

Using a co-occurrence network

Jim Brunner

June 28, 2017

1 Network Building

We are looking at creating ecological networks of a microbiome. Right now, I have built two networks, in the form of graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$. In both graphs, vertex are labeled by taxa name. I'm going to conflate the vertices and their label. The edge sets are defined by co-incidence and co-occurrence, respectively. Given a set of samples with abundances of organisms, we map the abundances to discrete levels, as proportions of the maximum abundance *of that organism*. Precisely, let the samples be s_i and raw abundance of organism j in sample i be r_{ji} . I map the abundances according to

$$a(r_{ji}) = \begin{cases} \lfloor \left(\frac{r_{ji}}{\max_{s_k}(r_{jk})} \right) n \rfloor + 1 & \frac{r_{ji}}{\max_{s_k}(r_{jk})} \geq m \\ 0 & \frac{r_{ji}}{\max_{s_k}(r_{jk})} < m \end{cases}$$

where m is some minimum. Then, I create weighted edges between vertices (where 0 weight means no edge exists) where the weights of edges in \mathcal{E}_1 are

$$w_{jk}^1 = \frac{\|\{i : a(r_{ji}) = a(r_{ki}) \neq 0\}\|}{S}$$

where S is the total number of samples. That is, we count the propotion of samples in which the two taxa appear at the same discretized level.

The second network accounts for random coincidence of taxa in a sample, following [3]. It begins with \mathcal{G}_1 , and compares to a null model N . The null model is defined in the following way.

$$A_j = \sum_{s_i} \mathbf{1}_{a(r_{ji}) \neq 0}$$

and

$$S_i^l = \sum_{v_j} \mathbf{1}_{a(r_{ji})=l}$$

Then, N assumes that if

$$X_{jil} \sim \text{binom} \left(A_j, \frac{S_i^l}{\sum_{il} S_i^l} \right)$$

then $P(a_{ji}^N = l) = 1 - P(X_{jil} = 0)$. This allows us to calculate the probability of co-incidence of taxa under the null model. Let w_{jk}^N be

$$w_{jk}^N = \|\{i : a_{ji}^N = a_{ki}^N \neq 0\}\|$$

This is the similar to the co-incidence model but now randomized. Then,

$$P(w_{jk}^N = K) = \sum_{\{A \subset \mathcal{V}_2 : |A|=K\}} \prod_{l \in A} a_{jl} a_{kl} \prod_{l \notin A} (1 - a_{jl} a_{kl})$$

Ideally, we would then define \mathcal{E}_2 by the weights

$$w_{jk}^2 = \begin{cases} 1 & P(w_{jk}^N \geq w_{jk}^1) \leq t \\ 0 & P(w_{jk}^N < w_{jk}^1) > t \end{cases}$$

However, that probability is intractible to compute. Instead, we take

$$\tilde{P}(w_{jk}^N = K) = \sum_{l=0}^i \binom{N_1}{l} \binom{N_2}{K-l} p_1^l p_2^{K-l} (1-p_1)^{N-l} (1-p_2)^{N-K+l}$$

where

$$p_1 = p_a - \left(\frac{N_2}{N_1} \frac{N(\mu - \sigma^2) - \mu^2}{N^2} \right)^{1/2}$$

and

$$p_2 = p_a - \left(\frac{N_1}{N_2} \frac{N(\mu - \sigma^2) - \mu^2}{N^2} \right)^{1/2}$$

Finally, N_1, N_2 are to ensure that $p_1, p_2 \in [0, 1]$. It turns out we need:

$$\frac{\mu N(1 - p_a) - N\sigma^2}{N(1 - p_a) - \sigma^2} \leq N_2 \leq \frac{\mu^2}{\mu - \sigma^2}$$

with μ the mean of the real distribution, σ^2 the variance, and $p_a = \frac{1}{S} \sum_i a_{ji} a_{ki}$. So, we take

$$w_{jk}^2 = \begin{cases} 1 & \tilde{P}(w_{jk}^N \geq w_{jk}^1) \geq t \\ 0 & \tilde{P}(w_{jk}^N < w_{jk}^1) > t \end{cases}$$

1.1 An alternative coincidence construction

A faster way would be to first normalize abundance vectors, and just take

$$W = AA^T$$

(with the diagonal reset to 0) to be the weighted adjacency matrix, where A is the matrix of abundances. Then, each entry is the cosine of the angle between the abundance vectors. This also allows us to look at negatively aligned abundance vectors.

How then would we construct the cooccurrence network? Basically, what would our null model be? Well, we would still randomize the sample data. That is, we could take X_{ij} be a random variable representing the abundance of taxa i in sample j , and require that $\|X_i\| = 1$. Then, we can ask $P(\mathbf{X}_i \cdot \mathbf{X}_k > w_{ik})$. I just don't know what distribution to put on the X_{ij} (the random abundances). I'll start with a binomial:

$$P(\tilde{X}_{ij} = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

where $p = \frac{\deg(s_j)}{\sum_l \deg(s_l)}$ and $n = \lfloor \frac{1}{\min_k(r_{ik})} r_i \rfloor$ where $r_i = \sum_k r_{ik}$. Then I normalize to the unit sphere. Finally, I can use a Monte-Carlo simulation to see which weights to check. To do this, I generate random matrices with the appropriate distributions, using a built in Numpy method. Then I check the proportion of random entries larger than the weights. If this is small, I keep the edge. Unfortunately, I can't use a control variate or otherwise try to reduce variance, because generating my own samples was far slower than the Numpy method. The Numpy method is pre-compiled in C. Variance was ~ 0.25 , so 1000 samples should be enough.

Similar to the above but maybe on more established footing is Pearson's correlation coefficient. That is:

$$\rho_{xy} = \frac{1}{N} \frac{(\mathbf{x} - \mu_x \mathbf{1}) \cdot (\mathbf{y} - \mu_y \mathbf{1})}{\sigma_x \sigma_y}$$

which can of course be constructed by modifying the data matrix and then multiplying with it's transpose. In expectation, this the covariance divided by the product of the variances. I can also use a Monte-Carlo approach to check significance. This ends up keeping a lot more edges than the binning procedure. I am considering doing a first pass of only keeping high weights. Low weights may be significant (i.e. higher than one could expect randomly) but are still low, and so maybe should be dropped. In the literature, people tend to only keep weights over 0.8 or something.

2 Network Analysis

We can cluster - community clustering, spectral clustering, comparing to random graphs. Comparison of clusters to grouping by sample type.

Community clustering minimizes a function of the graph called modularity [1][4]. That is

$$Q = \frac{1}{2m} \sum_{i,j} \left(w_{ij} + \frac{d_i d_j}{2m} \right) \delta(c_i, c_j)$$

where $m = \sum_{i \sim j} w_{ij}$, d_i is the (weighted) degree of vertex i , c_i is the community containing vertex i , and δ is the Kronecker δ . This done by a kind of gradient search/greedy algorithm.

Spectral clustering performs a k -means clustering on the rows of the matrix whose columns are the k eigenvectors of the graph laplacian corresponding to the smallest k eigenvalues [6]. Morally, this means it clusters points together that are close in the first k (slowest decaying!) modes of the diffusion equation on the graph.

3 Using the network to analyze a sample

The question now is what can we do these networks?

First, assessing GOTTCHA reads. A single GOTTCHA read would produce a network with each connected component complete. I think the co-incidence network is more appropriate. We want to assess the probability that you see this set together. We have the probability that you see any vertex pair (estimated) as the edge weights of \mathcal{G}_1 . Precisely, the edge weights are

$$w_{jk}^1 = P(j \& k \in S_i^l)$$

where S_i^l is sample i at discrete abundance level l . It might be useful to have the directed weight graph where

$$w_{jk}^3 = P(j \in S_i^l | k \in S_i^l)$$

but that wouldn't be hard, because then

$$w_{jk}^3 = S \frac{w_{jk}^1}{\|\{i : a(r_{ki}) > 0\}\|}$$

Anyway, let's start with the simplest case of one abundance level. Assume GOTTCHA found taxa a, b, c, \dots, n . Maybe the first thing we would want is

$$P(a|b, c, d, \dots, n), P(b|a, c, d, \dots, n), \text{ etc}$$

Clearly, we can see directly $P(a|b)$, etc. We can also get a bound for triplets (assuming $P(c, b) \neq 0$):

$$P(a|b, c) \leq \frac{\min_{(i,j) \subset \{a,b,c\}} (P(i, j))}{P(b, c)}$$

but we can't do any better than triplets explicitly, because we don't have any sort of independence (conditional or otherwise) and because our network is not acyclic.

We can probably learn something from asking about the connectivity of the induced subgraph. Notice that if it isn't complete, then one of the $P(a, b)$ is 0 above.

What does the connectivity of the induced subgraph of \mathcal{G}_2 tell us? If it is connected, that's good. We could use that network to identify vertices that are connected to many of the vertices in the induced subgraph - this might indicate that node should be in the sample.

I guess \mathcal{G}_2 is a markov random field [2]. This might give us a way to calculate the probability you see a group taxa (and maybe others). The main idea of a MRF is that nodes are conditionally independent of nodes they aren't neighbors of (conditioned on ones they are neighbors of). If c are the (maximal) cliques of the graph (complete subgraphs), then the probability of configuration \mathbf{x} is

$$P(\mathbf{x}) = \frac{1}{Z} \prod_c \psi_c(x_c)$$

where Z is a normalizing constant and ψ_c are potential functions I don't know how to come up with yet. Anyway, if we have a sample that contains (maybe as a subset) s , we can calculate something. Let C_s be the cliques represented in s .

$$P(s) = \frac{1}{Z} \sum_{\{\mathbf{x}: s \subset \mathbf{x}\}} \prod_{c \in C_s} \psi_c(\mathbf{x})$$

And we can ignore cliques not represented in s . We probably have to change Z . I guess we can also use neighbor pairs instead of maximal cliques. Either way we have to figure out what ψ_c are.

3.1 Determining ψ_c

Before figuring out ψ_c , it should be noted that we have a choice of configuration space of the network. We can use a binary $\{1, 0\}^N$ space to denote presence or absence, or we can choose a continuous space to include abundances. To begin, I will only consider presence & absence.

Snowshoe hairs and Canadian Lynx, CRNT

Now I'm very tempted to think about how this approach relates to mass action dynamical systems. I'm going to think about my very favorite model.

$$\dot{\mathbf{x}} = \kappa_1 x \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \kappa_2 xy \begin{pmatrix} -1 \\ 1 \end{pmatrix} + \kappa_3 y \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

Well. Yes. But there is only one maximal clique so $P(\mathbf{x}) = \psi(\mathbf{x})$ and that's just whatever it is. We know that it evolves according to the stochastic mass action equations. In fact, the result that the stationary distribution is a product of poissons for a complex balanced system can be thought of as fitting into this framework. I wonder if you could use this theory to re-prove that? Recall that the stationary distribution for a complex balanced system with equilibrium \mathbf{c} is

$$\pi(\mathbf{x}) = \frac{1}{Z} \prod_{i=1}^d \frac{c_i^{x_i}}{x_i!}$$

so here we have that

$$\psi_c = \prod_{i \in c} \frac{c_i^{x_i}}{x_i!}$$

or taking cliques to be just singletons

$$\psi_i = \frac{c_i^{x_i}}{x_i!}$$

I can't think of any way to arrive upon that directly, and so prove the result from this direction. Also, This is even more general than a MRF, as it has self-loops. Self loops play the role of time update. In fact, one might say that a MC is to a MRF as an ODE is to a PDE. Then, self loops in the MRF correspond to time derivatives or forcing appearing in the PDE.

It seems reasonable to use a pairwise RMF. It also seems reasonable to take a log linear function. My initial guess is

$$\psi_{s \sim t}(y_s, y_t) = \exp(\boldsymbol{\theta} \cdot \mathbf{1}_{y_s=i, y_t=j})$$

where $\boldsymbol{\theta} = (0, \log(P(s)), \log(P(t)), \log(1/2(P(s|t) + P(t|s))))$. But that's a guess. It makes sense we have only unconnected nodes. It also also penalizes leaving out nodes that are connected to the nodes we do have. However, this penalizes us too harshly if we include a hub node.

Let's play with a small network to try to get a handle on this. Consider the network shown in fig. 1 I think we can assume that

$$p(\mathbf{x}) = \prod_{s \sim t} \psi_{st}(\mathbf{x}) \prod_s \psi_s(\mathbf{x})$$

To start, we should think about the distribution of an independent node (T_1). Clearly

$$P(T_1 = x) = \psi_1(x)$$

Similarly, we should have

$$P(T_2 = x) = \psi_2(x) = \int \psi_2(x) \psi_{23}(y) dy = \mathbb{E}(P(T_2 = x | T_3))$$

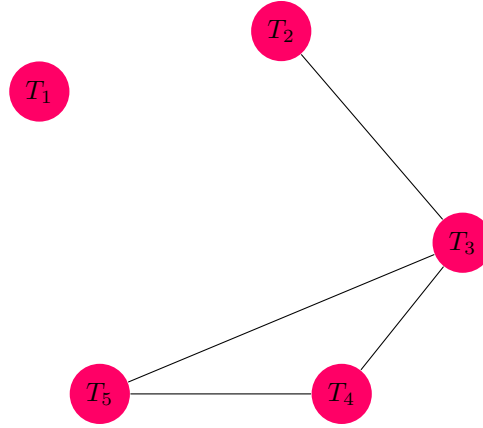


Figure 1: A small network example

Maybe a binomial is a good idea:

$$P\left(T_1 = \frac{k}{N}\right) = \binom{N}{k} p^k (1-p)^{N-k}$$

where p is something like the proportion of times T_1 appears in the data, the sum of abundances of T_1 in the data divided by N . N is some scaling parameter, like the largest total abundance or something. But then, it's natural to take N large and get a Gaussian by the CLT. So maybe we do a multivariate Gaussian. Then

$$\psi_{s \sim t}(\mathbf{x}) = \exp\left(-\frac{1}{2} \mathbf{x}_s \Lambda_{st} \mathbf{x}_t\right)$$

and

$$\psi_s(\mathbf{x}) = \exp(\eta_s x_s - \Lambda_{ss} x_s^2)$$

[2]. We already have the covariance (normalized by the product of the standard deviations) matrix Σ , in the Pearson coefficient. So, we have $\Lambda = \Sigma^{-1}$, and $\boldsymbol{\eta} = \Sigma^{-1} \boldsymbol{\mu}$. Unfortunately, Σ is singular, it's dimension is bounded by the number of samples I have. This of course leaves us in the wind when it comes to Λ . So, I suppose we have to try to approximate Λ subject to the constraint that if $s \not\sim t$, $\Lambda_{st} = 0$. We can use the SVD:

$$\Sigma = M \tilde{S} N^* \Rightarrow \Sigma^+ = N \tilde{S} M^*$$

where \tilde{S} is diagonal with $1/s_i$ until you get to zero singular values, where you just put a 0. Because Σ is symmetric, we have $M^* = N$. Then,

$$\Sigma^+ = N \tilde{S} N$$

I guess this is used, although it doesn't seem to preserve zeros. There's a function in scipy.

On the other hand, taking N large and p small we get an exponential random variable. So perhaps I want

$$\psi_1(x) = e^{-\lambda} \frac{\lambda^{Nx}}{(Nx)!}$$

or more directly

$$\psi_1(x) = \binom{N}{Nx} p^{Nx} (1-p)^{N-Nx}$$

and using Stirling's formula,

$$\log(\psi_1) \approx N \left[x \log\left(\frac{p}{x}\right) + (1-x) \log\left(\frac{1-p}{1-x}\right) \right]$$

3.2 Comparing configurations without specifying ψ

Maybe I don't need to figure out ψ_{st} and ψ_s . Maybe I can conclude something like

$$\frac{P(\mathbf{x})}{P(\mathbf{y})} > 1$$

without knowing them. For that, I would need to just identify which cliques change, and ask whether they increase or decrease. How exactly to decide that is a question. Probably single nodes we wouldn't change. For a clique, we could assume that ψ_c decreases if we have over half the nodes and reduce and still have at least half, but increase if the we have less than half and decrease. That is, let c be a clique with n members, and $\psi_c(k)$ be the value of ψ_c with k members "on". Then

$$\frac{\psi_c(k+1)}{\psi_c(k)} \begin{cases} > 1 & k > n/2 \\ < 1 & k \leq n/2 \end{cases}$$

Then, we are basically asserting the cliques like to either be all there or none there. Which seems good.

Here the algorithm for deciding which of two nodes should be on and which should be off would be:

- (i) Identify cliques c_1, \dots, c_m which contain one or both nodes
- (ii) compute $\frac{\psi_{c_i}(k^1)}{\psi_{c_i}(k^2)}$ for each $i \in \{1, \dots, m\}$, where k^j is the number of members present in configuration $j \in \{1, 2\}$, and this fraction is computed according to a simple rule of the kind above.
- (iii) multiply these ratios: $\frac{P(\text{configuration 1})}{P(\text{configuration 2})} = \prod_{i=1}^m \frac{\psi_{c_i}(k^1)}{\psi_{c_i}(k^2)}$

I need to specify that rule. The simplest reasonable thing is probably linear

$$\frac{\psi_c(k+1)}{\psi_c(k)} = \frac{2(1 - r_{min})}{n-1}k + r_{min}$$

but one could imagine some kind of sigmoidal rule as well, so that around $n/2$ this ratio is close to 1. This makes sense on a 2 clique, as it implies that

$$\psi(2) > \psi(1) \ \& \ \psi(0) > \psi(1)$$

and in general makes sense for n even. It makes sense for n odd as well. It asserts that it is equally likely to have on $n/2 \pm 1/2$. Finally, if we need to compare k and $k+2$, we simply take

$$\frac{\psi(k+2)}{\psi(k)} = \frac{\psi(k+2)}{\psi(k+1)} \frac{\psi(k+1)}{\psi(k)}$$

That is, a discrete analogue to the chain rule.

3.3 Diffusion based method.

Here's an idea inspired by spectral clustering: Solve the diffusion equation on the graph:

$$\frac{\partial}{\partial t} u(v, t) = Lu(v, t)$$

where v takes values in the vertex set of the graph. Put the initial mass on nodes you are confident in or something like that. Then, compare

$$U(v) = \int_0^\infty u(v, t) dt$$

Higher should be more likely.

Conjecture 1. Consider a random markov field with configurations in $\{0, 1\}^N$. Assume that $P(v_i = 1 | v_j = 1) \geq P(v_i = 1)$ and $P(v_i = 1 | v_j = 0) \leq P(v_i = 1)$, with strict inequality if there exists an edge connecting v_i and v_j .

Let $u_i(t)$ be the solution at node v_i to the discete diffusion problem

$$\frac{d}{dt} \mathbf{u}(t) = -L\mathbf{u}$$

where L is the graph laplacian with initial conditions given by a known set of nodes $\{v_l\}$. Suppose $v_1, v_2 \notin \{v_l\}$ (so the initial vector has support on the known nodes).

Then,

$$\int_0^\infty u_1(t) dt - \int_0^\infty u_2(t) dt > 0 \Leftrightarrow P(v_1 = 1 | \{v_l\}) > P(v_2 = 1 | \{v_l\})$$

If this is true we can very easily compute these comparisons. Solutions are

$$\mathbf{u} = \sum_{i=1}^a c_i \mathbf{1} + \sum_{i=a+1}^n c_i e^{\lambda_i t} \mathbf{v}_i$$

where a is the number of connected components of the graph. Note that each eigenvalue $\lambda_i \leq 0$, with $\lambda_i < 0$ for $i = a + 1, \dots, n$ [6]. Then, assuming there is some initial mass on the connected components containing v_1 and v_2 ,

$$\int_0^\infty u_1(t) dt - \int_0^\infty u_2(t) dt = \sum_{i=a}^n c_i (v_{i1} - v_{i2}) \int_0^\infty e^{\lambda_i t} dt = \sum_{i=a}^n \frac{-c_i}{\lambda_i} (v_{i1} - v_{i2})$$

and $\mathbf{c} = V^{-1} \mathbf{u}(0)$.

Alternatively:

Conjecture 2. Consider a random markov field with configurations in $\{0, 1\}^N$. Assume that $P(v_i = 1 | v_j = 1) \geq P(v_i = 1)$ and $P(v_i = 1 | v_j = 0) \leq P(v_i = 1)$, with strict inequality if there exists an edge connecting v_i and v_j . Suppose that m node values are known. Let $u_i(t)$ be the the solution to

$$\frac{d}{dt} \mathbf{u}(t) = -L \mathbf{u} + \mathbf{f}$$

where L is the graph Laplacian, we have initial conditions 0 for unknown nodes and $\mathbf{f} \in \mathbf{1}^\perp$ has support at known nodes (positive at known “on” nodes, and negative at known “off” nodes). Then,

$$\lim_{t \rightarrow \infty} u_i(t) > \lim_{t \rightarrow \infty} u_j(t) \Leftrightarrow P(v_i = 1 | \text{known node values}) > P(v_j = 1 | \text{known node values})$$

And the last version is the boundary value version:

Conjecture 3. Consider a random markov field with configurations in $\{0, 1\}^N$. Assume that $P(v_i = 1 | v_j = 1) \geq P(v_i = 1)$ and $P(v_i = 1 | v_j = 0) \leq P(v_i = 1)$, with strict inequality if there exists an edge connecting v_i and v_j . Suppose that m node values are known. Let $u_i(t)$ be the the solution to

$$\frac{d}{dt} \mathbf{u}(t) = -L|_{\text{unknown}} \mathbf{u} + \mathbf{b}$$

where L is the graph Laplacian with rows and columns corresponding to known nodes deleted, and \mathbf{b} is the “forcing” due to boundary conditions. Then,

$$\lim_{t \rightarrow \infty} u_i(t) > \lim_{t \rightarrow \infty} u_j(t) \Leftrightarrow P(v_i = 1 | \text{known node values}) > P(v_j = 1 | \text{known node values})$$

For a good answer, both the second two I think need the known nodes vector to live in $\mathbf{1}^\perp$. The first idea works, where we probably want to put -1 on known 0 nodes (which would morally correspond to the assumption that $P(v_i = 1 | v_j = 0) < P(v_i = 1)$ if i and j are connected).

We really just need the \Rightarrow direction. We need to decide on what assumptions on the various conditional probabilities make these true. Let me call the known set of node values ξ . To compare $P(v_1 = 1 | \xi)$ and $P(v_2 = 1 | \xi)$, let X be the configurations with ξ and $v_1 = 1$, and Y the configurations with ξ and $v_2 = 1$. Then,

$$P(v_1 = 1 | \xi) = \sum_{x \in X} \frac{P(x)}{P(\xi)}$$

and so

$$P(v_1 = 1 | \xi) > P(v_2 = 1 | \xi) \Leftrightarrow \sum_{x \in X \setminus Y} P(x) > \sum_{y \in Y \setminus X} P(y)$$

but inspecting X and Y , we see that this is equivalent to

$$\sum_{x \in X \setminus Y} (P(x) - P(\tilde{x})) > 0$$

where $\tilde{x} = x$ for all nodes except v_1 and v_2 , while $x(v_1) = \tilde{x}(v_2) = 1$ and $x(v_2) = \tilde{x}(v_1) = 0$. This can be written

$$\sum_{x \in X \setminus Y} \prod_{c: v_1, v_2 \notin c} \psi_c(x) \left(\prod_{c: v_1 \text{ or } v_2 \in c} \psi(x) - \prod_{c: v_1 \text{ or } v_2 \in c} \psi(\tilde{x}) \right) > 0 \quad (1)$$

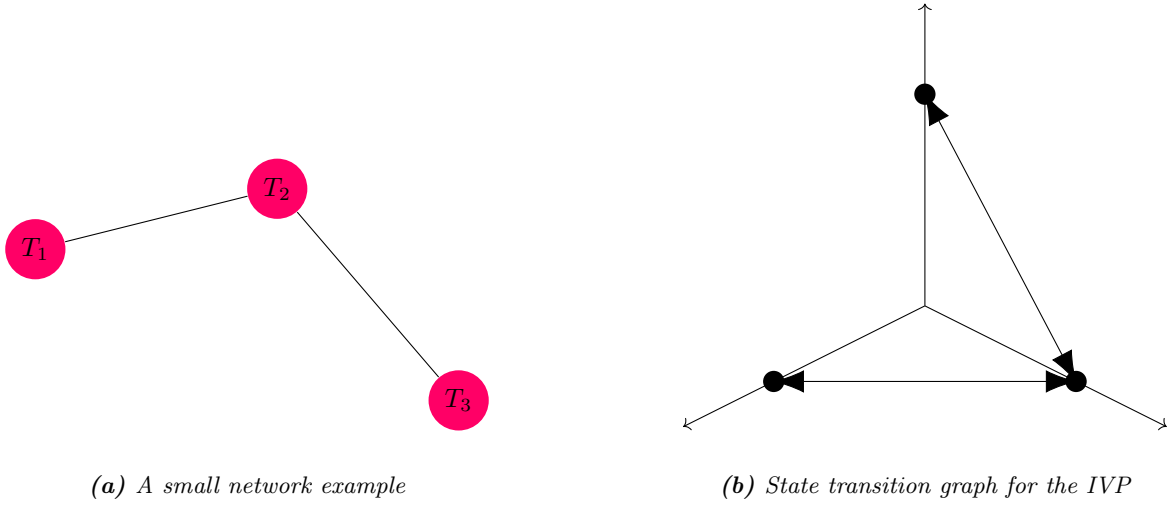


Figure 2: Diffusion is equivalent to the Kolmogorov forward equations for the above system.

where c are maximal cliques of the RMF.

We can regard the initial value problem version as the Kolmogorov forward equation of a jump process that transitions from taxa to taxa which have an edge between them at a linear rate (see fig. 2). This inspires one last version of this idea, similar to the forced idea, which is to use the forward equations for a jump process with inflow and outflow at known on and off nodes, respectively. However, that would be a lot more complicated because of the infinite state space. The forward equations would not be closed. We could deal directly with that model, I suppose, but that doesn't seem worth it.

I suppose I need to ask the question of whether or not the boundary value version will always have an equilibrium. That is a question about the kernel of L . A “boundary value” set on the graph is a set of fixed node values (probably $1, 0, -1$). Clearly, this defines an affine subspace, the span of the standard basis vectors in the unknown dimensions plus the specified point (putting 0 in the unknown dimensions). This doesn't have to intersect the kernel of L (which always includes $\mathbf{1}$). If it doesn't, there is not an equilibrium solution to the diffusion problem which gives those values at boundary nodes. However, this doesn't mean the boundary value problem (or, more precisely, the equivalent forced problem on the unknown subset) doesn't have an equilibrium solution. Take for example a complete graph on three nodes. If we prescribe one node as 1 and one as 0, the reduced problem is

$$\frac{d}{dt}u = -2u + 1$$

which has an equilibrium at $u = 1/2$. This is not an equilibrium to the entire problem, because in that case

$$-L \begin{pmatrix} 1 \\ 1/2 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{pmatrix} \begin{pmatrix} 1 \\ 1/2 \\ 0 \end{pmatrix} = \begin{pmatrix} -3/2 \\ 0 \\ 3/2 \end{pmatrix}$$

The nullity of L is the number of connected components of the graph. Removing a column from a matrix doesn't lower the rank of that matrix if the column was a linear combination of some others, which is the case precisely when the columns is the first one removed that corresponds to a connected component of the graph. Therefore, as long as we specify at least one “boundary value” from each connected component, the restricted matrix is non-singular and there exists a unique equilibrium to the boundary value problem.

The boundary value problem can be interpreted as the forward equations for the same process as the IVP, but with a “walker” added to the known “on” nodes whenever one leaves, and any walker who enters a known “off” node is removed from the problem. The result is not a probability distribution, but it can be scaled to be one.

3.3.1 Tests of these methods

I tested these three ideas on two small (unweighted) graphs, shown in fig. 3.

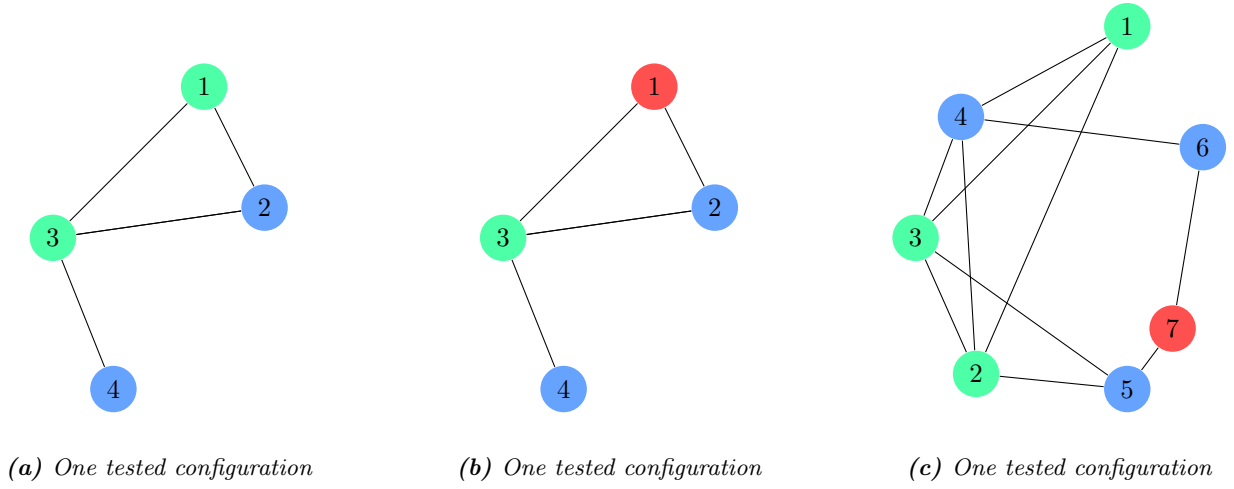


Figure 3: Test networks. Blue nodes were “unknown”, red nodes “off”, and green “on”.

Configuration	Method	Ranking	Ties
fig. 3(a)	IVP	2, 4	none
	BVP	4, 2	4, 2
	Forcing	2, 4	none
fig. 3(b)	IVP	4, 2	4, 2
	BVP	4, 2	none
	Forcing	4, 2	none
fig. 3(c)	IVP	4, 5, 6	none
	BVP	4, 5, 6	none
	Forcing	4, 5, 6	none

Table 1: Results of ranking nodes by likelihood of being “on” by the three above methods

4 Comparison of Networks

At some point here, I'm going to want to compare networks. Luckily, there's things to do. There are of course the simple and obvious: size, radius, various connectivity and clustering metrics. There are also more interesting things, like the random walk kernel on graphs. Graph kernels are similar to an inner product on two graphs.

A graph kernel $k(G_1, G_2)$ must be symmetric and positive definite (an inner product must also be bilinear) [5]. To compute the most common type (random walk kernels) we need to define the direct product of two graphs G and G' . That is the graph G_{\times} with vertices

$$(v_i, v'_j) \in V \times V'$$

and edges

$$(v_i, v'_j) \sim (v_k, v'_l) \Leftrightarrow v_i \sim v_k \& v'_j \sim v'_l$$

We can compute the adjacency matrix of this by computing the Kronecker product of the adjacency matrices of G and G' . The Kronecker product of matrices A and B is

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1m}B \\ a_{21}B & \ddots & & \\ \vdots & & & \\ a_{n1}B & & & a_{nm}B \end{pmatrix}$$

A random walk on this graph is isomorphic to a random walk on either graph, and so one can consider a random walk on this graph as a simultaneous random walk on both. We can take the adjacency matrices normalized so row sums are 1, call those A , A' , and then

$$W_{\times} = A \otimes A'$$

This gives a stochastic matrix W_{\times} . Given distributions p and p' , define $p_{\times} = p \otimes p'$. Define the kernel

$$k(G, G') = \sum_{k=0}^{\infty} \mu(k) q_{\times}^T W_{\times}^k p_{\times}$$

where q, q' are stopping probabilities, and p, p' are initial distributions. This counts all the common random walks. The coefficients $\mu(k)$ serve two purposes. One is to make sure the sum is finite. The other allows one to tune the kernel to emphasize walks of certain lengths. I think it's called geometric if $\mu(k) = c^k$. However, this seems like it will overemphasize short walks, of which there will be many in common.

References

- [1] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, Dec 2004.
- [2] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
- [3] Heiko Mller and Francesco Mancuso. Identification and analysis of co-occurrence networks with netcutter. *PLOS ONE*, 3(9):1–16, 09 2008.
- [4] M. E. J. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70:056131, Nov 2004.
- [5] S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242, August 2010.
- [6] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.