

# A Uniform Methodology for Ranking Internet Topology Models

Ivona Bezáková  
Department of Computer Science  
University of Chicago  
Chicago, Illinois 60637  
Email: ivona@cs.uchicago.edu

Adam Kalai  
Toyota Technological Institute at Chicago  
Chicago, Illinois 60637  
Email: kalai@tti-c.org

Rahul Santhanam  
Department of Computer Science  
University of Chicago  
Chicago, Illinois 60637  
Email: rahul@cs.uchicago.edu

**Abstract**—In recent years, there has been a proliferation of theoretical graph models, e.g., preferential attachment, motivated by real-world graphs such as the Web or Internet topology. Typically these models are designed to mimic particular properties observed in the graphs, such as power-law degree distribution or the small-world phenomenon. The mainstream approach to comparing models for these graphs has been somewhat subjective and very application dependent. Comparisons are often based on specific graph properties, without adequate justification for prioritizing some properties over others.

We propose to use the Maximum Likelihood Estimation (MLE) principle to compare graph models: models are scored by the probability with which they generate the real data. Our methodology has several advantages. It is uniform, in that its definition does not presuppose any information about the data or the models. It is unambiguous, in that it yields a clearly defined score for each model, and thus an ordering of models. Moreover, it can be used to determine the best values of the parameters for a given model.

We demonstrate the feasibility of the approach by designing and implementing algorithms computing the probability for four natural models: a power-law random graph model, a preferential attachment model, a small-world model, and a uniform random graph model. We tested our algorithms on three different snapshots of the AS-level Internet topology. We found that the preferential attachment model performed the best, closely followed by the power-law model, with the other two models lagging behind. An interesting aspect of the findings is the fact that the optimal parameters for the power-law models have not changed significantly over time, even though the size of the data has grown by an order of magnitude.

## I. INTRODUCTION

The Internet has undergone rapid growth in the past decade. The problem of accurately modeling the Internet topology has assumed great importance. To take just one example, the AS-level topology is critical to the convergence of the BGP inter-domain routing protocol [1]. Incorporating an accurate model into topology generators is key to correctly determining the performance and scaling of network protocols.

Early topology generators [2], [3] used either hierarchical models or geometric random graph models. In a seminal paper, Faloutsos et al. [4] observed that the AS-level Internet topology had a power-law degree distribution, a property that none of the existing topology generators was able to produce. A similar observation was made by Adamic and Huberman [5] about the graph defined by the link structure of the World

Wide Web. Following this observation, a number of random graph models [6], [7], [8], [9], [10], [11], [12], [13] such as preferential attachment (PA) models, power law random graph (PRG) models and models based on heuristically-optimized tradeoffs (HOT) have been defined which attempt to replicate and explain the power-law degree distribution of the Internet graph and the Web graph. Several topology generators have also been proposed based on these models [14], [15], [10].

However, it remains unclear which of these several models is the “best” one for the Internet topology. It may be possible to prove mathematical theorems establishing that a model has “nice” properties, but this does not say much about how well the model fits the actual Internet data. Also, to say that one model is better than another because it replicates a certain property A of the Internet topology is unsatisfactory, because there may be a different property B on which the second model does better, and it is unclear how to assess the relative significance of properties A and B. For instance, it has been observed that the Internet topology exhibits the small-world phenomenon [10] which is found in several large self-organizing networks [16], and there is a well-known model of networks exhibiting the small-world phenomenon [17]. Distinguishing between different models generating graphs with power-law degree distributions is difficult enough; how do we compare a model designed to replicate the power-law behavior with a model designed to produce small-world behavior? There is a real need for a methodology for comparing Internet topology models which is uniform, unambiguous and fair.

In this paper, we propose the use of the Maximum Likelihood Estimate (MLE) principle from statistics [18], [19] to rank random graph models. Each random graph model, by definition, determines a probability distribution over graphs. Given this observation, our criterion for ranking random graph models is both natural and simple: we prefer models which assign larger probability to the real-world Internet graph.

There are several advantages to using the MLE principle, apart from naturalness and simplicity. There is a well-defined score associated with each model, which implies an unambiguous ranking of models. The MLE principle is very general, and hence can be used to compare diverse models in a uniform way - the definition of the principle does not depend on specific properties of the models or of the data.

Also, the principle has a sound theoretical basis - if one generates several graphs using a certain model and then computes the average score with respect to various models, the model that will score the highest will be the original model using which the graphs were generated. Intuitively, this is a property that any good scoring principle should possess. Imagine that there is some “correct” model  $M$  for the Internet topology, and that the Internet topology is a “typical” graph generated by this model. In such a situation, we would like the model  $M$  to score better on the Internet topology than any other model, an expectation that is satisfied when our scoring principle is used.

Further evidence of the power of the MLE principle is its cross-disciplinary impact. Metrics such as log-loss, perplexity and cross-entropy, which derive from the MLE principle, have had proven success in fields such as machine learning, language modeling and stochastic optimization.

We emphasize that we are concerned with the *predictive* power of a model, i.e., the degree to which it is successful in making predictions about real-world graphs. We are not judging the *explanatory* power of a model, if any. The validity of explanations and causal relationships is much more difficult to quantify.

From a theoretical point of view, the MLE principle is compelling; the challenge is in applying it to rank graph models. The main difficulty is computational. Popular random graph models are often generative, i.e., they describe a process for generating a random graph but not necessarily how to calculate this probability for a given graph. Naive methods for calculating the probability typically take exponential time and are therefore infeasible. We design novel algorithms and heuristics based on Monte Carlo Markov Chains (MCMC) to solve this problem.

There are also certain conceptual issues that arise when applying the MLE principle in the context of networking. First, there is the question of whether connectivity-only information is adequate when representing the Internet topology. Since all common probabilistic models for the Internet topology only generate such information, when we apply our methodology, we must represent the real-world data in the same way. Another set of issues has to do with what kind of graph we consider - whether to interpret the real-world graph as labelled or unlabelled, directed or undirected, having simple edges or multi-edges. We discuss these issues in more detail in the Preliminaries section. Yet another issue has to do with the accuracy of the data, which is in our case the AS-level Internet topology. Our findings are relative to the technology we apply for inferring AS-level peering relationships from traceroute measurements [20]; as this technology improves, our conclusions will gain in accuracy.

To illustrate the feasibility of our approach for existing models, we implement our algorithms for computing the probability for four random graph models. The selection of models represents different streams of graph modeling research. We include a power-law model (PRG), a preferential attachment model (PA), a small-world model (SW), and for comparison,

the Erdős-Rényi (ER) random graph model. Based on experiments on three snapshots of the AS-level Internet topology, we find that the preferential attachment model ranks highest, closely followed by the power-law model, with the random graph model coming last. Moreover the optimal parameters for the preferential attachment model and the power-law model do not change significantly between snapshots, although the sizes of the snapshots differ by an order of magnitude.

Our algorithms have the additional benefit that they can be used for estimating the best parameters for models. The need for such an estimation procedure may arise when setting parameters for a simulator based on a given model.

We do not claim that the preferential attachment model is the best available model for the Internet topology. This model ranked highest among models for which we implemented probability estimation procedures; as and when we are able to feasibly implement such procedures for other kinds of models, certain of the latter may emerge superior. Our main goal in this paper is to demonstrate that an objective comparison of models is possible, not only in theory but also in practice. We hope that in addition to enabling current models to be ranked, this will stimulate the development of improved models.

## II. RELATED WORK

Our paper is primarily concerned with the issue of how different models can be compared. This question has received a great deal of attention in the literature on models for the Internet topology. One natural approach is to identify key topology metrics such as clustering coefficient, average path length, and degree distribution, and compare different models by assessing how accurately they reproduce the properties observed in the real-world Internet topology [9], [10], [21]. The main problem with this approach is that it is subjective. The choice of properties can vary and the comparison may depend critically on this choice.

A completely different approach is advocated by Willinger et al. [22], who distinguish between two kinds of models, “evocative” models which attempt to replicate the properties of the data, and “explanatory” models which attempt to replicate the mechanisms producing the data. They propose a “closing-the-loop” approach to model validation for explanatory models. In this approach, an explanatory model can be tested by determining whether the mechanisms it proposes are reflected in the evolution of the data over time. Chen et al. [23] use this approach on the preferential attachment model of [6], and conclude that the observed data do not validate the model.

The closing-the-loop approach is an attractive one for explanatory models, but it differs from ours in that it is used for validating models rather than ranking them.

Recently, HOT models (highly-optimized tolerance/heuristically optimal tradeoffs) [24], [25] have gained in popularity. Such models attempt to abstract out the constraints and objectives driving the participation of a node in the network. A comparison of these models to standard models has been attempted by Li et al. [26]. They criticize approaches to topology modeling which focus on

macroscopic statistics such as degree distribution. They define “performance” and “likelihood” metrics for a network, where the likelihood of a network is essentially the probability that it is generated by a certain power law random graph model. They show that HOT networks which incorporate technological constraints and economic considerations into their design at the router-level have high performance and low likelihood, while several typical high-likelihood networks have low performance. They conclude that traditional graph models are inadequate to capture the essential characteristics of the router-level Internet topology.

Our approach addresses the main criticism of previous approaches in [26], in that it is based on the probability of generating the real network, rather than just measuring how accurately some macroscopic statistic is reproduced. Moreover, our approach can be used to put their observations about performance and likelihood into context. HOT networks may have low likelihood with respect to power law random graph models, but the method of [26] for producing these networks implicitly defines a model, with respect to which the networks will have high likelihood. If high performance of a network is taken to indicate similarity to the Internet, then their experiments suggest the possibility that their implicitly defined model outperforms preferential attachment and power law random graph models when our ranking principle is applied. It would be interesting to test this by formally defining their model and computing the probability of generating the Internet topology using their model. The challenge here is to do the computation of the probability in an efficient way, since HOT models tend to be rather complicated. The question remains whether it is useful to compare models which focus more on the role of randomness than of design, such as the ones considered in the present paper. We believe this is still useful for the case of the AS-level topology, where macroscopic characteristics may play a bigger role than in the router-level case, and where design tradeoffs may be harder to formulate.

An additional point worth making is that HOT models aim for a detailed causal explanation of the evolution of the Internet topology. Thus, while it would be interesting to assess their predictive power using our methodology, their ultimate validation must derive from a different kind of approach, such as the “closing-the-loop” approach.

### III. PRELIMINARIES

In this section, we formally introduce our ranking methodology and describe the models we rank with respect to the Internet topology using our methodology.

#### A. Ranking Methodology

We use the MLE principle, which scores models based on the probability with which they generate the real-world Internet topology, and then rank the models in order of their scores. The actual score we assign a model is not the probability itself but the negative logarithm of the probability. We refer to this quantity as the *negative log-likelihood*. The negative log-likelihood has the property that the average score

of a model on graphs generated by the model exceeds the average score of any other model (which we argued for in the Introduction). In addition, working with logarithms is computationally convenient.

**Input:** A graph  $G$  and a model  $M$  with parameter vector  $\mathbf{p}$ .

**Output:**

$$\begin{aligned} \text{neg-log-likelihood}(G, M, \mathbf{p}) = \\ -\log \Pr(M \text{ with parameter setting } \mathbf{p} \text{ generated } G) \end{aligned}$$

To obtain the score for a model  $M$  with respect to the graph  $G$ , we minimize  $\text{neg-log-likelihood}(G, M, \mathbf{p})$  over all possible settings of the parameter vector  $\mathbf{p}$  (note that maximizing the probability corresponds to minimizing the negative log-likelihood).

One point worth mentioning is that this scoring method gives an advantage to models with more parameters. To take an extreme case, for each graph  $G$ , one can define a model  $M(G)$  which generates the graph  $G$  with probability 1 and all other graphs with probability 0. As per our ranking principle, the model  $M(G)$  should be ranked above all other models with respect to the graph  $G$ . However,  $M(G)$  is clearly an unsatisfactory model, because the model needs to include a description of the graph  $G$  if it is to generate only that graph. We may think of the graph as a parameter of the model; the problem is that the description of the graph may be massive, hence the parameter size is large. To avoid such pathologies when defining the general methodology, we need some way to penalize models for which the parameters cannot be described in a relatively small number of bits. This can be done by adopting the *Minimum Description Length* (MDL) principle: each model is assigned a score which is the sum of the negative log-likelihood and the size of the parameter description.

In our application, the models are simple and the parameters can be described succinctly. Indeed, this is true of any reasonable model for the Internet topology. Thus the parameter description term is negligible, implying that the MDL and MLE principles yield identical rankings. For the sake of conceptual simplicity, we identify our methodology with the MLE principle rather than the more sophisticated MDL principle.

#### B. Graph Models

We want to generate a directed<sup>1</sup> graph  $G = (V, E)$  with vertex set  $V$  and edge set  $E$ , where  $n = |V|$  is the number of vertices and  $m = |E|$  is the number of edges. For a node  $v$ , let  $\text{in}(v)$  be its indegree, i.e., the number of edges pointing into that node and  $\text{out}(v)$  be its outdegree, i.e., the number of edges pointing out of that node.

We assume that the generated graphs are unlabeled. Of course, in our datasets each vertex is specified by a unique number from the set  $\{1, \dots, n\}$  but typically these labels do

<sup>1</sup>Analogous models and algorithms can be defined for undirected graphs. We chose the directed case, since in our application, modeling the Internet topology, the AS-level graph can naturally be seen as directed.

not bear any information relevant to the generating process. Therefore the models we consider are symmetric in the sense that they generate all vertex labelings (each of the  $n!$  permutations of vertex labels) of a given graph with the same probability.

We consider models which represent different streams of graph modeling research. We include a power-law model (PRG), a preferential attachment model (PA), a small-world model (SW), and, for comparison, the Erdős-Rényi (ER) random graph model. By a power-law probability distribution with exponent  $\beta$  and cutoff  $c$  (possibly  $\infty$ ), we mean the distribution over integers that assigns probability to  $i \in \{1, 2, \dots, c\}$  of,

$$\Pr(i) = \frac{i^{-\beta}}{\sum_{j=1}^c j^{-\beta}}.$$

Many graph models are unrealistic in the sense that they would assign probability 0 to most real-world graphs. For example, many preferential attachment models never generate any cycles, and many small-world models only generate graphs which have the grid as a subgraph. In most cases, slight variations on these models are more appealing in that they retain the essential features of the model while assigning positive probability to every graph. We have modified the PA model and the SW model for this purpose.

- **PA model.** The PA model, inspired by [27], simulates a graph growth process where incoming nodes connect to existing nodes with probability proportional to their degree. It is parameterized by probabilities  $p, q$  satisfying  $p + q < 1$ , and a parameter  $\gamma > 0$ . The graph is created iteratively. We start with a single vertex. In each iteration  $i = 2, \dots, n$  a random vertex “appears” and edges between the new vertex and already existing nodes are generated by the following process. With probability  $p$  (resp.  $q$ ) an edge from (resp. to) the new vertex is created and the other end-point  $v$  is chosen with probability proportional to  $\text{in}(v) + \gamma$  (resp.  $\text{out}(v) + \gamma$ ). The edge-generating process is then repeated (possibly resulting in multi-edges). Otherwise, with probability  $1 - p - q$  the process stops and new iteration  $i + 1$  begins. It can be shown that this model produces with high probability a graph whose indegree and outdegree distributions are governed by power laws with exponents  $\beta_{\text{in}}$  and  $\beta_{\text{out}}$ , where  $\beta_{\text{in}}$  and  $\beta_{\text{out}}$  are functions of  $p, q$  and  $\gamma$ .
- **PRG model.** This model is based on the model for random graphs with power-law degree distribution proposed by Aiello, Chung and Lu [7], which is a special case of the model for random graphs with a given degree sequence due to Bollobás [28]. The motivation for this model is that many real-world networks have been observed to exhibit power-law distribution of degrees [6], [4]. Input parameters are  $\beta_{\text{in}}, \beta_{\text{out}}$ , the intended power-law exponents, and cutoffs  $c_{\text{in}}, c_{\text{out}}$ . First we generate the indegrees and outdegrees of each of the  $n$  vertices independently at random according to power-law distributions with exponents  $\beta_{\text{in}}$  and  $\beta_{\text{out}}$ , respectively, and maximum

indegrees and outdegrees  $c_{\text{in}}$  and  $c_{\text{out}}$ , respectively. Then we connect the vertices randomly to match the selected indegrees and outdegrees.

To be precise, we need to specify exactly how the vertices are matched. If the sum of the indegree sequence equals the sum of the outdegree sequence, we connect the vertices as follows. For every vertex, we create  $\text{in}(v)$  copies of  $v: v_1, \dots, v_{\text{in}(v)}$ . Let  $A$  denote the set of all these vertex copies. Similarly, for out-degrees we create a set  $B$ . We then pair up the vertices, one to one, randomly from these two sets. For each pair  $v_i \in A$  and  $u_j \in B$ , we create an edge from  $v$  to  $u$  in the original graph. This process creates a directed graph (possibly a multi-graph). If the sums of the indegrees and outdegrees do not equal, we output the empty graph (a graph with  $n$  nodes and no edges).<sup>2</sup>

- **SW model.** Many real-world networks have been observed to have low diameter and high clustering coefficient [10], [29]. This is known as the small-world phenomenon. Our model for graphs exhibiting the small-world phenomenon is inspired by the Watts-Strogatz and Kleinberg models [17], [30]. The basic idea is to add random links to an underlying well-structured graph, in our case the grid. The parameters of our model are  $s$ , the side of an underlying grid topology, and constants  $\alpha, \beta$ . There are  $s \times s$  vertices arranged in a grid. For every pair of vertices  $u, v$ , an edge from  $u$  to  $v$  is added with probability  $\alpha \text{dist}(u, v)^{-\beta}$ , where  $\text{dist}(u, v)$  is the Manhattan distance on the grid, i.e.,  $|u_x - v_x| + |u_y - v_y|$ . All vertices with no incident edges are omitted. Note that not all the grid edges are necessarily present in the generated graph.
- **ER model.** The Erdős-Rényi model is parameterized by a probability  $p \in [0, 1]$ . For every pair of vertices  $u, v$  an edge from  $u$  to  $v$  appears independently with probability  $p$ .

The PA and PRG models can technically produce graphs with multiple copies of a single edge, which do not occur in our data sets. We have two options - to consider all such graphs with multiple copies of an edge as equivalent to the corresponding simple graph, or to distinguish them. We choose the second option because it makes the probability calculation significantly easier. Choosing the first option would only increase the scores of the PA and PRG models, and hence would not affect the rankings of these models relative to the other models.

#### IV. ALGORITHMS

In this section we present an algorithm for each of the models which estimates the probability that the model (with a specified parameter setting) generates a given directed graph  $G$ . From this we obtain the negative log-likelihood, which

<sup>2</sup>It can be shown that the negative log-likelihoods assigned by this model and the more natural model which generates the degree sequences until the sums match are almost identical. The advantage of our model is that we can calculate the negative log-likelihood exactly.

---



---

PRG( $G, \beta_{\text{in}}, \beta_{\text{out}}, c_{\text{in}}, c_{\text{out}}$ )

- $Z_{\text{in}} := \sum_{d=0}^{c_{\text{in}}} d^{-\beta_{\text{in}}}$ , similarly for  $Z_{\text{out}}$ .
  - $P_{\text{in}} := \prod_{v \in V} \frac{(\text{in}(v))^{-\beta_{\text{in}}}}{Z_{\text{in}}}$ , similarly for  $P_{\text{out}}$ .
  - return  $P_{\text{in}} P_{\text{out}} \frac{1}{m!} \prod_{v \in V} \text{in}(v)! \text{out}(v)!$
- 
- 

Fig. 1. PRG model computation

defines the score of the model. To obtain the best possible score, we need to search the parameter space for the optimal parameters.

The algorithms range from easy (for the ER model) to challenging (for the PA model). The reason is that in the PA model the order in which vertices appear in the graph matters. Most graphs can be generated by using many vertex orderings (some more likely than others) and thus, once we have just the graph (i.e. a snapshot of the Internet topology), we have to consider all possible vertex permutations. Usually, the computation for any specific permutation is relatively easy and the hard part is to average the values over all permutations. We solve this problem by designing a non-trivial MCMC algorithm. We believe the algorithms we use are interesting on their own and the methods might be applicable when evaluating other graph models.

#### A. ER algorithm

Recall that  $n$  is the number of nodes and  $m$  is the number of edges in  $G$ . In the ER model, each edge is generated independently with probability  $p$  and each non-edge with probability  $1 - p$ . There are  $n(n - 1)$  possible (directed) edges, therefore the ER model generates  $G$  with probability  $\Pr(G) = p^m (1 - p)^{n(n-1)-m}$ . For this model we can not only compute the probability exactly but also compute the best parameter  $p = \frac{m}{n(n-1)}$ , which maximizes the above probability.

#### B. PRG algorithm

The algorithm for computing the probability of a graph according to the PRG model is given in Figure 1. Fixing the parameters  $\beta_{\text{in}}, \beta_{\text{out}}, c_{\text{in}}, c_{\text{out}}$ , the probability that the PRG model generates  $G$  can be computed as the product of the probability of generating the in-degree sequence, the probability of generating the out-degree sequence, and the probability of matching up the right vertices.

In the PRG model a vertex gets assigned an indegree  $d \in \{0, \dots, c_{\text{in}}\}$  with probability  $d^{-\beta_{\text{in}}}/Z_{\text{in}}$ , where  $Z_{\text{in}} = \sum_{d=0}^{c_{\text{in}}} d^{-\beta_{\text{in}}}$  is the normalization factor. Therefore, the probability of generating the particular indegree sequence is  $P_{\text{in}}$ , as defined in Figure 1.

Given a simple graph  $G$  and indegree and outdegree sequences whose sums agree, there are  $\prod_{v \in V} \text{in}(v)! \prod_{u \in V} \text{out}(u)!$  matchings of edges in  $A$  and  $B$  that corresponds to  $G$  because for each matching. To see this, notice that for any matching which gives rise to  $G$ ,

we can permute any of the  $\text{in}(v)$  vertices corresponding to vertex  $v \in A$  or any of the  $\text{out}(u)$  vertices corresponding to vertex  $u \in B$  to get another matching corresponding to  $G$ . Moreover, these permutations correspond exactly to the set of matchings that give rise to  $G$ .

Since there are  $n!$  matchings in total between two sets of size  $n$ , the probability of a matching producing  $G$  is exactly  $\frac{1}{n!} \prod_{v \in V} \text{in}(v)! \text{out}(v)!$ . Finally, the probability of generating the graph  $\Pr(G)$  is the product of the probabilities of choosing the degree sequence and choosing a corresponding matching. We cannot easily compute the optimal parameters  $\beta_{\text{in}}, \beta_{\text{out}}, c_{\text{in}}, c_{\text{out}}$ , but, since the computation is fast, we quickly search the (appropriately discretized) parameter space.

#### C. PA algorithm

Given a permutation of the vertices  $\pi$  (the order of their appearance), it is not difficult to compute the probability that the PA model generates  $G$ :

$$\Pr(G|\pi) = \prod_{i=2}^n d_{\pi}(i)! r \prod_{\substack{j < i: \\ (\pi_j, \pi_i) \in E}} p \frac{\text{in}_i^{\pi}(\pi_j) + \gamma}{m_i^{\pi}} \prod_{\substack{j < i: \\ (\pi_i, \pi_j) \in E}} q \frac{\text{out}_i^{\pi}(\pi_j) + \gamma}{m_i^{\pi}},$$

where,

$$\begin{aligned} \text{in}_i^{\pi}(v) &= |\{j : j < i \wedge (\pi_j, v) \in E\}| \\ \text{out}_i^{\pi}(v) &= |\{j : j < i \wedge (v, \pi_j) \in E\}| \\ m_i^{\pi} &= \sum_{k=1}^{i-1} (\text{in}_i^{\pi}(\pi_k) + \gamma) = \sum_{k=1}^{i-1} (\text{out}_i^{\pi}(\pi_k) + \gamma) \\ d_{\pi}(i) &= \text{in}_{i+1}^{\pi}(\pi_i) + \text{out}_{i+1}^{\pi}(\pi_i) \\ r &= 1 - p - q \end{aligned}$$

In words,  $\text{in}_i^{\pi}(v)$  and  $\text{out}_i^{\pi}(v)$  are the in- and out-degrees of vertex  $v$  just *before* the  $i$ th node appears and  $d_{\pi}(i)$  is the number of edges incident to the  $i$ th node generated before the  $(i + 1)$ st iteration begins (before the  $(i + 1)$ st node appears). These edges can be generated in any of the  $d_{\pi}(i)!$  orders. An edge from  $\pi_i$  to  $\pi_j$  (for  $j < i$ ) is generated with probability

$$\Pr(\text{adding edge } (\pi_i, \pi_j)) = p \frac{\text{in}_i^{\pi}(\pi_j) + \gamma}{m_i^{\pi}}$$

and an edge from  $\pi_j$  to  $\pi_i$  appears with probability

$$\Pr(\text{adding edge } (\pi_j, \pi_i)) = q \frac{\text{out}_i^{\pi}(\pi_j) + \gamma}{m_i^{\pi}}.$$

Finally, with probability  $r$  we move to the next iteration. This justifies the above expression for  $\Pr(G|\pi)$ .

For any particular permutation  $\pi$ , the computation of  $\Pr(G|\pi)$  is relatively straightforward compared to the calculation of  $\Pr(G)$ , which is the average over all permutations,

$$\Pr(G) = \frac{1}{n!} \sum_{\pi} \Pr(G|\pi). \quad (1)$$

Obviously, summing over all  $n!$  permutations is not feasible and traditional sampling methods (pick a large sample of

---



---

PA( $G, p, q, \gamma, T$ )

- Let  $\sigma$  be the permutation of vertices sorted by the total degree  $\text{in}(v) + \text{out}(v)$  in decreasing order.
  - return  $\frac{\Pr(G|\sigma)}{n! \prod_{i=1}^n \text{Self-iter}(G, \sigma, i, p, q, \gamma, T)}$ .
- 

Self-iter( $G, \sigma, i, p, q, \gamma, T$ ) // estimates  $\Pr(\sigma_i|G, \sigma_1 \dots \sigma_{i-1})$ .

- $\pi := \sigma$
  - est := 0
  - for  $t=1$  to  $T$ 
    - $v :=$  random vertex from  $\{\sigma_i, \sigma_{i+1}, \dots, \sigma_n\}$
    - $a[j] := \Pr(G|\text{Shift}(\pi, v, j))$  for  $j \in \{i, i+1, \dots, n\}$
    - choose  $j$  at random from  $\{i, \dots, n\}$  with probability  $a[j] / \sum_{k=i}^n a[k]$ .
    - $\pi := \text{Shift}(\pi, v, j)$  // random walk step
    - $b[j] := \Pr(G|\text{Shift}(\pi, \sigma_i, j))$  for  $j \in \{i, i+1, \dots, n\}$
    - est := est +  $b[i] / \sum_{k=1}^n b[k]$  // estimation step
  - return  $\frac{\text{est}}{T}$ .
- 

Shift( $\pi, v, j$ ) is the permutation obtained from  $\pi$  by inserting  $v$  at position  $j$ .

---



---

Fig. 2. PA model computation

uniformly random permutations and average the corresponding  $\Pr(G|\pi)$ 's) do not approximate the sum well. More precisely, for a sufficiently large sample, this would give a good approximation which, with high probability, would be extremely close to the true  $\Pr(G)$ . However, some permutations have  $\Pr(G|\pi)$  exponentially larger than the rest (typically, the good permutations are those where high-degree nodes appear early). The variance is very large — the good permutations are difficult to find, like needles in a haystack. Thus the averaging approach would require exponentially many samples.

Instead, we use an MCMC method to compute  $\Pr(G)$ . This method enables us to sample permutations from the weighted distribution over permutations, with probability of generating  $\pi$  equal to  $\Pr(\pi|G)$ . The calculation is based on Bayes' rule:

$$\Pr(\pi|G) = \frac{\Pr(G|\pi) \Pr(\pi)}{\Pr(G)}$$

Then,

$$\begin{aligned} \Pr(G) &= \frac{\Pr(G|\pi) \Pr(\pi)}{\Pr(\pi|G)} \\ &= \frac{\Pr(G|\pi) \Pr(\pi)}{\prod_{i=1}^n \Pr(\pi_i|G, \pi_1 \dots \pi_{i-1})} \end{aligned}$$

where  $\Pr(\pi) = 1/n!$  and we can calculate  $\Pr(G|\pi)$  explicitly for any  $G$ , by the earlier formula.

Our algorithm is summarized in Figure 2. The algorithm works by fixing a pivot permutation  $\pi := \sigma$  and computing a sequence of conditional probabilities  $\Pr(\sigma_i|G, \sigma_1 \dots \sigma_{i-1})$ , in the Self-iter routine. Then,

$$\Pr(G) = \frac{\Pr(G|\sigma)}{n! \prod_{i=1}^n \text{Self-iter}(G, \sigma, i, p, q, \gamma, T)}$$

To compute each of these probabilities, we run a random walk and estimate the conditional probability from the observed permutations in the walk. We describe the Self-iter routine in more detail below. We note that even though the above derivation works for any pivot permutation  $\pi$ , we use a specific permutation  $\sigma$  for which the algorithm converges significantly faster than for an ordinary  $\pi$ .

*Algorithm correctness:* We need to argue that the routine Self-iter correctly approximates  $\Pr(\sigma_i|G, \sigma_1 \dots, \sigma_{i-1})$ , i.e., the probability that  $\pi_i = \sigma_i$  where  $\pi$  shares the first  $i-1$  nodes with  $\sigma$  and it is chosen with probability proportional to  $\Pr(\pi|G)$ .

Fix the parameters to the Self-iter routine. It performs a random walk on permutations which have  $\pi_1 \dots \pi_{i-1} = \sigma_1 \dots \sigma_{i-1}$ . One step of the walk consists of choosing a random vertex  $v$  appearing on a position  $\geq i$  and considering shifting the element to every position  $j \geq i$ . The final position  $j$  is chosen proportionally to the probability that the resulting permutation generates  $G$ . Of course, we have to argue that such a walk generates a permutation  $\pi$  with probability proportional to  $\Pr(\pi|G)$  (and that not too many steps of the walk are necessary). Before we do this, let us give some intuition for the walk we use.

Imagine trying to shuffle a deck of  $n$  cards uniformly at random. The natural random walk to do (and the first one we tried) would be to pick an arbitrary pair of adjacent cards and swap them. Unfortunately, this would require approximately  $\theta(n^3)$  steps until it is approximately random, because each card would need to be swapped  $\theta(n^2)$  times to be in a random place. In contrast, imagine picking a random card, removing it, and reinserting it in a random place in the deck. The cards will be random as soon as each one has been removed once, which happens in approximately  $\theta(n \log n)$  steps. While the steps in the latter case take  $O(n)$  times longer than the steps in the first case, the second algorithm is still faster by nearly a factor of  $n$ . Since  $n$  is large, this difference amounts to hundreds of hours of computation.

As the random walk progresses, it approaches a *stationary distribution*. For sufficiently large  $T$ , the distribution over  $\pi$ 's in the walk will be arbitrarily close to the stationary distribution. We first argue that the stationary distribution of Self-iter is  $\mu_i^\sigma$  which assigns probability to permutation  $\pi$  of  $\mu_i^\sigma(\pi) = \Pr(\pi|G, \sigma_1 \dots \sigma_{i-1})$ , i.e., the distribution conditioned on the first  $i-1$  elements of the permutation. Note that Self-iter never changes any of the first  $i-1$  elements of the permutation. To see that  $\mu_i^\sigma$  is the stationary distribution, consider what happens when we take a random permutation from the stationary distribution and then take a step. The probability of being in permutation  $\pi$  is:

$$\frac{1}{n-i+1} \sum_{\substack{v \in \{\sigma_i, \dots, \sigma_n\} \\ j \in \{i, \dots, n\}}} \mu_i^\sigma(\text{Shift}(\pi, v, j)) \frac{\mu_i^\sigma(\pi)}{\sum_{k=i}^n \mu_i^\sigma(\text{Shift}(\pi, v, k))}$$

This is because there are  $n-i+1$  vertices that can be inserted during the step. For each such vertex  $v$ , there are  $n-i+1$  positions  $j$  which it could have been in before

moving to permutation  $\pi$ . The probability of being in such a  $v, j$  shifted state is  $\mu_i^\sigma(\text{Shift}(\pi, v, j))$ , and the probability of moving to  $\pi$  is the fraction in the displayed equation. Simple algebraic manipulation shows that the above quantity is  $\mu_i^\sigma(\pi)$ , as required. This implies that  $\mu_i^\sigma$  is the stationary distribution. Finally, one can verify that the  $\frac{\text{est}}{T}$  is a good estimate of  $\Pr(\sigma_i|G, \sigma_1 \dots \sigma_{i-1})$ , for sufficiently large  $T$ .

Theoretical upper bounds can be given on the number of steps (mixing time) sufficient for the walk to converge and the estimates to be accurate. These bounds are too pessimistic for our purposes. Instead, for every graph instance we heuristically estimate the mixing time for a sample of self-iteration steps  $i \in \{1, 2, \dots, n\}$ . We also performed extensive tests to ensure that the number of steps we take is sufficient and that our estimates are very accurate. We discuss these issues in detail in Section V.

#### D. SW algorithm

We do not know how to accurately (and in a reasonable time) estimate the correct negative log-likelihood by the SW model, rather we bound it from below and from above. This information is sufficient to conclude that the SW model ranks worse than the PA and PRG models but not worse than the ER model. Notice that the SW model performs at least as well as the ER model. In particular, setting  $\beta = 0$  gives exactly the ER model with parameter  $p = \alpha$ . It remains to bound the negative log-likelihood by the SW model from below.

We attempt to use a similar MCMC approach as for the PA model. If we knew the initial grid alignment of the vertices  $g$ , then computing the probability of generating  $G$  from  $g$  would be straightforward:

$$\Pr(G|g) = \prod_{(u,v) \in E} \alpha \text{dist}(u,v)^{-\beta} \prod_{(u,v) \notin E} (1 - \alpha \text{dist}(u,v)^{-\beta}),$$

where  $\text{dist}(u, v) = |u_x^g - v_x^g| + |u_y^g - v_y^g|$  is the Manhattan-distance of vertices  $u$  and  $v$  in  $g$ , and  $(v_x^g, v_y^g)$  are the coordinates of vertex  $v$  in  $g$ . As before, we want to compute

$$\Pr(G) = \frac{1}{n!} \sum_g \Pr(G|g)$$

where the sum ranges over all possible grid alignments. An MCMC approach analogous to the one presented for the PA model, a (self-reducible) walk on the grid alignments with stationary distribution proportional to  $\Pr(G|g)$ , converges to the correct negative log-likelihood as the number of steps goes to infinity. Unfortunately, this walk does not mix rapidly, both in practice and theoretically (consider two cliques of size  $n/2$ , they will position themselves on opposite sides of the grid and it will take exponentially long for them to switch).

In lieu of estimating this probability, using standard techniques we give an upper bound on the performance of the algorithm as follows,

$$\Pr(G) = \frac{1}{n!} \sum_g \Pr(G|g) \leq \max_g \Pr(G|g)$$

In other words, the probability of the best grid is an upper bound on the average probability of random grid.

To estimate the probability of the best grid, we use Simulated Annealing [31], a well-known optimization method. It searches among a set of states  $S$  with a real-valued energy function  $c(s)$ . It consists of several phases, each characterized by a temperature  $T$ . For a given  $T$  a Markov chain is run with stationary distribution proportional to  $e^{-c(s)/T}$ . Starting with a high temperature, the distribution is nearly uniform, the temperature is then decreased, biasing the distribution towards low-energy states. Simulated Annealing has met with great empirical success [32].

In our case,  $S$  is the set of all grid alignments and  $c(g) = -\ln \Pr(G|g)$ . The stationary distribution at temperature  $T$  is proportional to  $\Pr(G|g)^{1/T}$ . The transitions are swaps of arbitrary grid vertices. The algorithm, specified in Figure 3, is a standard annealing algorithm with initial temperature  $T_0$  and a geometric cooling schedule.

---



---

```

SW( $G, \alpha, \beta, T_0, R, \delta$ ) // Finds  $\max_g \Pr(G|g)$ .
1)  $T := T_0$  // temperature
2)  $g :=$  random initial assignment to the grid.
3) repeat: // until convergence of  $\Pr(G|g)$ .
   a)  $r := 0$  // step counter
   b) Let  $u$  and  $v$  be two vertices on grid  $g$ . Let  $g'$  be the grid  $g$  with  $u$  and  $v$  swapped.
   c) with probability  $\max \left\{ 1, \left( \frac{\Pr(G|g')}{\Pr(G|g)} \right)^{1/T} \right\}$ , do:
      •  $g := g'$ 
      •  $r := r + 1$ 
      • if  $r > R$ , then  $r := 0$ ;  $T := T(1 - \delta)$ 

```

---



---

Fig. 3. SW model computation

Brute force search over the (discretized) parameter space is used to find optimal parameters  $\alpha, \beta$ . Of course, the result is only an upper-bound on performance, but, in our case, it's enough to rank the SW model with respect to the others.

## V. EXPERIMENTS

We ran our algorithms on three publicly available snapshots of the AS-level Internet topology, data from 1997, 1999, and 2001 [33]. As suggested by [20], for each year we combined five snapshots taken in a short time span into one representative snapshot. The datasets are summarized in Figure 4. We

Year	Vertices	Edges
1997	3117	6024
1999	6266	13681
2001	11080	25485

Fig. 4. AS-level Internet topology datasets

ranked all models with respect to all datasets, see Figure 5 for the raw results of our algorithms. For each model and graph pair we give the neg-log-likelihood (the bold number) of the

model together with the optimal parameters (the parameters which minimize the rank for this graph).

1997:			
PA	PRG	SW	ER
<b>49,999</b>	<b>51,806</b>	> <b>53,975</b>	<b>72,890</b>
$p = 0.58$	$\beta^{\text{in}} = 1.55$	$\alpha = 0.111$	$p = 6.2\text{e-}4$
$q = 0.08$	$\beta^{\text{out}} = 2.39$	$\beta = 1.9$	
$\gamma = 0.5$	$c^{\text{in}} = 10.13\%$		
	$c^{\text{out}} = 1.15\%$		
1999:			
PA	PRG	SW	ER
<b>116,973</b>	<b>120,803</b>	> <b>133,527</b>	<b>176,895</b>
$p = 0.61$	$\beta^{\text{in}} = 1.57$	$\alpha = 0.092$	$p = 3.5\text{e-}4$
$q = 0.08$	$\beta^{\text{out}} = 2.44$	$\beta = 1.8$	
$\gamma = 0.4$	$c^{\text{in}} = 10.31\%$		
	$c^{\text{out}} = 1.26\%$		
2001:			
PA	PRG	SW	ER
<b>218,661</b>	<b>225,542</b>	> <b>265,554</b>	<b>348,635</b>
$p = 0.63$	$\beta^{\text{in}} = 1.57$	$\alpha = 0.088$	$p = 2.1\text{e-}4$
$q = 0.07$	$\beta^{\text{out}} = 2.5$	$\beta = 1.8$	
$\gamma = 0.3$	$c^{\text{in}} = 9.5\%$		
	$c^{\text{out}} = 0.84\%$		

Fig. 5. Raw results of the four algorithms. Bold numbers represent the neg-log-likelihood (the score) rounded to the nearest integer, the rest are the best parameter values. To make comparison of parameters across the snapshots more meaningful, we state the parameters  $c_{\text{in}}$  and  $c_{\text{out}}$  as a percentage of the final edge count. Note: For SW we only state a lower bound on the score.

To visualize the relative scores, we depict them in a histogram shown in Figure 6. It includes an error estimate (a rather large uncertainty region) for the SW model. We scaled the neg-log-likelihood by the number of edges in the particular snapshot. This scaling has a special meaning: it captures the compression factor of the snapshot by the given model. We will briefly explain this connection.

There is an interesting relationship between the MLE principle and data compression. Consider a model  $M$  (suppose  $M$  includes a parameter setting). The model describes a probability distribution over graphs. Let us restrict ourselves to graphs with non-zero probability. Then, using the principle of arithmetic coding [34], each graph  $G$  can be encoded by a number of bit-length  $\text{neg-log-likelihood}(G, M)$ , from which  $G$  can be recovered uniquely. Thus, in a sense, the neg-log-likelihood computes the compressed size of  $G$  with respect to  $M$ . The number of edges of  $G$  reflects the original (uncompressed) description length of  $G$ . Thus, the ratio of neg-log-likelihood and the number of edges reflects the compression factor of the model on the graph instance.

As can be seen in the histogram, all three data sets produced the same ranking of models. This result is more meaningful than a comparison based on a single dataset alone. In all three cases, the PA model placed first, closely followed by the PRG model. The two models both attempt to reproduce the power-

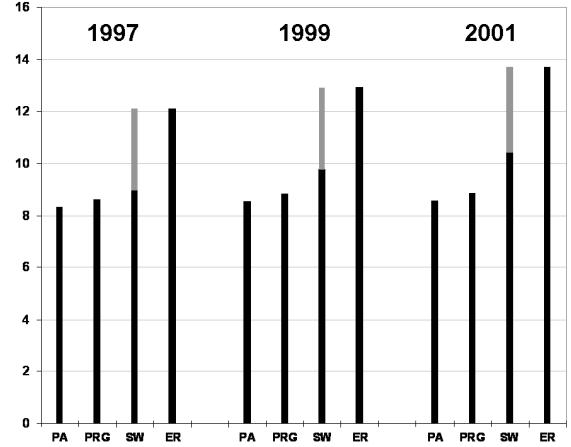


Fig. 6. The scores of the data sets using the four models. The SW model has an uncertainty region, but we can still rank it with respect to all the others. The scores reflect the neg-log-likelihood per edge.

law property. In the PRG, it is imposed directly, while in PA, it arises naturally. Nonetheless, the two scored similarly.

The SW model ranked third. With care, comparisons between models can be interpreted as statements about the relative importance of graph properties. In this case, one could conclude that the power-law degree distribution is a more significant predictor of Internet-topology data than the small-world phenomenon.

It is interesting to compare the compression factors across the datasets and it is intriguing to realize that the histograms for each snapshot look much alike. Together with the virtual invariance of the parameter settings this is further evidence of the scalability of the Internet topology. However, further work needs to be done to quantify this statement.

Clearly, the validity of our rankings depends on the accuracy of the data. Recently, this accuracy has been questioned [35]. As more up-to-date snapshots become available, it will be interesting to see whether our rankings change. Our method could help with measuring the significance of the inaccuracy in the present snapshots.

#### A. Implementation Details

In implementing these algorithms that combine theoretical analysis with various heuristic stopping rules, it is important to perform tests to see that the computations are indeed accurate. In this section we discuss the settings under which we ran our experiments, including optimizations in the code, choice of parameters and software and hardware setup, and validation of the correctness of our implementations.

1) *PA model*: There are several important implementation details to discuss. First of all, in the algorithm we do not recompute  $\Pr(G|\text{Shift}(\pi, v, j))$  from scratch as needed in the algorithm. Instead, such calculations can be quickly computed incrementally from the value  $\Pr(G|\pi)$ . This is essential to the efficiency of the algorithm, as it shaves off a linear factor from the running time. Second, we do not do estimation every step but rather every ten steps. Since consecutive steps are

dependent, the benefit of estimating *every* step is not worth the cost.

Third, we parallelize the algorithm by computing, separately for each  $i$ ,  $\text{Self-iter}(G, i, p, q, \gamma, T)$ . Usually when one runs a self-iterating Markov chain algorithm, the pivot permutation  $\sigma$  is not fixed in advance. Rather, the  $(i + 1)$ -st element is determined as the most likely element on the  $(i + 1)$ -st position in the  $i$ -th self-iteration. Thus, the most likely element is guaranteed to appear on the  $(i + 1)$ -st position with probability at least  $1/(n - i)$ , which in turn implies that the number of steps needed to estimate the  $i$ -th self-iteration is small. However, such approach would require a sequential algorithm which we could not afford for larger datasets. We overcome this problem by choosing a likely pivot permutation  $\sigma$  explicitly.

Fourth, we have to estimate the convergence time. We do this heuristically and conservatively. We ran the Self-iter procedure independently four times for  $i \in \{1, \dots, 10\}$  and a sample of twenty  $i$ 's from  $\{10, \dots, n\}$  for different settings of parameters  $p, q$  and  $\gamma$ . We let the chain run long enough so that all four executions stabilized and the resulting four numbers differed minimally. We determined  $T$ , the number of steps we take, as the time when the estimators of the four runs were within  $(1 \pm \varepsilon)$  factor of each other (for a small  $\varepsilon$ ) at time  $T/4$  and they remained within the range for all remaining  $3T/4$  steps. As expected, fewer steps  $T$  were required for convergence for later (larger  $i$ ) Self-iter's. Thus we used decreasing values of  $T$  for larger  $i$ .

To run the Self-iter steps in parallel we used Condor [36] and the University of Chicago Condor pool consisting of about 70 2-GHz computers, out of which about 35-50 were available at a time. We used the "vanilla" universe, recording the current state of the walk and the estimator every minute. This way we kept extensive log-files about the computation which allowed us to check the precision of our computations by progressively increasing the number of steps of the chain  $T$  without the need for restarting the chain from scratch every time. Our computations for the 1997 data took around 40 hours total, compared with twice as long for the 1999 data and about a week for the 2001 data. However, as we discuss below, we were very conservative in our stopping heuristics.

To ensure that estimates in each phase were converging, for a sampling of the Self-iter's, we ran the walks for significantly longer than our original bound on  $T$  and the values remained within .1%. Note that margin of error of the total estimate is actually lower than the margin of error of any individual Self-iter. This is because the total is the sum of thousands of independent random variables. Hence, as long as each estimate has the correct expectation, the margin of error of the total estimate will be significantly smaller (in percentage).

To verify the correctness of our implementation, we performed two checks. First, we implemented an algorithm which computes the rank exactly by a brute force enumeration of all permutations of vertices. We checked that for small graphs we obtained the same scores (the scores differed less than %.001). For the large graphs we ran the algorithm with presumably

one of the worst pivot permutations – the original  $\sigma$  reversed. We used the same estimate methods for  $T$ , thus we needed more samples for larger values of  $i$  compared to the pivot permutation  $\sigma$ . The scores we obtained were within .1% of the corresponding scores with the original pivot permutation  $\sigma$ . The precision of the two scores points to the fact that we were very conservative when determining the steps bound  $T$ .

To search for optimal parameters, we noticed that  $\Pr(G|\sigma)$  is quite close to  $\Pr(G)$  (their logarithms are within 5% of each other). Since the computation of  $\Pr(G|\sigma, p, q, \gamma)$  is fast, we could find the parameters that maximize this probability and then we searched the parameter space close to these values.

2) *SW model*: In Figure 3 we incorporated a standard cooling schedule which met with great empirical success in optimization. It works by specifying parameters  $\delta$  and  $R$  with the following meaning: The temperature decreases every  $R$  "true" steps by a factor of  $(1 - \delta)$ . A step of the chain is "true" if the state of the chain changes after the step (notice that the Metropolis walk often stays in the same state).

In most real world applications it is recommended to take  $\delta < 0.05$  and  $R$  (at least) linearly proportional to the size of the input. We conservatively chose  $\delta = 0.01$  and  $R = 1,000,000$ , resp.  $R = 2,000,000$ , resp.  $R = 4,000,000$  for the 1997, resp. 1999, resp. 2001 dataset. We chose a random grid alignment of vertices as a starting state of our algorithm. The algorithm converged in a few hours.

Our ranking depends crucially on the ability of simulated annealing to come close to the true maximum  $\Pr(G|g)$ . We performed two tests to validate the SW algorithm. First, we ran several independent runs of the algorithm to check that the outputs converge to the same answer. All the outputs fell in the interval of about 1% of each other. Second, we took the simple grid graphs (of size comparable to those of the actual snapshots) where each interior node has degree four. We permuted the nodes randomly and ran a search for the best configuration, with parameters of  $\delta$  and  $R$  set as above. As seen in Figure 7, the final answer is not perfect but very close to the true answer – the grid itself. The error of the simulated annealing estimate versus  $\Pr(G|\text{grid})$  was less than 1%. For practical purposes, we conclude that the negative log-likelihood assigned by the SW model is between 99% of the raw score of SW (which computes a bound on the SW model's performance) and the score of ER, which can never be better.

For completeness we include the optimal configuration computed by the SW algorithm for all three datasets, see Figure 8.

## VI. VALIDATION

To further validate our methodology, we took all four models and generated a representative graph. Under our paradigm, the model which generated the graph should get the best score.

Figure 9 summarizes the rankings which we pictorially show in the histograms in Figure 10. As before, the score of the SW model falls in a rather large interval. Unlike previously, this time the interval does not imply the precise position of

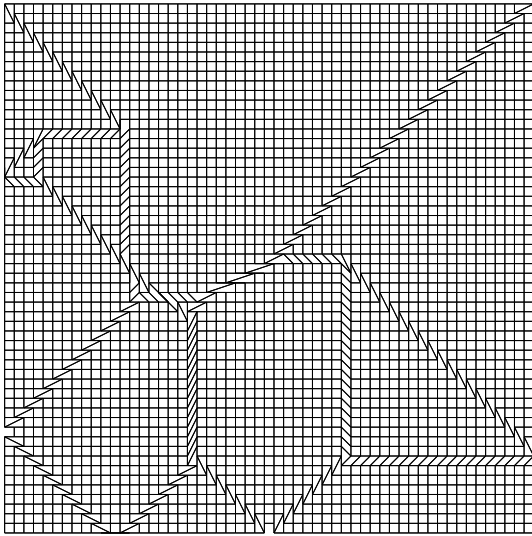


Fig. 7. The results of simulated annealing on a grid graph.

the SW model with respect to the other models. Nevertheless, it behaves consistently with our ranking paradigm.

To make our case stronger, we compared the scores of the SW model and the ER model on many 9-node graphs. We chose the size so that we can compute the SW score exactly by evaluating all  $n!$  grid configurations. We generated the graphs by our models, as well as by hand to include a wide selection of graphs. For all graphs the ER score was only 5% worse than the SW score while the lower bound of our algorithm was often off by more than 30%. Thus our bound is very pessimistic and this experiment gives an evidence for the conjecture that the actual score is just below the ER score.

As a side note, we chose the parameters for the graphs rather arbitrarily, as the specific choice is not important. We did not make any effort to make the graphs connected as this would only skew the original distribution defined by the model. Again, the connectivity property is irrelevant for the validation.

## VII. CONCLUSIONS AND FUTURE WORK

We have proposed a uniform methodology for ranking probabilistic graph models, which compares models based on the negative log-likelihood of the real-world graphs with respect to each model. As a proof of concept, we have designed and implemented algorithms for computing the negative log-likelihood for four natural Internet topology models from various parts of the modeling spectrum.

We designed algorithms based on a variety of techniques, ranging from simple to complex, including an MCMC-based algorithm and Simulated Annealing, for estimating the probability assigned to a graph. While the algorithms we use are not always simple, we have demonstrated the feasibility of applying the approach even to models which were not designed with this metric in mind.

The results of our experiments ranked the PA model first, closely followed by the PRG model. Their close performance

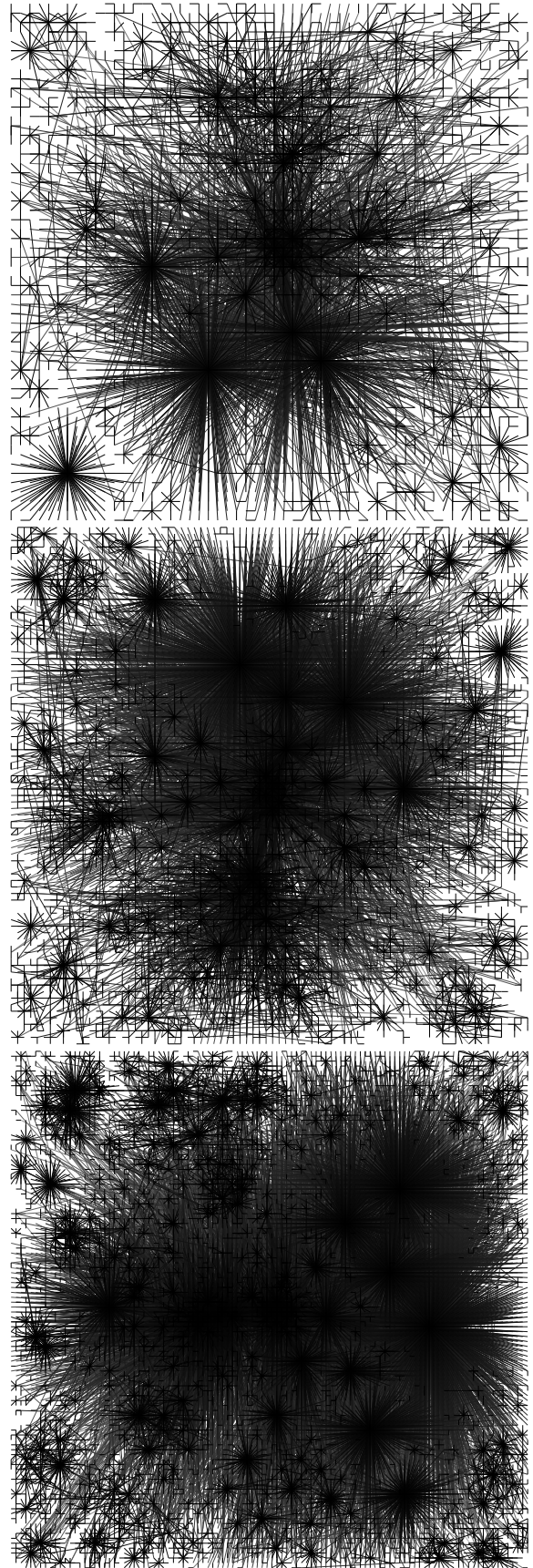


Fig. 8. The result of the simulated annealing on the 1997, 1999, and 2001 snapshot, respectively.

### PA graph, generated with parameters

$p = 0.3, q = 0.4$  and  $\gamma = 0.1$ :

PA	PRG	SW	ER
<b>1,299</b>	<b>1,337</b>	<b>&gt;1,211</b>	<b>1,565</b>
$p = 0.3$	$\beta^{\text{in}} = 1.4$	$\alpha = 0.3$	$p = 0.023$
$q = 0.4$	$\beta^{\text{out}} = 1.5$	$\beta = 2.1$	
$\gamma = 0.1$	$c^{\text{in}} = 10.1\%$		
	$c^{\text{out}} = 6.6\%$		

### PRG graph, generated with parameters

$\beta^{\text{in}} = \beta^{\text{out}} = 1.5$  and  $c^{\text{in}} = c^{\text{out}} = 30$ :

PA	PRG	SW	ER
<b>1,480</b>	<b>1,449</b>	<b>&gt;1,229</b>	<b>1,538</b>
$p = 0.2$	$\beta^{\text{in}} = 1.5$	$\alpha = 0.27$	$p = 0.023$
$q = 0.6$	$\beta^{\text{out}} = 1.5$	$\beta = 1.5$	
$\gamma = 0.1$	$c^{\text{in}} = 8.5\%$		
	$c^{\text{out}} = 8.8\%$		

### SW graph generated with parameters

$s = 11, \alpha = 0.1$  and  $\beta = 1.8$ :

PA	PRG	SW	ER
<b>988</b>	<b>1,014</b>	<b>&gt;538</b>	<b>956</b>
$p = 0.2$	$\beta^{\text{in}} = 0.8$	$\alpha = 0.21$	$p = 0.011$
$q = 0.3$	$\beta^{\text{out}} = 0.8$	$\beta = 3.5$	
$\gamma = 10,000$	$c^{\text{in}} = 3.3\%$		
	$c^{\text{out}} = 3.3\%$		

### ER graph generated with parameter $p = 0.1$ :

PA	PRG	SW	ER
<b>4,850</b>	<b>4,884</b>	<b>&gt;4,280</b>	<b>4,649</b>
$p = 0.5$	$\beta^{\text{in}} = 0$	$\alpha = 0.35$	$p = 0.1$
$q = 0.4$	$\beta^{\text{out}} = 0$	$\beta = 0.8$	
$\gamma = 10,000$	$c^{\text{in}} = 1.7\%$		
	$c^{\text{out}} = 1.7\%$		

Fig. 9. Validation: Raw scores of the four algorithms on 100-node graphs generated by the models. As before, bold numbers represent the neg-log-likelihood (the score) rounded to the nearest integer, the rest are the best parameter values.

is interesting because the two take very different approaches to modeling power-law distributed data. The first is a explanatory model, while the second offers no explanation. The SW model ranked third. This may be interpreted as a statement that the power-law degree distribution is a more significant predictor of Internet-topology data than the small-world phenomenon. As expected, the ER model and LZW placed last.

Our ranking of models is not entirely surprising, given the impact of the discovery that the Internet topology obeys a power-law degree distribution [4]. Our contribution is that we have provided a way to both *verify* and *quantify* beliefs in the networking community about the relative strengths of different kinds of models. Our finding that the exponents of the best power-law models change very little with time complements the observation of [4] that the exponents of the power-law distributions that best fit the Internet topology change very little with time.

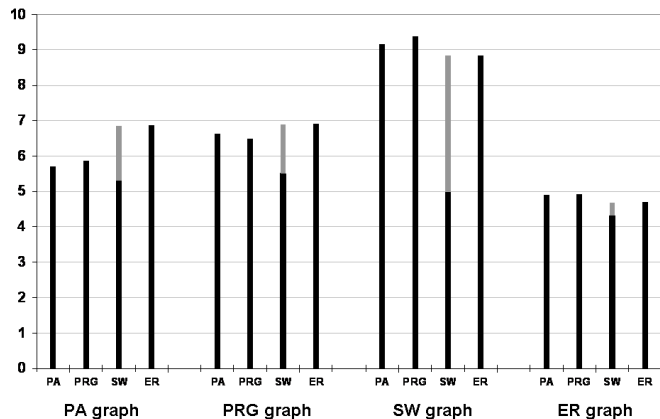


Fig. 10. Validation: The scores of the models on graphs generated by these models, scaled per edge.

For future work, one would like to design better scoring models on our data sets and others. We have additional algorithmic ideas based on recent fast MCMC techniques [37] which would enable the computation of scores for a wide range of models. This would make it easy to design and measure improved models.

Another future direction of our research is applying our methodology to HOT models. Although, as mentioned before, our methodology can't say much about the explanatory power of such models, it is still interesting whether such models excel power-law models in terms of predictive power.

Another possible direction is to extend our methodology to other contexts. It would be particularly interesting to implement our methodology for Web graph models, since again there is an abundance of such models and a lack of clear and objective criteria to distinguish between them.

All of our code is publicly available for download.

### ACKNOWLEDGMENT

The authors would like to thank Milena Mihail, Matei Ripeanu and Ian Foster for helpful feedback and encouragement throughout this work.

### REFERENCES

- [1] C. Labovitz, A. Ahuja, R. Wattenhofer, and S. Venkataschary, "The impact of internet policy and topology on delayed routing convergence," in *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM-01)*, 2001, pp. 537–546.
- [2] B. M. Waxman, "Routing of multipoint connections," *IEEE Jour. Selected Areas in Communications (Special Issue: Broadband Packet Communications)*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.
- [3] E. Zegura, K. Calvert, and M. Donahoo, "A quantitative comparison of graph-based models for internet topology," *Transactions on Networking*, pp. 770–783, 1997.
- [4] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos, "Power laws and the AS-level internet topology," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 514–524, 2003.

- [5] L. Adamic and B. Huberman, "Scaling behavior of the world wide web," *Science*, vol. 287, p. 2115, 2000.
- [6] A. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, 1999.
- [7] W. Aiello, F. Chung, and L. Lu, "A random graph model for massive graphs," in *Proceedings of ACM Symposium on Theory of Computing*, 2000.
- [8] —, "Random evolution in massive graphs," in *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS-01)*, 2001, pp. 510–521.
- [9] A. Medina, I. Matta, and J. Byers, "On the origin of power laws in internet topologies," *Computer Communications Review*, vol. 30, no. 2, pp. 18–28, Apr. 2000.
- [10] T. Bu and D. F. Towsley, "On distinguishing between internet power law topology generators," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-02)*, 2002, pp. 638–647.
- [11] A. Fabrikant, E. Koutsoupias, and C. Papadimitriou, "Heuristically optimized trade-offs: A new paradigm for power laws in the internet," *Lecture Notes in Computer Science*, vol. 2380, pp. 110–??, 2002.
- [12] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal, "Stochastic models for the web graph," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, 2000, pp. 57–65.
- [13] C. Cooper and A. Frieze, "A general model of web graphs," *Random Structures & Algorithms*, vol. 22, 2003.
- [14] C. Jin, Q. Chen, and S. Jamin, "Inet: Internet topology generator," Tech. Rep., Oct. 05 2000. [Online]. Available: <http://citeseer.ist.psu.edu/490530.html>; <http://www.eecs.umich.edu/techreports/cse/2000/CSE-TR-433-00.pdf>
- [15] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: Universal topology generation from a user's perspective," Tech. Rep., Apr. 26 2001. [Online]. Available: <http://citeseer.ist.psu.edu/443553.html>; <http://www.cs.bu.edu/techreports/pdf/2001-003-brite-user-manual.pdf>
- [16] M. E. J. Newman, "Models of the small world - A review," 2001.
- [17] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.
- [18] R. Fisher, "On the mathematical foundations of theoretical statistics," *Philosophical Transactions of the Royal Society of London, Series A*, vol. 222, pp. 309–368, 1922.
- [19] A. Edwards, *Likelihood*. Cambridge University Press, 1974.
- [20] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 733–745, Dec. 2001.
- [21] H. Tangmunarunkit, R. Govindan, S. Shenker, S. Jamin, and W. Willinger, "Network topology generators: Degree-based vs. structural," in *Proceedings of ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2002, pp. 147–160.
- [22] W. Willinger, R. Govindan, S. Jamin, V. Paxson, and S. Shenker, "Scaling phenomena in the internet: Critically examining criticality," *Proceedings of the National Academy of Science, USA*, vol. 99, pp. 2573–2580, 2002.
- [23] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "The origin of power-laws in internet topologies revisited," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Society*, 2002, pp. 608–617.
- [24] J. M. Carlson and J. Doyle, "Highly optimized tolerance: a mechanism for power laws in designed systems," *Physical Review E*, vol. 60, no. 2, pp. 1412–1427, 1999.
- [25] D. Alderson, J. Doyle, R. Govindan, and W. Willinger, "Toward an optimization-driven framework for designing and generating realistic internet topologies," 2002. [Online]. Available: <http://citeseer.ist.psu.edu/546510.html>; <http://www.cs.washington.edu/hotnets/papers/alderson.pdf>
- [26] L. Li, D. Alderson, W. Willinger, and J. Doyle, "A first-principles approach to understanding the internet's router-level topology," in *Proceedings of ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2004.
- [27] M. Mitzenmacher, "A brief history of generative models for power law and log normal distributions," in *Proceedings of the 39th Annual Allerton Conference on Communication, Control, and Computing*, 2001, pp. 182–191.
- [28] B. Bollobás, *Random graphs*. Academic Press, 1985.
- [29] A. L. Barabási, R. Albert, and H. Jeong, "Scale-free characteristics of random networks: topology of the world-wide web," *Physica A*, vol. 281, p. 69, 2000.
- [30] J. Kleinberg, "The small-world phenomenon: an algorithm perspective," in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 2000, pp. 163–170.
- [31] S. Kirkpatrick, C.D.Gellatt, and M. Vecchi, "Simulated annealing," *Science*, vol. 220, no. 671, 1983.
- [32] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 2nd ed. Cambridge (UK) and New York: Cambridge University Press, 1992.
- [33] NLANR, "National Laboratory for Applied Network Research Routing data," <http://moat.nlanr.net/Routing/rawdata/>, 2001.
- [34] I. Witten, R. Neal, and J. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, pp. 520–540, 1987.
- [35] Z. Mao, J. Rexford, J. Wang, and R. Katz, "Towards an accurate AS-level traceroute tool," in *Proceedings of ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2003.
- [36] CONDOR, "University of Wisconsin at Madison," <http://www.cs.wisc.edu/condor/>, 2005.
- [37] L. Lovász and S. Vempala, "Simulated annealing in convex bodies and an  $O^*(n^4)$  volume algorithm," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computing*, 2003, pp. 650–.