Journal of Complex Networks (2014) **2**, 1–18
doi:10.1093/comnet/cnt011
Advance Access publication on 27 September 2013

Oriented and degree-generated block models: generating and inferring communities with inhomogeneous degree distributions

Yaojia Zhu* and Xiaoran Yan
Computer Science, University of New Mexico, Albuquerque, NM, USA
*Corresponding author: yaojia.zhu@gmail.com

and

Cristopher Moore
Santa Fe Institute and University of New Mexico, Santa Fe, NM, USA

Edited by: Ernesto Estrada

[Received on 25 May 2013; accepted on 16 July 2013]

The stochastic block model is a powerful tool for inferring community structure from network topology. However, it predicts a Poisson degree distribution within each community, while most real-world networks have a heavy-tailed degree distribution. The degree-corrected (DC) block model can accommodate arbitrary degree distributions within communities. But since it takes the vertex degrees as parameters rather than generating them, it cannot use them to help it classify the vertices, and its natural generalization to directed graphs cannot even use the orientations of the edges. In this paper, we present variants of the block model with the best of both worlds: they can use vertex degrees and edge orientations in the classification process, while tolerating heavy-tailed degree distributions within communities. We show that for some networks, including synthetic networks and networks of word adjacencies in English text, these new block models achieve a higher accuracy than either standard or DC block models.

Keywords: complex networks; community detection; generative model; stochastic block model; degree distribution.

1. Introduction

In many real-world networks, vertices can be divided into *communities* based on their connections. Social networks can be forged by daily interactions such as karate training [1], the blogosphere contains groups of linked blogs with similar political views [2], words can be tagged as different parts of speech based on their adjacencies in large texts [3], and so on. Communities range from assortative clumps, where vertices preferentially attach to others of the same type, to *functional* communities of vertices that connect to the rest of the network in similar ways, such as groups of predators in a food web that feed on similar prey [4,5]. Understanding various community structures, and their relations to the functional roles of vertices and edges, is crucial to understanding network data.

The *stochastic block model* (SBM) [6–9] is a popular and highly flexible generative model for community detection. It partitions the vertices into communities or *blocks*, where vertices belonging to the same block are *stochastically equivalent* [10] in the sense that the probabilities of a connection with all other vertices are the same for all vertices in the same block. With this rather general definition of community, block models can capture many types of community structure, including assortative, disassortative and satellite communities and mixtures of them [5,11–15].

The SBM assumes that each edge is generated independently conditioned on the block memberships. Each entry A_{uv} of the adjacency matrix is then Bernoulli-distributed, where the probability that $A_{uv} = 1$ depends solely on the block memberships g_u and g_v of its endpoints. Since every pair of vertices in a given pair of blocks are connected with the same probability, for large n the degree distribution within each block is Poisson. As a consequence, vertices with very different degrees are unlikely to be in the same block. This leads to problems with modelling real networks, which often have heavy-tailed degree distributions within each community. For instance, both liberal and conservative political blogs range from high-degree 'leaders' to low-degree 'followers' [2].

To avoid this effect, and allow degree inhomogeneity within blocks, there is a long history of generative models where the probability of an edge depends on vertex attributes θ_u as well as their block memberships (e.g. [13,16]). A particularly elegant variant is the *degree-corrected* (DC) block model of Karrer and Newman [17]. They consider random multigraphs, where A_{uv} is Poisson-distributed with mean $\theta_u\theta_v\omega_{g_u,g_v}$. The most-likely value of θ_u is the observed degree d_u, and this model can thus generate graphs with arbitrary (expected) degree distributions within each community.

On the other hand, the DC model cannot use the vertex degrees to help it classify the vertices, precisely because it takes the degrees as parameters rather than as data that need to be explained. For this reason, the DC model may actually fail to recognize communities that differ significantly in their degree distributions. Thus, we have two extremes: the SBM separates vertices by degree even when it should not, and the DC model fails to do so even when it should.

We have a similar problem for directed graphs. The natural generalization of the DC model, the *directed degree-corrected* (DDC) block model, has two parameters for each vertex: the expected in-degree and out-degree. But this model cannot even take advantage of edge orientations. For instance, in English, adjectives usually precede nouns but rarely vice versa. Thus the ratio of each vertex's in- and out-degree is strongly indicative of its block membership. But the DDC model takes these degrees as parameters, so it is unable to use this part of the data to classify words according to their parts of speech.

In this paper, we propose two new types of block model, which combine the strengths of the DC and uncorrected block models. The *oriented degree-corrected* (ODC) block model is able to utilize the edge orientations for community detection by only correcting the total degrees. We show that for networks with strongly asymmetric behaviour between communities, including synthetic networks and some real-world networks, ODC achieves a higher accuracy than SBM or DDC.

We also propose the *degree-generated* (DG) block model, which treats the expected degree of each vertex as generated from a prior distribution in each block, such as a power law whose exponent varies from one community to another. By including the probability of these degrees in the likelihood of a given block assignment, the DG model captures the interaction between the degree distribution and the community structure. In particular, it automatically strikes a balance between allowing vertices of different degrees to coexist in the same community on the one hand, and using vertex degrees to separate vertices into communities on the other. Our experiments show that DG works especially well in networks where communities have highly inhomogeneous degree distributions, but where the degree distributions differ significantly between communities. In some cases, DG has a further advantage in faster convergence as it reshapes the parameter space, providing the algorithm a shortcut to the correct community structure.

These new variants of the block model give us the best of both worlds. They can not only tolerate heavy-tailed degree distributions within communities, but can also use degrees and edge orientations to help classify the vertices. In addition to their performance on real and synthetic networks, our models illustrate a valuable point about generative models and statistical inference: when inferring the structure of a network, you can only use the information that you try to generate.

2. The models

2.1 *Background: degree-corrected block models*

Throughout, we use n and m to denote the number of vertices and edges, respectively, and k to denote the number of blocks. The problem of determining k is a crucial model selection problem. In some cases, we can use prior domain knowledge, such as the number of different parts of speech, or the number of different factions into which a network splits over time. In the absence of such knowledge, a variety of methods have been proposed; in particular, we could compute the likelihood of our various models with different values of k, and apply a suitable penalty term as in the Akaike information criterion (AIC) [18] or Bayesian information criterion (BIC) [19] to discourage overfitting. However, in this paper, we assume that k is given.

In the original SBM, the entries A_{uv} of the adjacency matrix are independent and Bernoulli-distributed, with $\Pr[A_{uv} = 1] = p_{g_u,g_v}$. Here, g_u is the block to which u belongs, where p is a $k \times k$ matrix. In the DC block model of Karrer and Newman [17] we consider random multigraphs where the A_{uv} are independent and Poisson-distributed, $A_{uv} \sim \mathrm{Poi}(\theta_u\theta_v\omega_{g_u,g_v})$. Here, ω replaces p, and θ_u is an overall propensity for u to connect to other vertices. Note that since the A_{uv} are independent, the degrees d_u will vary somewhat around their expectations; however, the resulting model is much simpler to analyse than the one that controls the degree of each vertex exactly.

Ignoring self-loops, the likelihood with which the DC model generates an undirected multigraph G is then

$$P(G \mid \theta, \omega, g) = \prod_{u<v} \frac{(\theta_u\theta_v\omega_{g_ug_v})^{A_{uv}}}{A_{uv}!} \exp(-\theta_u\theta_v\omega_{g_ug_v}). \tag{2.1}$$

To remove the obvious symmetry where we multiply the θ's by a constant C and divide ω by C^2, we can impose a normalization constraint $\sum_{u:g_u=r} \theta_u = \kappa_r$ for each block r, where $\kappa_r = \sum_{u:g_u=r} d_u$ is the total degree of the vertices in block r. Under these constraints, the maximum likelihood estimates (MLEs) for the θ parameters are $\hat{\theta}_u = d_u$. For each pair of blocks r, s, the MLE for ω_{rs} is

$$\hat{\omega}_{rs} = \frac{m_{rs}}{\kappa_r\kappa_s}, \tag{2.2}$$

where m_{rs} is the number of edges connecting block r to block s (and edges within blocks are counted twice). Substituting these MLEs for θ and ω then gives the profile log-likelihood [20]

$$\log P(G \mid g) = \frac{1}{2} \sum_{r,s=1}^{k} m_{rs} \log \frac{m_{rs}}{\kappa_r\kappa_s}. \tag{2.3}$$

Here we ignore constants that are independent of g, namely $\sum_u d_u \log d_u$, $\sum_{uv} \log A_{uv}!$, and $-m/2$.

2.2 *Directed and oriented degree-corrected block models*

The natural extension of DC to directed networks, which we call the DDC block model, has two parameters θ_u^{out} and θ_u^{in} for each vertex. The number of directed edges from u to v is again Poisson-distributed, $A_{uv} \sim \mathrm{Poi}(\theta_u^{\mathrm{out}}\theta_v^{\mathrm{in}}\omega_{g_u,g_v})$. We impose the constraints $\sum_{u:g_u=r} \theta_u^{\mathrm{out}} = \kappa_r^{\mathrm{out}}$ and $\sum_{u:g_u=r} \theta_u^{\mathrm{in}} = \kappa_r^{\mathrm{in}}$ for each block r, where $\kappa_r^{\mathrm{out}} = \sum_{u:g_u=r} d_u^{\mathrm{out}}$ and $\kappa_r^{\mathrm{in}} = \sum_{u:g_u=r} d_u^{\mathrm{in}}$ denote the total out- and in-degree of block r.

As before, let m_{rs} denote the number of directed edges from block r to block s. Then the likelihood is

$$P(G \mid \theta, \omega, g) = \prod_{uv} \frac{(\theta_u^{\text{out}} \theta_v^{\text{in}} \omega_{g_u g_v})^{A_{uv}}}{A_{uv}!} \exp(-\theta_u^{\text{out}} \theta_v^{\text{in}} \omega_{g_u g_v})$$
$$= \frac{\prod_u (\theta_u^{\text{out}})^{d_u^{\text{out}}} (\theta_u^{\text{in}})^{d_u^{\text{in}}} \prod_{rs} \omega_{rs}^{m_{rs}} \exp(-\kappa_r^{\text{out}} \kappa_s^{\text{in}} \omega_{rs})}{\prod_{uv} A_{uv}!}. \tag{2.4}$$

Ignoring the constant $\sum_{uv} \log A_{uv}!$, the log-likelihood is

$$\log P(G \mid \theta, \omega, g) = \sum_u (d_u^{\text{out}} \log \theta_u^{\text{out}} + d_u^{\text{in}} \log \theta_u^{\text{in}}) + \sum_{rs} (m_{rs} \log \omega_{rs} - \kappa_r^{\text{out}} \kappa_s^{\text{in}} \omega_{rs}). \tag{2.5}$$

The MLEs for the parameters (see Appendix A) are

$$\hat{\theta}_u^{\text{out}} = d_u^{\text{out}}, \quad \hat{\theta}_u^{\text{in}} = d_u^{\text{in}}, \quad \hat{\omega}_{rs} = \frac{m_{rs}}{\kappa_r^{\text{out}} \kappa_s^{\text{in}}}. \tag{2.6}$$

Ignoring constants again and substituting these MLEs gives

$$\log P(G \mid g) = \sum_{r,s=1}^{k} m_{rs} \log \frac{m_{rs}}{\kappa_r^{\text{out}} \kappa_s^{\text{in}}}. \tag{2.7}$$

In the DDC model, the expected in- and out-degrees of each vertex are completely specified by the θ parameters. Thus DDC allows vertices with arbitrary degrees to fit comfortably together in the same block. On the other hand, since the degrees are given as parameters, rather than as data that the model must generate and explain, DDC cannot use them to infer vertex labels. Indeed, it cannot even take advantage of the orientations of the edges, as shown below by its poor performance on networks with strongly asymmetric community structure.

To deal with this, we present a partially DC block model capable of taking advantage of edge orientations, which we call the ODC block model. Following the maxim that we can only use the information that we try to generate, we correct only for the total degrees of the vertices, and generate the edges' orientations.

Let $\bar{G}$ denote the undirected version of a directed graph G, i.e., the multigraph resulting from erasing the arrows for each edge. Its adjacency matrix is $\bar{A}_{uv} = A_{uv} + A_{vu}$, so, for instance, $\bar{G}$ has two edges between u and v if G had one pointing in each direction. The ODC model can be thought of as generating $\bar{G}$ according to the undirected DC model, and then choosing the orientation of each edge according to another matrix ρ_{rs}, where an edge (u, v) is oriented from u to v with probability ρ_{g_u, g_v}. Thus the total log-likelihood is

$$\log P(G \mid \theta, \omega, \rho, g) = \log P(\bar{G} \mid \theta, \omega, g) + \log P(G \mid \bar{G}, \rho, g). \tag{2.8}$$

Writing $\bar{m}_{rs} = m_{rs} + m_{sr}$ and $\kappa_r = \kappa_r^{\text{in}} + \kappa_r^{\text{out}}$, we can set θ_u and ω_{rs} for the undirected model to their MLEs as in Section 2.1, giving

$$\log P(\bar{G} \mid g) = \frac{1}{2} \sum_{r,s=1}^{k} \bar{m}_{rs} \log \frac{\bar{m}_{rs}}{\kappa_r \kappa_s}. \tag{2.9}$$

The orientation term is

$$\log P(G \mid \bar{G}, \rho, g) = \sum_{rs} m_{rs} \log \rho_{rs}. \tag{2.10}$$

For each r and s, we have $\rho_{rs} + \rho_{sr} = 1$, and the MLEs for ρ are

$$\hat{\rho}_{rs} = m_{rs}/\bar{m}_{rs}. \tag{2.11}$$

Note that $\hat{\rho}_{rr} = \frac{1}{2}$ for any r. Substituting the MLEs for ρ and combining (2.9) with (2.10) gives the profile log-likelihood for the ODC model as follows:

$$\log P(G \mid g) = \sum_{r,s=1}^{k} m_{rs} \log \frac{m_{rs}}{\kappa_r \kappa_s}. \tag{2.12}$$

In order to understand ODC better, we analyse the edge orientation term (2.10) more carefully. Substituting the MLEs for ρ in (2.10) gives

$$\begin{aligned} \log P(G|\bar{G}, g) &= \frac{1}{2} \sum_{rs} (m_{rs} \log \hat{\rho}_{rs} + m_{sr} \log \hat{\rho}_{sr}) \\ &= \frac{1}{2} \sum_{r \neq s} \bar{m}_{rs} (\hat{\rho}_{rs} \log \hat{\rho}_{rs} + \hat{\rho}_{sr} \log \hat{\rho}_{sr}) + \sum_{r} m_{rr} \log \hat{\rho}_{rr} \\ &= -\sum_{r<s} \bar{m}_{rs} \tau(\hat{\rho}_{rs}) - (\log 2) \sum_{r} m_{rr}. \end{aligned} \tag{2.13}$$

Here $\tau(x) = -x \log(x) - (1-x) \log(1-x)$ is the entropy function. The total number of inter-block edges is $\sum_{r<s} \bar{m}_{rs}$, and the total number of intra-block edges is $\sum_r m_{rr}$.

Examining (2.13), we see that the edge orientation term prefers highly directed inter-block connections, i.e., such that $\hat{\rho}_{rs}$ are near 0 or 1, so that $\tau(\hat{\rho}_{rs})$ is minimized. However, as $\tau(\hat{\rho}_{rs}) \leqslant \log 2$, it also prefers disassortative structures, in which the number of intra-block edges m_{rr} is as small as possible; it has no basis on which to orient these edges, so they contribute a negative term to the log-likelihood.

Thus, while ODC can detect assortative structures due to the undirected term (2.9), and may do better than DC or DDC if the connections between blocks are highly directed (for instance, if there are three blocks, and all inter-block connections are oriented from the 'lower' block to the 'higher' one), it performs best in disassortative networks with highly directed connections between blocks, so that the orientation of most edges is determined by the block assignment of their endpoints. We will see an example of this in a real-world network in Section 4.1.

We note that we could reduce ODC's preference for disassortative structure by simply ignoring the second term in (2.13). This would correspond to a generative model where inter-block edges are directed, but intra-block edges are undirected. We have not pursued this.

We can also view ODC as a special case of DDC, where we add the constraint $\theta_u^{\text{in}} = \theta_u^{\text{out}}$ for all vertices u (see Appendix B). Moreover, if we set $\theta_u = 1$ for all u, we obtain the original block model, or rather its Poisson multigraph version where each A_{uv} is Poisson-distributed with mean ω_{g_u,g_v}. Thus, SBM $\leqslant$ ODC $\leqslant$ DDC, where $A \leqslant B$ means that model A is a special case of model B, or that B is an elaboration of A. We will see below that since it is forced to explain edge orientations, ODC performs better on some networks than either SBM or DDC.

2.3 *Degree-generated block models*

Another way to utilize vertex degrees for community detection is to require the model to generate them, according to some prior degree distribution derived from domain knowledge. For instance, many real-world networks have a power-law degree distribution, but with parameters (such as the exponent, minimum degree or leading constant) that vary from community to community. In that case, the degree of a vertex gives us a clue as to its block membership. This yields our proposed DG block models. They can tolerate heavy-tailed degree distributions within communities, but can also use degrees and edge orientations to help classify the vertices.

In a DG model, we first generate the θ parameters of one of the DC block models discussed above, i.e., the expected vertex degrees, and then use them to generate a random multigraph. Specifically, each θ_u is generated independently according to some distribution whose parameters ψ depend on the block g_u to which u belongs. Thus DG is a hierarchical model, which extends the previous DC block models by adding a degree generation stage on top, treating the θ's as generated by the block assignment g and the parameters ψ rather than as parameters.

We can apply this approach to the undirected, directed, or oriented versions of the DC model; at the risk of drowning the reader in acronyms, we denote these DG-DC, DG-DDC and DG-ODC. In each case, the total log-likelihood of a graph G is

$$\log P(G \mid \psi, \omega, g) = \log \int \mathrm{d}\theta P(G \mid \theta, \omega, g) P(\theta \mid \psi, g), \tag{2.14}$$

where

$$P(\theta \mid \psi, g) = \prod_u P(\theta_u \mid \psi_{g_u}). \tag{2.15}$$

For the directed models, we use θ_u as a shorthand for θ_u^{in} and θ_u^{out}.

As in many hierarchical models, computing this integral appears to be difficult, except when $P(\theta \mid \psi)$ has the form of a conjugate prior such as the Gamma distribution (see Appendix C). We approximate it with a point estimate by assuming that it is dominated by the most-likely value of θ,

$$\log P(G \mid \psi, \omega, g) \approx \log P(G \mid \hat{\theta}, \omega, g) + \log P(\hat{\theta} \mid \psi, g). \tag{2.16}$$

However, even determining $\hat{\theta}$ is challenging when $P(\theta \mid \psi)$ is, say, a power law with a minimum-degree cutoff. Thus we make a further approximation, setting $\hat{\theta}$ just by maximizing the block model term $\log P(G \mid \hat{\theta}, \omega, g)$ as we did before, using (2.6) or the analogous equations for the DC or ODC. In essence, these approximations treat $P(\hat{\theta} \mid \psi, g)$ as a penalty term, imposing a prior on the degree distribution of each community with hyperparameters ψ. This leads to community structures that might not be as good a fit to the edges, but compensate with a much better fit to the degrees.

We can either treat the degree-generating parameters ψ as fixed—say, as predicted by a theoretical model of network growth [21–23]—or infer them by finding the $\hat{\psi}$ that maximizes $P(\hat{\theta} \mid \psi)$. For instance, suppose the θ_u in block $g_u = r$ are distributed as a continuous power law with a lower cutoff $\theta_{\min,r}$. Specifically, let the parameters in each block r be $\psi_r = (\alpha_r, \beta_r, \theta_{\min,r})$, and let

$$P(\theta_u \mid \psi_r) = \begin{cases} \beta_r & \theta_u = 0, \\ 0 & 0 < \theta_u < \theta_{\min,r}, \\ \dfrac{(1-\beta_r)(\alpha-1)}{\theta_{\min,r}} \left(\dfrac{\theta_u}{\theta_{\min,r}} \right)^{-\alpha_r} & \theta_u \geqslant \theta_{\min,r}. \end{cases} \tag{2.17}$$

In the directed case, we have $\psi_r^{\text{in}} = (\alpha_r^{\text{in}}, \beta_r^{\text{in}}, \theta_{\text{min},r}^{\text{in}})$ and $\psi_r^{\text{out}} = (\alpha_r^{\text{out}}, \beta_r^{\text{out}}, \theta_{\text{min},r}^{\text{out}})$. Allowing β_r^{out} to be non-zero, for instance, lets us directly include vertices with no outgoing neighbours; we find this useful in some networks. Alternately, we can choose $(\theta_u^{\text{in}}, \theta_u^{\text{out}})$ from some joint distribution, allowing in- and out-degrees to be correlated in various ways.

We fix $\theta_{\text{min},r} = 1$. Given the degrees and the block assignment, let $Y_r = \{u : g_u = r \text{ and } \theta_u \neq 0\}$, and let $y_r = |Y_r|$. The MLE for α_r is [24]

$$\hat{\alpha}_r = 1 + \frac{y_r}{\sum_{u \in Y_r} \ln \theta_u}. \tag{2.18}$$

The MLE for $\hat{\beta}_r$ is simply the fraction of vertices in block r with degree zero.

3. Experiments on synthetic networks

In order to understand under what circumstances our models out-perform previous variants of the block model, we performed experiments on synthetic networks, varying the degree distributions in communities, the degree of directedness between communities, and so on.

First, we generated undirected networks according to the DG-DC model, with two blocks or communities of equal size $n/2$. In order to confound the block model as much as possible, we deliberately designed these networks so that the two blocks have the same average degree. The degree distribution in block 1 is a power law with exponent $\alpha = 1.7$, with an upper bound of 1850, so that the average degree is 20. The degree distribution in block 2 is Poisson, also with mean 20. As described in Appendix D, the upper bound on the power law is larger than any degree actually appearing in the network; it just changes the normalizing constant of the power law, and the MLE for α can still be calculated using (2.18). We assume the algorithm knows that one block has a power-law degree distribution and the other is Poisson, but we force it to infer the parameters of these distributions.

As in [17], we use a parameter λ to interpolate linearly between a fully random network with no community structure and a 'planted' one where the communities are completely separated. Thus,

$$\omega_{rs} = \lambda \omega_{rs}^{\text{planted}} + (1 - \lambda)\omega_{rs}^{\text{random}}, \tag{3.1}$$

where

$$\omega_{rs}^{\text{random}} = \frac{\kappa_r \kappa_s}{2m}, \quad \omega^{\text{planted}} = \begin{pmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{pmatrix}. \tag{3.2}$$

We inferred the community structure with various models. We ran the Kernighan–Lin (KL) heuristic first to find a local optimum [17], and then ran the heat-bath Markov Chain Monte Carlo (MCMC) algorithm with a fixed number of iterations to further refine it if possible. We initialized each run with a random block assignment; to test the stability of the models, we also tried initializing them with the correct block assignment. Since isolated vertices do not participate in the community structure, giving us little or no basis on which we can classify them, we remove them and focus on the giant component. For $\lambda = 1$, where the community structure is purely the 'planted' one, we kept two giant components, one in each community.

We measured accuracy by the normalized mutual information (NMI) [25] between the most-likely block assignment found by the model and the correct assignment. To make this more concrete, if there are two blocks of equal size and 95% of the vertices in each block are labelled correctly, the NMI is 0.714. If 90% in each group are labelled correctly, the NMI is 0.531. For groups of unequal size, the NMI is a better measure of accuracy than the fraction of vertices labelled correctly, since one can make this fraction fairly large simply by assigning every vertex to the larger group.

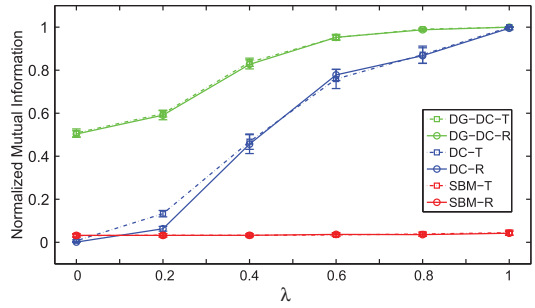

FIG. 1. Tests on synthetic networks generated by the DG-DC model. Each point is based on 30 randomly generated networks with $n = 2400$. For each network and each model, we choose the best result from 10 independent runs, initialized either with random assignments (the suffix *R*) or the true block assignment (the suffix *T*). Each run consisted of the KL heuristic followed by 10^6 Markov Chain Monte Carlo (MCMC) steps. Our DG block model performs much better on these networks than the DC model. The non-DC (SBM) model does not work at all.

As shown in Fig. 1, DG-DC works very well even for small λ. This is because it can classify most of the vertices simply based on their degrees; if d_u is far from 20, for instance, then u is probably in block 1. As λ increases, it uses the connections between communities as well, giving near-perfect accuracy for $\lambda \geqslant 0.6$. It does equally well whether its initial assignment is correct or random.

The DC model, in contrast, is unable to use the vertex degrees, and has accuracy near zero (i.e., not much better than a random block assignment) for $\lambda \leqslant 0.2$. Like the SBM [11,12], it may have a phase transition at a critical value of λ below which the community structure is undetectable. Initializing it with the correct assignment helps somewhat at these values of λ, but even then it settles on an assignment far from the correct one.

The original SBM, as discussed above, separates vertices with high degrees from vertices with low degrees. Thus it cannot find the correct group structure even for large λ. Our synthetic tests are designed to have a broad degree distribution in block 1, and thus make SBM fail. Note that if the degree distribution in block 1 is a power law with a larger exponent α, then most of the degrees will be much lower than 20, in which case SBM works reasonably well.

Next, we generated directed networks according to the DG-DDC model. We again have two blocks of equal size, with degree distributions similar to the undirected networks tested above. In block 1, both out- and in-degrees are power-law distributed with $\alpha = 1.7$, with an upper bound of 1850 so that the expected degree is 20. In block 2, both out- and in-degrees are Poisson-distributed with mean 20. To test our oriented and directed models, we interpolate between a random network $\omega_{rs}^{\text{random}} = \kappa_r \kappa_s / 4m$ and a planted network with completely asymmetric connections between the blocks,

$$\omega^{\text{planted}} = \begin{pmatrix} (\kappa_1 - \omega_{12})/2 & \omega_{12} \\ 0 & (\kappa_2 - \omega_{12})/2 \end{pmatrix}, \tag{3.3}$$

where $\omega_{12} \leqslant \min(\kappa_1, \kappa_2)$. We choose $\omega_{12} = \frac{1}{2}\min(\kappa_1, \kappa_2)$.

As Fig. 2 shows, DG-ODC and DG-DDC have very similar performance at the extremes where $\lambda = 0$ and 1. However, DG-ODC works better than DG-DDC for other values of λ, and both of them achieve much better accuracy than the ODC or DDC models. As in Fig. 1, the DG models can achieve a high accuracy based simply on the vertex degrees, and as λ grows they leverage this information further to achieve near-perfect accuracy for $\lambda \geqslant 0.8$.

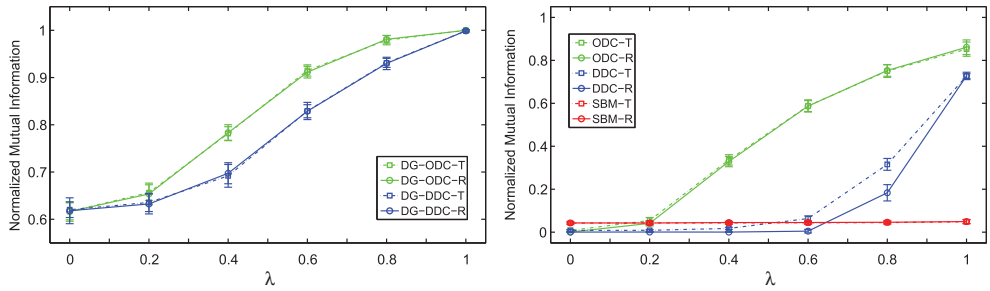

FIG. 2. Tests on synthetic directed networks with $n = 2400$. Left, DG-ODC and DG-DDC; right, ODC and DDC. The DG models again perform very well even for small λ, since they can use in- and out-degrees to classify the vertices. ODC performs significantly better than DDC for $\lambda \geqslant 0.4$, since it can use the edge orientations to distinguish the two blocks. The number of networks, runs, and MCMC steps per run are as in Fig. 1.

TABLE 1 *Basic statistics of the three word adjacency networks. S and M denote the simple and multigraph versions, respectively*

Network	No. of words	No. of adjectives	No. of nouns	No. of edges (S)	No. of edges (M)
David	112	57	55	569	1494
News	376	91	285	1389	2411
Brown	23258	6235	17023	66734	88930

Among the non-DC models, ODC performs significantly better than DDC for $\lambda \geqslant 0.4$. Edges are more likely to point from block 1 to block 2 than vice versa, and ODC can take advantage of this information while DDC cannot. As we will see in the next section, ODC performs well on some real-world networks precisely for this reason.

4. Experiments on real networks

We studied three word adjacency networks, where vertices are separated into two blocks: adjectives and nouns. The first consists of common words in Dickens' novel *David Copperfield* [26]. The other two are built from the Brown corpus, which is a tagged corpus of present-day edited American English across various categories, including news, novels, documents and many others [27]. The smaller one contains words in the News category (45 archives) that appeared at least 10 times; the larger one contains all the adjectives and nouns in the giant component of the entire corpus.

We considered both the simple version of these networks where $A_{uv} = 1$ if u and v ever occur adjacently in that order, and the multigraph version where $A_{uv} \geqslant 0$ is the number of adjacent cooccurrences. The sizes, block sizes and number of edges of these networks are shown in Table 1. In 'News' and 'Brown', the block sizes are quite different, with more nouns than adjectives. As discussed above, the NMI is a better measure of accuracy than the fraction of vertices labelled correctly, since we could make the latter fairly large by labelling everything a noun.

In each network, both blocks have heavy-tailed in- and out-degree distributions (Fig. 3). The connections between them are disassortative and highly asymmetric: since in English adjectives precede nouns more often than they follow them, and more often than adjectives precede adjectives or nouns precede nouns, ω_{12} is roughly 10 times larger than ω_{21}, and ω_{12} is larger than either ω_{11} or ω_{22}. The

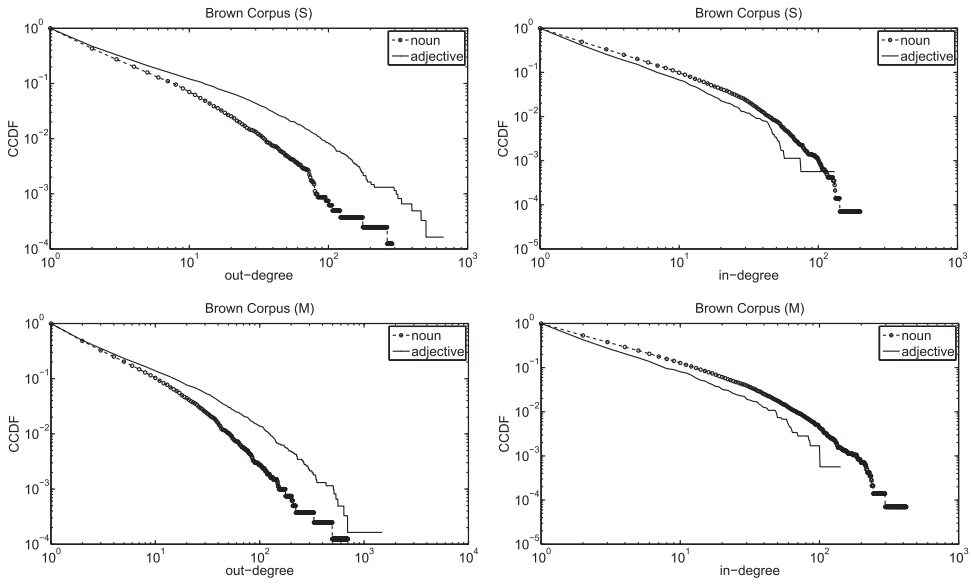

FIG. 3. Degree distributions in the Brown network.

TABLE 2 *The matrices $\omega_{rs} = m_{rs}/(n_r n_s)$ for the most-likely block assignment according to the stochastic block model*

	David(S)	David(M)	News(S)	News(M)	Brown(S)	Brown(M)
ω_{11}	0.039	0.080	0.010	0.012	9.1e−05	1.1e−04
ω_{12}	0.118	0.358	0.015	0.028	3.4e−04	4.4e−04
ω_{21}	0.018	0.025	0.002	0.003	2.0e−05	2.4e−05
ω_{22}	0.006	0.011	0.010	0.019	8.8e−05	1.2e−04

ω for each network corresponding to the correct block assignment (according to the SBM) is shown in Table 2.

4.1 *Performance of oriented and degree-corrected models*

Table 3 compares the performance of non-DG block models, including SBM, DC, ODC, and DDC. (Under DC, we ignore the edge orientations, and treat the graph as undirected. Note that the resulting network may contain multi-edges even though the directed one does not.)

In our experiments, we started with a random initial block assignment, ran the KL heuristic to find a local optimum, and then ran the heat-bath MCMC algorithm. We also tested a naive heuristic NH which simply labels a vertex v as an adjective if $d_v^{\text{out}} > d_v^{\text{in}}$, and a noun if $d_v^{\text{in}} > d_v^{\text{out}}$. If $d_v^{\text{out}} = d_v^{\text{in}}$, NH labels v randomly with equal probabilities.

For 'David', DC and ODC work fairly well, and both are better than the naive heuristic NH. Moreover, the mistakes they make are instructive. There are three adjectives with out-degree zero: 'full', 'glad' and 'alone'. ODC mislabels these since it expects edges to point away from adjectives, while DC

TABLE 3 *For each model and each network, we pick the block assignment with highest likelihood and compute its NMI with the correct block assignment. Each run consisted of the KL heuristic, starting with a random block assignment, followed by* 10^6 *Markov Chain Monte Carlo (MCMC) steps. The results for 'David' and 'News' are based on 100 independent runs; for 'Brown',* 50 *runs are executed. The best NMI for each network is shown in bold*

	David(S)	David(M)	News(S)	News(M)	Brown(S)	Brown(M)
SBM	0.423	0.051	0.006	0.018	0.001	7e−04
DC	**0.566**	**0.568**	0.084	0.083	0.020	0.015
ODC	0.462	0.470	0.084	0.029	**0.311**	**0.318**
DDC	0.128	8e−04	0.084	0.091	0.016	0.012
NH	0.395	0.449	**0.215**	**0.233**	0.309	0.314

TABLE 4 *Results using the naive NH assignment as the initial condition, again followed by* 10^6 *MCMC steps. This hint now lets ODC outperform the other models on 'News'. The best NMI for each network is shown in bold*

	David(S)	David(M)	News(S)	News(M)	Brown(S)	Brown(M)
SBM	0.423	0.051	0.006	0.021	0.001	7e−04
DC	**0.566**	**0.568**	0.084	0.015	0.160	0.155
ODC	0.462	0.470	**0.247**	**0.270**	**0.311**	**0.318**
DDC	0.015	0.060	0.084	0.005	0.005	0.070
NH	0.395	0.449	0.215	0.233	0.309	0.314

labels them correctly by using the fact that edges are disassortative, tending to cross from one block to the other.

The standard SBM works well on 'David(S)' but fails on 'David(M)' because the degrees in the multigraph are more skewed than those in the simple one. Finally, DDC performs the worst; by correcting for in- and out-degrees separately, it loses any information that the edge orientations could provide, and even fails to notice the disassortative structure that DC uses. Thus full degree-correction in the directed case can make things worse, even when the degrees in each community are broadly distributed.

For 'Brown', all these models fail except ODC, although it does only slightly better than the naive NH. For 'News', all these models fail, even ODC. Despite the degree correction, the most-likely block assignment is highly assortative, with high-degree vertices connecting each other. However, we found that in most runs on 'News', ODC used the edge orientations successfully to find a block assignment close to the correct one; it found the assortative structure only occasionally. This suggests that, even though the 'wrong' structure has a higher likelihood, we can do much better if we know what kind of community structure to look for; in this case, disassortative and directed.

To test this hypothesis, we tried giving the models a hint about the community structure by using NH to determine the initial block assignment. We then performed the KL heuristic and the MCMC algorithm as before. As Table 4 shows, this hint improves ODC's performance on 'News' significantly; it is able to take the initial naive classification, based solely on degrees, and refine it using the network's structure. Note that this more accurate assignment actually has lower likelihood than the one found in Table 3 using a random initial condition—so NH helps the model stay in a more accurate, but less likely, local optimum. Starting with NH improves DC's performance on 'Brown' somewhat, but DC still ends up with an assignment less accurate than the naive one.

TABLE 5 *MLEs for the degree generation parameters in the Brown network, given the correct assignment*

	Brown(S)				Brown(M)			
Block	$\hat{\alpha}_{\text{in}}$	$\hat{\alpha}_{\text{out}}$	$\hat{\beta}_{\text{in}}$	$\hat{\beta}_{\text{out}}$	$\hat{\alpha}_{\text{in}}$	$\hat{\alpha}_{\text{out}}$	$\hat{\beta}_{\text{in}}$	$\hat{\beta}_{\text{out}}$
Adjective	2.329	2.629	0.161	0.527	2.136	2.326	0.161	0.527
Noun	2.721	2.248	0.716	0.021	2.576	2.134	0.716	0.021

TABLE 6 *Performance of degree-generated models. KL indicates that we applied the KL heuristic before* 10^6 *MCMC steps. DG indicates degree generation. Each number gives the NMI for the most-likely assignment found in* 50 *independent runs. The best model is DG-ODC. Moreover, degree generation helps ODC converge, providing much of the benefit of the KL heuristic while avoiding its long running time* (*see bold numbers*)

		Brown(S)			Brown(M)		
		DC	ODC	DDC	DC	ODC	DDC
–	–	0.010	0.188	0.008	0.007	0.203	0.011
KL	–	0.020	0.311	0.016	0.015	0.318	0.012
–	DG	0.267	0.302	0.213	0.278	0.310	0.149
KL	DG	0.271	**0.312**	0.225	0.284	**0.320**	0.195

4.2 *Performance of degree-generated models*

In this section, we measure the performance of DG models on the Brown network, and compare them to their non-DG counterparts. According to Fig. 3, the in- and out-degree distributions in each block have heavy tails close to a power-law. Moreover, the out-degrees of the adjectives have a heavier tail than those of the nouns, and vice versa for the in-degrees. This is exactly the kind of difference in the degree distributions between communities that our DG block models are designed to take advantage of.

Setting $\theta_{\min} = 1$, we can estimate the parameters α and β for these distributions as discussed in Section 2.3. We show the most likely values of these parameters, given the correct assignment, in Table 5.

As Table 6 shows, degree generation improves DC and DDC significantly, letting them find a good assignment as opposed to one with NMI near zero. For ODC, the slight performance improvement makes DG-ODC the best model overall. We compare performance starting with the KL heuristic to performance using MCMC alone. We see that degree generation gives ODC almost as much benefit as the KL heuristic does. In other words, it speeds up the MCMC optimization process, letting ODC find a good assignment without the initial help of the computationally expensive KL heuristic.

5. Conclusions

DC SBMs are powerful tools for dealing with networks with inhomogeneous degree distributions. However, since DC models are given the vertex degrees as parameters and are under no obligation to explain

them, they cannot use degrees to help them classify vertices. A generative model can only learn from the data that it is required to generate.

We have introduced two new kinds of block models that allow for broad or heavy-tailed degree distributions, while still being able to take vertex degrees into account when inferring communities. First, the ODC model is forced to generate edge orientations. Unlike the DDC block model, which takes both in- and out-degrees as parameters, ODC is able to capture certain correlations between the in- and out-degrees. Simply put, for ODC, two vertices are unlikely to be in the same community if one has high in-degree and low out-degree while another has high out-degree and low in-degree. If the network is highly directed or asymmetric, the edge orientations can help ODC find community structures that DDC fails to perceive.

Second, we consider DG models. These use DC block models as a subroutine, but they first generate the expected degree of each vertex from a prior distribution in each community. DG models can achieve high accuracy even when the density of connections within or between communities is close to uniform, as we illustrated in synthetic networks for small λ. Augmenting block models, such as ODC, with degree generation also appears to speed up their convergence in some cases, helping simple algorithms such as MCMC handle large networks without the benefit of expensive preprocessing steps like the KL heuristic. However, the effectiveness of DG depends heavily on knowing the correct form of the degree distribution in each community.

With all these variants of the block model, ranging from the 'classic' version to DC and DG variants, we now have a wide variety of tools for inferring structure in network data. Each model will perform better on some networks and worse on others. A better understanding of the strengths and weaknesses of each one—which kinds of structure they can see and which they are blind to—will help us select the right algorithm each time we meet a new network.

Acknowledgement

We are grateful to Terran Lane, Ben Edwards, Aaron Clauset, and Mark Newman for helpful conversations. This work was supported by the McDonnell Foundation, and by AFOSR and DARPA under grant FA9550-12-1-0432.

References

1. Zachary, W. W. (1977) An information flow model for conflict and fission in small groups. *J. Anthropol. Res.*, **33**, 452–473.
2. Adamic, L. & Glance, N. (2005) The political blogosphere and the 2004 US election: divided they blog. *Proceedings of the 3rd International Workshop on Link Discovery*, pp. 36–43.
3. Newman, M. & Leicht, E. (2007) Mixture models and exploratory analysis in networks. *Proc. Natl Acad. Sci. USA*, **104**, 9564–9569.
4. Allesina, S. & Pascual, M. (2009) Food web models: a plea for groups. *Ecol. Lett.*, **12**, 652–662.
5. Moore, C., Yan, X., Zhu, Y., Rouquier, J.-B. & Lane, T. (2011) Active learning for node classification in assortative and disassortative networks. *Proc. 17th KDD*, pp. 841–849.
6. Airoldi, E., Blei, D., Fienberg, S. & Xing, E. (2008) Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.*, **9**, 1981–2014.
7. Fienberg, S. & Wasserman, S. (1981) Categorical data analysis of single sociometric relations. *Sociol. Methodol.*, pp. 156–192.
8. Holland, P., Laskey, K. & Leinhardt, S. (1983) Stochastic blockmodels: first steps. *Social Networks*, **5**, 109–137.

9. Snijders, T. & Nowicki, K. (1997) Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *J. Classif.*, **14**, 75–100.
10. Wasserman, S. & Anderson, C. (1987) Stochastic a posteriori blockmodels: construction and assessment. *Social Networks*, **9**, 1–36.
11. Decelle, A., Krzakala, F., Moore, C. & Zdeborová, L. (2011a) Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Phys. Rev. E*, **84**.
12. Decelle, A., Krzakala, F., Moore, C. & Zdeborová, L. (2011b) Inference and phase transitions in the detection of modules in sparse networks. *Phys. Rev. Lett.*, **107**.
13. Mørup, M. & Hansen, L. (2009) Learning latent structure in complex networks. *NIPS Workshop on Analyzing Networks and Learning with Graphs*.
14. Newman, M. (2002) Assortative mixing in networks. *Phys. Rev. Lett.*, **89**, 208701.
15. Newman, M. (2003) Mixing patterns in networks. *Phys. Rev. E*, **67**, 026126.
16. Reichardt, J., Alamino, R. & Saad, D. (2011) The interplay between microscopic and mesoscopic structures in complex networks. *PloS One*, **6**, e21282.
17. Karrer, B. & Newman, M. (2011) Stochastic blockmodels and community structure in networks. *Phys. Rev. E*, **83**.
18. Akaike, H. (1974) A new look at the statistical model identification. *Autom. Control IEEE Trans.*, **19**, 716–723.
19. Schwarz, G. (1978) Estimating the dimension of a model. *Ann. Statist.*, **6**, 461–464.
20. Bickel, P. J. & Chen, A. (2009) A nonparametric view of network models and Newman–Girvan and other modularities. *Proc. Natl Acad. Sci.*, **106**, 21068–21073.
21. Albert, R. & Barabási, A.-L. (2002) Statistical mechanics of complex networks. *Rev. Mod. Phys.*, **74**, 47–97.
22. Bauke, H., Moore, C., Rouquier, J.-B. & Sherrington, D. (2011) Topological phase transition in a network model with preferential attachment and node removal. *Eur. Phys. J. B*, **83**, 519–524.
23. Moore, C., Ghoshal, G. & Newman, M. E. J. (2006) Exact solutions for models of evolving networks with addition and deletion of nodes. *Phys. Rev. E*, **74**, 036121.
24. Clauset, A., Shalizi, C. & Newman, M. (2009) Power-law distributions in empirical data. *SIAM Rev.*, **51**, 661–703.
25. Danon, L., Díaz-Guilera, A., Duch, J. & Arenas, A. (2005) Comparing community structure identification. *J. Statist. Mech.: Theory Experiment*, **2005**, P09008.
26. Newman, M. (2006) Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, **74**, 036104.
27. Francis, W. & Kucera, H. (1979) Brown Corpus Manual. *Technical Report*, Department of Linguistics, Brown University, Providence, Rhode Island, US.

Appendix A. Maximum Likelihood Estimators for the directed degree-corrected (DDC) block model

We maximize the log-likelihood function (2.5),

$$\log P(G \mid \theta, \omega, g) = \sum_u (d_u^{\text{out}} \log \theta_u^{\text{out}} + d_u^{\text{in}} \log \theta_u^{\text{in}}) + \sum_{rs} (m_{rs} \log \omega_{rs} - \kappa_r^{\text{out}} \kappa_s^{\text{in}} \omega_{rs}), \tag{A.1}$$

where we have imposed the constraints on the θ parameters

$$\sum_{u:g_u=r} \theta_u^{\text{out}} = \kappa_r^{\text{out}} \quad \text{and} \quad \sum_{u:g_u=r} \theta_u^{\text{in}} = \kappa_r^{\text{in}}. \tag{A.2}$$

For each block r, we associate Lagrange multipliers λ_r^{out} and λ_r^{in} with these constraints. For each vertex u, taking the partial derivative of the log-likelihood with respect to θ_u^{out} and θ_u^{in} gives

$$\frac{d_u^{\text{out}}}{\theta_u^{\text{out}}} = \lambda_{g_u}^{\text{out}} \quad \text{and} \quad \frac{d_u^{\text{in}}}{\theta_u^{\text{in}}} = \lambda_{g_u}^{\text{in}}. \tag{A.3}$$

To satisfy the constraints (A.2), we take $\lambda_r^{\text{out}} = \lambda_r^{\text{in}} = 1$ for all r, so that

$$\hat{\theta}_u^{\text{out}} = d_u^{\text{out}} \quad \text{and} \quad \hat{\theta}_u^{\text{in}} = d_u^{\text{in}}, \tag{A.4}$$

$$\hat{\omega}_{rs} = \frac{m_{rs}}{\kappa_r^{\text{out}} \kappa_s^{\text{in}}}. \tag{A.5}$$

Appendix B. Another view of the ODC model

Here we show that the ODC model is a special case of the DDC model. Recall that the ODC model first generates an undirected graph according to the DC model with parameters θ_u and ω_{rs}, and then orients each edge (u, v) from u to v with probability ρ_{g_u,g_v}. The number of directed edges from u to v is then Poisson-distributed as

$$A_{uv} \sim \text{Poi}(\theta_u \theta_v \omega_{g_u,g_v} \rho_{g_u,g_v}). \tag{B.1}$$

But if we write

$$\omega'_{rs} = \omega_{rs} \rho_{rs}, \tag{B.2}$$

then

$$A_{uv} \sim \text{Poi}(\theta_u \theta_v \omega'_{g_u,g_v}). \tag{B.3}$$

Thus ODC is the special case of DDC where $\theta_u^{\text{in}} = \theta_u^{\text{out}} = \theta_u$ for all vertices u.

For completeness, we check that the two models correspond when we set these parameters equal to their MLEs. We impose the constraint $\sum_{u:g_u=r} \theta_u = \kappa_r = \kappa_r^{\text{out}} + \kappa_r^{\text{in}}$ for all blocks r. Ignoring constants, the log-likelihood is then

$$\log P(G \mid \theta, \omega', g) = \sum_u d_u \log \theta_u + \sum_{rs} (m_{rs} \log \omega'_{rs} - \kappa_r \kappa_s \omega'_{rs}), \tag{B.4}$$

where $d_u = d_u^{\text{out}} + d_u^{\text{in}}$. The MLEs for θ_u and ω'_{rs} are then

$$\hat{\theta}_u = d_u, \quad \hat{\omega}'_{rs} = \frac{m_{rs}}{\kappa_r \kappa_s}. \tag{B.5}$$

Thus $\hat{\omega}'_{rs} = \hat{\omega}_{rs} \hat{\rho}_{rs}$ where

$$\hat{\omega}_{rs} = \frac{\bar{m}_{rs}}{\kappa_r \kappa_s} \quad \text{and} \quad \hat{\rho}_{rs} = \frac{m_{rs}}{\bar{m}_{rs}}, \tag{B.6}$$

recovering (2.12).

Appendix C. Bayesian estimation for DG models

Bayesian inference focuses on posterior distributions of parameters rather than on point estimates. In hierarchical models such as DG-DDC, the full Bayesian posterior of the θ parameters (omitting the other parameters g and ω) is

$$P(\theta \mid G) = \int P(\theta \mid G, \psi) P(\psi \mid G)\, \mathrm{d}\psi. \tag{C.1}$$

Here we employ the Empirical Bayesian method, and use point estimates for the hyperparameters ψ, namely their MLEs $\hat{\psi}$,

$$\begin{aligned} \hat{\psi} &= \operatorname*{argmax}_{\psi} P(G \mid \psi) \\ &= \operatorname*{argmax}_{\psi} \int P(G \mid \theta, \psi)\, P(\theta \mid \psi)\, \mathrm{d}\theta. \end{aligned} \tag{C.2}$$

With this approximation, we have

$$\begin{aligned} P(\theta \mid G) &\approx P(\theta \mid G, \hat{\psi}) \\ &= \frac{P(G \mid \theta)\, P(\theta \mid \hat{\psi})}{P(G \mid \hat{\psi})} \\ &= \frac{P(G \mid \theta)\, P(\theta \mid \hat{\psi})}{\int P(G \mid \theta, \hat{\psi})\, P(\theta \mid \hat{\psi})\, \mathrm{d}\theta}, \end{aligned} \tag{C.3}$$

where we used Bayes' rule in the second line.

Computing the posterior $P(\theta \mid G)$ is usually difficult, as the integral in the denominator of (C.3) is often intractable. However, with a clever choice of the prior distribution $P(\theta \mid \psi)$, we can work out an analytic solution. It is called the *conjugate prior* of the likelihood term. We focus here on DG-DDC; the calculations for other DG models are similar.

Say that a random variable X is Gamma-distributed with parameters α and β, and write $X \sim \Gamma(\alpha, \beta)$, if its probability distribution is

$$f(x; \alpha, \beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha - 1}\, \mathrm{e}^{-\beta x}. \tag{C.4}$$

In DG-DDC, the likelihood (2.4) can be written (where we have plugged in the MLEs for ω, and substituted $\kappa_r^{\mathrm{out}} = \sum_{u:g_u = r} d_u^{\mathrm{out}}$)

$$P(G \mid \theta^{\mathrm{out}}) = \frac{\prod_u (\theta^{\mathrm{in}})^{d_u^{\mathrm{in}}} \prod_{rs} \omega_{rs}^{m_{rs}}}{\prod_{uv} A_{uv}!} \prod_u (\theta_u^{\mathrm{out}})^{d_u^{\mathrm{out}}} \exp(-\theta_u^{\mathrm{out}}). \tag{C.5}$$

If we assume that the θ^{in} and θ^{out} for each u are independent, this is proportional to a product of Gamma distributions with parameters $\alpha = d_u^{\mathrm{out}} + 1$ and $\beta = 1$ for each θ_u^{out}.

A natural conjugate prior for Gamma distributions is the Gamma distribution itself. Let the hyperparameters ψ_r^{out} for each block r consist of a pair $(\alpha_r^{\text{out}}, \beta_r^{\text{out}})$, and consider the prior

$$\theta_u^{\text{out}} \sim \Gamma(\alpha_{g_u}^{\text{out}}, \beta_{g_u}^{\text{out}}). \tag{C.6}$$

That is,

$$P(\theta_u^{\text{out}} \mid \psi_{g_u}^{\text{out}}) = \frac{(\beta_{g_u}^{\text{out}})^{\alpha_{g_u}^{\text{out}}}}{\Gamma(\alpha_{g_u}^{\text{out}})} (\theta_u^{\text{out}})^{\alpha_{g_u}^{\text{out}}-1} \exp(-\beta_{g_u}^{\text{out}}\theta_u^{\text{out}}). \tag{C.7}$$

Multiplying this prior by the likelihood (C.5) stays within the family of Gamma distributions, and simply updates the parameters:

$$\begin{aligned} P(\theta_u^{\text{out}} \mid G) &\propto P(\theta_u^{\text{out}} \mid \psi_{g_u}^{\text{out}}) P(G \mid \theta^{\text{out}}) \\ &\propto (\theta_u^{\text{out}})^{\alpha_{g_u}^{\text{out}}+d_u^{\text{out}}-1} \exp(-\theta_u^{\text{out}}(\beta_{g_u}^{\text{out}}+1)). \end{aligned} \tag{C.8}$$

Thus, the posterior distribution is

$$\theta_u^{\text{out}} \sim \Gamma(\alpha_{g_u}^{\text{out}} + d_u^{\text{out}}, \beta_{g_u}^{\text{out}} + 1). \tag{C.9}$$

Note that if we use a uninformative prior, i.e., in the limit $\alpha_{g_u}^{\text{out}} = 1$ and $\beta_{g_u}^{\text{out}} = 0$, the Gamma prior reduces to a uniform prior. The maximum *a posteriori* (MAP) estimate of θ_u^{out} is

$$\hat{\theta}_u^{\text{out}} = d_u^{\text{out}}, \tag{C.10}$$

and similarly for θ_u^{in}, just as we obtained for the MLEs in (2.6).

However, our goal is to integrate over θ, not focus on its MAP estimate. So let us continue the Bayesian analysis. Assuming the θ parameters are independent, then their joint posterior is simply a product of their individual posteriors

$$\begin{aligned} P(\theta|G) &= \prod_u P(\theta_u^{\text{out}}|G) P(\theta_u^{\text{in}}|G) \\ &= \prod_u f(\theta_u^{\text{out}}; \alpha_{g_u}^{\text{out}} + d_u^{\text{out}}, \beta_{g_u}^{\text{out}} + 1) f(\theta_u^{\text{in}}; \alpha_{g_u}^{\text{in}} + d_u^{\text{in}}, \beta_{g_u}^{\text{in}} + 1). \end{aligned} \tag{C.11}$$

Then we can calculate the integral in (C.2) and (C.3) by the simple algebra:

$$\begin{aligned} &\int P(G \mid \theta, \psi) P(\theta \mid \psi) \, \mathrm{d}\theta \\ &= \frac{P(G \mid \theta) P(\theta \mid \psi)}{P(\theta \mid G)} \\ &= \frac{\prod_u f(\theta_u^{\text{out}}; d_u^{\text{out}} + 1, 1) f(\theta_u^{\text{in}}; d_u^{\text{in}} + 1, 1) f(\theta_u^{\text{out}}; \alpha_{g_u}^{\text{out}}, \beta_{g_u}^{\text{out}}) f(\theta_u^{\text{in}}; \alpha_{g_u}^{\text{in}}, \beta_{g_u}^{\text{in}})}{\prod_u f(\theta_u^{\text{out}}; \alpha_{g_u}^{\text{out}} + d_u^{\text{out}}, \beta_{g_u}^{\text{out}} + 1) f(\theta_u^{\text{in}}; \alpha_{g_u}^{\text{in}} + d_u^{\text{in}}, \beta_{g_u}^{\text{in}} + 1)} \\ &= \frac{\prod_u {\beta_{g_u}^{\text{out}}}^{\alpha_{g_u}^{\text{out}}} {\beta_{g_u}^{\text{in}}}^{\alpha_{g_u}^{\text{in}}} \Gamma(\alpha_{g_u}^{\text{out}} + d_u^{\text{out}}) \Gamma(\alpha_{g_u}^{\text{in}} + d_u^{\text{in}})}{\prod_u (\beta_{g_u}^{\text{out}} + 1)^{\alpha_{g_u}^{\text{out}}+d_u^{\text{out}}} (\beta_{g_u}^{\text{in}} + 1)^{\alpha_{g_u}^{\text{in}}+d_u^{\text{in}}} \Gamma(d_u^{\text{out}} + 1) \Gamma(d_u^{\text{in}} + 1) \Gamma(\alpha_{g_u}^{\text{out}}) \Gamma(\alpha_{g_u}^{\text{in}})}. \end{aligned} \tag{C.12}$$

Now that the dependence of the numerator and denominator on θ has cancelled out, the integral is a function only of the hyperparameters ψ, making it possible to do the point estimate of ψ in (C.2). In our case, optimizing for $\hat{\psi}$ requires some numeric techniques, but it is nonetheless doable.

Empirical Bayesian solution not only gives better approximation to the original problem, it also makes it possible to integrate prior knowledge if available. On top of that, because the posterior is now a direct function of the hyperparameters ψ, we no longer have to worry about the Poisson noise when estimating ψ indirectly from degrees.

On a final note, the above result only holds for Gamma priors. With any other prior, the integral may not be this simple.

Appendix D. Power-law distribution with upper bound

In this section, we show that imposing an upper bound on our power-law distributions in order to ensure a certain average degree does not appreciably change the procedure of [24] for estimating the exponent. Suppose x is distributed as a power-law lower bound $x_{\min}$, upper bound $x_{\max}$, and exponent $\alpha > 0$. Then

$$p(x) = \frac{\alpha - 1}{x_{\min}^{1-\alpha} - x_{\max}^{1-\alpha}} x^{-\alpha}, \quad x_{\min} \leqslant x \leqslant x_{\max}. \tag{D.1}$$

Given a random sample $\mathbf{x} = \{x_1, \ldots, x_n\}$ drawn from this distribution independently, the likelihood function is

$$p(\mathbf{x}) = \prod_{i=1}^{n} \frac{\alpha - 1}{x_{\min}^{1-\alpha} - x_{\max}^{1-\alpha}} x_i^{-\alpha} = \left(\frac{\alpha - 1}{x_{\min}^{1-\alpha} - x_{\max}^{1-\alpha}} \right)^n \prod_{i=1}^{n} x_i^{-\alpha}. \tag{D.2}$$

Thus, the log-likelihood is

$$\log p(\mathbf{x}) = n(\log(\alpha - 1) - \log(x_{\min}^{1-\alpha} - x_{\max}^{1-\alpha})) - \alpha \sum_{i=1}^{n} \log x_i. \tag{D.3}$$

Taking the derivative with respect to α gives

$$\frac{\partial \log p(\mathbf{x})}{\partial \alpha} = n \left(\frac{1}{\alpha - 1} + \frac{x_{\min}^{1-\alpha} \log x_{\min} - x_{\max}^{1-\alpha} \log x_{\max}}{x_{\min}^{1-\alpha} - x_{\max}^{1-\alpha}} \right) - \sum_{i=1}^{n} \log x_i. \tag{D.4}$$

Setting (D.4) to zero, we obtain

$$\frac{1}{\alpha - 1} + \frac{x_{\min}^{1-\alpha} \log x_{\min} - x_{\max}^{1-\alpha} \log x_{\max}}{x_{\min}^{1-\alpha} - x_{\max}^{1-\alpha}} = \frac{\sum_{i=1}^{n} \log x_i}{n}. \tag{D.5}$$

If $x_{\min} = 1$ and $x_{\max} \to \infty$, then solving (D.5) gives the MLE for α just as in (2.18).