart: Parameter-free graph partitioning and outlier detection. In *PKDD*, 2004.

[8] D. Chakrabarti, S. Papadimitriou, D. Modha, and C. Faloutsos. Fully automatic Cross-associations. In *KDD*, 2004.

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

[10] P. Erdös and A. Rényi. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Acadamy of Science*, 5:17–61, 1960.

[11] A. Fabrikant, E. Koutsoupias, and C. H. Papadimitriou. Heuristically Optimized Trade-offs: A new paradigm for power laws in the Internet (extended abstract), 2002.

[12] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *SIGCOMM*, pages 251–262, 1999.

[13] P. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. IrisNet: An architecture for a world-wide sensor web. *IEEE Pervasive Computing*, 2003.

[14] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating congestion in wireless sensor networks. In *SenSys*, pages 134–147, 2004.

[15] http://db.lcs.mit.edu/labdata/labdata.html.

[16] A. Medina, I. Matta, and J. Byers. On the origin of power laws in Internet topologies. In *SIGCOMM*, pages 18–34, 2000.

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

[18] 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(8):5207–5211, 2002.

[19] 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(1), 2002.

[20] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos. Epidemic spreading in real networks: An eigenvalue viewpoint. In *SRDS*, 2003.

[21] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.