Hierarchical structure and the prediction of missing links in networks

Aaron Clauset,[1,2] Cristopher Moore,[1,2,3] and M. E. J. Newman[2,4]

[1]*Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, USA*
[2]*Santa Fe Institute, 1399 Hyde Park Road, Santa Fe NM, 87501, USA*
[3]*Department of Physics and Astronomy, University of New Mexico, Albuquerque, NM 87131, USA*
[4]*Department of Physics and Center for the Study of Complex Systems, University of Michigan, Ann Arbor, MI 48109, USA*

Networks have in recent years emerged as an invaluable tool for describing and quantifying complex systems in many branches of science. Recent studies suggest that networks often exhibit hierarchical organization, where vertices divide into groups that further subdivide into groups of groups, and so forth over multiple scales. In many cases these groups are found to correspond to known functional units, such as ecological niches in food webs, modules in biochemical networks (protein interaction networks, metabolic networks, or genetic regulatory networks), or communities in social networks. Here we present a general technique for inferring hierarchical structure from network data and demonstrate that the existence of hierarchy can simultaneously explain and quantitatively reproduce many commonly observed topological properties of networks, such as right-skewed degree distributions, high clustering coefficients, and short path lengths. We further show that knowledge of hierarchical structure can be used to predict missing connections in partially known networks with high accuracy, and for more general network structures than competing techniques. Taken together, our results suggest that hierarchy is a central organizing principle of complex networks, capable of offering insight into many network phenomena.

Supplementary Information

Appendix A: Hierarchical random graphs

Our model for the hierarchical organization of a network is as follows.[1] Let G be a graph with n vertices. A *dendrogram* D is a binary tree with n leaves corresponding to the vertices of G. Each of the $n-1$ internal nodes of D corresponds to the group of vertices that are descended from it. We associate a probability p_r with each internal node r. Then, given two vertices i, j of G, the probability p_{ij} that they are connected by an edge is $p_{ij} = p_r$ where r is their lowest common ancestor in D. The combination $(D, \{p_r\})$ of the dendrogram and the set of probabilities then defines a *hierarchical random graph*.

Note that if a community has, say, three subcommunities, with an equal probability p of connections between them, we can represent this in our model by first splitting one of these subcommunities off, and then splitting the other two. The two internal nodes corresponding to these splits would be given the same probabilities $p_r = p$. This yields three possible binary dendrograms, which are all considered equally likely.

We can think of the hierarchical random graph as a variation on the classical Erdős–Rényi random graph $G(n, p)$. As in that model, the presence or absence of an edge between any pair of vertices is independent of the presence or absence of any other edge. However, whereas in $G(n, p)$ every pair of vertices has the same probability p of being connected, in the hierarchical random graph the probabilities are inhomogeneous, with the inhomogeneities controlled by the topological structure of the dendrogram D and the parameters $\{p_r\}$. Many other models with inhomogeneous edge probabilities have, of course, been studied in the past. One example is a structured random graph in which there are a finite number of types of vertices with a matrix p_{kl} giving the connection probabilities between them.[2]

Appendix B: Fitting the hierarchical random graph to data

Now we turn to the question of finding the hierarchical random graph or graphs that best fits the observed real-world network G. Assuming that all hierarchical random graphs are *a priori* equally likely, the probability that a given model $(D, \{p_r\})$ is the correct explanation of the data is, by Bayes' theorem, proportional to the posterior probability or *likelihood* $\mathcal{L}$ with which that model generates the observed network.[3] Our goal is to maximize $\mathcal{L}$ or, more generally, to sample the space of all models with probability proportional to $\mathcal{L}$.

Let E_r be the number of edges in G whose endpoints have r as their lowest common ancestor in D, and let L_r and R_r, respectively, be the numbers of leaves in the left and right subtrees rooted at r. Then the likelihood of the hierarchical random graph is

$$\mathcal{L}(D, \{p_r\}) = \prod_{r \in D} p_r^{E_r} \left(1 - p_r\right)^{L_r R_r - E_r} \qquad \text{(B1)}$$

with the convention that $0^0 = 1$.

If we fix the dendrogram D, it is easy to find the probabilities $\{\overline{p}_r\}$ that maximize $\mathcal{L}(D, \{p_r\})$. For each r, they are

[1] Computer code implementing many of the analysis methods described in this paper can be found online at `www.santafe.edu/~aaronc/randomgraphs/`.

[2] F. McSherry, "Spectral Partitioning of Random Graphs," *Proc. Foundations of Computer Science (FOCS)*, pp. 529–537 (2001)

[3] G. Casella and R. L. Berger, "Statistical Inference." Duxbury Press, Belmont (2001).

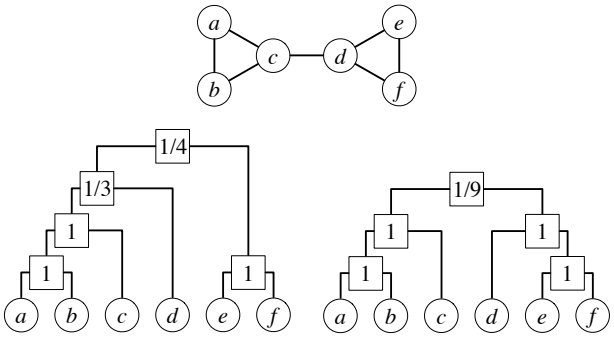

FIG. 1: An example network G consisting of six vertices, and the likelihood of two possible dendrograms. The internal nodes r of each dendrogram are labeled with the maximum-likelihood probability $\overline{p}_r$, i.e., the fraction of potential edges between their left and right subtrees that exist in G. According to Eq. (B3), the likelihoods of the two dendrograms are $\mathcal{L}(D_1) = (1/3)(2/3)^2 \cdot (1/4)^2(3/4)^6 = 0.00165\ldots$ and $\mathcal{L}(D_2) = (1/9)(8/9)^8 = 0.0433\ldots$ The second dendrogram is far more likely because it correctly divides the network into two highly-connected subgraphs at the first level.

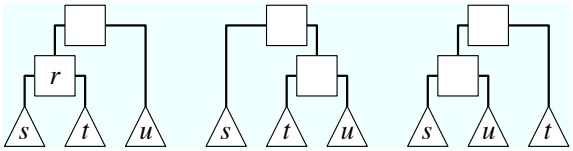

FIG. 2: Each internal node r of the dendrogram has three associated subtrees s, t, and u, which can be placed in any of three configurations. (Note that the topology of the dendrogram depends only on the sibling and parent relationships; the order, left to right, in which they are depicted is irrelevant).

given by

$$\overline{p}_r = \frac{E_r}{L_r R_r}, \tag{B2}$$

the fraction of potential edges between the two subtrees of r that actually appear in the graph G. The likelihood of the dendrogram evaluated at this maximum is then

$$\mathcal{L}(D) = \prod_{r \in D} \left[\overline{p}_r^{\,\overline{p}_r} \left(1 - \overline{p}_r\right)^{1-\overline{p}_r} \right]^{L_r R_r} . \tag{B3}$$

Figure 1 shows an illustrative example, consisting of a network with six vertices.

It is often convenient to work with the logarithm of the likelihood,

$$\log \mathcal{L}(D) = -\sum_{r \in D} L_r R_r h(\overline{p}_r), \tag{B4}$$

where $h(p) = -p \log p - (1-p)\log(1-p)$ is the Gibbs-Shannon entropy function. Note that each term $-L_r R_r h(\overline{p}_r)$ is maximized when $\overline{p}_r$ is close to 0 or to 1, i.e., when the entropy is minimized. In other words, high-likelihood dendrograms are those that partition the vertices into groups between which connections are either very common or very rare.

We now use a Markov chain Monte Carlo method to sample dendrograms D with probability proportional to their likelihood $\mathcal{L}(D)$. To create the Markov chain we need to pick a set of transitions between possible dendrograms. The transitions we use consist of rearrangements of subtrees of the dendrogram as follows. First, note that each internal node r of a dendrogram D is associated with three subtrees: the subtrees s, t descended from its two daughters, and the subtree u descended from its sibling. As Figure 2 shows, there are two ways we can reorder these subtrees without disturbing any of their internal relationships. Each step of our Markov chain consists first of choosing an internal node r uniformly at random (other than the root) and then choosing uniformly at random between the two alternate configurations of the subtrees associated with that node and adopting that configuration. The result is a new dendrogram D'. It is straightforward to show that transitions of this type are *ergodic*, i.e., that any pair of finite dendrograms can be connected by a finite series of such transitions.

Once we have generated our new dendrogram D' we accept or reject that dendrogram according to the standard Metropolis–Hastings rule.[4] Specifically, we accept the transition $D \to D'$ if $\Delta \log \mathcal{L} = \log \mathcal{L}(D') - \log \mathcal{L}(D)$ is nonnegative, so that D' is at least as likely as D; otherwise we accept the transition with probability $\exp(\log \Delta\mathcal{L}) = \mathcal{L}(D')/\mathcal{L}(D)$. If the transition is not accepted, the dendrogram remains the same on this step of the chain. The Metropolis-Hastings rule ensures detailed balance and, in combination with the ergodicity of the transitions, guarantees a limiting probability distribution over dendrograms that is proportional to the likelihood, $P(D) \propto \mathcal{L}(D)$. The quantity $\Delta \log \mathcal{L}$ can be calculated easily, since the only terms in Eq. (B4) that change from D to D' are those involving the subtrees s, t, and u associated with the chosen node.

The Markov chain appears to converge relatively quickly, with the likelihood reaching a plateau after roughly $O(n^2)$ steps. This is not a rigorous performance guarantee, however, and indeed there are mathematical results for similar Markov chains that suggest that equilibration could take exponential time in the worst case.[5] Still, as our results here show, the method seems to work quite well in practice. The algorithm is able to handle networks with up to a few thousand vertices in a reasonable amount of computer time.

We find that there are typically many dendrograms with roughly equal likelihoods, which reinforces our contention that it is important to sample the distribution of dendrograms rather than merely focusing on the most likely one.

[4] M. E. J. Newman and G. T. Barkema. (Clarendon Press, Oxford, 1999).
[5] E. Mossel and E. Vigoda, "Phylogenetic MCMC Are Misleading on Mixtures of Trees." *Science* **309**, 2207 (2005)

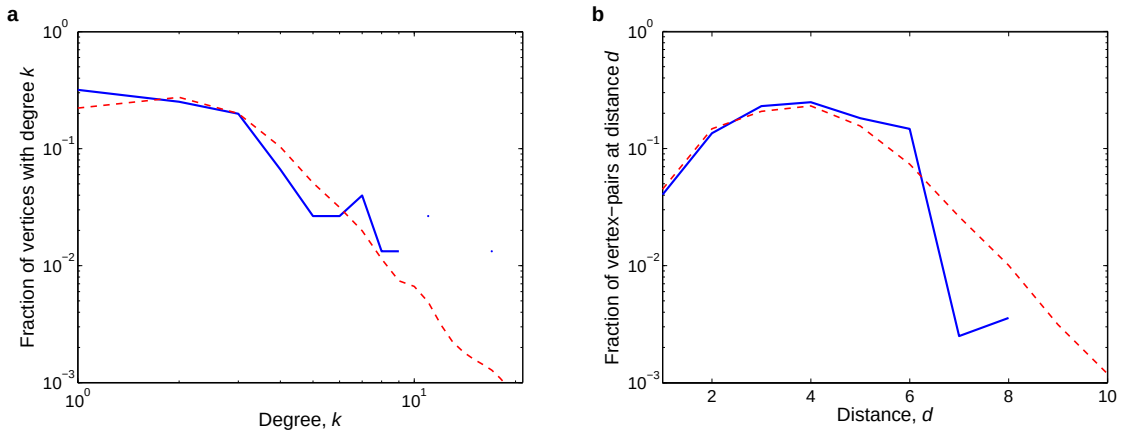

FIG. 3: Application of our hierarchical decomposition to the network of grassland species interactions. **a**, Original (blue) and resampled (red) degree distributions. **b**, Original and resampled distributions of vertex-vertex distances.

Appendix C: Resampling from the hierarchical random graph

The procedure for resampling from the hierarchical random graph is as follows.

1. Initialize the Markov chain by choosing a random starting dendrogram.
2. Run the Monte Carlo algorithm until equilibrium is reached.
3. Sample dendrograms at regular intervals thereafter from those generated by the Markov chain.
4. For each sampled dendrogram D, create a resampled graph G' with n vertices by placing an edge between each of the $n(n-1)/2$ vertex pairs (i,j) with independent probability $p_{ij} = \overline{p}_r$, where r is the lowest common ancestor of i and j in D and $\overline{p}_r$ is given by Eq. (B2). (In principle, there is nothing to prevent us from generating many resampled graphs from a dendrogram, but in the calculations described in this paper we generate only one from each dendrogram.)

After generating many samples in this way, we can compute averages of network statistics such as the degree distribution, the clustering coefficient, the vertex-vertex distance distribution, and so forth. Thus, in a way similar to Bayesian model averaging,[6] we can estimate the distribution of network statistics defined by the equilibrium ensemble of dendrograms.

For the construction of consensus dendrograms such as the one shown in Fig. 2a, we found it useful to weight the most likely dendrograms more heavily, giving them weight proportional to the square of their likelihood, in order to extract a coherent consensus structure from the equilibrium set of models.

Appendix D: Predicting missing connections

Our algorithm for using hierarchical random graphs to predict missing connections is as follows.

1. Initialize the Markov chain by choosing a random starting dendrogram.
2. Run the Monte Carlo algorithm until equilibrium is reached.
3. Sample dendrograms at regular intervals thereafter from those generated by the Markov chain.
4. For each pair of vertices i,j for which there is not already a known connection, calculate the mean probability $\langle p_{ij} \rangle$ that they are connected by averaging over the corresponding probabilities p_{ij} in each of the sampled dendrograms D.
5. Sort these pairs i,j in decreasing order of $\langle p_{ij} \rangle$ and predict that the highest-ranked ones have missing connections.

In general, we find that the top 1% of such predictions are highly accurate. However, for large networks, even the top 1% can be an unreasonably large number of candidates to check experimentally. In many contexts, researchers may want to consider using the procedure interactively, i.e., predicting a small number of missing connections, checking them experimentally, adding the results to the network, and running the algorithm again to predict additional connections.

The alternative prediction methods we compared against, which were previously investigated in[7], consist of giving each pair i,j of vertices a score, sorting pairs in decreasing order of

[6] T. Hastie, R. Tibshirani and J. Friedman, "The Elements of Statistical Learning." Springer, New York (2001).

[7] D. Liben-Nowell and J. Kleinberg. *Proc. Internat. Conf. on Info. and Know. Manage.* (2003).

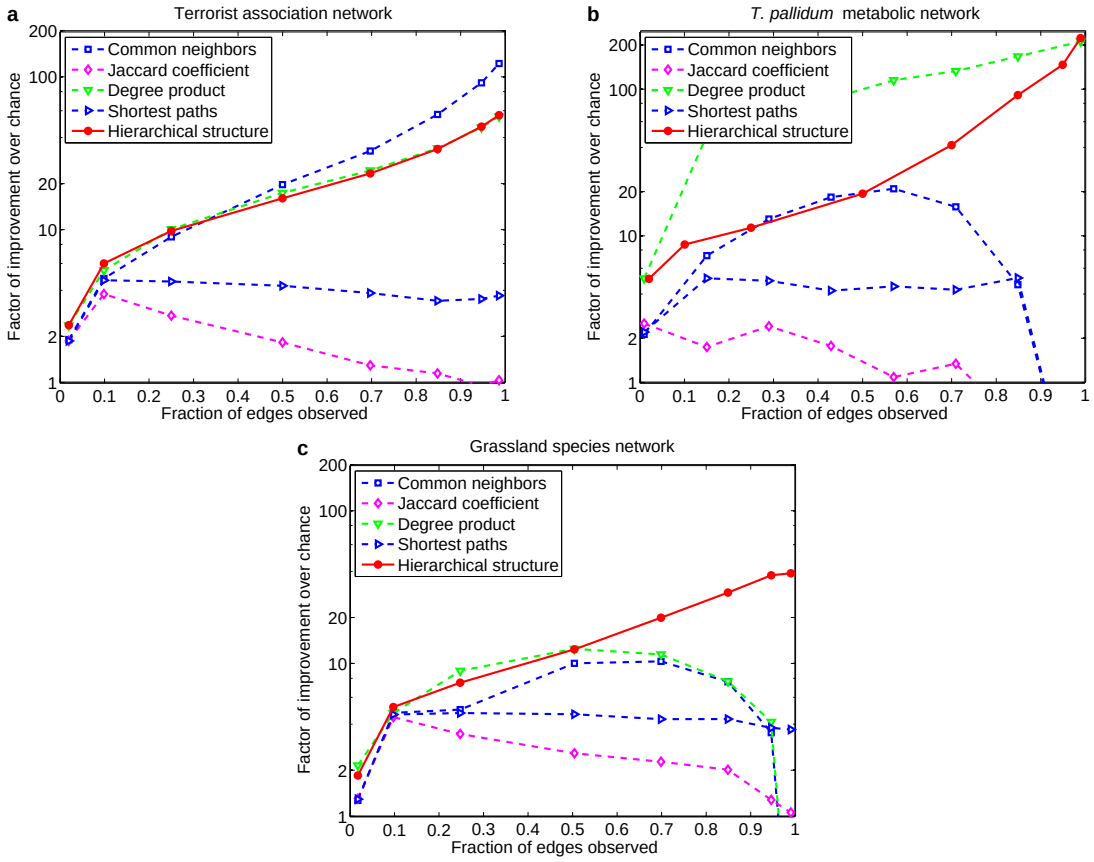

FIG. 4: Further comparison of link prediction algorithms. Data points represent the average ratio between the probability that the top-ranked pair of vertices is in fact connected and the corresponding probability for a randomly-chosen pair, as a function of the fraction of the connections known to the algorithm. For each network, (**a**, Terrorist associations; **b**, *T. pallidum* metabolites; and **c**, Grassland species interactions), we compare our method with simpler methods such as guessing that two vertices are connected if they share common neighbors, have a high degree product, or have a short path between them.

their score, and predicting that those with the highest scores are the most likely to be connected. Several different types of scores were investigated, defined as follows, where $\Gamma(j)$ is the set of vertices connected to j.

1. Common neighbors: $\text{score}(i,j) = |\Gamma(i) \cap \Gamma(j)|$, the number of common neighbors of vertices i and j.

2. Jaccard coefficient: $\text{score}(i,j) = |\Gamma(i) \cap \Gamma(j)| \,/\, |\Gamma(i) \cup \Gamma(j)|$, the fraction of all neighbors of i and j that are neighbors of both.

3. Degree product: $\text{score}(i,j) = |\Gamma(i)|\,|\Gamma(j)|$, the product of the degrees of i and j.

4. Short paths: $\text{score}(i,j)$ is 1 divided by the length of the shortest path through the network from i to j (or zero for vertex pairs that are not connected by any path).

One way to quantify the success of a prediction method, used by previous authors who have studied link prediction problems[7], is the ratio between the probability that the top-ranked pair is connected and the probability that a randomly chosen pair of vertices, which do not have an observed connection between them, are connected. Figure 4 shows the average value of this ratio as a function of the percentage of the network shown to the algorithm, for each of our three networks. Even when fully 50% of the network is missing, our method predicts missing connections about ten times better than chance for all three networks. In practical terms, this means that the amount of work required of the experimenter to discover a new connection is reduced by a factor of 10, an enormous improvement by any standard. If a greater fraction of the network is known, the accuracy becomes even greater, rising as high as 200 times better than chance when only a few connections are missing.

We note, however, that using this ratio to judge prediction algorithms has an important disadvantage. Some missing connections are much easier to predict than others: for instance, if a network has a heavy-tailed degree distribution and we remove a randomly chosen subset of the edges, the chances are excellent that two high-degree vertices will have a missing connection and such a connection can be easily predicted by simple heuristics such as those discussed above. The AUC

statistic used in the text, by contrast, looks at an algorithm's overall ability to rank all the missing connections over nonexistent ones, not just those that are easiest to predict.

Finally, we have investigated the performance of each of the prediction algorithms on purely random (i.e., Erdős–Rényi) graphs. As expected, no method performs better than chance in this case, since the connections are completely independent random events and there is no structure to discover. We also tested each algorithm on a graph with a power-law degree distribution generated according to the configuration model.[8] In this case, guessing that high-degree vertices are likely to be connected performs quite well, whereas the method based on the hierarchical random graph performs poorly since these graphs have no hierarchical structure to discover.

[8] M. Molloy and B. Reed, "A critical point for random graphs with a given degree sequence", *Random Structures and Algorithms* **6**, 161–179 (1995)