A physical model for efficient ranking in networks

Caterina De Bacco,1,∗ Daniel B. Larremore,2,3,1,† and Cristopher Moore1,‡

1Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA
2Department of Computer Science, University of Colorado, Boulder, CO 80309, USA
3BioFrontiers Institute, University of Colorado, Boulder, CO 80303, USA

We present a physically-inspired model and efficient algorithm to infer hierarchical rankings of nodes in directed networks. Unlike other methods such as minimum violation ranking, it assigns real-valued ranks to nodes rather than simply ordinal ranks, and it formalizes the assumption that interactions are more likely to occur between individuals with similar ranks. It provides a natural framework for a statistical significance test for distinguishing when the inferred hierarchy is due to the network topology or is instead due to random chance, and it can be used to perform inference tasks such as predicting the existence or direction of edges. The ranking is inferred by solving a linear system of equations, which is sparse if the network is; thus the resulting algorithm is extremely efficient and scalable. We illustrate these findings by analyzing real and synthetic data, including datasets from animal behavior, faculty hiring, and social support networks, and show that our method outperforms others, in both speed and accuracy, in recovering the underlying ranks and predicting edge directions.

Introduction

In systems of many individual entities, interactions and their outcomes are often correlated with these entities’ ranks or positions in a hierarchy. While in most cases these rankings are hidden from us, their presence is nevertheless revealed in the asymmetric patterns of interactions that we observe. For example, social groups of birds, primates, and elephants are organized according to dominance hierarchies in which more powerful animals assert themselves over less powerful ones [1]. Social positions are not directly visible to researchers, but we can infer each animal’s position in the hierarchy by observing a sufficient number of interactions. Similar latent hierarchies exist in systems of endorsement in which status is due to prestige or reputation. For example, in academia, universities are more likely to hire faculty candidates from equally or more prestigious universities [2].

In all these cases, the direction of the interactions is affected by the prestige, desirability, or social position of the entities involved. But it is often the case that even the existence of an interaction, rather than its direction, contains some information about those entities’ relative prestige. For example, in some species, animals are more likely to interact with others who are close in dominance rank [3–7], and human beings tend to claim friendships with others of similar or slightly higher status [8]. This suggests that we can infer the ranks of individuals in a social hierarchy from both the existence and the direction of their pairwise interactions. It also suggests a need to assign real-valued ranks to entities rather than simply ordinal rankings, for instance in order to infer clusters of roughly equal entities with gaps between them.

In this work we introduce a physical model that addresses the problems of hierarchy inference, edge prediction, and significance testing. The model, which we call SpringRank, maps each directed edge to a directed spring between the nodes that it connects, and finds real-valued positions of the nodes that minimizes the total energy of these springs. Because this optimization problem requires only linear algebra, it can be solved for networks of millions of nodes and edges in seconds.

We also introduce a generative model for hierarchical networks in which the existence and direction of edges depend on the relative ranks of the nodes. This model formalizes the assumption that individuals tend to interact with others of similar rank, and it can be used to create synthetic benchmark networks with tunable levels of hierarchy and noise. It can also predict unobserved edges, allowing us to use cross-validation as a test of accuracy and statistical significance. Moreover, the maximum likelihood estimates of the ranks coincides with SpringRank asymptotically.

SpringRank and its generative model version can be used to infer hierarchy from data, predict the existence and direction of as-yet unobserved edges, and test hierarchies for statistical significance. In particular, we find that SpringRank predicts the direction of unobserved edges more accurately than existing methods.

Related work

Ranking entities in a system from pairwise comparisons is a fundamental problem in many contexts, and a number of methods have been proposed. One family consists of spectral methods like eigenvector centrality [9] and PageRank [10]. However, by design these methods tend to give high ranks to a small number of nodes, giving us little information about the lower-ranked nodes.

A second family focuses on ordinal rankings—permutations that minimize various penalty functions—
such as Minimum Violation Rank [11–13]. These do not make probabilistic predictions about missing data, so we cannot use cross-validation as a significance test. Moreover, for common choices of the penalty function, minimization can be computationally difficult [12, 14], forcing us to use simple heuristics that find local minima. In addition, as discussed above, we are interested in real-valued ranks rather than just ordinal ones.

Random Utility Models [15], such as the Bradley-Terry-Luce (BTL) model [16, 17], are designed to infer real-valued ranks from data on pairwise preferences. These models assign a probability to the direction of an edge conditioned on its existence, but they do not assign a probability to the existence of an edge. They are appropriate, for instance, when an experimenter presents subjects with choices between pairs of items, and asks them which they prefer. Our model focuses on endogenous interactions in which entities are more likely to interact if their ranks are close.

Methods like David’s Score [18] and the Colley matrix [19] compute rankings from proportions of wins and losses. The latter, which was originally developed by making mathematical adjustments to winning percentages, is equivalent to a particular case of the general method we introduce below. Elo score [20], Go Rank [21], and TrueSkill [22] are also widely used win-loss methods, but these schemes update the ranks after each match rather than taking all previous interactions into account. This specialization makes them useful when ranks evolve over sequential matches, but less useful otherwise.

Finally, there are fully generative models such the Probabilistic Niche Model of ecology [23–25], models of friendship based on social status [8], and more generally latent space models [26] which assign probabilities to the existence and direction of edges based on real-valued positions in social space. However, inference of these models tends to be difficult, with many local optima. Our generative model can be viewed as a special case of these models for which inference is especially easy.

**The SpringRank model**

We represent interactions between entities as a weighted directed network, where is the number of interactions suggesting that is ranked above ; for instance, the number of fights between and that has won, or the number of times that has endorsed . Given the adjacency matrix , our goal is to find a ranking of the nodes. To do so, the SpringRank model computes the optimal location of nodes in a hierarchy by imagining the network as a physical system. Specifically, each node is embedded at a real-valued position or rank , and each directed edge becomes an oriented spring with a nonzero resting length and displacement . Since we are free to rescale the latent space and the energy scale, we set the spring constant and the resting length to 1. Thus, the spring corresponding to a link has energy 

$$\frac{1}{2}(s_i - s_j - 1)^2$$,

which is minimized when .

This version of the model has no tunable parameters. In principle, we could allow different links to have different rest lengths or spring constants, based on the strength or confidence of each link. However, this would create a large number of parameters, which we would have to either infer from the data or choose a priori. We do not explore this here.

According to this model, the optimal rankings of the nodes are the ranks which minimize the total energy of the system given by the Hamiltonian

$$H(s) = \frac{1}{2} \sum_{i,j=1}^{N} A_{ij} (s_i - s_j - 1)^2,$$

(1)

Since this Hamiltonian is convex in , we can find by setting , yielding the linear system

$$[D^\text{out} + D^\text{in} - (A + AT^T)] s^* = [D^\text{out} - D^\text{in}] \mathbf{1}$$

(2)

where is the all-ones vector and are diagonal matrices whose entries are the weighted in- and out-degrees, . See SI Text S1 for detailed derivations.

The matrix on the left side of (2) is not invertible. This is because is translation-invariant: it depends only on the relative ranks , so that if minimizes then so does for any constant . One way to break this symmetry is to invert the matrix in the subspace orthogonal to its nullspace. If the network is connected, this is the one-dimensional subspace spanned by the eigenvector , in which case this method finds the where the average rank is minimized. Alternately, we can simply fix the rank of one of the nodes.

Another way to break translation invariance is to introduce an “external field” , affecting each node, so that the combined Hamiltonian is

$$H_{\text{full}}(s) = H(s) + \alpha \sum_{i=1}^{N} s_i^2.$$  

(3)

The field corresponds to a spring that attracts every node to the origin. The spring constant is a tunable parameter: we can think of it as imposing a Gaussian prior on the ranks, or as a regularization term that quadratically penalizes ranks with large absolute values. Since scales with the total edge weight , while scales with , for a fixed value of this regularization becomes less relevant as networks become more dense and the average (weighted) degree increases. For there is a unique which minimizes given by

$$[D^\text{out} + D^\text{in} - (A + AT^T) + \alpha \mathbb{I}] s^* = [D^\text{out} - D^\text{in}] \mathbf{1},$$

(4)

where is the identity matrix. The matrix on the left side is now invertible, since the eigenvector has eigen-
values \( \alpha \) instead of 0. In the limit \( \alpha \rightarrow 0 \), we recover (2); the value \( \alpha = 2 \) corresponds to the Colley matrix method [19].

Minimizing \( H(s) \), or the regularized version \( H_{\text{full}}(s) \), corresponds to finding the “ground state” \( s^* \) of the model. In the next section we show that this corresponds to a maximum-likelihood estimate of the ranks in a generative model. However, we can use SpringRank not just to maximize the likelihood, but to compute a joint distribution of the ranks as a Boltzmann distribution with Hamiltonian (3), and thus estimate the uncertainty and correlations between the ranks. In particular, the ranks \( s_i \) are random variables following an \( N \)-dimensional Gaussian distribution with mean \( s^* \) and (SI Text S4) covariance matrix

\[
\Sigma = \frac{1}{\beta} [D_{\text{out}} + D_{\text{in}} - (A + AT + \alpha I)]^{-1}.
\]

Here \( \beta \) is an inverse temperature controlling the amount of noise in the model. In the limit \( \beta \rightarrow \infty \), the rankings are sharply peaked around the ground state \( s^* \), while for \( \beta \rightarrow 0 \) they are noisy. As we discuss below, we can estimate \( \beta \) from the observed data in various ways.

The rankings given by SpringRank (2) and its regularized form (4) are easily and rapidly computed by standard linear solvers. In particular, iterative solvers that take advantage of the sparsity of the system can find \( s^* \) for networks with millions of nodes and edges in seconds [27]. However, as defined above, SpringRank is not a fully generative model that assigns probabilities to the data and allows for Bayesian inference, or that can predict unobserved interactions. In the next section we introduce a generative model for hierarchical networks and show that it converges to SpringRank in the limit of strong hierarchy or sparse networks.

A generative model

In this section we propose a probabilistic generative model that takes as its input a set of node ranks \( s_1, \ldots, s_N \) and produces a directed network with adjacency matrix \( A \). In this formulation, edges are generated stochastically, and each edge is conditionally independent of all others. We let the expected number of edges between nodes \( i \) and \( j \) be proportional to the Boltzmann weight of the spring energy:

\[
E[A_{ij}] = c \exp \left[ -\frac{1}{2} \beta(s_i - s_j - 1)^2 \right],
\]

with the actual edge weight \( A_{ij} \) drawn from a Poisson distribution with this mean. The constant \( c \) controls the overall density of the network, giving an expected number of edges

\[
E[M] = \sum_{i,j} E[A_{ij}] = c \sum_{i,j} \exp \left[ -\frac{1}{2} \beta(s_i - s_j - 1)^2 \right],
\]

while the inverse temperature \( \beta \) controls the extent to which edges respect (or defy) the ranks \( s \). For smaller \( \beta \), edges are more likely to violate the hierarchy or to connect distant nodes, decreasing the correlation between the ranks and the directions of the interactions: for \( \beta = 0 \) the model generates a directed Erdős-Rényi graph, while in the limit \( \beta \rightarrow \infty \) edges only exist between nodes \( i, j \) with \( s_i - s_j = 1 \), and only in the direction \( i \rightarrow j \).

The Poisson distribution may generate multiple edges between a pair of nodes, so this model may create multigraphs. This is consistent with the interpretation that \( A_{ij} \) is the number, or total weight, of links from \( i \) to \( j \). If we wish to generate binary networks where \( A_{ij} \in \{0,1\} \), we use the fact that the Poisson and Bernoulli distributions become close in the sparse limit where \( c \) is small.

The likelihood of observing a network \( A \) given ranks \( s \), inverse temperature \( \beta \), and density \( c \) is

\[
P(A \mid s, \beta, c) = \prod_{i,j} \left[ \frac{\exp \left[ -\frac{\beta}{2}(s_i - s_j)^2 \right]}{A_{ij}} \right] \exp \left[ -c \frac{\beta}{2}(s_i - s_j - 1)^2 \right].
\]

Taking logs, substituting the maximum-likelihood value of \( c \), and discarding constants that do not depend on \( s \) or \( \beta \) yields a log-likelihood (see Supplemental Text S2)

\[
\mathcal{L}(A \mid s, \beta) = -\beta H(s) - M \log \left[ \sum_{i,j} e^{\frac{\beta}{2}(s_i - s_j)^2} \right].
\]

Note the appearance of the SpringRank energy \( H(s) \). For large \( \beta \) or small \( M \), i.e., when the hierarchical structure is strong or the network is sparse, the \( \hat{s} \) that maximizes (7), i.e., the maximum likelihood estimate (MLE) of the ranks, approaches the solution \( s^* \) of (2) that minimizes \( H(s) \).

As discussed above, we can break translational symmetry by adding a field \( H_0 \) that attracts the ranks to the origin. This is equivalent to imposing a prior \( P(s) \propto \prod_{i=1}^N e^{-\frac{\beta}{2}(s_i - \bar{s})^2} \). The maximum a posteriori estimate \( \hat{s} \) then approaches the ground state \( s^* \) of the Hamiltonian in (3), given by (4).

This model belongs to a larger family of generative models considered in ecology and network theory [8, 23, 24], and more generally the class of latent space models [26], where \( i \) links to \( j \) with probability \( f(s_i - s_j) \) for some function \( f \). These models typically have complicated posterior distributions with many local optima, requiring Monte Carlo methods (e.g. [25]) that do not scale efficiently to large networks. In our case, \( f(s_i - s_j) \) is a Gaussian centered at 1, and the posterior converges to the multivariate Gaussian (5) in the limit of strong structure or sparse networks.

Predicting edge directions

Hierarchical structure should allow us to predict the direction of previously unobserved interactions, such as
the winner of an upcoming match, or which direction social support will flow between two individuals. This is a kind of cross-validation, which lets us test the statistical significance of hierarchical structure. It is also a principled way of comparing the accuracy of various ranking methods for datasets where no ground truth for the ranks is known.

Edge prediction can be formulated as follows: what is the probability of an edge \( i \rightarrow j \) conditioned on the event that these two interact, i.e., that there is an edge in one direction or the other? In our model this conditional probability can be written using the Boltzmann distribution,

\[
P_{ij}(\beta) = \frac{e^{-\beta H_{ij}}}{e^{-\beta H_{ij}} + e^{-\beta H_{ji}}} = \frac{1}{1 + e^{-2\beta (s_i - s_j)}}.
\]  

(8)

where \( H_{ij} = (1/2)(s_i - s_j - 1)^2 \) is the term of the Hamiltonian (1) corresponding to that edge. Note that \( P_{ij} \) is a logistic curve, is monotonic in the rank difference \( s_i - s_j \), and has width determined by the inverse temperature \( \beta \). SpringRank has this in common with two other ranking methods: setting \( \gamma_i = e^{2\beta s_i} \) recovers the Bradley-Terry-Luce model [16, 17] for which \( P_{ij} = \gamma_i/(\gamma_i + \gamma_j) \), and setting \( k = 2\beta \) recovers the probability that \( i \) beats \( j \) in the Go rank [21], where \( P_{ij} = 1/(1 + e^{-k(s_i - s_j)}) \). Note, however, that SpringRank differs from these methods in how it infers the ranks from observed interactions.

In our experiments below, we test various ranking methods for edge prediction by giving them access to 80% of the edges in the network (the training data) and then asking them to predict the direction of the remaining edges (the test data). In order to learn the parameter \( \beta \) from the training data, and measure the accuracy on the test data, we consider two goodness-of-fit measures: \( \sigma_a \) is the number of edges whose direction is correctly predicted, and \( \sigma_L \) is the log-likelihood of generating the directed edges given their existence. For simple directed graphs where \( A_{ij} + A_{ji} \in \{0, 1\} \), these are

\[
\sigma_a = \sum_{i,j} A_{ij} P_{ij} \quad \text{and} \quad \sigma_L = \sum_{i,j} A_{ij} \log P_{ij}.
\]

In the multigraph case, we ask how well \( P_{ij} \) approximates the fraction of interactions between \( i \) and \( j \) that point from \( i \) to \( j \) [see Eqs. (10) and (11)].

We perform our prediction experiments as follows. Given the training data, we derive the ranks using (4). Since \( \beta \) affects the predictions \( P_{ij} \), we then set \( \beta \) to maximize either \( \sigma_a \) or \( \sigma_L \) on the training data while holding the ranks fixed. The resulting values of \( \beta \), which we denote \( \hat{\beta}_a \) and \( \hat{\beta}_L \) respectively, are distinct (SI Text S7).

This is intuitive, since a single severe mistake where \( A_{ij} = 1 \) but \( P_{ij} \approx 0 \) reduces the likelihood by a large amount, while only reducing the accuracy by one edge. As a result, predictions using \( \hat{\beta}_a \) produce fewer incorrectly oriented edges, achieving a higher \( \sigma_a \) on the test set, while predictions using \( \hat{\beta}_L \) will produce fewer dramatically incorrect predictions where \( P_{ij} \) is very low, and thus achieve higher \( \sigma_L \) on the test set (Fig. S3).

Statistical significance using the ground state energy

We can measure statistical significance using any test statistic, by asking whether its value on a given dataset would be highly improbable in a null model. One such statistic is the accuracy of edge prediction using a method such as the one described above; but this takes some time to compute, if only to average over many trials of cross-validation. Here we propose a test statistic which is very easy to compute, inspired by the physical model behind SpringRank: namely, the ground state energy. For the unregularized version Eq. (1), the energy per edge is (see SI Text S3)

\[
H(s^*) = \frac{1}{2M} \sum_i (d_i^+ - d_i^-) s_i^* + \frac{1}{2}.
\]

(9)

Since the ground state energy depends on many aspects of the network topology, and since hierarchical structure is statistically significant if it helps us predict edge directions, like [28] we focus on the following null model: we randomize the direction of each edge while preserving the total number \( A_{ij} = A_{ji} + A_{ij} \) of edges between each pair of vertices. If the real network has a ground state energy which is much lower than typical draws from this null model, we can conclude that the hierarchical structure is statistically significant.

This test correctly concludes that directed Erdős-Rényi graphs have no significant structure. It also finds no significant structure for networks created using the generative model (6) with \( \beta = 0.1 \), i.e., when the temperature or noise level \( 1/\beta \) is large (Fig. S2). However, we see in the next section that it shows statistically significant hierarchy for a variety of real-world datasets, showing that \( H(s^*) \) is both useful and computationally efficient as a test statistic.

Results on real and synthetic data

Having introduced SpringRank, an efficient procedure for inferring real-valued ranks, a corresponding generative model, a method for edge prediction, and a test for the statistical significance of hierarchical structure, we now demonstrate it by applying it to both real and synthetic data. Real-world datasets are often noisy, due to missing, spurious, or reversed interactions, but it is nevertheless reasonable to hope that a ranking algorithm will perform well in noisy conditions, both inferring the underlying ranks and predicting the direction of previously observed edges.

In the real-world datasets available to us, we only have a directed network of observed interactions—there is no available ground truth for the actual ranks. Thus we first examine the ability of SpringRank and competing algorithms to infer ranks from synthetic networks with known ranks, produced by the generative model described above with various prior distributions of ranks and various lev-
In our tests below, we compare SpringRank to other widely used methods: the spectral methods PageRank [10] and eigenvector centrality [9], minimum violation ranking (MVR) [11, 12], David’s score [18], and the BTL random utility model [16, 17] using the algorithm proposed in [29]. We also compare unregularized SpringRank with the regularized version $\alpha = 2$, corresponding to the Colley matrix method [19]. Unfortunately, eigenvector centrality, David’s score, and BTL are undefined when the network is not strongly connected, such as when there are “dangling nodes” with zero in- or out-degree. Thus we follow the common procedure of adding low-weight edges between every pair of nodes (SI Text S9).

**Performance for synthetic networks**

We study two types of synthetic networks, generated by the model described above. Of course, since the log-likelihood in this model corresponds asymptotically to the SpringRank energy, we expect SpringRank to do well on these networks, and its performance should be viewed largely as a consistency check. But by varying the distribution of ranks and the noise level, we can illustrate types of structure that may exist in real-world data, and test each algorithm’s ability to identify them.

In the first type, the ranks are normally distributed with mean zero and variance one (Fig. 1A). In the second type, the ranks are drawn from an equal mixture of three Gaussian with different means and variances, so that nodes cluster into high, middle, and low tiers (Fig. 1C). This second type is intended to focus on the importance of real-valued ranks, and to measure the performance of algorithms that (implicitly or explicitly) impose strong priors on the ranks when the data defy their expectations. In both cases, we vary the amount of noise by changing $\beta$ while keeping the total number of edges constant (see Materials and Methods).

Since we wish to compare SpringRank to methods such as MVR that only produce ordinal rankings, and to those like PageRank and David’s Score that produce ranks in an ad-hoc space without an interpretable metric, we measure the accuracy of each algorithm according to the Spearman correlation $\rho$ between its inferred rank order and the true one. If we instead measure the algorithms’ ability to infer the true real-valued ranks as defined by the Pearson correlation, this strongly favors SpringRank and BTL (Fig. S4).

We find that all the algorithms do well on the first type of synthetic network. As $\beta$ increases so that the network becomes more structured, with fewer edges (shown in red in Fig. 1A) pointing in the “wrong” direction, all algorithms infer ranks that are more correlated with the ground truth. SpringRank has the highest accuracy, followed closely by the Colley matrix method and BTL (Fig. 1B). Presumably the Colley matrix works well here because the ranks are in fact drawn from a Gaussian prior, as it implicitly assumes.

Results for the second type of network are more nuanced. The accuracy of SpringRank increases rapidly with $\beta$ with exact recovery around $\beta = 1$, while all other methods except eigenvector centrality are non-monotonic in their accuracy (Fig. 1D). This suggests that these other algorithms become confused when the nodes are clustered into tiers, even when the noise is small enough that most edges have directions consistent with the hierarchy. SpringRank takes advantage of the fact that edges are more likely between nodes in the same tier (Fig. 1C), so the mere existence of edges helps it cluster the ranks. In contrast, other algorithms may view the lack of between-tier edges as providing less information about the relative ranks between tiers.

These synthetic tests suggest that real-valued ranks...
capture information that ordinal ranks do not, and that many ranking systems perform poorly when there exist substructures such as tiered groups within the overall ranking. When such structures exist in real-world datasets, these tests clearly indicate that SpringRank or BTL should be preferred. Of course, in most real-world scenarios, the ground-truth ranks are not known, and thus edge prediction and other forms of cross validation should be used instead. For better comparison, edge prediction in synthetic networks is discussed in the context of real-world networks in the next section.

Real-world networks and edge prediction

As discussed above, in most real-world networks, we have no ground truth for the ranks. Thus we focus on our ability to predict edge directions from a subset of the data, and measure the statistical significance of the inferred hierarchy.

We apply our methods to datasets from a diverse set of fields, with sizes ranging up to $N = 415$ nodes and up to 7000 links (see Table S1): three North American academic hiring networks where $A_{ij}$ is the number of faculty at university $j$ who received their doctorate from university $i$, for History (illustrated in Figs. 2A and B), Business, and Computer Science departments [2]; two networks of animal dominance among captive monk parakeets [4] and one among Asian elephants [28] where $A_{ij}$ is the number of dominating acts by animal $i$ toward animal $j$; and social support networks from two villages in Tamil Nadu referred to (for privacy reasons) by the pseudonyms “Tenpat’ti” and “A’lakāpuram,” where $A_{ij}$ is the number of distinct social relationships (up to five) through which person $i$ supports person $j$ [30].

Together, these examples cover prestige, dominance, and social hierarchies. In each of these domains, inferring ranks from interactions is key to further analysis. Prestige hierarchies play an unequivocal role in the dynamics of academic labor markets [31]: in behavioral ecology, higher ranked individuals in dominance hierarchies have higher fitness [1, 32]; and patterns of aggression reveal animal strategies and cognitive capacities [3–7]. Finally, in social support networks, higher ranked individuals have greater social capital and reputational standing [33, 34], particularly in settings in which social support is a primary way to express and gain respect and prestige [35].

We first applied our ground state energy test for the presence of a statistically significant hierarchy, rejecting the null hypothesis with $p < 10^{-4}$ in all cases (e.g., for History faculty hiring, see Fig. 2C) except the Asian Elephants network for which $p > 0.4$. This corroborates the original study of this network [28], which found that counting triad motifs shows no significant hierarchy [36]. This is despite the fact that one can find an appealing ordering of the elephants with the Minimum Violation Rank method, with just a few violating edges (SI Fig. S8). Thus the hierarchy found by MVR may well be illusory.

As described above, we performed edge prediction experiments using 5-fold cross-validation, where 80% of the edges are available to the algorithm as training data, and a test set consisting of 20% of the edges is held out (see Materials and Methods). Since most of the competing methods do not make probabilistic predictions about edge directions, we focused on SpringRank and the BTL model. We found that SpringRank systematically outperforms BTL, both in terms of the accuracy $\sigma_a$ (Fig. 3A)
FIG. 3. Edge prediction accuracy over BTL. Distribution of differences in performance of edge prediction of SpringRank compared to BTL on real and synthetic networks defined as (A) edge-prediction accuracy \( \sigma_a \) (10) and (B) the conditional log-likelihood \( \sigma_L \) (11). Error bars indicate quartiles and markers show medians, corresponding to 50 independent trials of 5-fold cross-validation, for a total of 250 test sets for each network. The two synthetic networks are generated with \( N = 100 \), average degree 5, and Gaussian-distributed ranks as in Fig. 1A, with inverse temperatures \( \beta = 1 \) and \( \beta = 5 \). For each experiment shown, the fractions of trials in which each method performed equal to or better than BTL are shown in Table. I.

![Diagram A: Accuracy improvement over BTL](image)

![Diagram B: Log-likelihood improvement over BTL](image)

TABLE I. Edge prediction with BTL as a benchmark. During 50 independent trials of 5-fold cross-validation (250 total folds), columns show the percentages of instances in which SpringRank Eq. (2) and regularized SpringRank Eq. (4) with \( \alpha = 2 \) produced equal or higher accuracy predictions than BTL. Distributions of accuracy improvements are shown in Fig. 3. Center columns show accuracy \( \sigma_a \) and right columns show \( \sigma_L \) (Materials and Methods). Italics indicate where BTL outperformed SpringRank for more than 50% of tests. † Dagger symbols indicate tests that are shown in detail in Fig. 4.

| Dataset       | Type            | \% trials higher \( \sigma_a \) vs BTL | SpringRank +regularization | \% trials higher \( \sigma_L \) vs BTL |
|---------------|-----------------|---------------------------------------|---------------------------|---------------------------------------|
| Alakapurum    | Social Support  | 99.2†                                  | 99.6                     | 100.0                                 |
| Tenpatti      | Social Support  | 88.8                                   | 93.6                     | 100.0                                 |
| Parakeet G1   | Animal Dominance| 71.2†                                  | 56.8                     | 41.2                                  |
| Parakeet G2   | Animal Dominance| 62.0                                   | 51.6                     | 47.6                                  |
| Business      | Faculty Hiring  | 66.8†                                  | 59.2                     | 39.2                                  |
| Comp. Sci.    | Faculty Hiring  | 100.0                                  | 97.2                     | 100.0                                 |
| History       | Faculty Hiring  | 97.6†                                  | 96.8                     | 98.8                                  |
| Synthetic \( \beta = 1 \) | Synthetic | 83.2                                   | 65.2                     | 98.4                                  |
| Synthetic \( \beta = 5 \) | Synthetic | 98.4                                   | 63.2                     | 76.4                                  |

and, for most networks, the log-likelihood \( \sigma_L \) (Fig. 3B).

As shown in the scatterplots in Fig. 4, the edge prediction accuracy of both methods has a fairly broad distribution over the trials of cross-validation, since for each network some subsets of the edges are harder to predict than others when they are held out; but in most trials SpringRank was more accurate than BTL. As a result, Fig. 3A and Table I show that SpringRank predicts edge directions more accurately in the majority of trials of cross-validation for all 9 networks, where this majority ranges from 62% for the parakeet networks to 100% for the Computer Science hiring network.

In terms of the log-likelihood \( \sigma_L \), SpringRank outperformed BTL for 6 of the 9 networks (Table I). We found similar results for a more coarse-grained measure of accuracy (Fig. S1). Regularizing SpringRank with \( \alpha = 2 \) does not appear to improve either measure of accuracy (Fig. 3). We did not attempt to tune the regularization parameter \( \alpha \).

Finally, the real-valued ranks found by SpringRank shed light on the organization and assembly of real-world networks. For example, we found that ranks in the faculty hiring networks have a long tail at the top, suggesting that the most prestigious universities are more separated from those below them than an ordinal ranking would reveal. In contrast, ranks in the social support networks have a long tail at the bottom, suggesting a set of people who do not have sufficient social status to provide support to others. SpringRank’s ability to find real-valued ranks makes these patterns obvious from visual inspection (see Figs. S5, S6, S7, S11, and S12).
SpringRank prediction accuracy on real-world networks of various sizes.

were chosen to illustrate the range of relative performances. Results are shown in plot legends, matching Table I. Networks were accurate than BTL; the fractions for which each method is superior are shown in plot legends, matching Table I. Networks were chosen to illustrate the range of relative performances on real-world networks of various sizes.

**Conclusions**

SpringRank is a mathematically principled, physics-inspired model for hierarchical structure in networks of directed interactions. It yields a simple, highly scalable and nonparametric algorithm that infers real-valued ranks, requiring only sparse linear algebra, which enables analysis of networks with millions of nodes and edges in seconds. The corresponding generative model allows one to create synthetic networks with tunable levels of hierarchy and noise, whose posterior coincides with SpringRank in the limit where these networks are sparse or strongly structured.

In addition to providing a simple and natural test for statistically significant hierarchy, this model allows probabilistic predictions of edge directions for unobserved interactions. While SpringRank is nonparametric, a parameterized regularization can be included as well, corresponding to a Gaussian prior. Although regularization is common—indeed, it is often required for BTL and eigenvector centrality (Supplemental Text S9)—it is not necessary for SpringRank and our tests indicate that its effects are mixed.

We tested this model for its ability to infer ranks from synthetic networks where ground truth ranks are known, and to predict edge directions in real networks. We found that in networks of faculty hiring, animal interactions, and social support, it consistently outperforms the popular Bradley-Terry-Luce model at edge prediction. In synthetic networks, it significantly outperforms a wide variety of methods at inferring the ranks, including BTL, spectral algorithms, and Minimum Violation Ranking. Given its simplicity, speed, and high performance, we believe that SpringRank will be useful in a wide variety of fields where hierarchical structure appears due to dominance, social status, or prestige.

**Materials and methods**

**Synthetic network generation**

Networks were generated in three steps. First, node ranks $s_{\text{planted}}$ were drawn from a chosen distribution. For Test 1, $N = 100$ ranks were drawn from a standard normal distribution, while for Test 2, 34 ranks were each drawn from $N(-4, 2)$, $N(0, \frac{1}{2})$, and $N(4, 1)$ for a total of $N = 102$. Second, an average degree $\langle k \rangle$ and a value of the inverse temperature $\beta$ were chosen. Third, edges were drawn according to (6), with $c = \langle k \rangle N / \sum_{i,j} \exp[\frac{-\beta}{2}(s_i - s_j - 1)^2]$, ensuring that synthetically generated networks have mean degree $\langle k \rangle$ in expectation (see SI Text S6). This procedure resulted in a directed network with the desired hierarchy, number of edges, and noise level. Tests were conducted for $\langle k \rangle \in [5, 15]$, $\beta \in [0.1, 5]$, and all performance plots show mean and standard deviations for 100 replicates.

**Performance measures for edge prediction**

In the multigraph case, we define the accuracy of an edge prediction as the extent to which $P_{ij}$ is a good estimate of the fraction of interactions between $i$ and $j$ that point from $i$ to $j$. If this prediction were perfect, we would have $A_{ij} = \hat{A}_{ij} P_{ij}$ where $\hat{A}_{ij} = A_{ij} + A_{ji}$. We define $\sigma_A$ as 1 minus the sum of the absolute values of the difference between $A_{ij}$ and this estimate,

$$\sigma_A = 1 - \frac{1}{2M} \sum_{i,j} |A_{ij} - (A_{ij} + A_{ji}) P_{ij}|$$  

where $M$ is the number of directed edges in the subset of the network under consideration, e.g., the training or test set). If $P_{ij} = \hat{A}_{ij}/\hat{A}_{ij}$ for all $i,j$, then $\sigma_A = 1$.

To measure the performance via the conditional log-likelihood, we ask with what probability we would get the directed network $A$ from the undirected network $\hat{A}$ if each edge between $i$ and $j$ points from $i \rightarrow j$ with probability $P_{ij}$ and from $j \rightarrow i$ with probability $P_{ji} = 1 - P_{ij}$. This gives

$$\sigma_L = \log \Pr[A | \hat{A}] = \sum_{i,j} \log\left(\frac{A_{ij} + A_{ji}}{A_{ij}}\right) + \log\left(P_{ij}^{A_{ij}} [1 - P_{ij}]^{A_{ji}}\right)$$

where $\binom{x}{y}$ is the binomial coefficient. We disregard the first term of this sum since it does not depend on $P$. If we wish to compare networks of different sizes as in Fig. 3, we can normalize $\sigma_L$ by the number of edges.
Statistical significance of ranks

We compute a standard left-tailed $p$-value for the statistical significance of the ranks $s^*$ by comparing the ground state energy (9) of the real network $A$ with the null distribution of ground state energies of an ensemble of edge direction-randomized networks $	ilde{A}$.

$$p\text{-value} = \Pr[H(s^*; A) \leq H(\tilde{s}^*; \tilde{A})].$$ (12)

In practice, this $p$-value is estimated by drawing many samples from the null distribution by randomizing the edge directions of $A$ to produce $\tilde{A}$, computing the ranks $\tilde{s}^*$ from (2), and then computing the ground state energy (9) of each.

Cross validation tests

We performed edge prediction using 5-fold cross-validation. In each realization, we divide the interacting pairs $i,j$, i.e., those with nonzero $\tilde{A}_{ij} = A_{ij} + A_{ji}$, into five equal groups. We use four of these groups as a training set, inferring the ranks and setting $\beta$ to maximize $\sigma_a$ or $\sigma_L$ (on the left and right of Fig. 3 respectively). We then use the fifth group as a test set, asking the algorithm for $P_{ij}$ for each pair $i,j$ in that group, and report $\sigma_a$ or $\sigma_L$ on that test set. By varying which group we use as the test set, we get 5 trials per realization: for instance, 50 realizations give us 250 trials of cross validation.

Acknowledgements

CDB and CM were supported by the John Templeton Foundation. CM was also supported by the Army Research Office under grant W911NF-12-R-0012. DBL was supported by NSF award SMA-1633747 and the Santa Fe Institute Omidyar Fellowship. We thank Aaron Clauset and Johan Ugander for helpful comments.

[1] C. Drews, Behaviour 125, 283 (1993).
[2] A. Clauset, S. Arbesman, and D. B. Larremore, Science advances 1, e1400005 (2015).
[3] S. D. Côté and M. Festa-Bianchet, Animal Behaviour 62, 173 (2001).
[4] E. A. Hobson and S. DeDeo, PLoS Comput Biol 11, e1004411 (2015).
[5] C. J. Dey and J. S. Quinn, Behavioral Ecology 25, 1402 (2014).
[6] C. J. Dey, A. R. Reddon, C. M. O’Connor, and S. Balshine, Animal Behaviour 85, 395 (2013).
[7] M. A. Cant, J. B. Llop, and J. Field, The American Naturalist 167, 837 (2006).
[8] B. Ball and M. E. Newman, Network Science 1, 16 (2013).
[9] P. Bonacich, American journal of sociology 92, 1170 (1987).
[10] L. Page, S. Brin, R. Motwani, and T. Winograd, The PageRank citation ranking: Bringing order to the web., Tech. Rep. (Stanford InfoLab, 1999).
[11] I. Ali, W. D. Cook, and M. Kress, Management Science 32, 660 (1986).
[12] P. Slater, Biometrika 48, 303 (1961).
[13] M. Gupte, P. Shankar, J. Li, S. Muthukrishnan, and L. Iftode, in Proc. 20th Intl. Conf. on the World Wide Web (ACM, 2011) pp. 557–566.
[14] N. Tatti, Data Mining and Knowledge Discovery 31, 702 (2017).
[15] K. E. Train, Discrete choice methods with simulation (Cambridge university press, 2009).
[16] R. A. Bradley and M. E. Terry, Biometrika 39, 324 (1952).
[17] R. D. Luce, Psychological Review 66, 81 (1959).
[18] H. A. David, Biometrika 74, 432 (1987).
[19] W. N. Colley, (2002).
[20] A. E. Elo, The rating of chessplayers, past and present (Arco Pub., 1978).
[21] R. Coulom, in International Conference on Computers and Games (Springer, 2008) pp. 113–124.
[22] R. Herbrich, T. Minka, and T. Graepel, in Advances in neural information processing systems (2007) pp. 569–576.
[23] R. J. Williams, A. Anandanadesan, and D. Purves, PloS one 5, e12092 (2010).
[24] R. J. Williams and D. W. Purves, Ecology 92, 1849 (2011).
[25] A. Z. Jacobs, J. A. Dunne, C. Moore, and A. Clauset, arXiv preprint arXiv:1505.04741 (2015).
[26] P. D. Hoff, A. E. Raftery, and M. S. Handcock, J. American Statistical Association 97, 1090 (2001).
[27] Open-source code in Python and MATLAB available at https://github.com/cdebacco/SpringRank.
[28] S. de Silva, V. Schmid, and G. Wittemyer, Behavioral Ecology , arw153 (2016).
[29] D. R. Hunter, Annals of Statistics , 384 (2004).
[30] E. A. Power, Nature Human Behaviour 1, 0057 (2017).
[31] S. F. Way, D. B. Larremore, and A. Clauset, in Proc. 25th Intl Conf on World Wide Web (2016) pp. 1169–1179.
[32] B. Majolo, J. Lehmann, A. de Bortoli Vizioli, and G. Schino, American Journal of Physical Anthropology 147, 652 (2012).
[33] N. Lin, Social capital: A theory of social structure and action, Vol. 19 (Cambridge University Press, 2002).
[34] K. S. Cook, M. Levi, and R. Hardin, Whom can we trust?: How groups, networks, and institutions make trust possible (Russell Sage Foundation, 2009).
[35] M. Mimes, Public faces, private lives: Community and individuality in South India (Univ of California Press, 1994).
[36] D. Shizuka and D. B. McDonald, Animal Behaviour 83, 925 (2012).
Supporting Information (SI)

S1. Deriving the linear system minimizing the Hamiltonian

The SpringRank Hamiltonian (1) is convex in $s$ and we set its gradient $\nabla H(s) = 0$ to obtain the global minimum:

$$\frac{\partial H}{\partial s_i} = \sum_j [A_{ij} (s_i - s_j - 1) - A_{ji} (s_j - s_i - 1)] = 0 . \quad (S1)$$

Let the weighted out-degree and in-degree be $d_i^\text{out} = \sum_j A_{ij}$ and $d_i^\text{in} = \sum_j A_{ji}$, respectively. Then (S1) can be written as

$$(d_i^\text{out} + d_i^\text{in}) s_i - (d_i^\text{out} - d_i^\text{in}) - \sum_j [A_{ij} + A_{ji}] s_j = 0 . \quad (S2)$$

We now write the system of $N$ equations together by introducing the following matrix notation. Let $D^\text{out} = \text{diag}(d_1^\text{out}, \ldots, d_N^\text{out})$ and $D^\text{in} = \text{diag}(d_1^\text{in}, \ldots, d_N^\text{in})$ be diagonal matrices, let $1$ be the $N$-dimensional vector of all ones. Then (S2) becomes

$$[D^\text{out} + D^\text{in} - (A + A^T)] s = [D^\text{out} - D^\text{in}] \ 1 . \quad (S3)$$

This is a linear system of the type $B\ s = b$, where $B = [D^\text{out} + D^\text{in} - (A + A^T)]$ and $b = [D^\text{out} - D^\text{in}] \ 1$. The rank of $B$ is at most $N - 1$ and more generally, if the network represented by $A$ consists of $C$ disconnected components, $B$ will have rank $N - C$. In fact, $B$ has an eigenvalue 0 with multiplicity $C$, and the eigenvector $1$ is in the nullspace. $B$ is not invertible, but we can only invert in the $N - C$-dimensional subspace orthogonal to the nullspace of $B$. The family of translation-invariant solutions $s^*$ is therefore defined by

$$s^* = [D^\text{out} + D^\text{in} - (A + A^T)]^{-1} [D^\text{out} - D^\text{in}] \ 1 , \quad (S4)$$

in which the notation $[\cdot]^{-1}$ should be taken as the pseudo inverse.

In practice, rather than constructing the pseudo-inverse, it will be more computationally efficient (and for large systems, more accurate) to solve the linear system in an iterative fashion. Since we know that solutions may be translated up or down by an arbitrary constant, the system can be made full-rank by fixing the position of an arbitrary node 0. Without loss of generality, let $s_N = 0$. In this case, terms that involve $s_N$ can be dropped from (S2), yielding

$$(d_i^\text{out} + d_i^\text{in}) s_i - (d_i^\text{out} - d_i^\text{in}) - \sum_{j=1}^{N-1} [A_{ij} + A_{ji}] s_j = 0 , \quad i \neq N \quad (S5)$$

$$- (d_N^\text{out} - d_N^\text{in}) - \sum_{j=1}^{N-1} [A_{Nj} + A_{jN}] s_j = 0 . \quad (S6)$$

Adding (S5) to (S6) yields

$$(d_i^\text{out} + d_i^\text{in}) s_i - (d_i^\text{out} + d_N^\text{out} - d_i^\text{in} - d_N^\text{in}) - \sum_{j=1}^{N-1} [A_{ij} + A_{Nj} + A_{ji} + A_{jN}] s_j = 0 \quad (S7)$$

which can be written in matrix notation as

$$[D^\text{out} + D^\text{in} - \hat{A}] \ s = [D^\text{out} - D^\text{in}] \ 1 + (d_N^\text{out} - d_N^\text{in}) \ 1 \ . \quad (S8)$$

where

$$\hat{A}_{ij} = A_{ij} + A_{Nj} + A_{ji} + A_{jN} \ . \quad (S9)$$

In this formulation, (S8) can be solved to arbitrary precision using iterative methods that take advantage of the sparsity of $\hat{A}$. The resulting solution may then be translated by an arbitrary amount as desired.
S2. Poisson generative model

The expected number of edges from node $i$ to node $j$ is $c \exp \left[ -\beta H(s_i - s_j - 1)^2 \right]$ and therefore the likelihood of observing a network $A$, given parameters $\beta$, $s$, and $c$ is

$$ P(A | s, \beta, c) = \prod_{i,j} \left[ \frac{c e^{-\frac{\beta}{2} (s_i - s_j - 1)^2}}{A_{ij}!} \right] A_{ij} \exp \left[ -c e^{-\frac{\beta}{2} (s_i - s_j - 1)^2} \right]. $$

(S10)

Taking logs yields

$$ \log P(A | s, \beta, c) = \sum_{i,j} A_{ij} \log c - \frac{\beta}{2} A_{ij} (s_i - s_j - 1)^2 - \log [A_{ij}!] - ce^{-\frac{\beta}{2} (s_i - s_j - 1)^2}. $$

(S11)

Discarding the constant term $\log [A_{ij}!]$, and recognizing the appearance of the SpringRank Hamiltonian $H(s)$, yields

$$ \mathcal{L}(A | s, \beta, c) = -\beta H(s) + \sum_{i,j} A_{ij} \log c - \sum_{i,j} ce^{-\frac{\beta}{2} (s_i - s_j - 1)^2}. $$

(S12)

Taking $\partial \mathcal{L} / \partial c$ and setting it equal to zero yields

$$ \hat{c} = \frac{\sum_{i,j} A_{ij}}{\sum_{i,j} e^{-\frac{\beta}{2} (s_i - s_j - 1)^2}}, $$

(S13)

which has the straightforward interpretation of being the ratio between the number of observed edges and the expected number of edges created in the generative process for $c = 1$. Substituting in this solution and letting $M = \sum_{i,j} A_{ij}$ yields

$$ \mathcal{L}(A | s, \beta) = -\beta H(s) + M \log \hat{c} - \sum_{i,j} \hat{c} e^{-\frac{\beta}{2} (s_i - s_j - 1)^2} $$

$$ = -\beta H(s) + M \log M - M \log \left[ \sum_{i,j} e^{-\frac{\beta}{2} (s_i - s_j - 1)^2} \right] - M. $$

(S14)

The terms $M \log M$ and $M$ may be neglected since they do not depend on the parameters, and we divide by $\beta$, yielding a log-likelihood of

$$ \mathcal{L}(A | s, \beta) = -H(s) - \frac{M}{\beta} \log \left[ \sum_{i,j} e^{-\frac{\beta}{2} (s_i - s_j - 1)^2} \right]. $$

(S15)

Note that the SpringRank Hamiltonian may be rewritten as $H(s) = \beta \left[ \frac{1}{2} \langle (s_i - s_j - 1)^2 \rangle_E \right]$ where $\langle \cdot \rangle_E$ denotes the average over elements in the edge set $E$. In other words, $H(s)$ scales with $\beta$ and half of the average spring length. Finally, this substitution for $H(s)$ and dividing by $M$ allows us to make sense of the behavior of the log-likelihood

$$ \mathcal{L}(A | s, \beta) = -M \left\{ \frac{1}{2} \left\langle (s_i - s_j - 1)^2 \right\rangle_E - \frac{1}{\beta} \log \left[ \sum_{i,j} e^{-\frac{\beta}{2} (s_i - s_j - 1)^2} \right] \right\}. $$

(S16)

In other words, the second term decays rapidly as $\beta$ increases, indicating that for well-resolved hierarchies, the maximum likelihood ranks $\hat{s}$ approach the ranks $s^*$ found by minimizing the Hamiltonian. In practice, exactly maximizing the likelihood would require expensive Monte Carlo sampling.
S3. Rewriting the energy

The Hamiltonian (1) can be rewritten as:

\[
2H(s) = \sum_{i,j=1}^{N} A_{ij} (s_i - s_j - 1)^2 \tag{S17}
\]

\[
= \sum_{i,j=1}^{N} A_{ij} \left( s_i^2 + s_j^2 - 2s_i s_j + 1 - 2s_i + 2s_j \right) \tag{S18}
\]

\[
= \sum_{i=1}^{N} s_i^2 \sum_{j=1}^{N} A_{ij} + \sum_{j=1}^{N} s_j^2 \sum_{i=1}^{N} A_{ij} - 2 \sum_{i=1}^{N} s_i \sum_{j=1}^{N} A_{ij} s_j + M \tag{S19}
\]

\[
-2 \sum_{i=1}^{N} s_i \sum_{j=1}^{N} A_{ij} + 2 \sum_{j=1}^{N} s_j \sum_{i=1}^{N} A_{ij} \tag{S20}
\]

\[
= \sum_{i=1}^{N} s_i^2 \left( d_i^{\text{out}} + d_i^{\text{in}} \right) - 2 \sum_{i=1}^{N} s_i \left( d_i^{\text{out}} - d_i^{\text{in}} \right) + M - 2 \sum_{i=1}^{N} s_i \sum_{j=1}^{N} A_{ij} s_j \tag{S21}
\]

From (S2) we have:

\[
\sum_{j=1}^{N} s_i \left( d_i^{\text{out}} - d_i^{\text{in}} \right) = \sum_{i=1}^{N} s_i \sum_{j=1}^{N} [A_{ij} + A_{ji}] s_j \tag{S22}
\]

We can substitute this into (S21):

\[
2H(s) = \sum_{i=1}^{N} s_i \sum_{j=1}^{N} [A_{ij} + A_{ji}] s_j - \sum_{i=1}^{N} s_i \left( d_i^{\text{out}} - d_i^{\text{in}} \right) + M - 2 \sum_{i=1}^{N} s_i \sum_{j=1}^{N} A_{ij} s_j \tag{S23}
\]

\[
= \sum_{i=1}^{N} s_i \left( d_i^{\text{in}} - d_i^{\text{out}} \right) + M \tag{S24}
\]

\[
= \sum_{i=1}^{N} h_i s_i + M \tag{S25}
\]

where \( h_i \equiv d_i^{\text{in}} - d_i^{\text{out}} \).

S4. Ranks distributed as a multivariate Gaussian distribution

Assuming that the ranks are random variables distributed as a multivariate Gaussian distribution of average \( \bar{s} \) and covariance matrix \( \Sigma \), we have:

\[
P(s) \propto \exp \left( -\frac{1}{2} (s - \bar{s})^T \Sigma^{-1} (s - \bar{s}) \right) \tag{S26}
\]

We can obtain this formulation by considering a Boltzmann distribution with the Hamiltonian (1) as the energy term and inverse temperature \( \beta \) so that:

\[
P(s) \propto \exp \left( -\beta \sum_{i,j=1}^{N} A_{ij} (s_i - s_j - 1)^2 \right) \tag{S27}
\]

Developing the exponent of expression (S26) we get:

\[
\frac{1}{2} (s - \bar{s})^T \Sigma^{-1} (s - \bar{s}) = \frac{1}{2} \left( s^T \Sigma^{-1} s - 2 s^T \Sigma^{-1} \bar{s} + \bar{s}^T \Sigma^{-1} \bar{s} \right) \tag{S28}
\]

whereas the one of (S27) is

\[
\frac{\beta}{2} \sum_{i,j=1}^{N} A_{ij} (s_i - s_j - 1)^2 = \frac{\beta}{2} \left[ s^T (D^{\text{out}} + D^{\text{in}} - A^T - A) s + 2 s^T (D^{\text{in}} - D^{\text{out}}) 1 + M \right] \tag{S29}
\]

where \( h_i = d_i^{\text{in}} - d_i^{\text{out}} \).
where $\mathbf{1}$ is a vector of ones and $D^i$ are diagonal matrices whose entries are the in- and out-degrees, $D^o_i = \sum_j A_{ij}$ and $D^i_i = \sum_j A_{ji}$ and $M = \sum_{i,j} A_{ij}$. Comparing these last two expressions and removing terms that do not depend on $s$ because irrelevant when accounting for normalization, we obtain:

$$\Sigma = 1/\beta \left( D^o + D^i - A^T - A \right)^{-1}$$

(S30)

$$\bar{s} = \beta \Sigma \left( D^o - D^i \right) \mathbf{1} = s^*$$

(S31)

S5. Bayesian SpringRank

Adopting a Bayesian approach with a factorized Gaussian prior for the ranks, we obtain that the $s$ that maximizes the posterior distribution is the one that minimizes the full SpringRank Hamiltonian (3), i.e. the $s$ that solves the linear system (4). In fact, defining:

$$P(s) = Z^{-1}(\beta, \alpha) \prod_{i \in V} e^{-\beta \frac{2}{2} (s_i - \bar{s})^2} = Z^{-1}(\beta, \alpha) \prod_{i \in V} e^{-\beta \alpha H_0(s_i)}$$

(S32)

where $Z(\beta, \alpha) = \left[ \frac{2\pi}{\beta \alpha} \right]^{N/2}$ is a normalization constant that depends on $\alpha$ and $\beta$, and following the same steps as before we get:

$$\log P(s|A) = \sum_{i,j} \log P(A_{ij}|s) - \beta \alpha \sum_{i \in V} H_0(s_i) + \log(Z(\beta, \alpha)) \sim -\beta \left[ H(s) + \alpha \sum_{i \in V} H_0(s_i) \right]$$

(S33)

where we neglected the term containing the normalization because it does not depend on the parameters.

S6. Fixing $c$ to control for sparsity

The parameter $c$ included in the generative model (6) controls for network’s sparsity. We can indeed fix it so to obtain a network with a desired expected number of edges $\langle M \rangle$ as follows:

$$\langle M \rangle \equiv \sum_{i,j} \langle A_{ij} \rangle = c \sum_{i,j} e^{-\beta \frac{2}{2} (s_i - s_j - 1)^2}$$

For a given vector of ranks $s$ and inverse temperature $\beta$, the $c$ realizing the desired sparsity will then be:

$$c = \frac{\langle M \rangle}{\sum_{i,j} e^{-\beta \frac{2}{2} (s_i - s_j - 1)^2}} = \frac{\langle k \rangle N}{\sum_{i,j} e^{-\beta \frac{2}{2} (s_i - s_j - 1)^2}}$$

(S34)

where $\langle k \rangle$ is the expected node degree $\langle k \rangle = \sum_{i=1}^N [d^i + d^o_i]$. Similar arguments apply when considering a generative model with Bernoulli distribution.

S7. Comparing optimal $\beta$ for predicting edge directions

In the main text, (10) and (11) define the accuracy of edge prediction, in terms of the number of edges predicted correctly in each direction and the log-likelihood conditioned on the undirected graph. Here we compute the optimal values of $\beta$ for both notions of accuracy. In both computations that follow, the following two facts will be used:

$$P'_{ij}(\beta) = 2(s_i - s_j) e^{-2\beta(s_i - s_j)} P^2_{ij}(\beta)$$

(S35)

$$1 = \frac{P_{ij}(\beta)}{1 - P_{ij}(\beta)} e^{-2\beta(s_i - s_j)}$$

(S36)
A. Choosing $\beta$ to optimize edge direction accuracy

We take the derivative of (10) with respect to $\beta$, set it equal to zero, and partially solve as follows.

$$0 = \frac{\partial \sigma_a(\beta)}{\partial \beta} = \frac{\partial}{\partial \beta} \left[ 1 - \frac{1}{2m} \sum_{i,j} |A_{ij} - (A_{ij} + A_{ji}) P_{ij}(\beta)| \right]. \quad (S37)$$

In preparation to take the derivatives above, note that $P_{ij}(\beta) = -P_{ji}(\beta)$ and that whenever the $(i, j)$ term of $\sigma_a(\beta)$ takes one sign, the $(j, i)$ term takes the opposite sign,

$$A_{ij} - (A_{ij} + A_{ji}) P_{ij}(\beta) = - [A_{ji} - (A_{ij} + A_{ji}) P_{ji}(\beta)] . \quad (S38)$$

Without loss of generality, assume that the $(i, j)$ term is positive and the $(j, i)$ term is negative. This implies that

$$\frac{\partial}{\partial \beta} |A_{ij} - (A_{ij} + A_{ji}) P_{ij}(\beta)| = -(A_{ij} + A_{ji}) P'_{ij}(\beta) \quad (S39)$$

$$\frac{\partial}{\partial \beta} |A_{ji} - (A_{ij} + A_{ji}) P_{ji}(\beta)| = -(A_{ij} + A_{ji}) P'_{ij}(\beta) . \quad (S40)$$

In other words, the derivatives of the $(i, j)$ and $(j, i)$ terms are identical, and the sign of both depends on whether the quantity $[A_{ij} - (A_{ij} + A_{ji}) P_{ij}(\beta)]$ is positive or negative. We can make this more precise by directly including the sign of the $(i, j)$ term, and by using (S35), to find that

$$\frac{\partial}{\partial \beta} |A_{ij} - (A_{ij} + A_{ji}) P_{ij}(\beta)| = -2(A_{ij} + A_{ji})(s_i - s_j) e^{-2\beta(s_i - s_j)} P_{ij}^2 \times \text{sign}\{A_{ij} - (A_{ij} + A_{ji}) P_{ij}(\beta)\} . \quad (S41)$$

Expanding $P_{ij}^2$ and reorganizing yields

$$\frac{\partial}{\partial \beta} |A_{ij} - (A_{ij} + A_{ji}) P_{ij}(\beta)| = -2(A_{ij} + A_{ji})(s_i - s_j) \frac{2}{\cosh[2\beta(s_i - s_j)] + 2} \times \text{sign}\{A_{ij} - (A_{ij} + A_{ji}) P_{ij}(\beta)\} . \quad (S42)$$

Combining terms $(i, j)$ and $(j, i)$, the optimal inverse temperature for local accuracy $\hat{\beta}_a$ is defined by

$$0 = \sum_{(i,j) \in U(E)} \frac{(A_{ij} + A_{ji})(s_i - s_j)}{\cosh[2\hat{\beta}_a(s_i - s_j)] + 1} \times \text{sign}\{A_{ij} - (A_{ij} + A_{ji}) P_{ij}(\hat{\beta}_a)\} , \quad (S43)$$

which may be solved using standard root-finding codes.

B. Choosing $\beta$ to optimize the conditional log likelihood

We take the derivative of (11) with respect to $\beta$, set it equal to zero, and partially solve as follows.

$$0 = \frac{\partial \sigma_L(\beta)}{\partial \beta} = \frac{\partial}{\partial \beta} \left[ \sum_{i,j} \log \left( \frac{A_{ij} + A_{ji}}{A_{ij}} \right) + \log \left[ P_{ij}(\beta)^{A_{ij}} [1 - P_{ij}(\beta)]^{A_{ji}} \right] \right] . \quad (S44)$$

Combining the $(i,j)$ and $(j,i)$ terms, we get

$$0 = \frac{\partial}{\partial \beta} \sum_{(i,j) \in U(E)} \log \left( \frac{A_{ij} + A_{ji}}{A_{ij}} \right) + \log \left( \frac{A_{ij} + A_{ji}}{A_{ji}} \right) + [A_{ij} \log P_{ij}(\beta) + A_{ji} \log [1 - P_{ij}(\beta)]]$$

$$= \sum_{(i,j) \in U(E)} \left[ \frac{A_{ij}}{P_{ij}(\beta)} - \frac{A_{ji}}{1 - P_{ij}(\beta)} \right] \frac{\partial P_{ij}(\beta)}{\partial \beta}$$

$$= \sum_{(i,j) \in U(E)} 2(s_i - s_j) [A_{ij} - (A_{ij} + A_{ji}) P_{ij}(\beta)] \frac{P_{ij}(\beta)}{1 - P_{ij}(\beta)} e^{-2\beta(s_i - s_j)} . \quad (S45)$$
Applying both (S35) and (S36), the optimal inverse temperature for the conditional log likelihood $\beta^\sigma L$ is defined by

$$0 = \sum_{(i,j) \in U(E)} 2 (s_i - s_j) \left[ A_{ij} - (A_{ij} + A_{ji}) P_{ij}(\beta^\sigma L) \right]$$  \hspace{1cm} (S46)

Comparing equations (S43) and (S46), we can see that the $\beta$ maximizing the two measures in general might be different.

S8. Coarsened (bit-wise) accuracy $\sigma_b$

Some methods provide rankings but do not provide a model to estimate $P_{ij}$, meaning that (10) and (11) cannot be used. Nevertheless, such methods still estimate one bit of information about each pair $(i, j)$: whether the majority of the edges are from $i$ to $j$ or vice versa. This motivates the use of a bit-wise or coarsened version of $\sigma_a$, which we call $\sigma_b$,

$$\sigma_b = 1 - \frac{1}{n^2 - t} \sum_{i,j} \Theta(s_i - s_j) \Theta(A_{ji} - A_{ij})$$  \hspace{1cm} (S47)

where $\Theta(x) = 1$ if $x > 0$ and $\Theta(x) = 0$ otherwise, and $n$ is the number of nodes and $t$ is the number of instances in which $A_{ij} = A_{ji}$; there are $n^2 - t$ total bits to predict. Results in terms of this measure on the networks considered in the main text are shown in Figure S1.

S9. Parameters used for regularizing ranking methods

During comparison of SpringRank to other methods, some methods were unable to return a valid output for certain data types. Eigenvector Centrality cannot, for example, be applied to directed trees, yet this is precisely the sort of structure that one might expect when hierarchy becomes extreme. Therefore, in our comparisons, we performed small amounts of regularization, described below.

A. Regularized Eigenvector Centrality

When regularization is necessary for Eigenvector Centrality, we use Katz centrality instead, so that

$$s_i^{KC} = \alpha \sum_{i,j} A_{ij} s_j^{KC} + b.$$  \hspace{1cm} (S48)

Here we chose $b = 1$ and attenuation factor $\alpha = 0.5/\lambda_{\text{max}}$, where $\lambda_{\text{max}}$ is the maximum eigenvalue of the matrix $A$.

B. Regularized PageRank

Nodes that have no out degree are sometimes called “dangling nodes.” We add an edge of weight $1/N$ from each dangling node to any other node in the network.

C. Regularized Bradley-Terry-Luce (BTL)

The Minimization-Maximization algorithm to fit the BTL model to real data proposed in [29] requires a regularization in the presence of dangling nodes. In this case we set the total number of out-edges $d_i^{\text{out}} = 10^{-6}$ for nodes that would have $d_i = 0$ otherwise. This corresponds to $W_i$ in Eq.(3) of [29].
FIG. S1. **One-bit edge direction prediction.** Symbols show averages of one-bit edge prediction accuracies (S47) over 50 realization of 5-fold cross validation (for a total of 250 trials), compared with the same accuracy for BTL; error bars indicate quartiles. Thus, points above the dashed line at zero indicate better predictions than BTL, while values below indicate that BTL performed better.

FIG. S2. **Null model distribution of energies.** Results are 1000 realizations of the null model where edge directions are randomized, for real and synthetic networks: a-c) US History (HS), Business (BS) and Computer Science (CS) departments faculty hiring networks [2]; d-e) social support networks of two Indian villages [30] considering 5 types of interactions (see main manuscript); f,g) aggression network of parakeet Group 1 and 2 (as in [4]); h,i) planted network using SpringRank generative model with \( N = 100, (k) = 5 \), Gaussian prior for the ranks with average \( \mu = 0.5 \) and variance \( \sigma ^{2} = 1 / (\alpha (\beta)) \) and two noise levels \( \beta = 5.0 \) and \( \beta = 0.1 \); j) dominance network of asian elephants [28]; k) Erdős-Rényi directed random network with \( N = 100 \) and \( (k) = 3 \). The vertical line is the energy obtained on the real network. In the last two cases we cannot reject the null hypothesis.
FIG. S3. Comparing goodness-of-fit measures We show the histograms of accuracy penalties in terms of σ_a (10) obtained on an instance of a planted network built using the SpringRank generative model with N = 100, ⟨k⟩ = 5, β = 1 and Gaussian factorized priors for the ranks with average 0.5 and variance 1. Colors referred to binning the errors by the rank difference between source and target nodes of the corresponding edges (the accuracy is the sum over undirected edges of the penalty on an edge). Results are obtained using two different inferred optimal β: a) ˆβ_a, the accuracy, i.e. optimizing σ_a; b) ˆβ_l, the conditional log likelihood, i.e. optimizing σ_L. We can notice that optimizing in terms of the accuracy (10) leads to a smaller number of errors but with higher impact. Vice versa, optimizing in terms of the conditional log likelihood (11) leads to a bigger amounts of errors but each having less impact.
S10. Supplemental Tables

| DataSet                | Type          | N  | M    | H/m  | Acc. $\sigma_a$ | $\beta_L$ | $\beta_a$ | Viol. (%) / Bound | Wt. viol. (per viol.) | Depth $p$-value |
|------------------------|---------------|----|------|------|----------------|-----------|-----------|-------------------|---------------------|-----------------|
| Parakeet G1 [4]        | Anim. Dom.    | 21 | 838  | 0.174| 0.930          | 2.70      | 6.03      | 76 (9.1%) / 42    | 0.008 (0.089)      | 2.604 $< 10^{-4}$|
| Parakeet G2 [4]        | Anim. Dom.    | 19 | 961  | 0.193| 0.932          | 2.78      | 18.12     | 75 (7.8%) / 36    | 0.011 (0.139)      | 1.879 $< 10^{-4}$|
| Asian Elephants [28]   | Anim. Dom.    | 20 | 23   | 0.078| 0.923          | 2.33      | 3.44      | 2 (8.7%) / 0      | 0.001 (0.040)      | 3.000 0.4466     |
| Business [2]           | Fac. Hiring   | 112| 7353 | 0.251| 0.881          | 2.04      | 3.14      | 1171 (15.9%) / 808| 0.019 (0.119)      | 2.125 $< 10^{-4}$|
| Computer Science [2]   | Fac. Hiring   | 205| 4033 | 0.220| 0.882          | 2.23      | 8.74      | 516 (12.8%) / 255 | 0.013 (0.105)      | 2.423 $< 10^{-4}$|
| History [2]            | Fac. Hiring   | 144| 3921 | 0.186| 0.909          | 2.39      | 5.74      | 397 (10.1%) / 227 | 0.012 (0.119)      | 2.234 $< 10^{-4}$|
| Alakăpuram [30]        | Soc. Support  | 415| 2497 | 0.222| 0.867          | 1.98      | 7.95      | 347 (13.9%) / 120 | 0.011 (0.079)      | 3.618 $< 10^{-4}$|
| Tenpahti [30]          | Soc. Support  | 361| 1809 | 0.241| 0.858          | 1.89      | 8.20      | 262 (14.5%) / 120 | 0.012 (0.082)      | 3.749 $< 10^{-4}$|

**TABLE S1. Statistics for SpringRank applied to real-world networks.** Column details are as follows: $N$ is the number of nodes; $M$ is the number of edges; $H/m$ is the ground-state energy per edge; Accuracy $\sigma_a$ refers to accuracy in 5-fold cross validation tests using temperature $\hat{\beta}_a$; $\hat{\beta}_L$ and $\hat{\beta}_a$ are temperatures optimizing edge prediction accuracies $\sigma_L$ and $\sigma_a$ respectively; Violations refers to the number of edges that violate the direction of the hierarchy as a number, as a percentage of all edges, with a lower bound provided for reference, computed as the number of unavoidable violations due to reciprocated edges; Weighted violations are the sum of each violation weighted by the difference in ranks between the offending nodes; Depth is $s_{max} - s_{min}$; $p$-value refers to the null model described in the Materials and Methods.

S11. Supplemental Figures

**FIG. S4. Performance (Pearson correlation) on synthetic data.** Tests were performed as in Fig. 1, but here performance is measured using Pearson correlation instead of Spearman rank correlation, and thus unfairly favors SpringRank and BTL—ordinal ranking schemes like Minimum Violation Ranking are not expected to recover latent positions. (A) Linear hierarchy diagrams show latent ranks $s_{planted}$ of 100 nodes, drawn from a standard normal distribution, with edges drawn via the generative model (6) for indicated $\beta$ (noise) values. Blue edges point down the hierarchy and red edges point up, indicated by arrows. (B) Mean accuracies ± one standard deviation (symbols ± shading) are measured as the Pearson correlation between method output and $s_{planted}$ for 100 replicates. (C, D) Identical to A and B but for hierarchies of $N = 102$ nodes divided into three tiers. All plots show $\langle k \rangle = 5$; see Fig. 1 for performance curves for Spearman correlation $r$. See Materials and Methods for synthetic network generation.
FIG. S5. Summary of SpringRank applied to Computer Science faculty hiring network [2]. (left) A linear hierarchy diagram showing inferred SpringRank scores. Blue edges point down the hierarchy and red edges point up. The histogram shows the empirical distribution of ranks: the y axis is the rank $s_i$ (binned) and the x axis is the count of nodes having a rank in that bin. (right top) sparsity plot of rank-ordered adjacency matrix; blue and red dots represent non-zero entries going down and up the hierarchy, respectively. (right middle) Results of statistical significance test with randomized edge directions. The histogram represents the energies obtained in the randomized samples: the dotted line is the ground energy obtained on the observed real network. (bottom): universities ordered by rank, with top rank on the left and bottom rank on the right, shaded by tier. The tiers are calculated by the k-means algorithm.
FIG. S6. Summary of SpringRank applied to History faculty hiring network [2]. (left) A linear hierarchy diagram showing inferred SpringRank scores. Blue edges point down the hierarchy and red edges point up. The histogram shows the empirical distribution of ranks: the y-axis is the rank $s_j$ (binned) and the x-axis is the count of nodes having a rank in that bin. (right top) sparsity plot of rank-ordered adjacency matrix; blue and red dots represent non-zero entries going down and up the hierarchy, respectively. (right middle) Results of statistical significance test with randomized edge directions. The histogram represents the energies obtained in the randomized samples: the dotted line is the ground energy obtained on the observed real network. (bottom): universities ordered by rank, with top rank on the left and bottom rank on the right, shaded by tier. The tiers are calculated by the k-means algorithm.
FIG. S7. Summary of SpringRank applied to Business faculty hiring network [2]. (left) A linear hierarchy diagram showing inferred SpringRank scores. Blue edges point down the hierarchy and red edges point up. The histogram shows the empirical distribution of ranks: the $y$ axis is the rank $s_j$ (binned) and the $x$ axis is the count of nodes having a rank in that bin. (right top) Sparsity plot of rank-ordered adjacency matrix; blue and red dots represent non-zero entries going down and up the hierarchy, respectively. (right middle) Results of statistical significance test with randomized edge directions. The histogram (right top) sparsity plot of rank-ordered adjacency matrix; blue and red dots represent non-zero entries going down and up the hierarchy, respectively. (right middle) Results of statistical significance test with randomized edge directions. The histogram shows the energies obtained in the randomized samples: the dotted line is the ground energy obtained on the observed real network. (bottom): universities ordered by rank, with top rank on the left and bottom rank on the right, shaded by tier. The tiers are calculated by the $k$-means algorithm.
FIG. S8. Summary of SpringRank applied to Asian Elephants network [28]. (left) A linear hierarchy diagram showing inferred SpringRank scores. Blue edges point down the hierarchy and red edges point up. The histogram shows the empirical distribution of ranks: the y axis is the rank $s_i$ (binned) and the x axis is the count of nodes having a rank in that bin. (right top) sparsity plot of rank-ordered adjacency matrix; blue and red dots represent non-zero entries going down and up the hierarchy, respectively. (right middle) Results of statistical significance test with randomized edge directions. The histogram represents the energies obtained in the randomized samples: the dotted line is the ground energy obtained on the observed real network. (bottom): universities ordered by rank, with top rank on the left and bottom rank on the right, shaded by tier. The tiers are calculated by the $k$-means algorithm.
FIG. S9. Summary of SpringRank applied to Parakeet G1 network [4]. (left) A linear hierarchy diagram showing inferred SpringRank scores. Blue edges point down the hierarchy and red edges point up. The histogram shows the empirical distribution of ranks: the $y$ axis is the rank $s_i$ (binned) and the $x$ axis is the count of nodes having a rank in that bin. (right top) sparsity plot of rank-ordered adjacency matrix; blue and red dots represent non-zero entries going down and up the hierarchy, respectively. (right middle) Results of statistical significance test with randomized edge directions. The histogram represents the energies obtained in the randomized samples: the dotted line is the ground energy obtained on the observed real network. (bottom): universities ordered by rank, with top rank on the left and bottom rank on the right, shaded by tier. The tiers are calculated by the $k$-means algorithm.
FIG. S10. **Summary of SpringRank applied to Parakeet G2 network** [4]. (left) A linear hierarchy diagram showing inferred SpringRank scores. Blue edges point down the hierarchy and red edges point up. The histogram shows the empirical distribution of ranks: the $y$ axis is the rank $s_i$ (binned) and the $x$ axis is the count of nodes having a rank in that bin. (right top) sparsity plot of rank-ordered adjacency matrix; blue and red dots represent non-zero entries going down and up the hierarchy, respectively. (right middle) Results of statistical significance test with randomized edge directions. The histogram represents the energies obtained in the randomized samples: the dotted line is the ground energy obtained on the observed real network. (bottom): universities ordered by rank, with top rank on the left and bottom rank on the right, shaded by tier. The tiers are calculated by the $k$-means algorithm.
FIG. S11. **Summary of SpringRank applied to Tenpaṭṭi social support network** [30]. (left) A linear hierarchy diagram showing inferred SpringRank scores. Blue edges point down the hierarchy and red edges point up. The histogram shows the empirical distribution of ranks: the $y$ axis is the rank $s_i$ (binned) and the $x$ axis is the count of nodes having a rank in that bin. (right top) sparsity plot of rank-ordered adjacency matrix; blue and red dots represent non-zero entries going down and up the hierarchy, respectively. (right middle) Results of statistical significance test with randomized edge directions. The histogram represents the energies obtained in the randomized samples: the dotted line is the ground energy obtained on the observed real network. (bottom): universities ordered by rank, with top rank on the left and bottom rank on the right, shaded by tier. The tiers are calculated by the $k$-means algorithm.
FIG. S12. Summary of SpringRank applied to Alakāpuram social support network [30]. (left) A linear hierarchy diagram showing inferred SpringRank scores. Blue edges point down the hierarchy and red edges point up. The histogram shows the empirical distribution of ranks: the $y$ axis is the rank $s_i$ (binned) and the $x$ axis is the count of nodes having a rank in that bin. (right top) Sparsity plot of rank-ordered adjacency matrix; blue and red dots represent non-zero entries going down and up the hierarchy, respectively. (right middle) Results of statistical significance test with randomized edge directions. The histogram represents the energies obtained in the randomized samples: the dotted line is the ground energy obtained on the observed real network. (bottom): universities ordered by rank, with top rank on the left and bottom rank on the right, shaded by tier. The tiers are calculated by the $k$-means algorithm.