RTM: Laws and a Recursive Generator for Weighted Time-Evolving Graphs

Submitted for Blind Review

Abstract

How do real, weighted graphs change over time? What patterns, if any, do they obey? Earlier studies focus on unweighted graphs, and, with few exceptions, they focus on static snapshots. Here, we report patterns we discover on several real, weighted, time-evolving graphs. The reported patterns can help in detecting anomalies in natural graphs, in making link prediction and in providing more criteria for evaluation of synthetic graph generators. We further propose an intuitive and easy way to construct weighted, time-evolving graphs. In fact, we prove *that our generator will produce graphs which obey many patterns and laws observed to date. We also provide empirical evidence to support our claims.*

1 Introduction

Static, unweighted graphs have attracted a lot of interest recently, with several fascinating discoveries, such as small diameters [19], power-law degree distributions [2, 6], skewed eigenvalues and eigenvector scores. Numerous generators also try to mimick these patterns. Very recently, time-evolving graphs have attracted some attention, with the 'densification' power law, shrinking diameters [10], and related generators.

However, graphs that are both weighted and dynamic have been relatively unexplored. Here we focus on weighted graphs - both static snapshots of them, as well as dynamic properties. Just as the edges and nodes in a network have different meanings in different contexts, *edge weights* also have different meanings. For instance, in a network of computers, nodes may be IP addresses (sender/receiver), an edge may indicate a transaction of packets, and the "weight" on each edge may represent the total number of packets shipped. Alternatively, the size of the packet may be used, depending on what sort of knowledge one wants to extract from the graph.

Given a set of edge weights on a graph, a few questions come to mind. What patterns do the weights obey? Do they follow a Gaussian distribution, for a given snapshot in time? How, if at all, is the edge weight related to the popularity of its adjacent nodes? Which of these static patterns persist over time?

In this work, we answer all these questions, and we show that there are some unexpected patterns. In summary, the contributions are the following:

1. We present several new patterns for weighted, time-evolving graphs.

2. We give a simple generative model (*RTM*, for *Recursive Tensor Multiplication*) that generates weighted, time-evolving graphs that obey all the old and new properties.

3. We also prove that *RTM* produces several desired characteristics.

The most striking patterns we discover here are: (a) the first eigenvalue $\lambda_{1,w}(t)$ of the *weighted* adjacency matrix at time t, follows a power law with respect to the total number of edges $E(t)$ at time t, for several time-ticks. (b) a similar power law holds for the first eigenvalue $\lambda_1(t)$ of the 0-1 adjacency matrix, with different slope, of course. (c) for a given time snapshot, the weight $w_{i,j}$ for edge (i, j) is closely related to the total weights w_i and w_j of its adjacent nodes i and j.

Our experiments on multiple, real datasets show that these patterns hold; they also confirm that the edge-addition and the weight-addition follow a self-similar, bursty traffic pattern, as was observed earlier for blog activity [13].

The rest of the paper is organized as follows: Section 2 surveys the earlier work. Section 3 provides preliminary background material as well as description of the datasets we used. Section 5 describes our *RTM* generator and provides proofs showing that it reproduces several properties observed in real weighted graphs. Section 6 lists experimental results. We conclude in Section 7.

2 Related Work

We next list static and temporal properties that most real-world graphs were found to have in common, followed by a survey of earlier work on graph generators.

Real-world graph properties: One of the most impressive patterns that real-world graphs obey is a small diameter, which is also known as "small-world phenomenon" or "six degrees of separation" [1]. Diameter is defined as the minimum number of hops that will connect all of the nodes. The *effective diameter*, then, is the minimum number of hops in which 90% of connected node pairs can be reached; this measure is often used as a less computationally-intense estimate. Surprising recent work showed that not only is the diameter of graphs small, but it also *shrinks* and then *stabilizes* over time [10].

Many other patterns regarding power laws have been discovered. A power law relation between two variables x and y can be defined as $y \propto x^a$, where a is the power law exponent. Time-evolving graphs follow the "Densification Power Law" with the equation $E(t) = cN(t)^a$ [10]. Degree distribution of graphs also obey a power law of the form $N_d = cd^{-a}$, with $a > 0$, and N_d being the number of nodes with degree d. Such power law relations as well as many more have been reported in [4, 6, 8, 15].

There have been studies on spectral properties of power-law graphs in [7, 11, 14, 16]. Faloutsos et al. [16] examined the spectrum of the adjacency matrix of the Autonomous System (AS) Internet topology and they reported that the 20 or so largest eigenvalues of the Internet graph are power-law distributed with exponent between .45 and .5. Michail et al. [14] later provided an explanation for the eigenvalue law phenomenon, showing that Eigenvalues Power Law is a consequence of the Degree Power Law. Farkas et al. [7] studied the numerical and analytical properties of the adjacency matrices of complex networks and reported surprising results on the spectra of adjacency matrices corresponding to several models of real-world graphs. Moreover, Chung et al. [11] analyzed the eigenvalues of random graphs for which the number of nodes of degree d follow a power law and reported bounds on the first and second eigenvalues of such graphs for certain parameters.

Graph generators: Modeling real-world graphs successfully is an elusive task. A vast majority of earlier graph generators have focused on modeling a small number of common properties, but fail to mimic others. The very first generative model was proposed by Erdos & Renyi [5]. The model begins with a fixed number of nodes, and adds edges, where any pair of nodes has the same and independent probability of being linked by an edge. While it has some interesting provable properties, it fails to produce a number of realistic properties, most notably the heavy-tailed degree distribution. Another striking generator is the *preferential attachment* model, where at each time step nodes are added and 'prefer' to link to high-degree nodes. This in turn leads to small diameter and heavy-tailed degree distributions; however, this and related models lack the *shrinking diameter* property. There exists a group of other generators such as the "small-world" and "forest fire" models [19, 10]. In addition, recursive models using Kronecker multiplication have proved useful for generating self-similar properties of graphs [9]. Chakrabarti et. al. provide a detailed survey on graph generators in [3].

In our study, we explicitly concentrate on the first (principal) eigenvalue of real-world graphs and how it changes over time with the growing size of the graph. We find a correlation between the weight of edge (i, j) and the total weights w_i and w_j of its adjacent nodes. While previous work has often failed to match patterns for weighted time-evolving graphs, we propose a generative model for which we can show that the resulting graphs will exhibit certain laws such as bursty edge/weight additions and the Weight Power Law, as well as many other patterns discovered to date.

3 Background

Throughout this paper we will use the graph representation of the datasets we study. A weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V}$, connected by a set of undirected, weighted edges $\mathcal{E}$. No multiple edges between two nodes are allowed - however, we will account for repeated edges in edge weights. By nature, none of the datasets we study contain self-loops.

In a weighted graph $\mathcal{G}$, let $e_{i,j}$ be the edge between node i and node j. We shall refer to these two nodes as the *'neighboring nodes'* or *'incident nodes'* of edge $e_{i,j}$. Let $w_{i,j}$ be the weight on edge $e_{i,j}$.

The *total weight* w_i of node i is defined as the sum of weights of all its incident edges, that is $w_i = \sum_{k=1}^{d_i} w_{i,k}$, where d_i denotes its degree. As we show later, there is a relation between a given edge weight $w_{i,j}$ and the weights of its neighboring nodes w_i and w_j.

A graph $\mathcal{G}$ can be represented by its adjacency matrix $\mathbf{A}(\mathcal{G})$, which is a symmetric matrix with $\mathbf{A}_{ij} = 1$, if nodes i and j are connected, or 0, otherwise. For bipartite graphs with an $N \times M$ adjacency matrix $\mathbf{A}$, we will define a new symmetric square matrix $\mathbf{B} = [\mathbf{A}\ 0; 0\ \mathbf{A}^T]$ and compute the first eigenvalue of this new matrix $\mathbf{B}$, which would be the same as the first singular value of the original matrix $\mathbf{A}$.

A complete list of the symbols used throughout text is listed in Table 1.

3.1 Burstiness and Entropy Plots

We will show that the addition of weights and edges in our model is often bursty and self-similar. Among many methods that measure self-similarity and burstiness, we use the *entropy plot* [17], which plots entropy $H(r)$ versus resolution r. The resolution, or the aggregation level, is the

Symbol	Description
$\mathcal{G}$	Graph representation of datasets
$\mathcal{V}$	Set of nodes for graph $\mathcal{G}$
$\mathcal{E}$	Set of edges for graph $\mathcal{G}$
N	Number of nodes, or $\|V\|$
E	Number of edges, or $\|E\|$
$e_{i,j}$	Edge between node i and node j
$w_{i,j}$	Weight on edge $e_{i,j}$
w_i	Weight of node i (sum of weights of incident edges)
$\mathbf{A}$	0-1 Adjacency matrix of the un-weighted graph
$\mathbf{A}_w$	Real-value adjacency matrix of the weighted graph
$a_{i,j}$	Entry in matrix $\mathbf{A}$
λ_1	Principal eigenvalue of unweighted graph
$\lambda_{1,w}$	Principal eigenvalue of weighted graph
$\mathcal{A}, \mathcal{B}, \mathcal{C}$	Tensors used to illustrate recursive tensor product
$a_{i,j,k}$	Entry of a tensor
$\mathcal{I}$	Initial tensor in *RTM* model
$\mathcal{G}_\mathcal{A}$	t-graph (time-evolving graph) represented by tensor $\mathcal{A}$
$\mathbf{D}_t$	t^{th} slice of final tensor $\mathcal{D}$ in *RTM*
s_t	Total weight of $\mathbf{D}_t$
e_t	Number of edges of $\mathbf{D}_t$
$W_\mathcal{D}$	Total weight of a tensor $\mathcal{D}$, or $\sum_t s_t$
$\mathbf{s}_{\mathcal{D},r}$	Temporal profile of $\mathcal{D}$ at resolution r
$\mathbf{p}_{\mathcal{D},r}$	Normalized temporal profile of $\mathcal{D}$ at resolution r

Table 1. Table of symbols used in notation.

scale - that is, for resolution r, we divide our time interval into 2^r equal sub-intervals. For instance, suppose we would like to measure the entropy of weight additions - that is, the total weight of all edges added at each time t (this sum will be denoted $\Delta W(t)$). For each resolution r, we sum the weight-additions $\Delta W(t)$ in each sub-interval k, for $k = 1 \dots 2^r$, normalize into fractions p_k (= $\Delta W(t)/W_{total}$), and compute the Shannon entropy of the sequence p_k: $H(r) = -\sum_k p_k \log_2 p_k$. If the plot $H(r)$ is linear in some range of resolutions, the corresponding time sequence is said to be *fractal* in that range, and the slope of the plot is defined as the *intrinsic* (or *fractal*) dimension D of the time sequence. Notice that a uniform edge/weight-addition distribution yields D=1; a lower value of D corresponds to a more bursty time sequence with a single burst having the lowest D=0.

3.2 Data Description

We studied several large real-world weighted graphs described in detail in Table 2. In particular, *BlogNet* contains blog-to-blog links, *NetworkTraffic* records IP-source/IP-destination pairs, along with the number of packets sent. Bipartite networks *Auth-Conf*, *Keyw-Conf*, and *Auth-Keyw* are all from DBLP and have the submission records of authors to conferences with specified keywords. *CampaignOrg* is from the U.S. Federal Election Commission, a public record of donations between political candidates and organizations.

For *NetworkTraffic* and *CampaignOrg* datasets, the weights on the edges are actual weights representing number of packets and donation amounts, respectively. For the remaining datasets, the edge weights are simply the number of occurences of the edges. For instance, if author i submits a paper to conference j for the first time, the weight of $e_{i,j}$ is set to 1. If the same author later submits another paper to the same conference, the edge weight becomes 2.

4 Laws and Observations

4.1 LPL: Principal eigenvalue over time

Plotting the largest(principal) eigenvalue of the 0-1 adjacency matrix of our datasets over time, we notice that the principal eigenvalue grows following a power law with increasing number of edges. This observation is true especially after the *gelling point*. The 'gelling point' is defined to be the point at which a giant connected component (GCC) appears in real-world graphs - after this point, properties such as densification and shrinking diameter become increasingly evident. See [10] for details.

Observation 1 (λ_1 Power Law (LPL)) *In real graphs, the principal eigenvalue $\lambda_1(t)$ and the number of edges $E(t)$ over time follow a power law with exponent less than 0.5, especially after the 'gelling point'. That is,*

$$\lambda_1(t) \propto E(t)^\alpha,\ \alpha \leq 0.5$$

We report the power law exponents in Fig. 1. Note that we fit the given lines *after* the gelling point which is shown by a vertical line for each dataset. Notice that the given slopes are less than 0.5, with the exception of the *CampaignOrg* dataset, with slope ≈ 0.53.

Given the theorem $\lambda_{max}(\mathcal{G}) \leq \left\{2\left(1 - \frac{1}{N}\right) E\right\}^{\frac{1}{2}}$ for a connected, undirected graph $\mathcal{G}$ without self-loops and multiple edges, with E edges and N nodes (See [20] for proof), for large N, s.t. $\frac{1}{N} \to 0$, we expect the power law exponent to be less than 0.5. By construction, there are no multiple edges in our graphs (that is, we work with a binary adjacency matrix), and the nodes do not have self-loops by nature. Finally, we claim that the graphs behave as a single connected component after the 'gelling point' at which point the GCC dominates other connected components. The only slight exception to the power law, the *CampaignOrg* graph, always has many number of disconnected components as well as a GCC. Thus, we conclude that our observation follows early theory.

4.2 LWPL: Weighted principal eigenvalue over time

Given that unweighted (0-1) graphs follow the λ_1 Power Law, one may ask if there is a corresponding law for

Name	N, E, time	Description
BlogNet	60K, 125K, 80 days	Social network of blogs based on citations
NetworkTraffic	21K, 2M, 52 mo.	Network traffic: packets sent from IP source to IP destination
AuthorConference	17K, 22K, 25 yr.	DBLP Author-to-Conference associations
KeywordConference	10K, 23K, 25 yr.	DBLP Keyword-to-Conference associations
AuthorKeyword	27K, 189K, 25 yr.	DBLP Author-to-Keyword associations
CampaignOrg	23K, 877K, 28 yr.	U.S. electoral campaign donations (available from FEC)

Table 2. Weighted datasets studied in this work.

weighted graphs. To this end, we also compute the largest eigenvalue $\lambda_{1,w}$ of the *weighted* adjacency matrix $\mathbf{A}_w$. The entries $w_{i,j}$ of $\mathbf{A}_w$ now represent the actual edge weight between node i and j. We notice that $\lambda_{1,w}$ increases with increasing number of edges following a power law with a higher exponent than that of its λ_1 Power Law. We show the experimental results in Fig. 2.

Observation 2 ($\lambda_{1,w}$ Power Law (LWPL)) *Weighted real graphs exhibit a power law for the largest eigenvalue of the weighted adjacency matrix $\lambda_{1,w}(t)$ and the number of edges $E(t)$ over time. That is,*

$$\lambda_{1,w}(t) \propto E(t)^{\beta}$$

In our experiments, the exponent β ranged from 0.5 to 1.6.

4.3 EWPL: Edge Weights Power Law

We observe that the weight of a given edge and weights of its neighboring two nodes are correlated. Our observation is similar to Newton's Gravitational Law stating that the gravitational force between two point masses is proportional to the product of the masses. Similarly, the tendency of two nodes to interact often would be related to the "popularity" of both.

For each edge (i, j) at the final time step in the graph, we plot $\sqrt{(w_i - w_{i,j}) * (w_j - w_{i,j})}$ versus its weight $w_{i,j}$ as a single point. Notice that we did not include the weight of the edge itself in the total weight of its incident nodes. Next, we fit a line to the median y-axis values after applying logarithmic binning on the x-axis and report the corresponding slopes for each dataset in Fig. 3. Note that, we omit the points which represent edges to avoid the confusion due to overplotting. Instead, we show the 75% and 25%-tile of the data with upper and lower vertical bars, respectively.

Observation 3 (Edge Weights Power Law(EWPL)) *Given a real-world graph $\mathcal{G}$, 'communication' defined as the weight of the link between two given nodes has a power law relation with the weights of the nodes. In particular, given an edge $e_{i,j}$ with weight $w_{i,j}$ and its two neighbor nodes i and j with weights w_i and w_j, respectively,*

$$w_{i,j} \propto \left(\sqrt{(w_i - w_{i,j}) * (w_j - w_{i,j})}\right)^{\gamma}$$

EWPL can be used in link prediction; that is, one can estimate the probable weight of a future link between two nodes of the graph, given their weights. Moreover, edges with weights deviating too much from the expected might be flagged for further consideration.

5 Generative Recursive Tensor Model (RTM)

How could we have a simple generative model that will obey all the patterns we know so far, as well as the newly discovered ones for weighted graphs? Specifically, we would like the model to exhibit:

1. SUGP: static unweighted graph properties:
   - small diameter [19]
   - power law degree distribution [2, 6]
2. SWGP: static weighted graph properties:
   - the edge weight power law (EWPL) (Observation 3)
   - the snapshot power law (SPL) [12]
3. DUGP: dynamic unweighted graph properties:
   - the densification power law (DPL) [10]
   - shrinking diameter [10]
   - the λ_1 power law (LPL)(Observation 1)
   - bursty edge additions [13]
4. DWGP: dynamic weighted graph properties:
   - the $\lambda_{1,w}$ power law (LWPL)(Observation 2)
   - bursty weight additions [12]
   - the weight power law (WPL) [12].

At the high level, our idea is to use recursion, in conjunction with tensors (n-dimensional extension of matrices). Recursion and self-similarity naturally leads to modular network behavior ("communities-within-communities") and power laws [9]; it also leads to bursty traffic [18]. Earlier work used self-similarity to generate static snapshots of unweighted graphs [4].

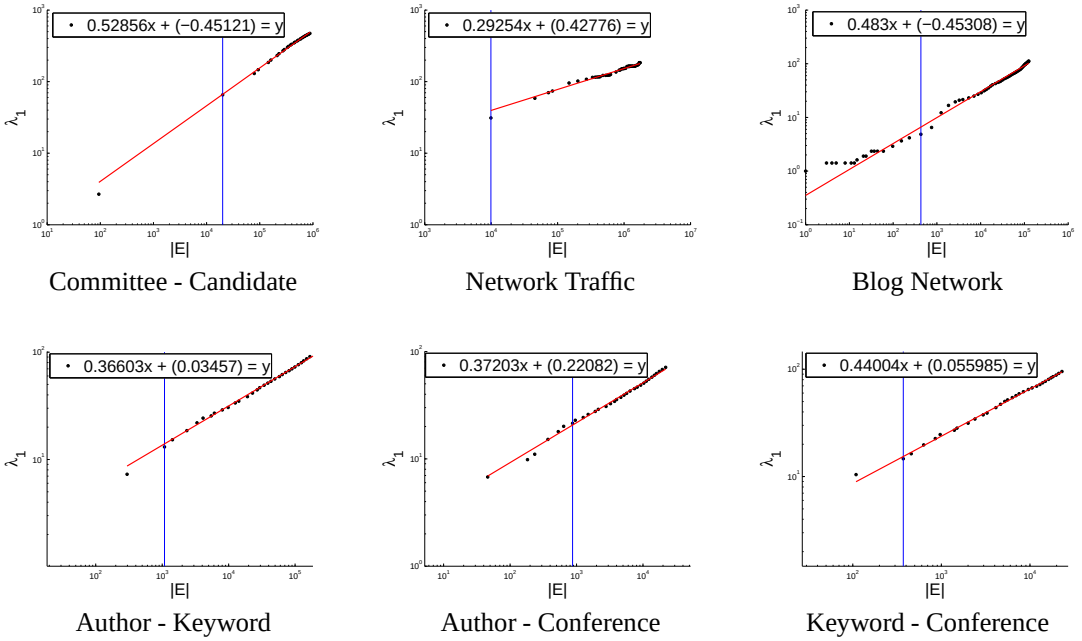


Figure 1. Illustration of the LPL. 1^{st} eigenvalue $\lambda_1(t)$ of the *0-1* adjacency matrix $\mathbf{A}$ versus number of edges $E(t)$ over time. The vertical lines indicate the gelling point.

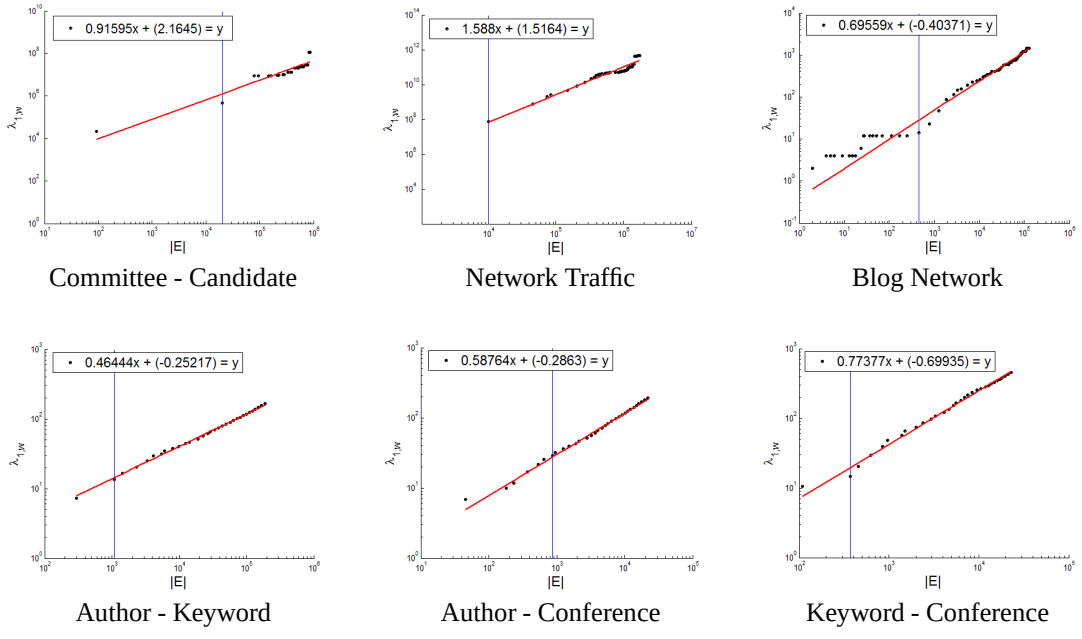
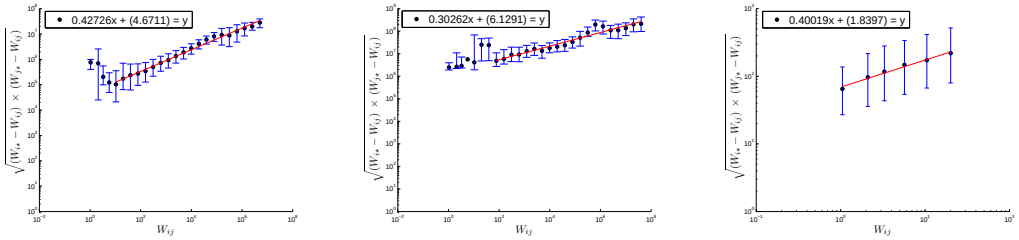


Figure 2. Illustration of the LWPL. 1^{st} eigenvalue $\lambda_{1,w}(t)$ of the *weighted* adjacency matrix $\mathbf{A}_w$ versus number of edges $E(t)$ over time. The vertical lines indicate the gelling point.



(a) Committee - Candidate (b) Network Traffic (c) Blog Network

(d) Author - Keyword (e) Author - Conference (f) Keyword - Conference

Figure 3. Illustration of the EWPL. Given the weight of a particular edge in the final snapshot of real graphs (x-axis), the multiplication of total weights(y-axis) of the edges incident to two neighboring nodes follow a power law. A line can be fit to the median values after logarithmic binning on the x-axis. Upper and lower bars indicate 75% and 25% of the data, respectively.

Here, we show how to build a generator that will match all of the properties listed. The idea is to use recursion not only on the adjacency matrix, but also on the *time* dimension. Specifically, we start with a small tensor $\mathcal{I}$ that has 3 sides ('modes'): (a) senders (b) recipients and (c) time. We call the graph represented by a tensor a 't-graph' that evolves over time (See Fig. 4(a-b)). Then, we recursively substitute every cell (i, j, t) of the original tensor $\mathcal{I}$, with a copy of itself, and multiply it with the value $a_{i,j,t}$ (See Fig. 4(c) for illustration and Definition 1 for full details). Thanks to the self-similarity of the construct, we expect the resulting tensor to have all the properties we want.

First, we give the details of the construction. Secondly, we provide proofs to show that our model will generate graphs with desired properties. Finally, we give our experimental results.

5.1 Description

For the construction, we choose an initial $(N \times N \times \tau)$ tensor $\mathcal{I}$ with nonzero cells (i, j, t) indicating an edge from node i to node j at time tick t. We initialize the cells so that the initial t-graph(t- for time-evolving) $\mathcal{G}_{\mathcal{I}}$ represented by $\mathcal{I}$ looks like a miniature real-world graph. We provide details for how to initialize $\mathcal{I}$ in Section 5.3.

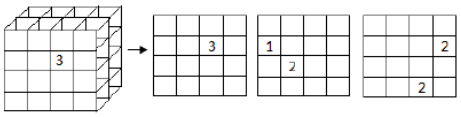
Note that *RTM* works for directed and/or bipartite t-graphs. A bipartite t-graph can be represented by an $(N \times M \times \tau)$ tensor. For simplicity, we focus on unipartite graphs in our work.

We propose to use *Recursive Tensor Multiplication* to produce a time-evolving graph. Our method extends Kronecker product [1] of two matrices by adding a third 'mode'. Kronecker product of two matrices is defined as follows: Given two matrices $\mathbf{A}$ and $\mathbf{B}$ of sizes $(N \times M)$ and $(N' \times M')$, respectively, the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$, namely matrix $\mathbf{C}$ of dimension $(N * N') \times (M * M')$ is given by

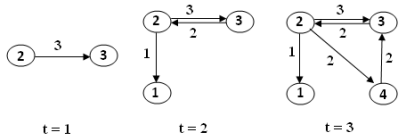
$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \dots & a_{1,M}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,2}\mathbf{B} & \dots & a_{2,M}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1}\mathbf{B} & a_{N,2}\mathbf{B} & \dots & a_{N,M}\mathbf{B} \end{pmatrix}$$

Definition 1 (Recursive Tensor Multiplication (RTM)) *Given two tensors $\mathcal{A}$ of size $(N \times M \times \tau)$ and $\mathcal{B}$ of size $(N' \times M' \times \tau')$, Recursive Tensor Multiplication $\mathcal{C}$*

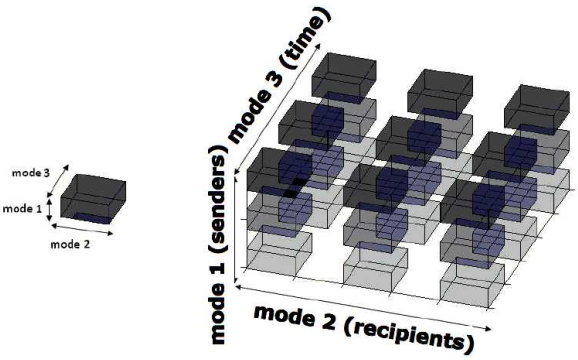
[1] Unfortunately, Kronecker product $\mathbf{C}$ of two matrices $\mathbf{A}$ and $\mathbf{B}$ is also called Kronecker Tensor multiplication, despite $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ are matrices. To disambiguate, we use the name *RTM* where $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ are in fact tensors.



(a) $(4 \times 4 \times 3)$ tensor → t-slices



(b) corresponding t-graph over time



(c) *RTM* of a tensor by itself

Figure 4. (a) An example for the initial tensor $\mathcal{I}$ of size $(4 \times 4 \times 3)$. The 't-slices' represent the changes on the adjacency matrix at every other time step. (b) The corresponding graph represented by the tensor in part (a). It changes according to the 't-slices' over time. (c) An example $(3 \times 3 \times 3)$ tensor $\mathcal{I}$ is given on the left. Recursive tensor product of $\mathcal{I}$ by itself, that is, the resulting $(3^2 \times 3^2 \times 3^2)$ tensor $\mathcal{D} = \mathcal{I} \textcircled{t} \mathcal{I}$ is given on the right.

of $\mathcal{A}$ and $\mathcal{B}$ is obtained by replacing each cell $a_{i,j,t}$ of tensor $\mathcal{A}$ with $a_{i,j,t} * \mathcal{B}$. The resulting tensor $\mathcal{C}$ is of size $(N * N') \times (M * M') \times (\tau * \tau')$ such that

$$c_{((i-1)*N+i'),((j-1)*M+j'),((k-1)*\tau+k')} = a_{i,j,k} * b_{i',j',k'}.$$

An example of the Recursive Tensor Multiplication of a $(3 \times 3 \times 3)$ tensor by itself is given in Fig. 4(c).

To generate a growing graph over time, we get the *'Recursive Tensor Multiplication'* of the initial $(N \times N \times \tau)$ tensor $\mathcal{I}$ by itself k times as:

$$\mathcal{I}^k = \mathcal{D} = \underbrace{\mathcal{I} \textcircled{t} \mathcal{I} \textcircled{t} \ldots \textcircled{t} \mathcal{I}}_{k\ times}$$

and then we take the final tensor $\mathcal{D}$ to represent our data. The data spans τ^k number of time ticks with N^k nodes. At every time step t ($t = \{1, 2, ..., \tau^k\}$), we get the t-slice (See Definition 2 below) $\mathbf{D}_t$ of $\mathcal{D}$, and for each nonzero cell $a_{i,j}$ of $\mathbf{D}_t$, we add an edge between node i and node j with weight $a_{i,j}$. If the edge already exists, we increase the weight $w_{i,j}$ by the same amount.

After giving the details of the construction, next we give proofs for the characteristics that *RTM* will generate. Before that, we define the terms we use throughout the proofs.

Definition 2 (t-slice of a tensor $\mathcal{T}$) *Given a tensor $\mathcal{T}$ of size $(N \times M \times \tau)$, t-slice of $\mathcal{T}$ is a matrix $\boldsymbol{T}_t$ such that*

$$T_t \equiv \mathcal{T}(i, j, t), \quad \forall i, \forall j, \quad 1 \leq i \leq N, 1 \leq j \leq M$$

Definition 3 ((Normalized) temporal (t-) profile of $\mathcal{T}$) *Given a tensor $\mathcal{T}$ of size $(N \times M \times \tau)$, let s_t denote the total weight of its t-slice. Then, the t-profile of $\mathcal{T}$ is a $(1 \times \tau)$ vector, such that*

$$\mathbf{s}_{\mathcal{T},0} \equiv (s_1, s_2, \ldots, s_\tau)$$

Total weight $W_\mathcal{T}$ of $\mathcal{T}$ can be written as $\sum_{t=1}^{\tau} s_t$. Then, normalized t-profile of $\mathcal{T}$ is a $(1 \times \tau)$ vector, such that

$$\mathbf{p}_{\mathcal{T},0} \equiv (\frac{s_1}{W_\mathcal{T}}, \frac{s_2}{W_\mathcal{T}}, \ldots, \frac{s_\tau}{W_\mathcal{T}})$$

5.2 Theorems and Proofs

Recursive Tensor graphs can be shown to exhibit several real-world graph properties. In particular, if we choose the initial graph to be a miniature of a real graph, after recursive iterations of *RTM*, the resulting graph will follow similar properties as of the initial graph due to self-similarity of the construction.

Theorem 1 (Self-similar and Bursty Edge/Weight Additions) *Let edge/weight additions for $\mathcal{I}$ with $\mathbf{p}_{\mathcal{I},0}$ be self-similar and bursty for which the slope of the entropy plot is*

$$slope = H(\mathbf{p}_{\mathcal{I},0}) = -\sum_{i=1}^{\tau} \mathbf{p}_{\mathcal{I},0}(i) log_2(\mathbf{p}_{\mathcal{I},0}(i)),$$

After k iterations of RTM*, edge/weight arrivals over time for $\mathcal{D}$ are also self-similar and bursty. The slope of the entropy plot over* all *aggregation levels r of $\mathcal{D}$ is equal to*

$$slope = H(\mathbf{p}_{\mathcal{D},r}) = H(\mathbf{p}_{\mathcal{I},0}), \quad \forall r$$

where $H(\mathbf{p}_{\mathcal{D},r})$ is the slope of the entropy plot at aggregation level r. Furthermore, the slope does not change with the value of k, that is, burstiness is independent of scale.

Proof We will prove for weight additions and similar arguments apply for edge additions.
After k iterations of *RTM*, total weight of $\mathcal{D}$ becomes

$$W_{\mathcal{D}} = W_{\mathcal{I}}^k = (s_1 + s_2 + \ldots + s_\tau)^k$$

At aggregation level (resolution) 1, we group slices by τ^{k-1} into τ groups. Then, $W_{\mathcal{D}}$ can be written as

$$W_{\mathcal{D}} = s_1 * W_{\mathcal{I}}^{k-1} + s_2 * W_{\mathcal{I}}^{k-1} + \ldots + s_\tau * W_{\mathcal{I}}^{k-1}$$

Then for $1 \leq t \leq \tau$,

$$\mathbf{p}_{\mathcal{D},1}(t) = \frac{s_t * W_{\mathcal{I}}^{k-1}}{W_{\mathcal{I}}^k} = \frac{s_t}{W_{\mathcal{I}}} = \mathbf{p}_{\mathcal{I},0}(t)$$

At aggregation level 2, we group slices by τ^{k-2} into τ^2 groups. Then,

$$\begin{aligned} W_{\mathcal{D}} &= s_1 * (s_1 * W_{\mathcal{I}}^{k-2} + s_2 * W_{\mathcal{I}}^{k-2} + \ldots + s_\tau * W_{\mathcal{I}}^{k-2}) \\ &+ s_2 * (s_1 * W_{\mathcal{I}}^{k-2} + s_2 * W_{\mathcal{I}}^{k-2} + \ldots + s_\tau * W_{\mathcal{I}}^{k-2}) \\ &+ \qquad \ldots \\ &+ s_\tau * (s_1 * W_{\mathcal{I}}^{k-2} + s_2 * W_{\mathcal{I}}^{k-2} + \ldots + s_\tau * W_{\mathcal{I}}^{k-2}) \end{aligned}$$

For any slice i at level 1 and t at level 2, $1 \leq t \leq \tau^2$,

$$\mathbf{p}_{\mathcal{D},2}(t) = \frac{s_i * s_t * W_{\mathcal{I}}^{k-2}}{s_i * W_{\mathcal{I}}^{k-1}} = \frac{s_t}{W_{\mathcal{I}}} = \mathbf{p}_{\mathcal{I},0}(t)$$

Finally, at aggregation level k, we group slices by τ^0 into τ^k groups as

$$\begin{aligned} W_{\mathcal{D}} &= (s_1)^{k-1} * (s_1 + s_2 + \ldots + s_\tau) \\ &+ (s_1)^{k-2} * s_2 * (s_1 + s_2 + \ldots + s_\tau) \\ &+ \qquad \ldots \\ &+ (s_\tau)^{k-1} * (s_1 + s_2 + \ldots + s_\tau) \end{aligned}$$

For all combinations of $(k-1)$ slices at levels from 1 to $(k-1)$, let c_j denote the corresponding coefficients, $1 \leq j \leq \tau^{k-1}$, and for any slice t at level k, $1 \leq t \leq \tau^k$

$$\mathbf{p}_{\mathcal{D},k}(t) = \frac{c_j * s_t}{c_j * W_{\mathcal{I}}} = \frac{s_t}{W_{\mathcal{I}}} = \mathbf{p}_{\mathcal{I},0}(t)$$

We showed that normalized t-profile, $\mathbf{p}_{\mathcal{D},r}$, of $\mathcal{D}$ remains the same at all aggregation levels r and is equal to that of the initial tensor $\mathcal{I}$. So, we conclude that bias of burstiness for $\mathcal{D}$ is the same as that of $\mathcal{I}$ for all values of k, since $H(\mathbf{p}_{\mathcal{D},r})$ would not change for $\forall r$. ∎

Starting with a $(10 \times 10 \times 2)$ $\mathcal{I}$, where $\mathbf{p}_{\mathcal{I},0}(1) : \mathbf{p}_{\mathcal{I},0}(2) = 0.175 : 0.825$; after $k = 3$ iterations, the slope of the entropy plot is obtained to be 0.669, which is equal to

$$H(\mathbf{p}_{\mathcal{I},0}) = .175 * log_2(.175) + .825 * log_2(.825)$$

See Fig. 5.2(c).

Theorem 2 (Weight Power Law (WPL)) *If the initial graph $\mathcal{G}_{\mathcal{I}}$ exhibits the WPL [12] at* all *time ticks, that is, number of edges $E(t)$ and total weight $W(t)$ over time follow a power law with exponent α, $\mathcal{G}_{\mathcal{D}}$ shows the same property at time ticks $1, \tau^1, \tau^2, \ldots, \tau^k$ with exactly the same exponent α.*

Proof We are given the condition that $\mathcal{G}_{\mathcal{I}}$ follows the *WPL* at *all* time ticks, that is,

$$e_1^\alpha = s_1, (e_1 + e_2)^\alpha = (s_1 + s_2), \ldots, (\sum_{t=1}^{\tau} e_t)^\alpha = (\sum_{1}^{\tau} s_t).$$

After k iterations of *RTM*, the resulting graph has $E^k = (\sum_{t=1}^{\tau} e_t)^k$ edges and $W_{\mathcal{I}}^k = (\sum_{t=1}^{\tau} s_t)^k$ total weight. At $t = \tau^k$,

$$E^\alpha = W_{\mathcal{I}} \Rightarrow (E^\alpha)^k = W_{\mathcal{I}}^k \Rightarrow (E^k)^\alpha = W_{\mathcal{I}}^k$$

At aggregation level 1, E^k can be written as,

$$E^k = e_1 * E^{k-1} + e_2 * E^{k-1} + \ldots + e_\tau * E^{k-1}$$

Same argument holds for $W_{\mathcal{I}}^k$. So, at $t = \tau^{k-1}$, number of edges is $(e_1 * E^{k-1})$ and total weight is $(s_1 * W_{\mathcal{I}}^{k-1})$. And,

$$(e_1 * E^{k-1})^\alpha = (e_1)^\alpha * (E^\alpha)^{k-1} = s_1 * W_{\mathcal{I}}^{k-1}$$

In general, at every aggregation level r, at $t = \tau^r$, the graph follows the *WPL* as

$$(e_1^r * E^{k-r})^\alpha = (e_1^\alpha)^r * (E^\alpha)^{k-r} = s_1^r * W_{\mathcal{I}}^{k-r}$$

∎

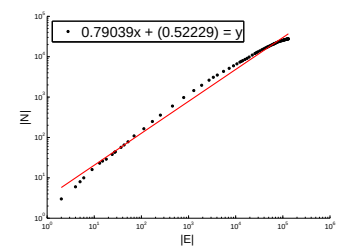
We observe that when we interpolate total weight $W(t)$ versus number of edges $E(t)$ at all time ticks $t \in \{1, 2, \ldots, \tau^k\}$ for the final graph $\mathcal{G}_{\mathcal{D}}$, the resulting exponent remains very close to α. In Fig. 5.2(b), the user-specified WPL exponent (See Section 5.3) is 1.5, which is equal to the slope when points at $t = \{1, 2, 4, 8\}$ are used to fit a line ($k = 3$). When all points are used, the slope is 1.47.

5.3 Initializing $\mathcal{I}$

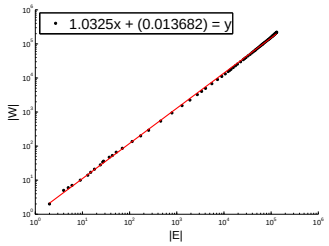
In order to take advantage of the self-similarity property of our construct, we want the initial graph $\mathcal{G}_{\mathcal{I}}$ to be a realistic graph itself. Basically, one can use any graph generator in the literature that is known to produce realistic graphs [5, 19, 10] to generate $\mathcal{G}_{\mathcal{I}}$.

To our knowledge, since *RTM* is the first *weighted* graph generator, we also take weights into consideration. Particularly, we force the initial graph to obey the WPL at all time ticks. That is, when a link occurs between two nodes, we put weight on the edge, so that number of edges $E(t)$ and total weight $W(t)$ over time follow a power law, with a user-specified exponent α.

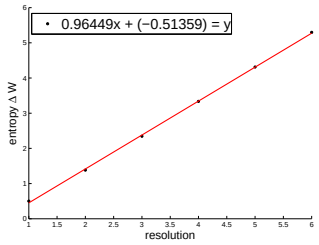
In our experiments, we used the *Butterfly model* [12] with weights. Note that this generator is shown to generate realistic un-weighted graphs. See [12] for more details.



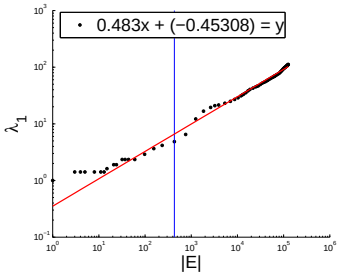
(a) Densification Power Law (DPL)



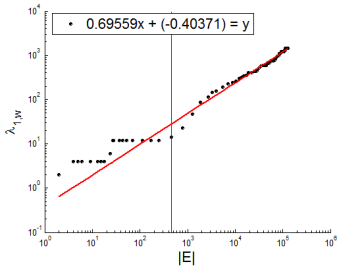
(b) Weight Power Law (WPL)



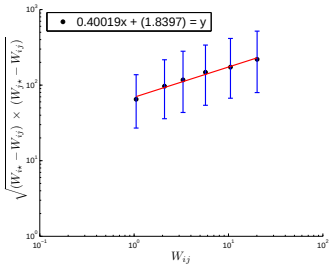
(c) ΔW entropy



(d) λ_1 Power Law (LPL)

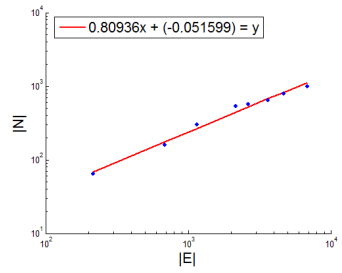


(e) $\lambda_{1,w}$ Power Law (LWPL)

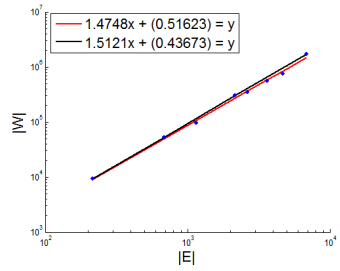


(f) Edge Weight Power Law (EWPL)

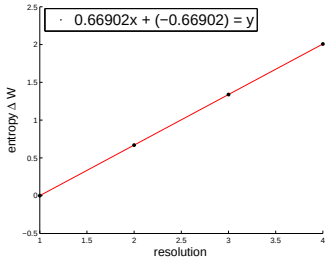
Figure 5. Plots showing related laws that real-world graphs obey for *BlogNet*. First row shows previous laws while second row shows observations of this work.



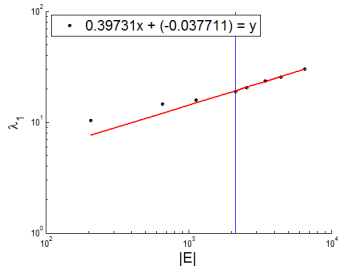
(a) Densification Power Law (DPL)



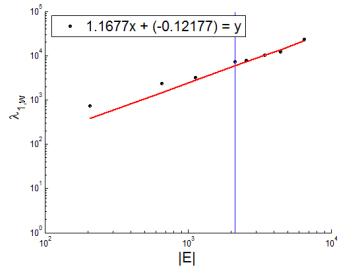
(b) Weight Power Law (WPL)



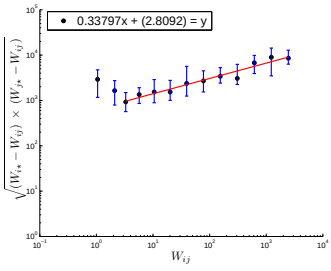
(c) ΔW entropy



(d) λ_1 Power Law (LPL)



(e) $\lambda_{1,w}$ Power Law (LWPL)



(f) Edge Weight Power Law (EWPL)

Figure 6. Plots showing related laws our *RTM* generator produced. Notice that they are very similar in all the listed properties to those of *BlogNet*.

6 Experimental Results

Having created the initial tensor, we take the Recursive Tensor Multiplication of $\mathcal{I}$ by itself k times. The final tensor $\mathcal{D}$ is a $(N^k \times N^k \times \tau^k)$ tensor spanning τ^k time ticks. At every time tick t, we take the t-slice $\mathbf{D}_t$ of $\mathcal{D}$. Next, we simply introduce an edge from node i to node j with weight $a_{i,j}$ for every nonzero entry of $\mathbf{D}_t$. If node i or node j did not exist, we introduce new node(s). If both the nodes and the edge inbetween existed, we only increase the edge weight by $a_{i,j}$.

We did several experiments for different values of N, τ and k. Our model produced realistic graphs for a wide range of parameters, the results being independent of the number of iterations k. In fact, k can be chosen as large as the size of the graph that one needs to generate.

As a comparison with real-world data, we give the plots showing reported laws for *BlogNet* in Fig.5. The plots our model generated for $N = 10$, $\tau = 2$ and $k = 3$ are shown in Fig.6. In particular, we show (a) the Densification Power Law (DPL); (b) the Weight Power Law (WPL); (c) bursty weight additions; (d) the λ_1 Power Law (LPL), (e) the $\lambda_{1,w}$ Power Law and finally, (f) the Edge Weight Power Law (EWPL). Other desired characteristics such as small and shrinking diameter, the gelling point and the Degree Power Law for degree distribution of nodes are also matched, but omitted here for brevity.

Note that characteristics matched by *RTM* include both those from previous work as well as additional patterns discovered in this work.

7 Conclusion

This is one of the few papers that focus on real, weighted, time-evolving graphs. The contributions are the following:

1. We discovered several patterns that such graphs follow, like the eigenvalue (λ_1) power law (*LPL*), and the edge-weight power law (*EWPL*).

2. We gave a simple, recursive generator, *Recursive Tensor Model* (RTM), that mimicks a *long* list of the power laws observed on weighted time-evolving graphs, as well as on unweighted and/or static graphs.

3. We rigorously proved that *RTM* produces several desired characteristics.

Future work abounds: weighted time-evolving graphs have only recently attracted attention. We believe that they will obey several more, fascinating patterns. Such patterns will help us understand the mechanism that make them evolve, will help us spot anomalies (fraud, spam, hardware failures) and they will help us do extrapolations, 'what if' scenarios, and fill in missing values.

References

[1] R. Albert, H. Jeong, and A.-L. Barabasi. Diameter of the world wide web. *Nature*, 401:130–131, 1999.

[2] A.-L. Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435:207, 2005.

[3] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1), 2006.

[4] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A recursive model for graph mining. *SIAM Int. Conf. on Data Mining*, Apr. 2004.

[5] P. Erdos and A. Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hungary. Acad. Sci.*, 5:17–61, 1960.

[6] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *SIGCOMM*, pages 251–262, Aug-Sept. 1999.

[7] I. J. Farkas, I. Derényi, A.-L. Barabási, and T. Vicsek. Spectra of 'real-world' graphs: Beyond the semi-circle law. *Physical Review E*, 64, 2001.

[8] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins. The Web as a graph: Measurements, models and methods. *Lecture Notes in Computer Science*, 1627:1–17, 1999.

[9] J. Leskovec, D. Chakrabarti, J. M. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In A. Jorge, L. Torgo, P. Brazdil, R. Camacho, and J. Gama, editors, *PKDD*, volume 3721 of *Lecture Notes in Computer Science*, pages 133–145. Springer, 2005.

[10] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proc. of ACM SIGKDD*, pages 177–187, Chicago, Illinois, USA, 2005. ACM Press.

[11] F. C. L. Lu and V. Vu. Eigenvalues of random power law graphs. *Annals of Combinatorics*, 33:7:21, 2003.

[12] M. McGlohon, L. Akoglu, and C. Faloutsos. Weighted graphs and disconnected components: Patterns and a generator. In *ACM SIGKDD*, Las Vegas, Aug 2008.

[13] M. McGlohon, J. Leskovec, C. Faloutsos, N. Glance, and M. Hurst. Finding patterns in blog shapes and blog evolution. In *International Conference on Weblogs and Social Media*, Boulder, Colo., Mar. 2007.

[14] M. Mihail and C. Papadimitriou. The eigenvalue power law, 2002.

[15] M. E. J. Newman. Power laws, pareto distributions and zipf's law, December 2004.

[16] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos. Powerlaws and the as-level internet topology, 2003.

[17] M. Wang, T. Madhyastha, N. H. Chang, S. Papadimitriou, and C. Faloutsos. Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic. *ICDE*, Feb. 2002.

[18] M. Wang, T. Madhyastha, N. H. Chang, S. Papadimitriou, and C. Faloutsos. Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic. *ICDE*, February 2002.

[19] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[20] H. S. Wilf. The eigenvalues of a graph and its chromatic number. *J. London Math. Soc.*, 42:330:332, 1967.