Connected Graphs with a Given Degree Sequence: Efficient Sampling, Correlations, Community Detection and Robustness

John H. Ring IV, Jean-Gabriel Young, and Laurent Hébert-Dufresne

Abstract Random graph models can help us assess the significance of the structural properties of real complex systems. Given the value of a graph property and its value in a randomized ensemble, we can determine whether the property is explained by chance by comparing its real value to its value in the ensemble. The conclusions drawn with this approach obviously depend on the choice of randomization. We argue that keeping graphs in one connected piece, or component, is key for many applications where complex graphs are assumed to be connected either by definition (e.g. the Internet) or by construction (e.g. a crawled subset of the World-Wide Web obtained only by following hyperlinks). Using an heuristic to quickly sample the ensemble of small connected simple graphs with a fixed degree sequence, we investigate the significance of the structural patterns found in real connected graphs. We find that, in sparse networks, the connectedness constraint changes degree correlations, the outcome of community detection with modularity, and the predictions of percolation on the ensemble.

Putting measurements into context is a crucial part of any network analysis. Suppose that we have some real network at our disposal and that we know the value of some of its properties—say its level of transitivity or its homophily with respect to some property [1]. It is clear that these values do not make much sense in and of themselves. Knowing that a network G_1 "has transitivity C," tells us far less than knowing that "network G_1 is more transitive than network G_2." In the first instance, we merely have an arbitrary number; it only begins to make sense once compared

J. H. Ring IV · L. Hébert-Dufresne (✉)
Vermont Complex Systems Center, University of Vermont, Burlington, VT, USA

Department of Computer Science, University of Vermont, Burlington, VT, USA
e-mail: Laurent.hebert-dufresne@uvm.edu

J.-G. Young
Center for the Study of Complex Systems, University of Michigan, Ann Arbor, MI, USA

N. Masuda et al. (eds.), *Proceedings of NetSci-X 2020: Sixth International Winter School and Conference on Network Science*, Springer Proceedings in Complexity,
https://doi.org/10.1007/978-3-030-38965-9_3

to other measurements, either implicitly (e.g., via a normalization) or explicitly, as in the second instance.

Null modeling is a technique whereby we create an ensemble of graphs that preserves some of the properties of the original network data, while randomizing the others. Computing the expected value of a property for the graphs of this ensemble gives a baseline against which to compare an original measurement. The baseline can be used to formulate a statistical test telling us which properties of an input G are "surprising" with respect to the model, and which are not. For example, if the transitivity $C(G)$ of a network takes roughly the same value under a null model that preserves the degree sequence $d(G)$ of the input G, then it is, as far as transitivity is concerned, typical of graphs with this degree sequence. Conversely, if the value of $C(G)$ greatly differs in the randomized ensemble, then G is atypical: The degree sequence $d(G)$ does not explain $C(G)$.

Similar inference can be made for any choice of null models and properties. Hence the more models we have, the more we can control for various features of real networks. Fixing only the number of vertices and edges leads to the classical random graph model of Erdős–Rényi [2, 3]. Further constraining the ensemble to graphs with a fixed degree distribution corresponds to generating graphs from the well-known Configuration Model (CM) [4–6]. Notwithstanding sampling problems [7], exponential random graphs [8] can be used to limit the ensemble to graphs with precise patterns and correlation structure. A wealth of other models allow to control for, e.g., arbitrary mesoscale patterns [9], degree correlations [10], or a centrality structure [11, 12].

All the models above are defined in terms of simple *local* connection rules. The main reason for this choice is that simple local rules usually lead to simple sampling algorithms and make the models analytically tractable [7]. A less desirable consequence of mathematical convenience, however, is that it dictates the models we have. Relying only on convenient models can leave important blind spots in our analyses, because unwieldy connection rules can—and do—lead to critically different null models [13].

An important example of unwieldy constraint, which will be the focus of the present paper, is connectedness. A few recent studies have shown that the connected subsets of random graphs can have a significantly different structure than the entire random graph itself [14–16]. Connectedness is, without a doubt, an important aspect to control for in null models, since it is so frequently found in real systems. It can arise for at least three different reasons. One, it may be a simple matter of perspective: If a food web (or a power grid) is split into independent components, then we are likely to consider them as separate food webs (or power grids) since what occurs in one component does not affect the other. Two, global connectivity may stem from the definition of the network: Such is the case of the Internet, which is a unique, global, connected network of computers. Three, some networks can never split into disconnected pieces because of the way they are sampled [17], including, for example: subsets of the World Wide Web obtained by crawlers that do not teleport [18], or social networks sampled by recursive nomination [19, 20]. If we do not use an appropriate random graph ensemble as a null model in our analysis, we are likely to overestimate the significance of some results or miss other important structural features.

The focus of our paper is twofold. First, we propose a simple heuristic to generate samples from the connected Configuration Model (connected CM), the natural null model for connected graphs with a fixed degree sequence. There are already algorithms that solve related problems—see Sect. 1 for an overview—but they are either inefficient or not adapted to the version of the problem we aim to tackle. Our algorithm is described in Sect. 2 and made available online in a reference implementation.[1] Second, we use this algorithm to quantify the impact of connectedness on applications that rely on comparison with random graphs, in Sect. 3. We show that using the connected version of the CM can change degree correlations, the outcome of community detection, and lead to qualitatively different predictions of percolation on graphs.

1 Connected Configuration Model and Related Work

It is somewhat imprecise to speak of *the* connected configuration model (CM) because there are, in fact, many versions of the classic configuration model [21]. In this paper, we will consider the so-called microcanonical variant, that assigns an identical probability to all graphs that contain a single component and whose nodes $1, \dots, N$ have degrees $d = (d_1, d_2, \dots, d_N)$, and zero probability to all other graphs. Formally, if we denote by $\Omega_N(d)$ the ensemble of connected graphs with degree sequence d, the probability of observing any given graph G with degree sequence d' is, under this model,

$$P(G) = \begin{cases} \frac{1}{|\Omega_N(d')|} & \text{if } d(G) = d', \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $|\cdot|$ denotes the cardinality of the ensemble. This definition of the connected CM is known as microcanonical by opposition to the canonical definition where the degree of nodes are fixed only on average, instead of exactly.

It turns out sampling that from this ensemble is much more challenging than sampling from the equivalent ensemble without the connectedness constraint [21]. There are a few approaches that solve the problem in different ways.

A sampling algorithm for generic connected models was recently proposed by Gray et al. [16]. Their idea is to first generate a (typically unconnected) initial graph from the model without connectedness constraints. They then add arbitrary links until the graph becomes connected, and they finally run a simple Monte-Carlo Markov chain (MCMC) algorithm with a target distribution that is now the

[1] Available at https://gitlab.com/jhring/connected_cm.

connected version of the model. This method unfortunately does not work when the target model is the *microcanonical* connected CM. The input degree sequence is modified when edges are added, and the resulting graphs therefore have zero probability under the connected model, and the MCMC can never leave the initial state, let alone approach a region of high probability.

Another recent algorithm proposed by Tishby et al. also makes use of the unconnected version of the CM as its starting point [15]. This approach differs from the previous one in what it does next. The idea, in this case, is to discard the disconnected components of the initial graph and to keep the largest one, which then constitutes a sample from the connected model. Tishby et al. show that by engineering the parameters of the generating (unconnected) ensemble, one can obtain the desired size and degree distribution for the large component, in expectation. However, this approach does not control the degree sequence exactly, and it is therefore again not applicable to the microcanonical CM.

The only method that specifically addresses the problem of sampling from the microcanonical CM is an algorithm by Viger and Latapy [22, 23] and its predecessor, proposed by Gkantsidis et al. [24]. Starting from an initial connected configuration generated (for example an input graph or constructed with a graphical test [25]), their algorithm proposes local randomizations that preserve connectivity *and* the degree sequence (double edge-swaps [21]). Since certifying connectivity is expensive, they only verify that the graph has remained connected every so often, and backtrack when necessary. They show that a good choice of monitoring interval can speed the computation up by a large constant factor [24].

2 Efficient Heuristic

Our algorithm is most closely related to the methods of Gkantsidist et al. [24] and Viger and Latapy [23]. In fact, it makes use of the same set of basic operations, but we swap their order to obtain a significant speed-up on smaller graphs such that we can quickly sample millions of instances.

To generate a graph with sequence d, we proceed as follows. (see Algorithm 1). We first use the Havel-Hakimi algorithm to create a graph with the appropriate degree sequence [25, 26]. We then shuffle its edges by applying T double edge swaps on random pairs of edges, where T is a tunable parameter. By construction, these swaps preserve the degree sequence and can be shown to generate ergodic and uniform chains over the set of graphs with a fixed degree sequence (with no connectedness constraints) [21]. Note that we disallow self-loops parallel edges—we are interested in the *simple* graph ensemble where these do not occur.

The innovation of our method lies in a final step where we connect our newly-shuffled graph with double edge swaps that connect components. These edge swaps are carried out with edges drawn uniformly at random within two different components. The components are themselves selected at random, with probability proportional to the number of nodes they contain. The algorithm terminates as soon as the graph becomes connected.

Algorithm 1 Connected configuration model

```
1: procedure CONNECTED_CM(DEG_SEQ)
2:     $G \leftarrow$ havelhakimi(DEGSEQ)
3:     while $swaps < T$ do
4:         $swaps$ += randomedgeswap($G$)
5:     while numberofcomponents($G$) $> 1$ do
6:         $c1 \leftarrow$ randomcomponent($G$)
7:         $c2 \leftarrow$ randomcomponent($G$)
8:         $e1 \leftarrow$ randomedge($c1$)
9:         $e2 \leftarrow$ randomedge($c2$)
10:        $G$.swapedges($e1$, $e2$)
```

All the steps of this algorithm are efficient. Constructing the initial graph and ran- domizing it with double edge swaps takes linear time in the number of edges, i.e., the sum of degrees $\sum_i k_i$ [21]. The connection step, step 3, also admits an efficient implementation: We precompute a list of components (say with union-find), and we then update the list every time an edge swap successfully merges two components. Verifying that a swap connects two components is not too costly, because we only need to verify that the swap did not disconnect the nodes from their original components. Altogether, exact sampling of small connected graphs (less than 200 nodes) with method of Viger and Latapy takes at best 50 times longer than with our approximate sampling. On larger graphs (e.g. over a couple of thousand nodes), our sampling heuristic currently requires similar or slightly larger amount of time as exact sampling, because we have yet to optimize our implementation. That said both approaches were found to scale linearly with graph size on perfect trees; we thus expect the reduction in overhead to be worth it in all regimes once optimized for larger graphs.

The price to pay for this speedup is exact uniformity over $\Omega_N(d)$. The proofs of Ref. [23], for example, do not generalize to our method. This is due to the fact that the swaps we make in step 3 are "unidirectional," in the sense that they move us from a space of disconnected graphs towards a space of connected ones, with no possibility of ever backtracking; classical proof techniques, in contrast, demonstrate the ergodicity of *reversible* chains over an ensemble. That said, our heuristic at least guarantees that (a) the degree sequence is preserved and (b) the graph is connected, and it finds these graphs rapidly in most cases. Furthermore, as we now show, a uniformity test and a comparison with the slower (but provably exact) algorithm of Viger and Latapy suggest that our algorithm samples from the target distribution to a close approximation.

As a first verification that our heuristic approximately generates graphs from the correct ensemble, we carry out a simple uniformity test. We consider the short degree sequence $k = (1, 1, 1, 2, 2, 2, 3)$, and enumerate all the labeled connected

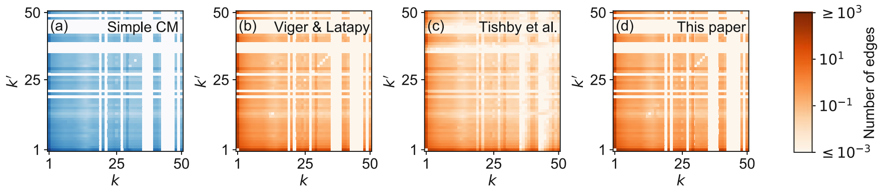
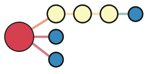
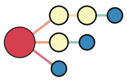
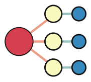


Fig. 1 Expected correlation matrix $\boldsymbol{e}$ for a real network (Norwegian board of directors [27]), as calculated empirically using 1000 draws from: (**a**) the simple CM, (**b**) the exact connected microcanonical CM [22], (**c**) the canonical connected CM [15], and our heuristics. Some features of the exact connected microcanonical sampling are recovered by our heuristics but not by the approach of Ref. [15]. Most relevant to our case studies is the weak diagonal, a signature of the disassortativity (negative degree correlations) of the true connected CM ensemble

Table 1 Frequencies of the isomorphism classes of connected graphs sharing the degree sequence $k = (1, 1, 1, 2, 2, 2, 3)$, compared against their empirical frequencies computed from a 10^6 graph samples

Isomorphism class	Uniform	Viger & Latapy	Our heuristic
	0.3	0.304	0.321
	0.6	0.599	0.609
	0.1	0.098	0.070

The calculation takes about 90 s with our non-uniform heuristic, versus hours with the exact sampler

graphs that are compatible with it—60 in this case. We then identify all the isomorphism classes up to a relabeling—there are three, see Fig. 1—and count the number of graphs in each class. The frequencies of the isomorphism classes are reported in Table 1. Since the microcanonical connected CM is a uniform distribution over the 60 labeled graphs (and not the isomorphism classes), a useful sampling algorithm should generate graphs with isomorphism classes that closely follow the true frequencies calculated in Table 1. And indeed, we find empirical frequencies that are not exactly equal to the true ones, but the deviations are small.

To further validate our heuristic, we carry out a second test where we randomize the connections of the giant component of a large connected graph. When the graph is large, there are far too many isomorphism classes to calculate their true frequencies, let alone evaluate their empirical frequencies to a reasonable degree of accuracy. Hence, we turn to a different test and analyze the correlations between the neighborhood of nodes, conditioned on their degrees. As we will see later in Sect. 3, this correlation structure intervenes in many of the applications of the connected CM as a null model—it is therefore essential to get it right.

We show the results in Fig. 1, with matrices $\boldsymbol{e}(k, k')$ whose entries $e(k, k')$ are the expected number of edges between nodes of degree k and k'. We use as input the degree sequence of a real network of Norwegian directors sitting on at least one corporate board together [27]. This graph is large, comprising 4475 nodes and 4652 edges, and its degree sequence is non-trivial, with degrees ranging from 1 to 552. We calculate the matrices using four sampling algorithms: The classical double-edge swap algorithm that samples from the simple CM (for the sake of comparison); the algorithm of Viger and Latapy (exact); the algorithm of Tishby et al. (canonical connected CM); and our heuristic. The results confirm that the connected constraints change the correlation structure significantly (compare Fig. 1a, d). They also show that hard constraints on the degree lead to a markedly different correlation matrix (compare Fig. 1b, c). And finally, they demonstrate that our heuristic finds correlations close to the exact ones (compare Fig. 1b, d).

3 On the Impact of Being Connected

3.1 Assortativity in Connected Graphs

Degree correlations aim to capture mixing patterns in a network [28]. Generally, these degree correlations can be captured by "joint-degree measures" such as the probability $\boldsymbol{e}(k, k')$ that a random edge joins nodes of degree k' and k', a measure of correlation we have used in Fig. 1. For any random graph with degree distribution p_k, we can expect the degree of a node at the end of a random edge to be distributed according to $q_k = kp_k/\langle k\rangle$ since a node of degree k participates to k times more edges than a node of degree 1. In a fully random graph with a given degree sequence, i.e., one drawn from the unconnected canonical CM, we could therefore expect $e_{\mathrm{CM}}(k, k') \propto q_j q_{k'}$, but that would assume that the degrees of neighboring nodes are uncorrelated. To correct for possible deviations from the CM one can therefore measure $e(k, k')$ from real datasets.

Edge matrices like $\boldsymbol{e}(k, k')$ can be used to parametrize random graph ensembles, as done in some degree-correlated version of the Configuration Model [10]. In practice, it is more parsimonious to coarse-grain this information and only specify a measure of correlation, the assortativity coefficient [28]. For a graph G, we write

$$r_{\mathrm{CM}}(G) = \frac{1}{\sigma_q^2} \sum_{k,k'} kk' \left[e(k, k') - q_k q_{k'}\right], \tag{2}$$

where $\sigma_q^2 = \sum k^2 q_k - \left(\sum k q_k\right)^2$ is the variance of the distribution q_k. Importantly, the negative term in Eq. (2) corresponds to correlations we would expect in the CM, meaning that we control for correlations that would emerge naturally given only the degree sequence of the graph. In other words, we control for our expectations from the CM.

In real graphs we typically do not expect $r = 0$ which would correspond to a typical graph drawn from the CM. Instead, as a general rule of thumb [28, 29], it is typically accepted that social graphs tend to display assortative mixing, or positive degree correlations $r > 0$, where high degree nodes tend to more connected than you would expect at random Conversely, technological graphs tend to display diassortative mixing, or negative degree correlations $r < 0$, where high degree nodes connect to low degree nodes more often than expected at random.

Technological and social datasets are also typically collected in very different ways. As mentioned in our introduction, some data on technological graphs, such as the structure of the Internet or power grids, tend to be connected per definition. For example, a power grid with two disconnected components would typically be considered as two distinct graphs. In other cases, like subsets of the World-Wide Web, data tends to produce connected graphs because they are often collected by crawling edges.

Given conventional wisdom on assortative mixing in complex graphs and on their different expected connectedness across domains, we ask: How does connectedness affect assortativity? In Fig. 2 we illustrate how connectivity might matter, using two trees as a simple example. Controlling for the wrong expectation means we might find a signal where there is none.

To study the interplay between assortativity and connectedness in real graphs, we calculate a new assortativity coefficient based on the connected CM rather than the typical CM null model. Because of the linearity of Eq. (2) we can write

$$r_{\mathrm{CCM}}(G) = r_{\mathrm{CM}}(G) - \langle r_{\mathrm{CM}}(G_{\mathrm{CCM}})\rangle \tag{3}$$

where we are controlling for our expectations using the connected CM (right-hand side), by measuring the classic assortativity coefficient (first term of left-hand side) and subtracting deviations between the new and old null models (second term of right-hand side). The final term is therefore the classic assortativity coefficients averaged over many graphs drawn from our sampling algorithm for the connected CM.

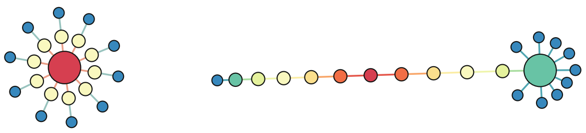


Fig. 2 Two simple examples of connected trees with the same degree sequence. The most assortative this degree sequence be, while remaining connected, is as a stretched star (left graph) with assortativity coefficient of $r = -0.23$. The most disassortative this degree sequence can be is as a star combined with a single chain (right graph), leading to $r = -0.50$. Without looking at the graph or degree sequence in detail, one might naturally conclude that both of these graphs are strongly diassortative. The disassortativity of trees stems from their high number of leaves that cannot be connected without disconnecting the graph

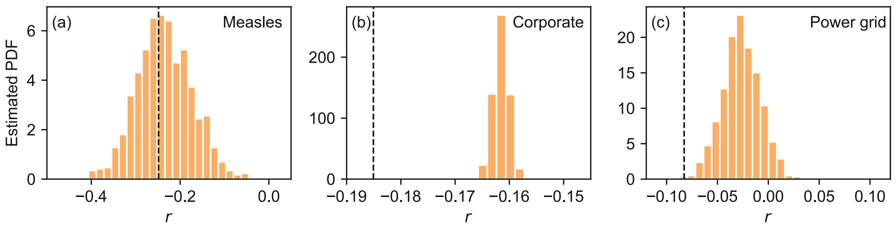


Fig. 3 Distribution of the degree correlation, for random graphs drawn from the connected CM, with degree sequences taken from three real graphs of increasing densities. (**a**) Phylogenetic tree of the Measles [30], of mean degree $\langle k \rangle = 1.99$, with 175 nodes and 174 edges. (**b**) Graphs of Norwegian directors sitting on at least one board together [27], of mean degree $\langle k \rangle = 2.08$, with 4475 nodes and 4652 edges. (**c**) Power grid of Poland during the winter [31], of mean degree $\langle k \rangle = 2.42$, with 2383 nodes and 2886 edges. The classic assortativity coefficients are shown as vertical lines at (**a**) −0.249, (**b**) −0.185, and (**c**) −0.0827; and become, under the connected CM, (**a**) −0.013, (**b**) −0.023 and (**c**) −0.056

We calculate the distribution of this corrected coefficient for a few real connected graphs in Fig. 3. As expected, on real trees such as the phylogenetic tree of the measles virus (Fig. 2a), we find strong disassortativity based on the classic assortativity coefficient $r_C M(G)$ (shown as a vertical line). This disassortativity, however, is not surprising when we compare it to the distribution of assortativity found for graphs generated from the correct, connected, null model (histogram). In sparse social systems with few loops (Fig. 2b), such as a corporate graph of board directors used in our example of Fig. 1, we similarly remove most of the signal when using the connected null model even if that graph can still be rejected as not being a typical instance of the connected CM. Finally, in denser cases such as power grids (Fig. 2c), we would conclude that the graph structure is disassortative by using either the CM or connected CM, because the null distribution is centered around $r = 0$ for the connected CM (and because the classical assortativity effectively compares the assortativity of real graphs to a distribution centered on $r = 0$).

3.2 Community Detection in Connected Graphs

Community detection [32]—and more generally mesoscopic pattern extraction [33]—refers to a wide variety of methods whose goal is to find structurally similar groups of nodes in a network, given only the structure of the network as input. Formally, these methods find assignments of the nodes of G to K communities of similar nodes, assigning precisely one community σ_i to each node $i = 1, \ldots, N$. They are among the most useful methods of network science, because communities can help us understand networks at the exploratory data analysis stage, act as the input of other network analysis methods [34], or even help us identify the fundamental building blocks of networks [35].

Modularity maximization methods were among the first proposed community detec- tion algorithms, and have since gained prominence among practitioners because of their clarity and ease-of-use [32, 36, 37]. These methods use a specific form of objective functions—modularities $Q(\sigma)$—to quantify the quality of partitions. The most general modularity function can be written as [38]

$$Q_P(\sigma) = \frac{1}{m} \sum_{i<j} \big(a_{ij} - \langle a_{ij} \rangle_P\big)\, \delta_{\sigma_i \sigma_j}, \tag{4}$$

where m is the number of edges, where the sum runs over all pairs of nodes, and where $\delta_{\sigma_i \sigma_j}$ is the Kronecker delta, equal to 1 if $\sigma_i = \sigma_j$ (i.e., node i and j are in the same community) and to zero otherwise. The two terms of Eq. (4) denote, respectively: Whether there is an edge between nodes i and j and the expected number of edges between the node i and j under the null model P, noted as $\langle a_{ij} \rangle_P$. According to this equation, a partition is deemed good if there are many more edges connecting nodes that are inside the same community than what we would expect to observe by chance, *given a null model* P for the graph. A specific modularity maximization *algorithm* consists of a particular choice of null model P and of maximizer[2] [33]—a search strategy for the space of possible partitions that converges to the partition with maximal modularity.

The canonical (unconnected) CM is almost always used as the null model P [36, 37], in part because the expectation $\langle a_{ij} \rangle$ is then given by the simple formula

$$\langle a_{ij} \rangle_{\text{CM}} = \frac{k_i\, k_j}{2m}, \tag{5}$$

where k_i is the degree of node i and $m = \frac{1}{2} \sum_i k_i$ is the total number of edges. But, as we have already argued, this null model is not always the most natural choice [21]. In fact, it is known that different choices of model will tend to resolve different types of communities [33, 41], with no obvious optimal choice on all inputs [42]. Futhermore, there are known applications where switching to a model P tailored to the class of graphs at hand leads more accurate and relevant inference results [21, 43]. So we ask: How do modularity-based algorithms behave when we choose the *connected* ensemble of graphs with fixed degree sequence as our null model?

To answer this question, we need to evaluate the average $\langle a_{ij} \rangle_{\text{CCM}}$ appearing in Eq. (4), with the connected CM as the null model. Just as in Sect. 3.1, we opt for a numerical average, computed with the efficient heuristic introduced in Sect. 2. If there are on average $e(k, k')$ edges connecting nodes of degrees k and k', then the expected number of edges between two nodes i and j of degree k_i and k_j is given by

[2]Many standard optimization methods are capable of fulfilling the role of optimizer: Spectral embedding [37, 39] and greedy maximization [40] are well-known examples.

$$\langle a_{ij}\rangle_{\text{CCM}} = \begin{cases} e(k_i,k_j)/n_{k_i}n_{k_j} & \text{if } i \neq j \text{ and } k_i \neq k_j \\ e(k_i,k_j)/n_{k_i}(n_{k_i}-1) & \text{if } i \neq j \text{ and } k_i = k_j \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where n_k is the number of nodes of degree k in the graph, and where the second expression accounts for the fact that there are no self-loops in the ensemble.

With Eq. (6) for $\langle a_{ij}\rangle_{\text{CCM}}$, we can readily evaluate the connected modularity of the different partitions of a network, and run optimization algorithms to find the best one. The number of communities K is not known a priori, so we run a double optimization[3] where we first find the best partition $\sigma^*(K)$ for many choices of K, and then find the K^* for which this optimal partition has the largest modularity. The results of one such experiment are shown in Fig. 4, where we find the communities of the network of directors (see also Figs. 1 and 3), using the standard modularity Q_{CM} (see Eq. (5)) and the connected modularity Q_{CCM} (see Eq. (6)).

We find a few differences between the analysis ran with the unconstrained and constrained CM as null model. Perhaps most noticeable is the fact that the optimal connected modularity stays close to its maximum for a much larger range of K than the standard modularity (Fig. 4a). This suggests that the number of communities is not as clearly defined once we account for connectivity. Furthermore the optimal partition $\sigma^*(K^*)$ under the connected CM occurs at a slightly larger number of communities K, which allows the algorithm to (correctly) resolve a few more communities (see Fig 4b, c). When we actually inspect these best partitions, however, we find that communities essentially consists of one node of high degree and its degree one neighbors—regardless of the choice of null model. This is a consequence of the fact that placing a node of degree one and its neighbor in the same communities is always good when the networks are near-trees. The two models do not put the exact same penalty on these connections, but the difference is not large enough to alter the optimal clustering choice significantly. And as a result, the communities are qualitatively unchanged when we switch the null model from the CM to the connected CM (the normalized reduced mutual information (nRMI) [46] of the optimal partitions is 0.93, while comparing the best partitions of size K^* yield nRMI $= 0.95$ ($K^* = 80$) and nRMI $= 0.98$ ($K^* = 106$), on a scale of 0–1).

3.3 *Robustness of Connected Graphs*

Percolation is a simple stochastic process in which edges or nodes are randomly removed from an existing network. It is an obvious model for the robustness of

[3]This strategy works thanks to the regularization properties of the modularity: The partition in a single community ($K = 1$) has zero modularity due to the negative contributions of the null model, and so does the partition in $K = n$ communities of 1 node since no two nodes share a community. It follows that there is an optimum K^* somewhere in between these two extremes.

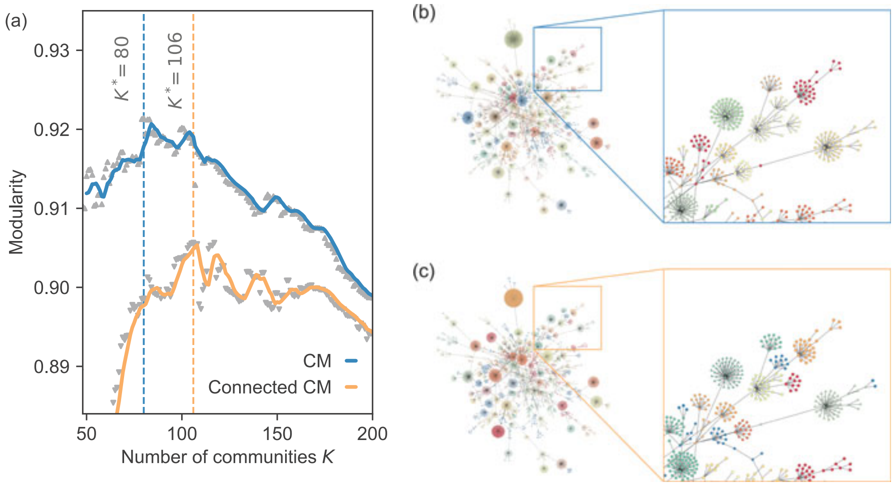


Fig. 4 Outcome of modularity maximization on the network of directors. (**a**) Modularity of the best partition in K communities found by using the CM and the connected CM as null models. We add a moving window average to guide the eye. The optima are identified with vertical lines. (**b**–**c**) Visualization [44] of the optimal partitions when we use as null model (**b**) the CM, and (**c**) the connected CM. The community of a node is indicated by a non-unique combination of node color and shape (but no two adjacent communities are shown with the same combination). These partitions are found as follows. For a fixed K, we first embed the nodes of G in $\mathbb{R}^K$, using the K largest eigenvectors of the modularity matrix $\boldsymbol{B} = \boldsymbol{A} - \langle \boldsymbol{A} \rangle_P$ [39], defined as the difference between the adjacency matrix $\boldsymbol{A}$ and the average adjacency matrix $\langle \boldsymbol{A} \rangle_P$ under null model P. We then create many candidate partitions by running agglomerative clustering on the embedded nodes [45], for various choices of affinity (ℓ_2 norm and cosine similarity) and linkage strategy. We finally select the candidate partition that maximizes the modularity as the optimum for that K. The partitions shown in (**b**–**c**) are the ones that maximize the modularity over all K

graphs to random failures, but also useful to study dynamical processes such as epidemics [47]. While one can certainly simulate percolation on a specific real graph to study its robustness or its ability to sustain an epidemics, it is often instructive to also study the percolation process on a series of random graph ensembles, because this lets us evaluate the impact of different structural properties on the outcome. Hence for example, an epidemic on a real graph could be first compared to the outcome of the percolation on Erdős–Rényi graphs with the same density, to evaluate the impact of density alone; then compared to the outcome of percolation on graphs drawn from the CM, to evaluate the impact of the contact distribution; and so on for higher order models.

Enforcing connectedness can, again, make a big difference on the outcome of percolation. To show this, we consider bond percolation where a fraction $p \in [0, 1]$ of edges are randomly removed from a network instance. We compute the size of the largest connected component (LCC) after this removal has taken place, and investigate its dependency on p. The original connectedness of a real system is

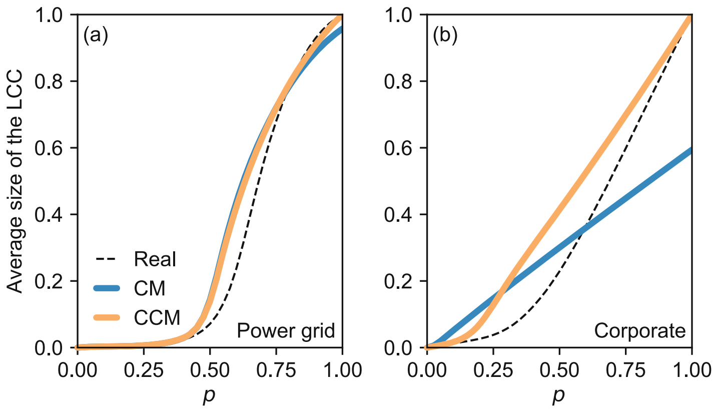


Fig. 5 Percolation on the CM and connected CM ensembles with degree sequences taken from real systems, compared with percolation on these systems. The size of the largest connected component (LLC) is shown as a function of the fraction of removed edges p

critical to account for, since the size of the LCC *after* some edges are removed is obviously bounded by the size of the LCC *before* this removal processes.

In Fig. 5, we show the outcome of bond percolation for two graphs studied in previous case studies: The social network of board directors sitting on common boards of Norwegian public limited companies, and the structure of a Polish power grid. In the first, a percolation model can be used to study pathways for information flows; and in the second, a naive model of robustness to failing power lines. We find that the connected CM delays the onset of the connected phase (i.e. the LCC starts growing at higher occupation probabilities) and forces fixation to full connectivity as the occupation probability goes to one. The delayed onset of the connected phase is most likely a consequence of the disassortativity of connected treelike graphs. The full connectivity at high occupation probability is a trivial consequence of the ensembles containing only connected graphs. Perhaps more interestingly, the connected CM also captures the convex relationship between the size of the LCC and the occupation probability at the onset of connectedness. This convex relationship is often a consequence of core–peri- phery structure, both in dense and sparse graphs, as shown in Refs. [11, 12]. Our results here show that the degree sequence and the connectedness can also explain this feature, and more parsimoniously so.

4 Discussion

In this paper, we have proposed an efficient heuristic that can generate samples from the connected configuration model with hard degree constraints. We have shown that this heuristic is fast on small graphs, and that it always returns a graph as long as the input degree sequence is graphical. By way of three case studies, we have then demonstrated how this algorithm can be used to analyze real systems where connectedness is key. In doing so, we have found, for example, that while graphs with a low density can seem disassortative at first glance, their disassortativity can be explained by the fact that they have to be connected in the first place. In the same manner, we have also found that connectivity can alter the communities found by modularity maximization algorithms or the interpretation of a percolation process.

Throughout the second part of this paper, we have found time and time again that connectedness constraints matter the most when the density of the modeled system is low. And this was, in a sense, expected. After all, two classical results from graph theory show that a high average degree automatically implies connectedness for a large portion of the nodes of a completely random graph [2] as well as for random graphs with fixed degree sequence [5]. Or in other words: Dense random graphs are already connected, whether we ask them to be or not. It follows that imposing connectedness is most important in the regime of sparse graphs. When modeling a sparse system, therefore, we should take special care and ask ourselves: Was this graph expected to be connected? Is there a sampling or construction mechanism that forces us to observe this graph in one piece? These questions help guide our choice of null model. And as we have shown, this choice can certainly change our conclusions—so beware.

Acknowledgements This work is supported by the James S. McDonnell Foundation (JGY) and Grant No. DMS-1622390 from the National Science Foundation (LHD). The authors contributed equally.

References

1. Newman, M.E.J.: Networks: An Introduction. Oxford University Press, Oxford (2010)
2. Erdős, P., Rényi, A.: Publ. Math. **6**, 290 (1959)
3. Gilbert, E.N.: Ann. Math. Stat. **30**(4), 1141 (1959)
4. Connor, E.F., Simberloff, D.: Ecology **60**(6), 1132 (1979)
5. Molloy, M., Reed, B.A., Random Struct. Algoritm. **6**(2/3), 161 (1995). https://doi.org/10.1002/rsa.3240060204
6. Newman, M.E.J., Strogatz, S.H., Watts, D.J.: Phys. Rev. E **64**, 026118 (2001). https://doi.org/10.1103/PhysRevE.64.026118
7. Coolen, T., Annibale, A., Roberts, E.: Generating Random Networks and Graphs. Oxford University Press, Oxford (2017)
8. Frank, O., Strauss, D.: J. Am. Stat. Assoc. **81**(395), 832 (1986)
9. Olhede, S.C., Wolfe, P.J.: Proc. Natl. Acad. Sci. U. S. A. **111**(41), 14722 (2014)
10. Vázquez, A., Moreno, Y.: Phys. Rev. E **67**(1), 015101 (2003)
11. Hébert-Dufresne, L., Allard, A., Young, J.G., Dubé, L.J.: Phys. Rev. E **88**(6), 062820 (2013)
12. Allard, A., Hébert-Dufresne, L.: Phys. Rev. X **9**(1), 011023 (2019)
13. Orsini, C., Dankulov, M.M., Colomer-de Simón, P., Jamakovic, A., Mahadevan, P., Vahdat, A., Bassler, K.E., Toroczkai, Z., Boguná, M., Caldarelli, G., et al.: Nat. Commun. **6**, 8627 (2015)
14. Tishby, I., Biham, O., Katzav, E., Kühn, R.: Phys. Rev. E **97**(4), 042318 (2018)
15. Tishby, I., Biham, O., Katzav, E., Kühn, R.: (2018). arXiv:1810.02198

16. Gray, C., Mitchell, L., Roughan, M.: J. Complex Netw. **7**(6), 896–912 (2019). https://doi.org/10.1093/comnet/cnz011
17. Crane, H.: Probabilistic Foundations of Statistical Network Analysis. Chapman and Hall/CRC, Boca Raton (2018)
18. Barabási, A.L., Albert, R.: Science **286**(5439), 509 (1999)
19. Erickson, B.H.: Sociol. Methodol. **10**, 276 (1979)
20. Lee, S.H., Kim, P.J., Jeong, H.: Phys. Rev. E **73**(1), 016102 (2006)
21. Fosdick, B.K., Larremore, D.B., Nishimura, J., Ugander, J.: SIAM Rev. **60**(2), 315 (2018)
22. Viger, F., Latapy, M.: (2005). arXiv:cs/0502085
23. Viger, F., Latapy, M.: J. Complex Netw. **4**(1), 15 (2015)
24. Gkantsidist, C., Mihail, M., Zegura, E.: In: Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments, vol. 111, p. 16. SIAM, Philadelphia (2003)
25. Havel, V.: Casopis Pest. Mat. **80**, 477 (1955)
26. Hakimi, S.L.: J. Soc. Ind. Appl. Math. **10**(3), 496 (1962). https://doi.org/10.1137/0110037
27. Seierstad, C., Opsahl, T.: Scand. J. Manag. **27**(1), 44 (2011)
28. Newman, M.E.J.: Phys. Rev. Lett. **89**(20), 208701 (2002)
29. Newman, M.E., Park, J.: Phys. Rev. E **68**(3), 036122 (2003)
30. Hadfield, J., Megill, C., Bell, S.M., Huddleston, J., Potter, B., Callender, C., Sagulenko, P., Bedford, T., Neher, R.A.: Bioinformatics **34**(23), 4121 (2018)
31. Zimmerman, R.D., Murillo-Sánchez, C.E., Thomas, R.J.: IEEE Trans. Power Syst. **26**(1), 12 (2010)
32. Fortunato, S., Hric, D., Phys. Rep. **659**, 1 (2016)
33. Young, J.G., St-Onge, G., Desrosiers, P., Dubé, L.J.: Phys. Rev. E **98**(3), 032309 (2018)
34. Hébert-Dufresne, L., Allard, A., Young, J.G., Dubé, L.J.: Sci. Rep. **3**, 2171 (2013)
35. Riolo, M.A., Newman, M.: (2019). arXiv:1908.09867
36. Newman, M.E.J., Girvan, M.: Phys. Rev. E **69**(2), 026113 (2004)
37. Newman, M.E.: Proc. Natl. Acad. Sci. U. S. A. **103**(23), 8577 (2006)
38. Reichardt, J., Bornholdt, S.: Phys. Rev. E **74**(1), 016110 (2006)
39. White, S., Smyth, P.: In: Proceedings of the 2005 SIAM International Conference on Data Mining, pp. 274–285. SIAM, Philadelphia (2005)
40. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: J. Stat. Mech. Theory Exp. **2008**(10), P10008 (2008)
41. Traag, V.A., Van Dooren, P., Nesterov, Y.: Phys. Rev. E **84**(1), 016114 (2011)
42. Peel, L., Larremore, D.B., Clauset, A.: Sci. Adv. **3**(5), e1602548 (2017)
43. Expert, P., Evans, T.S., Blondel, V.D., Lambiotte, R.: Proc. Natl. Acad. Sci. U. S. A. **108**(19), 7663 (2011)
44. Peixoto, T.P.: Figshare (2014)
45. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: J. Mach. Learn. Res. **12**, 2825 (2011)
46. Newman, M., Cantwell, G.T., Young, J.G.: (2019). arXiv:1907.12581
47. Newman, M.E.J.: Phys. Rev. E **66**(1), 016128 (2002)