Oriented and Degree-generated Block Models: Generating and Inferring Communities with Inhomogeneous Degree Distributions

Yaojia Zhu[1⋆], Xiaoran Yan[1], and Cristopher Moore[1,2]

[1] University of New Mexico
Albuquerque NM 87131 USA
[2] Santa Fe Institute
1399 Hyde Park Road, Santa Fe NM 87501, USA

Abstract. 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 block model can accommodate arbitrary degree distributions within communities. However, 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 orientation 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, these new block models achieve a higher accuracy than the standard or degree-corrected block models.

Keywords: We would like to encourage you to list your keywords within the abstract section

1 Introduction

In many real world networks, vertices can be divided into *communities* or *modules* based on their connecting patterns. Social communities can be forged by interactions in daily activities like karate training [22]. The blogosphere contains groups of linked blogs with similar political views [1]. English words can be tagged as different parts of speech based on their adjacencies in large texts [15]. Understanding these taxonomic structures is crucial in deciphering these topology data. There has been a great deal of work on efficient algorithms for community detection in networks (see [10, 19] for reviews).

The *Stochastic block model* (SBM) [9, 11, 20, 3] is a popular model-based approach for *functional* community detection. It partitions the vertices into communities or *blocks*, where vertices belonging to the same block are *stochastically*

⋆ This work was supported by the McDonnell Foundation.

equivalent [21] 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 general definition of *functional* communities, block models can capture various community structures, including assortative, disassortative, satellite communities and mixtures of them [16, 17, 14, 13, 8, 7].

Given the block memberships, the SBM assumes that each edge is generated independently, and follows a Bernoulli distribution solely determined by block memberships of its endpoints. Since edges in the SBM are independent, and since every pair of vertices in a given pair of blocks have a link with the same probability, for large n the degree distribution within each block is Poisson. As a consequence, the SBM dictates that vertices with very different degrees are unlikely to be in the same block. This leads to problems when modeling the networks like the political blogs, since within each community, e.g. liberal or conservative, there are both highly popular and isolated vertices at the same time.

Recently, Karrer and Newman [12] developed the *degree-corrected block model* for undirected networks (DC). They add a parameter for each vertex, which controls its expected degree. By setting these parameters equal to the observed degrees, the DC can accommodate arbitrary degree distributions within communities. This removes the model's tendency to separate high-degree and low-degree vertices into different communities.

On the other hand, the degree-corrected 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 needs to be explained. For this reason, DC 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 shouldn't, and DC fails to do so even when it should.

For directed graphs, the natural generalization of DC, which we call *directed degree-corrected block model* (DDC), 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 the other way around. The ratio of each vertex's in- and out-degree could be very indicative for its block membership, and leveraging this part of the degree information would be essential.

In this paper, we first propose the *oriented degree-corrected block model* (ODC), which combines the strengths of the degree-corrected and uncorrected block models. ODC is able to utilize the edge orientations for community detection by only correcting the total degrees instead of the in- and out-degrees separately. We show that for networks with strongly asymmetric behavior between communities, including synthetic networks and networks of word adjacencies in English text, ODC achieves a higher accuracy than either the original stochastic block model or the degree-corrected block model.

We then propose the *degree-generated block model* (DGBM), which treats the expected degree of each vertex as generated from prior distributions in each community, such as power laws with different exponent and cutoffs in each com-

munity. We then include the probability of these degrees in the likelihood of a given block assignment. In this way, the model captures the dependence of the degree distributions on the community structure. These degree-generated block models automatically strike 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 right communities on the other. DGBM works especially well for detecting community structures where communities have highly inhomogeneous degree distributions. These degree distributions differ enough between communities so that we should use vertex degrees to help us classify the vertices.

Empirical study show that DGBM are indeed a very robust choice especially when the connecting pattern alone is not enough to detect the community structure. DGBM has a further advantage in faster convergence as it reshapes the landscape of the parameter space, providing the searching algorithm a short cut to the desired community structure.

These new variants of the stochastic block model give us the best of both worlds: they can 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 these 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

In this section, we review the degree-corrected block model of [12], and present our variations on it, namely oriented and degree-generated block models.

2.1 Background: degree-corrected block models

Throughout, we use N and M to denote the number of vertices and edges, and K to denote the number of blocks. The problem of determining the number of blocks is a subtle model selection problem, which we do not address here.

In the original stochastic block model, the entries A_{uv} of the adjacency matrix are independent and Bernoulli-distributed, with $P(A_{uv} = 1) = p_{g_u,g_v}$. Here g_u is the block to which u belongs, where p is a $K \times K$ matrix. Karrer and Newman [12] 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 analyze than one that controls the degree of each vertex exactly.

The likelihood with which this model generates a graph G is then

$$P(G \mid \theta, \omega, g) = \prod_{u,v} \frac{(\theta_u \theta_v \omega_{g_u g_v})^{A_{uv}}}{A_{uv}!} \exp\left(-\theta_u \theta_v \omega_{g_u g_v}\right) . \tag{1}$$

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 $\sum_{u:g_u=r} \theta_u = 1$ for each block r. Under these constraints, the maximum likelihood estimates (MLEs) for the θ parameters are $\hat{\theta}_u = d_u/\kappa_{g_u}$ where $\kappa_r = \sum_{u:g_u=r} d_u$ is the total degree of the vertices in block r. For each pair of blocks r, s, the MLE for ω_{rs} is then m_{rs}, the number of edges connecting block r to block s (where edges within blocks are counted twice). Substituting these MLEs for θ and ω then gives the log-likelihood

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

2.2 Directed and oriented degree-corrected models

The natural extension of the degree-corrected model to directed networks, which we call the directed degree-corrected block model (DDC), has two parameters $\theta_u^{\text{in}}, \theta_u^{\text{out}}$ for each vertex. The number of directed edges from u to v is again Poisson-distributed,

$$A_{uv} \sim \text{Poi}(\theta_u^{\text{out}} \theta_v^{\text{in}} \omega_{g_u, g_v}) \,.$$

With the constraints $\sum_{u:g_u} \theta_u^{\text{in}} = \sum_{u:g_u} \theta_u^{\text{out}} = 1$ for each block r, the MLEs for these parameters (see the online version) are

$$\hat{\theta}_u^{\text{out}} = \frac{d_u^{\text{out}}}{\kappa_{g_u}^{\text{out}}} \,, \quad \hat{\theta}_u^{\text{in}} = \frac{d_u^{\text{in}}}{\kappa_{g_u}^{\text{in}}} \,, \quad \hat{\omega}_{rs} = m_{rs} \,, \tag{3}$$

where $\kappa_r^{\text{in}}, \kappa_r^{\text{out}}$ denote the total in- and out-degrees in block r and m_{rs} is the number of directed edges from block r to block s. Substituting these MLEs gives the log-likelihood

$$\log P(G \mid g) = \sum_{r,s=1}^{K} m_{rs} \log \frac{m_{rs}}{\kappa_r^{\text{out}} \kappa_s^{\text{in}}} \,. \tag{4}$$

In the DDC, the in- and out-degrees of each vertex are completely specified by the θ parameters, at least in expectation. Thus the DDC lets vertices with arbitrary in- and out-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, the DDC cannot use them to infer community structure. Indeed, it cannot even take advantage of the orientations of the edges, and as we will see below it performs quite poorly on networks with strongly asymmetric community structure.

To deal with this, we present a partially degree-corrected block model capable of taking advantage of edge orientations, which we call the *oriented degree-corrected block mode* (ODC). Following the maxim that we can only use the information that we try to generate, we separate the generation of the edge orientations from the degrees.

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 can be thought of as generating $\bar{G}$ according to the undirected degree-corrected 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 function for such a model is

$$\log P(G \mid g, \rho) = \log P(\bar{G} \mid g) + \log P(G \mid \bar{G}, g, \rho)\,. \tag{5}$$

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

$$\log P(G \mid g) = \frac{1}{2} \sum_{r,s=1}^{K} \bar{m}_{rs} \log \frac{\bar{m}_{rs}}{\bar{\kappa}_r \bar{\kappa}_s}\,. \tag{6}$$

The orientation term is

$$\log P(G \mid G, g, \rho) = \sum_{rs} m_{rs} \log \rho_{rs} = \frac{1}{2} \sum_{rs} (m_{rs} \log \rho_{rs} + m_{sr} \log \rho_{sr})\,, \tag{7}$$

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

$$\hat{\rho}_{rs} = m_{rs}/\bar{m}_{rs}\,. \tag{8}$$

As (7) is maximized when $\hat{\rho}_{rs}$ are near 0 or 1 for $r \neq s$, the edge orientation term prefers highly asymmetric inter-block connections. Since $\hat{\rho}_{rr} = 1/2$ for any block r, it also prefers disassortative mixing, with as few connections as possible within blocks.

Substituting the MLEs for ρ and combining (6) with (7), the total log-likelihood is

$$\log P(G \mid g) = \sum_{r,s=1}^{K} m_{rs} \log \frac{m_{rs}}{\kappa_r \kappa_s}\,. \tag{9}$$

As we show in the online version, we can also view the ODC as a special case of the DDC, where we add the constrain $\theta_u^{\text{in}} = \theta_u^{\text{out}}$ for all u. 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

$$\text{SBM} \leq \text{ODC} \leq \text{DDC}\,,$$

where $A \leq B$ means that model A is a special case of model B, or that B is an elaboration of A.

2.3 Degree-generated block model

Another way to utilize the degree information for community detection is through the degree distributions in each community. In all block models, the degree of each vertex is asymptotically Poisson-distributed. But in the SBM, all the vertices in each community have the same expected degree, while DC or DDC fully specify each vertex's expected degree using θ.

A natural way to make explicit use of the degree information is to force the model to generate the vertex degrees in each community according to some distribution whose parameters differ from one community to another. To maintain the tractability of the random multigraph, we generate the parameters θ, and thus the expected degree of each vertex, rather than the degrees themselves. Given a vertex u, we first generate its expected degree θ_u from a prior degree distribution according to its block membership, then use it as the θ parameter in the degree-corrected block model to generate edges. The *degree-generated block model* is thus a *hierarchical model*, which extends previous degree-corrected block models by adding a degree generation (DG) stage on top.

Likelihood functions and parameter estimation
Given the number of vertices N and number of blocks K, the generative process for undirected networks is described in Table. 1.

Table 1. Degree-generated block model (undirected)

For each vertex $u = 1, ..., N$
Generate $\theta_u \mid g_u, \psi_{1..K}, \sim \mathcal{F}_{g_u}(\cdot \mid \psi_{g_u})$
For each pair of vertices (u, v)
Generate $A_{uv} \sim \mathrm{Poi}(\theta_u \theta_v \omega_{g_u g_v})$

In the table, ψ_r are the hyper parameters of the degree distributions for block r. The form (cdf) of the degree distributions are denoted as $\mathcal{F}_r(\cdot|\psi_r)$, which are given using domain knowledge. If such prior knowledge does not exist, we shall pick distributions which best fit the observed degree sequence. Note that the second stage is just the DC model (1), and we call the whole model DG-DC.

Table 2. Degree-generated block model (directed)

For each vertex $u = 1, ..., N$
Generate $\theta_u^{\mathrm{in}} \mid g_u, \psi_{1..K}^{\mathrm{in}}, \sim \mathcal{F}_{g_u}^{\mathrm{in}}(\cdot \mid \psi_{g_u}^{\mathrm{in}})$
Generate $\theta_u^{\mathrm{out}} \mid g_u, \psi_{1..K}^{\mathrm{out}}, \sim \mathcal{F}_{g_u}^{\mathrm{out}}(\cdot \mid \psi_{g_u}^{\mathrm{out}})$
For each pair of vertices (u, v)
Generate $A_{uv} \sim \mathrm{Poi}(\theta_{uv} \omega_{g_u g_v})$
Generate $A_{vu} \sim \mathrm{Poi}(\theta_{vu} \omega_{g_v g_u})$

For directed graphs, we have ψ_r^{in} and ψ_r^{out} as the hyper parameters of the in- and out-degree distributions for block r respectively. Their degree distributions are denoted as $\mathcal{F}_{g_u}^{\text{in}}(\cdot|\psi_{g_u}^{\text{in}})$ and $\mathcal{F}_{g_u}^{\text{out}}(\cdot|\psi_{g_u}^{\text{out}})$. Both of them can be specified by domain knowledge. For directed graphs, DG stage can precede either DDC or ODC. This flexibility comes from the following options available for the degree-correction term θ_{uv}, which we should name accordingly:

$$\theta_{uv} = \begin{cases} \theta_u^{\text{out}}\theta_v^{\text{in}} & \text{DG-DDC} \\ \theta_u\theta_v & \text{DG-ODC} \end{cases}.$$

Here $\theta_u = \theta_u^{\text{out}} + \theta_u^{\text{in}}$ is the total degree of vertex u. We also have the option to generate total degrees instead, then generate directed networks with orientation parameters ρ as we did in (7). But for the rest of the paper, we use the above directed version for DG-ODC.

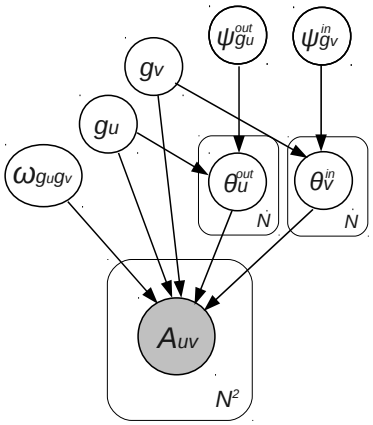

Fig. 1. Graphical model of DG-DDC

Under DG, θ parameters θ are now generated by hyper parameters ψ, as demonstrated by the graphical model in Fig. 1. The log-likelihood function is thus composed of two terms, for DG-DC:

$$\log P(G \mid g, \theta, \omega, \psi) = \log P(\theta \mid g, \psi) + \log P(G \mid \theta, g, \omega)\ . \tag{10}$$

where the first terms is for degree-generation stage, with the second term for edge-generation stage. Similarly, we have DG-DDC:

$$\log P(\boldsymbol{G} \mid g, \theta^{\text{out}}, \theta^{\text{in}}\omega, \psi) = \log P(\theta^{\text{out}}, \theta^{\text{in}} \mid g, \psi) + \log P(\boldsymbol{G} \mid \theta^{\text{out}}, \theta^{\text{in}}, g, \omega)\ . \tag{11}$$

And DG-ODC:

$$\log P(\boldsymbol{G} \mid g, \theta, \omega, \psi) = \log P(\theta^{\text{out}}, \theta^{\text{in}} \mid g, \psi) + \log P(\boldsymbol{G} \mid \theta, g, \omega)\ . \tag{12}$$

The inference of all 3 DG models are very similar. For simplicity, we shall focus on the parameter estimation of DG-DC. Readers can easily generalize the result to DG-DDC and DG-ODC.

In hierarchical models like DG-DC, estimating MLEs for both θ and ψ is usually difficult, as their joint distribution is often intractable. Here we shall propose an simple approximation that is intuitive and efficient.

We first estimate θ to maximize only the second term in (10). Notice that if we only consider the edge-generation term, we have the original DC model, and the MLEs for θ are the observed degrees. Substituting θ with these degrees, we can then estimate the hyper-parameters ψ to maximize the first term. As a result, we simply reuse the degree-corrected block models and the degree-generation term becomes the log-likelihood of generating the observed degrees.

As all the θ parameters are generated independently, for undirected networks the degree generation term is

$$\log P(\theta \mid g, \psi) = \sum_u \log \left(\mathcal{F}_{g_u}(d_u|\psi_{g_u})\right) . \tag{13}$$

Notice that if we use a uninformative piror, i.e., assuming $P(\theta \mid \psi)$ is a uniform distribution, this approximation is exact. With any other prior, the MLEs would be different. Under this approximation, the degree generation term acts merely as a penalty term rather than an intergral part of the likelihood function. However, it is enough to achieve our goal of leveraging the degree information for community detection.

Degree independent community structures favored by the DC model are no longer necessarily the most likely of the whole likelihood function. If they are too far off the prior degree distribution, they will be properly penalized for their poor fit. This would leave the door open for other community structures that might not be as a good fit to the edges, but compensates with a much better fit to the degrees.

Next, we shall show one of the most popular form for $\mathcal{F}_{g_u}(\cdot|\psi_{g_u})$, and the corresponding estimation of the hyper parameter ψ under the approximation $\hat{\theta_u} = d_u$.

Power-law degree generation

We present a power-law DG here to illustrate how we can use DG to handle degree sequences with power-law tail. Degree-generated block model for networks with other degree distributions can be established in the same way. We focus on power-law in this paper because it is prevalent in all kinds of real world networks. First popularized by the Preferential attachment model [4], power-law degree distributions has been hot topic in networks across different disciplines [10, 19, 2, 5].

With a power-law distribution, the degrees are highly skewed and degree-correction becomes necessary. Thus, power-law tail is an idea test bed for our degree-generated block model where degree-correction is crucial but it alone cannot achieve satisfying performance.

Although the degrees are discrete values, the θ parameters can be continuous. We fit the observed degrees to both discrete and continuous power-law distri-

butions. Empirically we find both of them perform very well. The only notable difference between them is the running time. Fitting degrees to a continuous power-law distribution can be faster because the MLEs of its parameters can be calculated analytically.

In the discrete case, for any vertex u in block r, the probability that it has degree d_u (here d_u can be in-, out- or total-degree of vertex u) is the following

$$p(d_u) = \begin{cases} \beta_r & d_u < d_{\min} \\ \frac{(1-\beta_r) d_u^{-\alpha_r}}{\zeta(\alpha_r, d_{\min})} & d_u \geq d_{\min} \end{cases} \tag{14}$$

Here we have the Riemann zeta function $\zeta(\alpha, x) = \sum_{n=x}^{\infty} n^{-\alpha}$. And $d_{\min}$ is the minimal degree of the power-law tail. The hyper parameters of the power-law degree distribution are $\psi_r = \{\alpha_r, \beta_r\}$.

The MLEs for β is straightforward, that is

$$\beta_r = \phi_r / n_r \,. \tag{15}$$

Here ϕ_r is the number of vertices in block r that have degree less than $d_{\min}$. The MLEs for α is a little more complicated. A detailed description about the MLEs for power-law distribution can be found in [6]. There is no analytical solution, but α can be estimated numerically. Given a power-law degree sequence in any block r, that is $d = \{d_1, d_2, ..., d_n\}$, here $d_i \geq d_{\min}$ are integers larger than or equal to $d_{\min}$, the log-likelihood function for the power-law parameter α is

$$\log P(\alpha) = -n \ln \zeta(\alpha, d_{\min}) - \alpha \sum_{i=1}^{n} \ln d_i \,. \tag{16}$$

Then, we can simply search α to maximize (16).

In the continuous case, for any vertex u, we have

$$p(d_u) = \begin{cases} \beta_r & d_u < d_{\min} \\ \frac{(1-\beta_r)(\alpha_r - 1)}{d_{\min}} \left(\frac{d_u}{d_{\min}} \right)^{-\alpha_r} & d_u \geq d_{\min} \end{cases} \tag{17}$$

The MLEs for β is same as (15). Now the MLEs for α can be solved analytically. Given a power-law degree sequence $d = \{d_1, d_2, ..., d_n\}$, $d_i \geq d_{\min}$ in any block r, the MLEs for α_r is

$$\hat{\alpha}_r = 1 + n \left[\sum_{i=1}^{n} \ln \frac{d_i}{d_{\min}} \right]^{-1} \tag{18}$$

3 Experiments

3.1 Generation of synthetic networks

Undirected networks with specific degree distributions in each community can be generated by DG-DC. For each vertex u, we first generate θ_u following the

distribution $\mathcal{F}_{g_u}(\cdot|\psi_{g_u})$. After generating θ parameters, we need to choose a symmetric ω matrix, which satisfies $\sum_s \omega_{rs} = \kappa_r$ for each block r. Here κ_r is the summation of the θ values in block r, that is $\kappa_r = \sum_{u:g_u=r} \theta_u$. Then the number of edges connecting each pair of vertices u, v is drawn from a Poisson distribution with mean $\theta_u\theta_v\omega_{g_ug_v}/(\kappa_{g_u}\kappa_{g_v})$ or $\theta_u\theta_v\omega_{g_ug_v}/(2\kappa_{g_u}\kappa_{g_v})$ if $u = v$. We can also generate the edges by first determining the number of edges for each air of blocks and then assigning the edge ends to vertices, which is described in [12].

Directed networks can be generated by DG-DDC or DG-ODC in the way illustrated in Table 2. For each vertex u, we first generate θ_u^{in} and θ_u^{out}, each of them follows a specific distribution: $\mathcal{F}_{g_u}^{\text{in}}(\cdot|\psi_{g_u}^{\text{in}})$ and $\mathcal{F}_{g_u}^{\text{out}}(\cdot|\psi_{g_u}^{\text{out}})$ respectively.

For DG-DDC, given a block assignment g, we have $\kappa_r^{\text{out}} = \sum_{u:g_u=r} \theta_u^{\text{out}}$ and $\kappa_r^{\text{in}} = \sum_{u:g_u=r} \theta_u^{\text{in}}$. Now ω_{rs} can be chosen to specify the community structure, and it should satisfy the constraints $\kappa_r^{\text{out}} = \sum_s \omega_{rs}$ and $\kappa_r^{\text{in}} = \sum_s \omega_{sr}$. After choosing ω_{rs}, the number of directed edges from vertex u to v is Poisson distributed with mean equal to $\theta_u^{\text{out}}\theta_v^{\text{in}}\omega_{g_ug_v}/(\kappa_{g_u}^{\text{out}}\kappa_{g_v}^{\text{in}})$.

As to DG-ODC, for each vertex u, we combine the parameters θ_u^{in} and θ_u^{out} into one by setting $\theta_u = \theta_u^{\text{out}} + \theta_u^{\text{in}}$. Given a block assignment g, we have $\kappa_r = \sum_{u:g_u=r} \theta_u$. Now ω_{rs} can be chosen to specify the community structure, and it should satisfy the constraints $\kappa_r = \sum_s (\omega_{rs} + \omega_{sr})$. After choosing ω_{rs}, the edges can be generated. The number of directed edges from vertex u to v is Poisson distributed with mean equal to $\theta_u\theta_v\omega_{g_ug_v}/(\kappa_{g_u}\kappa_{g_v})$.

3.2 Synthetic test on undirected networks

We generate undirected networks using DG-DC. The degree sequence in the first block follows a power-law tail with $\alpha = 2.5$ and $d_{\min} = 1$. In the second block, the degree sequence is Poisson-distributed with mean 20. We place about 1200 vertices in each block.

Just like in [12], we use a parameter λ to interpolate linearly between a fully random network with no community structure to some planted one as the following

$$\omega_{rs} = \lambda\omega_{rs}^{\text{planted}} + (1-\lambda)\omega_{rs}^{\text{random}}\,. \tag{19}$$

Here $\omega_{rs}^{\text{random}} = \kappa_r\kappa_s/2M$, and we set $\omega_{rs}^{\text{planted}}$ to be the following

$$\omega^{\text{planted}} = \begin{pmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{pmatrix}. \tag{20}$$

Thus, all edges are placed within communities.

We plotted the result in Fig. 2. Each point on the graph is based on 30 randomly generated networks. For each network, we choose the best result from 10 initials. For each initial, 1 million MCMC steps are executed. The green points are obtained from DG-DC and the red points are from the original DC without degree-generation. For each color, the square ones are from initials with true block assignment, and the circle ones are from random initials. DG-DC works very well even for very small λ values. DG-DC can classify most of the

vertices correctly even when there is no community structure for DC because the block memberships for most of the vertices can be well determined only based on the degree sequence information. True block assignment initialization cannot help DG-DC. It improves DC when λ is close to the phase transition point. We checked the likelihood values found by DC-T (DC initialized with true block assignment) and mboxDC-R (DC initialized randomly) at $\lambda = 0.4$, and found DC-R achieved higher likelihood value. That means DC-T actually got stuck into a local optima.

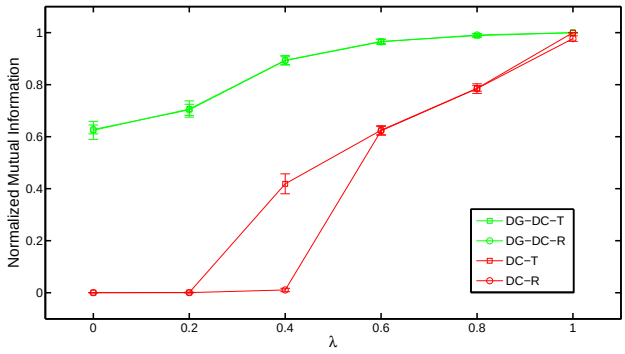

Fig. 2. Tests on networks generated by DG-DC

3.3 Synthetic test on directed networks

In our following synthetic test, we generate directed networks using DG-ODC. We place the vertices into two blocks and each block contains about 1200 vertices. In the first block, both out- and in-degrees are Poisson distributed with mean 20. In the second block, both out- and in-degrees are power-law distributed with $\alpha = 2.5$ and $d_{\min} = 1$. For ODC, $\omega_{rs}^{\text{random}}$ is the one that makes all ρ_{rs} to be equal, i.e., 1/2. In that case all edges are oriented randomly, and $\omega_{rs} = \omega_{sr}$. On the other hand, the corresponding undirected network should also be fully random with respect to DC, namely $\omega_{rs} + \omega_{sr} = \kappa_r\kappa_s/2M$. Thus, we have $\omega_{rs}^{\text{random}} = \kappa_r\kappa_s/4M$. We set $\omega_{rs}^{\text{planted}}$ to be totally asymmetric. For $K = 2$, we have

$$\omega^{\text{planted}} = \begin{pmatrix} (\kappa_1 - \omega_{12})/2 & \omega_{12} \\ 0 & (\kappa_2 - \omega_{12})/2 \end{pmatrix}, \tag{21}$$

where $\omega_{12} \leq \min(\kappa_1, \kappa_2)$. In this test, we choose $\omega_{12} = \frac{1}{2}\min(\kappa_1, \kappa_2)$.

We can see in Fig. 3, DG-ODC and DG-DDC have very similar performance all the way and both of them can achieve much better performance than the original block models. This meets our expectation very well. As now the degree sequences contain so much information about the block memberships, both

DG-ODC and DG-DDC works perfectly. DG-ODC doesn't outperform DG-DDC because the orientation information cannot help more if the degree sequence information is already used. Without degree generation, ODC outperforms DDC for large lambda values, this is because without leveraging the degree sequence information, edge orientations can still help.

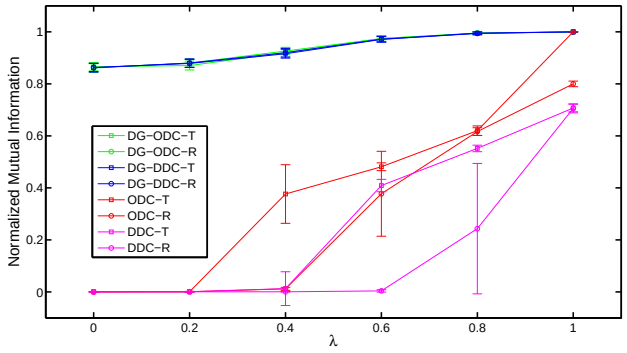

Fig. 3. Tests on networks generated by DG-ODC

In our following test, we generate directed networks using DG-DDC. We place the vertices into two blocks and each block contains about 1200 vertices. The out-degrees of block 1 and in-degrees of block 2 are Poisson-distributed with mean 20. The in-degrees of block 1 and out-degrees of block 2 are power-law distributed with $\alpha = 1.8$ and $d_{\min} = 1$. The power-law degree distributions are upper bounded to make sure the average degree is also 20 (same as the Poisson mean). For DDC, we have

$$\omega_{rs}^{\text{random}} = \kappa_r^{\text{out}} \kappa_s^{\text{in}} / M \,. \tag{22}$$

We set the planted ω matrix as the following one with only off-diagonal entries. Thus, the planted community structure is disassortative. For any λ, the connections between block 1 and block 2 are almost symmetric.

$$\omega^{\text{planted}} = \begin{pmatrix} 0 & \kappa_1^{\text{out}} \\ \kappa_2^{\text{out}} & 0 \end{pmatrix} . \tag{23}$$

As presented in Fig. 4, one interesting thing is ODC-R works equally well for all λ. Although $\omega_{rs}^{\text{random}}$ for DDC is also a random ω for ODC, for small λ, ODC can achieve very good performance due to the degree distributions we used. As both of the out-degrees in block 1 and the in-degrees in block 2 are power-law distributed, some vertices in block 1 will have very high out-degrees and some vertices in block 2 will have very high in-degrees. We can imagine a lot of edges from block 1 to block 2 are connecting these high degree vertices.

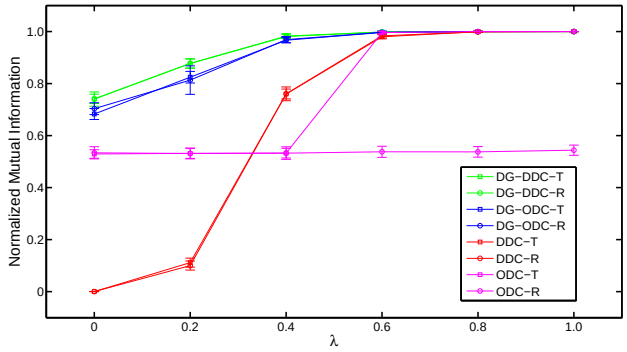

Fig. 4. Tests on networks generated by DG-DDC

If we exchange the block memberships of these vertices, the orientations of all those edges are reversed. This is a community structure preferred by ODC as now most of the inter-block edges are oriented to block 1. As the number of the high degree vertices is very small, mislabeling them does not matter much. That's why ODC can find a block assignment very close to the true one although mboxODC doesn't like the true block assignment either. When λ is large enough, ODC-T can find the planted community structure. We checked the log-likelihood values it found. For $\lambda \leq 0.8$, ODC-R actually found higher log-likelihood values than the one of the true block assignment. For $\lambda = 1$, ODC-R can only find lower log-likelihood values. That means when $\lambda = 0.6$ and 0.8, the true block assignment is actually a local optima for the ODC model. ODC-T got stuck there. But when $\lambda = 1$, namely the network is fully disassortative, then the true block assignment is a better, but ODC cannot find it within 1M MCMC steps if random initialization is adopted.

This phenomena is also found when we were trying to apply ODC on other symmetric connected networks, for example the political-blog network. Recall that the likelihood function for ODC, which is (5), has two parts. The first part generates the undirected network, which is DC. The second part generates the edge orientations. If the orientations are symmetric, there should be two modes in the landscape. One mode is the community structure preferred by the original DC, another mode is the one with highly asymmetric orientations. ODC will stuck into the second mode easily even if the network is highly assortative or disassortative and the true block assignment is the one with higher likelihood. ODC will also stuck into the first mode when true block assignment initials are used although the true block assignment may not necessary to be the one with maximum likelihood.

3.4 Empirical networks

In this section, we test the models on three word adjacency networks in which vertices are separated into two blocks: noun and adjective. "David" is the adjacency network of common adjectives and nouns in the novel David Copperfield by Charles Dickens [18]. "News" is the adjacency network of common (degree is larger than or equal to 10) adjectives and nouns obtained from News corpus. "Brown" is the giant component in the adjacency network of adjective and nouns in the Brown's corpus. All of them are asymmetric networks as adjectives are very likely to be followed by nouns, however nouns are not common to be followed by adjectives. Let's assume adjectives are in block 1 and nouns are in block 2, then we have $p_{12} > p_{21}$. For each network, we consider both multigraph (M) where multi-edges are included and simple graph (S) where multi-edges are ignored. Table. 3 revealed the basic information of these networks. The connection probability matrices are listed in Table. 4.

Table 3. Word adjacency networks

Network	#words	#adjective	#noun	#edges (S)	#edges (M)
David	112	57	55	569	1494
News	376	91	285	1389	2411
Brown	23258	6235	17023	66734	88930

Table 4. Connection probability matrices

David(S)		David(M)		News(S)		News(M)		Brown(S)		Brown(M)	
0.039	0.118	0.080	0.358	0.010	0.015	0.012	0.028	9.1e-05	3.4e-04	1.1e-04	4.4e-04
0.018	0.006	0.025	0.011	0.002	0.010	0.003	0.019	2.0e-05	8.8e-05	2.4e-05	1.2e-04

3.5 Comparison of degree-corrected models

Table. 5 compares the clustering performance for different block models, including SBM, DC, ODC and DDC. When applying DC to these directed networks, we simply ignore the edge orientations (the resulted network may contain multi-edges even though the original directed one doesn't). Both the percentage of correctly labeled vertices and the NMI value are listed for each model on each network. The results for "David" and "News" are based on 100 initials; for "Brown", 50 initials are used. All the initials are chosen randomly. For each model and each network, we take the best result among those initials. For each

initial we run the KL-heuristic [12] followed by 1 million MCMC steps. We also tested a naive heuristic (NH) algorithm which simply labels a vertex as adjective if its out-degree is larger than in-degree or noun if its out-degree is less than in-degree. For vertices with equal in- and out-degree, NH assigns its label randomly. We present the average performance of NH in Table. 5 based on 100 runs for "David" and "News" and 50 runs for "Brown". NH represents the performance we can achieve if we only use the edge orientations for community detection.

Table 5. Clustering results with random initialization

	David(S)	David(M)	News(S)	News(M)	Brown(S)	Brown(M)
SBM	84.8/0.423	57.1/0.051	62.2/0.006	57.2/0.018	70.0/0.001	70.1/7e-04
DC	91.1/0.566	91.1/0.568	56.4/0.084	56.6/0.083	55.6/0.020	55.5/0.015
ODC	87.5/0.462	87.5/0.470	56.4/0.084	56.4/0.029	75.3/0.311	80.3/0.318
DDC	70.5/0.128	51.8/8e-04	56.4/0.084	55.1/0.091	55.8/0.016	53.8/0.012
NH	84.4/0.395	86.6/0.449	72.8/0.215	73.8/0.233	78.0/0.309	78.1/0.314

a little better. ODC mislabeled 2731 while DC mislabeled 2621. But overall, ODC works much better than DDC on "News" and "Brown" networks.

For "David", DC works best and ODC also works pretty well. Both of them performs better than NH. After examining "David" more carefully, we found in this small network, three adjectives only have in-degree. They are "full", "glad" and "alone". ODC will mislabel them while DC labeled them correctly by just ignoring the edge orientations. In such a situation, we really cannot criticize ODC. As to SBM, it works well on "David(S)" but fails on "David(M)", this is because the degrees in the multigraph are more skewed than those in the simple one. It is a little surprising that DDC performs worst, and even worse than SBM. We learn a lesson here that a full degree-correction may make things worse even when the degrees in each community are quite inhomogeneous.

For "News", all the block models fail, even ODC doesn't work. We found there is a highly assortative community structure in the "News" network where about 90% edges are within commuities. All the block models will finally return a community structure very close to that one. As we mentioned earlier, ODC needs to make trade-off between the community structure preferred by DC and the one that is disassortative mixing and have highly asymmetric inter-block connections. Here, ODC sacrifices the second one. However, after checking the results returned by ODC for all initials, we found most of the times ODC works. Table. 6 presents the average clustering performances over 100 initials. The decimals outside the parentheses are the average percentages of correctly labelled vertices and the average NMI values between the best clustering the algorithm found in each

Table 6. Average clustering performances with random initialization

	David(S)	News(S)	Brown(S)
SBM	84.8(0)/0.423(0)	62.3(0.3)/0.005(0.001)	70.0(0)/0.001(0)
DC	91.1(0)/0.566(0)	56.4(0.08)/0.084(4e-04)	59.8(5.3)/0.052(0.037)
ODC	87.5(0)/0.462(0)	74.7(2.1)/0.237(0.020)	75.3(0)/0.311(0)
DDC	63.9(10.7)/0.096(0.108)	56.4(0.07)/0.084(4e-04)	53.5(2.0)/0.008(0.007)
NH	84.4(1.0)/0.395(0.023)	72.8(0.7)/0.215(0.012)	78.0(0.09)/0.309(0.003)

	David(M)	News(M)	Brown(M)
SBM	57.1(0)/0.051(0)	57.5(2.3)/0.017(0.006)	70.1(0)/7e-04(0)
DC	76.4(18.3)/0.344(0.264)	52.5(1.9)/0.025(0.030)	57.6(4.6)/0.032(0.028)
ODC	87.5(0)/0.470(0)	74.3(6.0)/0.232(0.058)	79.7(0.2)/0.317(6e-04)
DDC	59.4(7.6)/0.044(0.055)	52.4(1.9)/0.028(0.031)	52.5(1.6)/0.005(0.006)
NH	86.6(0)/0.449(0)	73.8(0.6)/0.233(0.009)	78.1(0.1)/0.314(0.002)

initial and the true clustering. The decimals in parentheses are the standard deviations. The average performance of ODC is pretty good. ODC returns that highly assortative community structure very occasionally during those initials, although that community structure has the highest likelihood value. On the other hand, if we really know something about the community structure we are looking for, finding out that one based on the results of those initials can be simple.

For "Brown", every block model fails except ODC. It seems SBM works, as it labels 70% vertices correctly. However SBM achieves this by simply putting almost all vertices in one block. That's why the NMI values are very low. ODC has very close performance to NH on both the simple and multigraph.

All the block model tests discussed so far are based on random initializations. However, NH is actually a perfect block assignment initializer for the block models on these word adjacency networks. The following tests are based on NH initializations, namely the NH results are used for block membership initializations.

Table 7. Clustering results with NH initialization

	David(S)	David(M)	News(S)	News(M)	Brown(S)	Brown(M)
SBM	84.8/0.423	57.1/0.051	62.2/0.006	56.1/0.021	70.0/0.001	70.1/7e-04
DC	91.1/0.566	91.1/0.568	56.4/0.084	55.3/0.015	70.6/0.160	70.2/0.155
ODC	87.5/0.462	87.5/0.470	75.3/0.247	77.9/0.270	75.3/0.311	80.3/0.318
DDC	57.1/0.015	64.3/0.060	56.4/0.084	52.9/0.005	54.0/0.005	64.0/0.070
NH	84.4/0.395	86.6/0.449	72.8/0.215	73.8/0.233	78.0/0.309	78.1/0.314

Table 8. Average clustering performances with NH initialization

	David(S)	News(S)	Brown(S)
SBM	84.8(0)/0.423(0)	62.2(0)/0.006(0)	70.0(0)/0.001(0)
DC	91.1(0)/0.566(0)	56.4(0.08)/0.084(4e-04)	71.1(0.2)/0.171(0.003)
ODC	87.5(0)/0.462(0)	75.3(0)/0.247(0)	75.3(0)/0.311(0)
DDC	84.2(6.1)/0.395(0.108)	56.4(0.06)/0.084(3e-04)	62.1(3.5)/0.060(0.023)
NH	84.4(1.0)/0.395(0.023)	72.8(0.7)/0.215(0.012)	78.0(0.09)/0.309(0.003)
	David(M)	News(M)	Brown(M)
SBM	57.1(0)/0.051(0)	58.7(2.0)/0.014(0.005)	70.1(0)/7e-04(0)
DC	91.1(0)/0.568(0)	52.5(1.9)/0.025(0.030)	70.2(0.3)/0.156(0.004)
ODC	87.5(0)/0.470(0)	75.3(1.2)/0.240(0.014)	79.7(0.08)/0.317(3e-04)
DDC	75.6(2.6)/0.204(0.037)	52.4(1.9)/0.028(0.031)	64.3(0.2)/0.073(0.002)
NH	86.6(0)/0.449(0)	73.8(0.6)/0.233(0.009)	78.1(0.1)/0.314(0.002)

In Table. 7, we found with NH initialization ODC works pretty well on "News". It returns the desired comunity structure in all 100 initials. DC also works much better on "Brown", and 70% vertices are correctly labelled. With random initialization, it's only around 55%. With NH initialization, the performances are more stable now. Table. 8 lists the average clustering performances and we can see the standard deviations are much lower than the ones in Table. 6. For "David", now DC works perfectly on "David(M)", but previously, it's average performance on "David(M)" is quite fair and not stable either. DDC also improves itself on both "David(S)" and "David(M)". ODC now works better on "News" with higher average percentages and NMI values and much lower deviations. DC also works much better on average while having much lower deviations on "David".

3.6 Results on degree generated models

In "Brown", the real data show that both the out- and in-degree distributions have heavy tails close to power-law. In Fig. 5 we plotted the complementary CDF of the out- and in-degrees for Brown(S) and Brown(M). We estimate the power-law θ parameters of the real data using both discrete (labeled with [D]) and continuous (labeled with [C]) methods, which are introduced in section 2.3. Setting $d_{\min} = 1$, the MLEs of the θ parameters are listed in Table 9.

We compare the performances of the power-law degree-generated block models with those original degree-corrected models. In Table 10 11, the first row lists the results of the original degree-corrected models without degree-generation, and the second row shows the results of the block models with power-law degree generation. We use KL-heuristic plus MCMC to infer the parameters and the community structures. All the results are obtained from 50 initials, each of them

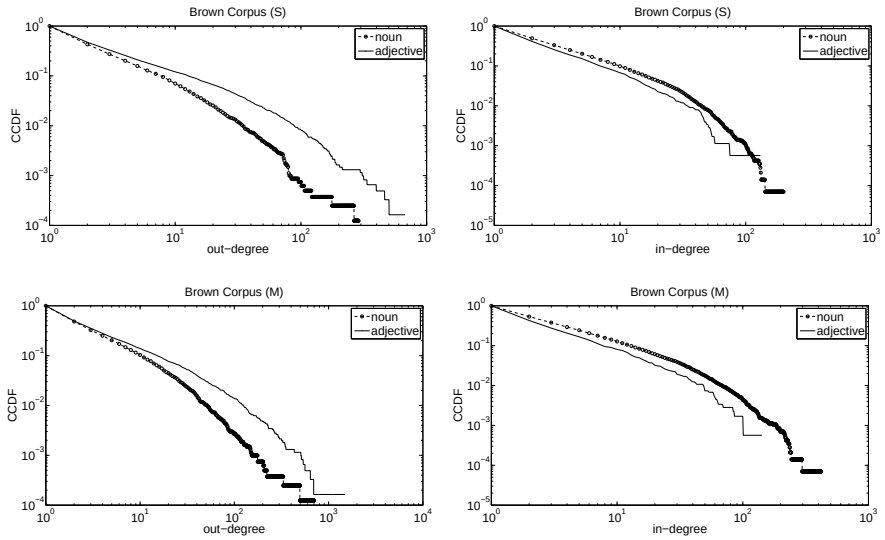

Fig. 5. Degree distributions in Brown network

Table 9. Degree parameter MLEs in Brown

	block	α_{in}[D]	α_{in}[C]	α_{out}[D]	α_{out}[C]	β_{in}	β_{out}
Brown(S)	adj	1.829	2.329	1.952	2.629	0.161	0.527
	noun	1.987	2.721	1.793	2.248	0.716	0.021
Brown(M)	adj	1.741	2.136	1.828	2.326	0.161	0.527
	noun	1.931	2.576	1.740	2.134	0.716	0.021

is initialized randomly. For each model and each network, we take the best result among those initials. In each initial we run the KL-heuristic followed by 1 million MCMC steps. Both the percentage of correctly labelled vertices and the NMI value are listed for each model on each network. We can see in Table 10, degree generation do improves the performance of DDC and DC. As to ODC, it already works pretty well by itself and degree generation doesn't help. However we also found power-law degree generation can actually speed up the searching performance. KL-heuristic has high complexity, which is $O(N^2 \log N)$. Although each MCMC step only takes $O(K^2)$, for large networks, MCMC converges slowly. If we give up KL-heuristic (this may be unavoidable for handling large networks), in other words, each initial only runs 1 million MCMC steps, we get the results in Table 11. We can see in Table 11, DDC, DC and even ODC becomes worse when there is no degree generation. With degree generation, the performances of all these models are very stable compared to the results with KL-heuristic in Table 10.

Table 10. Clustering results on Brown with KL-heuristic

	DC	ODC	DDC	DG-DC	DG-ODC	DG-DDC
Brown(S)	55.6/0.020	75.3/0.311	55.8/0.016	74.4/0.283	75.3/0.310	73.3/0.224
Brown(M)	55.5/0.015	80.3/0.318	53.8/0.012	75.6/0.290	76.0/0.320	72.2/0.210

Table 11. Clustering results on Brown without KL-heuristic

	DC	ODC	DDC	DG-DC	DG-ODC	DG-DDC
Brown(S)	54.3/0.010	72.0/0.188	53.7/0.008	75.1/0.275	76.5/0.297	71.4/0.224
Brown(M)	52.6/0.007	73.4/0.203	53.8/0.011	75.4/0.276	77.3/0.308	70.9/0.194

4 Conclusions

Degree-correction in stochastic block models provides a powerful approach to dealing with networks with inhomogeneous degree distributions. Partial degree-correction is a new idea proposed in this paper which tolerates highly skewed degrees within community while still be able to utilize degree information for community detection purpose. We demonstrated two ways to achieve this. One is the oriented degree-corrected block model which prefers highly asymmetric inter-block connections. Another is the degree-generated block model which fits the degree sequence in each community to a family of distributions before examining the edge connecting patterns. Thus, block assignments in which degree sequences are poorly fitted is thought to be unlikely.

We have shown that in certain networks these new block models can apply appropriate partial degree-correction, achieving higher accuracy. The oriented degree-corrected block model usually works well on asymmetric networks. However, when the misinformation in the edge connecting patterns is overwhelmingly strong, the orientation information might not be enough to compensate. There are also situations when the block connecting pattern looks fairly symmetric, OCD is still able to leverage highly asymmetric connections occurred between only a few vertices. Further study of different degree corrections is required to better understand their effectiveness for different networks.

The degree generated block models showed the most promise with their excellent performance across the board, especially with noisy data. Nonetheless, their effectiveness depends heavily on knowing the true form of the degree distribution in each community. Without the ground truth of block assignments, one would have to guess an appropriate form first, making it a much more difficult problem.

With multiple degree corrected models, and multiple ways to do degree generation for each of them, we now have a tough choice to make whenever we meet new networks. Better understanding of each model could help but real-world networks may exhibit structures too complicated to comprehend. A much better alternative is a model selection criterion which can predict relative performance of each model automatically based on the observed data.

5 Acknowledgments

We are grateful to Terran Lane, Ben Edwards, Aaron Clauset, and Mark Newman for helpful conversations. This work was supported by the McDonnell Foundation.

Bibliography

[1] L. Adamic and N. Glance. The political blogosphere and the 2004 US Election: Divided They Blog. In *Proc 3rd Intl Workshop on Link Discovery.*, 2005.

[2] W. Aiello, F. Chung, and L. Lu. A random graph model for power law graphs. *Experimental Mathematics*, 10(1):53–66, 2001.

[3] E.M. Airoldi, D.M. Blei, S.E. Fienberg, and E.P. Xing. Mixed membership stochastic blockmodels. *The Journal of Machine Learning Research*, 9:1981–2014, 2008.

[4] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002.

[5] Fan Chung and Linyuan Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences of the United States of America*, 99(25):15879–82, December 2002.

[6] A. Clauset, C.R. Shalizi, and M.E.J. Newman. Power-law distributions in empirical data. June 2007.

[7] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6), December 2011.

[8] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Inference and Phase Transitions in the Detection of Modules in Sparse Networks. *Physical Review Letters*, 107(6), August 2011.

[9] S.E. Fienberg and S. Wasserman. Categorical data analysis of single sociometric relations. *sociological Methodology*, pages 156–192, 1981.

[10] S. Fortunato. Community detection in graphs. *Physics Reports*, 2009.

[11] P.W. Holland, K.B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[12] B. Karrer and M. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1), 2011.

[13] Cristopher Moore, Xiaoran Yan, Yaojia Zhu, Jean-Baptiste Rouquier, and Terran Lane. Active learning for node classification in assortative and disassortative networks. page 841. ACM Press, 2011.

[14] M. Mørup and L.K. Hansen. Learning latent structure in complex networks. *NIPS Workshop on Analyzing Networks and Learning with Graphs*, 2009.

[15] M. Newman and E.A. Leicht. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences*, 104(23):9564–9569, 2007.

[16] M.E. Newman. Assortative mixing in networks. *Physical Review Letters*, 89(20):208701, 2002.

[17] M.E. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126, 2003.

[18] M.E. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006.
[19] M. A Porter, J. P Onnela, and P. J Mucha. Communities in networks. *Notices of the American Mathematical Society*, 56(9):1082–1097, 2009.
[20] T.A. Snijders and K. Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 14(1):75–100, 1997.
[21] S. Wasserman and C. Anderson. Stochastic a posteriori blockmodels: Construction and assessment. *Social Networks*, 9(1):1–36, 1987.
[22] W. W Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.