Improving mean-field network percolation models with neighbourhood information

Chris Jones [iD][1,2] and Karoline Wiesner [iD][2]

[1]School of Mathematics, University of Bristol, Bristol, UK
[2]Institute of Physics and Astronomy, University of Potsdam, Potsdam, Germany

Abstract

Mean field theory models of percolation on networks provide analytic estimates of network robustness under node or edge removal. We introduce a new mean field theory model based on generating functions that includes information about the tree-likeness of each node's local neighbourhood. We show that our new model outperforms all other generating function models in prediction accuracy when testing their estimates on a wide range of real-world network data. We compare the new model's performance against the recently introduced message passing models and provide evidence that the standard version is also outperformed, while the 'loopy' version is only outperformed on a targeted attack strategy. As we show, however, the computational complexity of our model implementation is much lower than that of message passing algorithms. We provide evidence that all discussed models are poor in predicting networks with highly modular structure with dispersed modules, which are also characterised by high mixing times, identifying this as a general limitation of percolation prediction models.

1 Introduction

Examples of complex networks may be found across many different real world systems, such as protein-protein interactions [1], online social networks [2], financial transactions [3], power grids [4] and ecosystems [5]. For many of these systems, one property of interest is robustness, which typically refers to how well a network can retain its structure and/or function as it undergoes some disruption, such as the removal of nodes or edges [6]. Robustness is of interest because we may want to know how real world systems can withstand or react to damage of some kind, whether that is from random events such as extreme weather or from targeted attacks such as the intentional taking down of internet servers. If we can predict how real-world networks will behave in these scenarios and identify what makes them robust or fragile, then we can potentially alter their structures to make them more or less robust in the future.

The most prevalent method for modelling network robustness is via the use of percolation theory, which may model phenomena such as the spreading of disease through a population [7], the dynamics of traffic in a city [8], or intentional attacks on internet servers [9]. With percolation theory, we consider the systematic removal of nodes or edges from a network in either a random order, or in a specific order according to some targeting. Percolation may be computationally simulated [10] but this approach has two crucial drawbacks: it is computationally expensive, and it cannot explain the structural properties of networks that affect robustness. If we wish to predict network robustness, it is convenient to have a less time consuming method, and if we want to alter network robustness then we want a method which will tell us the network properties which affect robustness.

A more efficient approach to predicting robustness is to use analytical mean-field theory models which make use of generating functions [11]. These generating function models work by using information about the local neighbourhoods of the nodes on the network and averaging this local information over the entire network. The generating function model of [11] assumes random network structure, and generating function models which include correlations and transitivity have since been developed [12,13]. To improve upon the generating function models, we introduce a new metric which measures how tree-like the local neighbourhoods of nodes in a network are. Based on this metric, which we call the *tree factor*, we introduce a new generating function model. We compute analytic estimates of percolation

1

robustness using this model and show that it outperforms all existing generating function models when tested on a large range of real-world network data.

Based on this success, we studied the remaining inaccuracy in the analytic estimates for some of the networks and looked for general

conditions under which generating function models are unable to effectively predict network robustness. The conditions we were able to identify are a highly modular structure with dispersed modules. We show that these poorly predicted networks also have high mixing times for simple random walks occurring on them, which gives us a simple, single metric which may be used as a diagnostic for when generating function model results can be considered reliable. Additionally, our results provide useful information about the conditions under which networks are fragile, and point towards possible methods for robustness enhancement.

Recently, mean-field methods have been supplanted in studies of network percolation by message passing methods [14–16]. The stated reasons for moving from generating function to message passing methods include the fact that generating function models assume infinite network size, and that they use summary data such as networks' degree distributions to make their predictions, omitting relevant structural information and making poor predictions [6, 14–16]. By contrast, message passing methods operate upon the adjacency matrix of networks, and so they capture the finite nature of real networks without having to use summary information.

Previous research, however, has established that simple message passing models fail to produce accurate predictions due to their assumptions of local tree-likeness, and even some more sophisticated models exhibit similar problems [15]. We, therefore, compare the accuracy of these methods to the performance of the generating function models. We consider the simplistic message passing model [14] and a recently developed model designed to include information about local loops [16]. We find that in all cases except two our tree-factor model outperforms the message passing algorithms. In all cases does our model provide a computationally much more efficient way of estimating network percolation robustness. Furthermore, we find that message passing models also make more inaccurate predictions for high mixing time networks, indicating a general limitation of percolation prediction methods.

2   Methods

2.1   Existing Generating Function Models

The predictions of robustness in this paper rely upon generating functions, which are a means of representing probability distributions as power series. The probability distributions of interest when predicting network robustness in the simplest generating function model are the degree distribution $p(k)$ and the remaining degree distribution $q(k')$. The degree distribution gives the probability of a randomly selected node having degree k, and the remaining degree distribution gives the probability of reaching a node with k' so-called remaining edges via a randomly selected edge, where k' is $k-1$, i.e. the node's degree with the traversed edge being discounted.

The generating function for the degree distribution is expressed as

$$g_0(z) = \sum_{k=0}^{\infty} p(k)z^k, \tag{1}$$

where z is some arbitrary number, and the generating function for the remaining degree distribution is given by

$$g_1(z) = \sum_{k'=0}^{\infty} q(k')z^{k'} = \frac{1}{\langle k \rangle} \sum_{k=1}^{\infty} (k-1)p(k-1)z^{k-1}, \tag{2}$$

where $\langle k \rangle$ is the average degree of nodes in the network. One approach to understanding network robustness is to consider how well connected a network remains as nodes or edges are removed from it, specifically by measuring the size of the largest connected component (LCC) in the network throughout the process of node or edge removal. Generating functions allow one to model this process by predicting the probabilities of nodes being in the LCC. If one starts with some average probability u that a node is not connected to LCC via one of its neighbours, then the total probability of a node of degree k not being connected to the LCC is u^k. Averaging over all nodes, this gives

$$g_0(u) = \sum_{k=0}^{\infty} p(k)u^k, \tag{3}$$

2

which is the average probability that a node is not connected to the LCC. The value of u will depend upon the ratio of nodes or edges which are still present in the network, defined as φ. The probability of one node not being connected to the LCC via one of its neighbours with k degree is $1 - \varphi + \varphi u^{k-1}$, and this may be averaged over the remaining degree distribution $q(k')$ to give

$$u = 1 - \varphi + \varphi g_1(u), \tag{4}$$

a self consistency relation which gives the average probability of a node not being connected to the LCC via one of its neighbours.

Finally, we may calculate the size of the LCC using Equations (3) and (4). In the case of random node removal, the size of the LCC, S_{node}, is given by

$$S_{node} = \varphi(1 - g_0(u)), \tag{5}$$

which is simply the ratio of removed nodes multiplied by the probability of a node being connected to the LCC. For random edge removal, the size of the LCC S_{edge} may be calculated as

$$S_{edge} = 1 - g_0(u), \tag{6}$$

differing from Equation (5) by a factor of φ. The preceding results apply for random node and edge removal, but we may also want to know how the size of the LCC changes when a network undergoes a series of targeted attacks. The most simplistic method of targeted attack is to remove nodes from the network in descending degree order, and this may also be modelled using generating functions.

Assuming random structure on a network undergoing targeted attacks, we may follow a similar logic to that which gave us $g_0(u)$ from Equation (3). If we start with some average probability u of not being in the LCC, then the probability that a node of degree k is not connected to the LCC via its neighbours is u^k. However, for targeted attacks we must explicitly factor in the probability φ_k of a node of degree k having been removed from the network, and so the probability of not being in the LCC is $\varphi_k u^k$ for a node of degree k. If we remove all nodes above some degree k_0, then $\varphi_k = 0$ when $k > k_0$, and $\varphi_k = 1$ when $k \leq k_0$. If we remove some nodes of degree k_0, then φ_{k_0} will be equal to the proportion of nodes of degree k_0 that have been removed.

Averaging over all nodes, we get the generating function for targeted attacks analogous to $g_0(z)$ from Equation (3)

$$f_0(u) = \sum_{k=0}^{\infty} p(k)\varphi_k u^k. \tag{7}$$

This generating function gives the average probability that a node is both present in the network and is not connected to the LCC. To calculate the probability of a node being in the LCC (and therefore the size of the LCC), we must subtract $f_0(u)$ from the average probability of a node being present in the network. Since we are considering targeted attacks, this probability is dependent upon the degree values of nodes and is given by

$$f_0(1) = \sum_{k=0}^{\infty} p(k)\varphi_k, \tag{8}$$

and therefore the expression for the size of the LCC is

$$S_{targ} = f_0(1) - f_0(u). \tag{9}$$

For targeted attacks, we must also update our expression for u. The probability of some node not being connected to the LCC via a particular neighbour of degree k is $1 - \varphi_k + \varphi_k u^{k-1}$, and averaging this over the remaining degree distribution $q(k')$ gives the self consistency relation

$$u = 1 - f_1(1) + f_1(u), \tag{10}$$

where

$$f_1(u) = \sum_{k'=0}^{\infty} q(k')\varphi_{k'+1} u^{k'}. \tag{11}$$

Note that all of Equations (1)-(11) are sourced from [17]. While these expressions are able to elegantly describe how a network behaves as it undergoes node or edge removal, this approach is only accurate by definition if the network in question is locally tree like. It assumes that the network is configured

3

"randomly", i.e. in keeping with the configuration model [18], and this is often not the case for real-world networks. A greater level of structural detail may be achieved by using generating function models which take correlations [12] or clustering [13] into account. In the following, Equations (12)-(15) are taken from or are derived from [12], and Equations (16)-(19) are from or follow directly from [13].

For random node or edge removal, when including correlations we calculate the quantity u_k, the average probability that an edge connected to a node of degree k connects to another node which is not part of the LCC. This is given by the equation

$$u_k = 1 - \varphi + \varphi \sum_{j=1}^{\infty} \pi(j|k)(u_j)^{j-1}, \tag{12}$$

where $\pi(j|k)$ is the conditional probability of traversing an edge from a node of degree k to a node of degree j. For random node and edge removal respectively, the size of the LCC is therefore given by

$$S_{node}^{corr} = \varphi(1 - \sum_{k=0}^{\infty} p(k)(u_k)^k), \tag{13}$$

$$S_{edge}^{corr} = 1 - \sum_{k=0}^{\infty} p(k)(u_k)^k. \tag{14}$$

For targeted attacks, the expression for $f_1(u)$ becomes

$$f_1(u_k) = \varphi_k \sum_{j=1}^{\infty} \pi(j|k)\varphi_j(u_j)^{j-1}. \tag{15}$$

Equations (7), (8), (9) and (10) may then be updated to include u_k accordingly, which allows us to calculate the size of the LCC for targeted attacks including information about correlations.

For clustering, when modelling random node and edge removal it is necessary to only substitute the expression for $g_1(u)$ with that of $g_C(u)$, where

$$g_C(u) = g_1(C + (1 - C)u), \tag{16}$$

where C is a measure of clustering called transitivity, given by the equation

$$C = \frac{3N_t}{N_3}, \tag{17}$$

where N_t is the number of triangles in the network and N_3 is the number of "triads" in the network. A triad is a formation of three nodes where one node, α, has edges leading to two neighbours, β and γ, which may or may not share an edge.

Consequently, when including clustering the expression for u becomes

$$u = 1 - \varphi + \varphi g_C(u), \tag{18}$$

and the calculations for LCC size under random node and edge removal remain the same as they are in Equations (5) and (6). Finally, when modelling targeted attacks while including information about clustering, the expression for $f_1(u)$ is replaced with $f_C(u)$, where

$$f_C(u) = f_1(C + (1 - C)u), \tag{19}$$

and this may be substituted for $f_1(u)$ into Equation (10) to calculate u and subsequently the size of the LCC for targeted attacks while including information about clustering.

2.2 Message Passing

The other group of models we consider in this paper are known as message passing models, which consider each node and edge in a network individually, as opposed to using a network's degree distribution. The message passing model simulates a message being sent between connected nodes, communicating whether or not they are in the LCC. The most simplistic message passing model for percolation assumes that the given network is locally tree like [14], we describe it here and refer to it as the tree like message passing method.

For a network undergoing percolation, each node i has a probability μ_i of not being in the LCC. This is dependent upon the probabilities that each of its neighbours j to which i is still connected in its local

neighbourhood $\mathcal{N}_i$ are not connected to the LCC via some route that does not include node i. For node and edge removal respectively, this may be written as

$$\mu_i(node) = 1 - \varphi + \varphi \prod_{j \in \mathcal{N}_i} \mu_{j \to i}(node), \tag{20}$$

$$\mu_i(edge) = \prod_{j \in \mathcal{N}_i} (1 - \varphi + \varphi \mu_{j \to i}(edge)), \tag{21}$$

where $\mu_{j \to i}$ is the probability that j is not connected to the LCC via any route which does not contain i. $\mu_{j \to i}$ itself is dependent upon the probability that each of j's neighbours, excluding i, is not connected to the LCC, and so we can define

$$\mu_{j \to i}(node) = 1 - \varphi + \varphi \prod_{k \in \mathcal{N}_j, k \neq i} \mu_{k \to j}(node), \tag{22}$$

$$\mu_{j \to i}(edge) = \prod_{k \in \mathcal{N}_j, k \neq i} (1 - \varphi + \varphi \mu_{k \to j}(edge)). \tag{23}$$

For a given ratio of removed nodes or edges φ, this gives us an LCC connection probability for each edge in the network. Equations (23) and (22) may be solved iteratively until they converge for some φ. We may then calculate the size of the LCC as

$$S(\varphi) = \frac{1}{N} \sum_{i=0}^{N} \mu_i, \tag{24}$$

for both node and edge removal. It is also straightforward to adapt the node removal equations for targeted node removal, where φ will vary depending on the total number of nodes removed and the degree value of the relevant node.

The main drawback of this model is that it assumes local tree-likeness, and only holds true if there are relatively few loops on the network in question. Here we consider a recently developed model which explicitly takes into account the loops in each node's local neighbourhood [16], referred to as the loopy message passing method.

In the loopy message passing model, each node's local neighbourhood is expanded to include loops of a certain length. In the original model, loops of length 2 are included, which describes moving to a node's neighbour and back. The newer model allows for loops of any length, but here we restrict ourselves to loops of length 4.

A node's neighbourhood $\mathcal{N}_i$ is expanded to include any node within a loop of length 4 starting from i. The neighbourhood may be in some configuration Γ_i depending on the nodes or edges removed, and we set $\sigma_{ij}(\Gamma_i) = 1$ if there is a path between i and j in state Γ_i, and 0 if not. The probability of i being in the LCC becomes

$$\mu_i = \sum_{\Gamma_i} P(\Gamma_i) \prod_{j \in \mathcal{N}_i} \mu_{j \to i}^{\sigma_{ij}(\Gamma_i)}, \tag{25}$$

where $P(\Gamma_i)$ is the probability of the configuration Γ_i occurring. While this expression averages over all possible Γ_i configurations, this is highly impractical and computationally expensive, and so Monte Carlo simulations may be used instead to estimate μ_i. As before, we may calculate $\mu_{j \to i}$, but the neighbourhood we consider is $\mathcal{N}_{j \to i}$, which is the neighbourhood of j excluding nodes or edges in the neighbourhood of i. $\mu_{j \to i}$ is given by

$$\mu_{j \to i} = \sum_{\Gamma_{j \to i}} P(\Gamma_{j \to i}) \prod_{j \in \mathcal{N}_{j \to i}} \mu_{k \to j}^{\sigma_{jk}(\Gamma_{j \to i})}, \tag{26}$$

where again Monte Carlo simulations are used to estimate $\mu_{j \to i}$ by averaging over different $\Gamma_{j \to i}$ configurations. As before, Equation (26) may be solved iteratively to convergence, and the LCC size may be calculated using Equation (24).

2.3 Simulating Percolation and Calculating Robustness

In order to evaluate how well the models we consider predict robustness, we must compare their results against direct simulations of node and edge removal. This may be done using an algorithm devised by Newman and Ziff [10], where nodes or edges are added to a network sequentially, and at each point the size of the LCC is measured. This is equivalent to a time reversed procedure of node or edge removal. This method of simulation requires several repetitions in order to get an accurate average for the size of the LCC at each stage of node or edge removal. From these simulations, we can plot a curve of the LCC size against the proportion of active nodes or edges in the network, and we show examples in Figure 1.
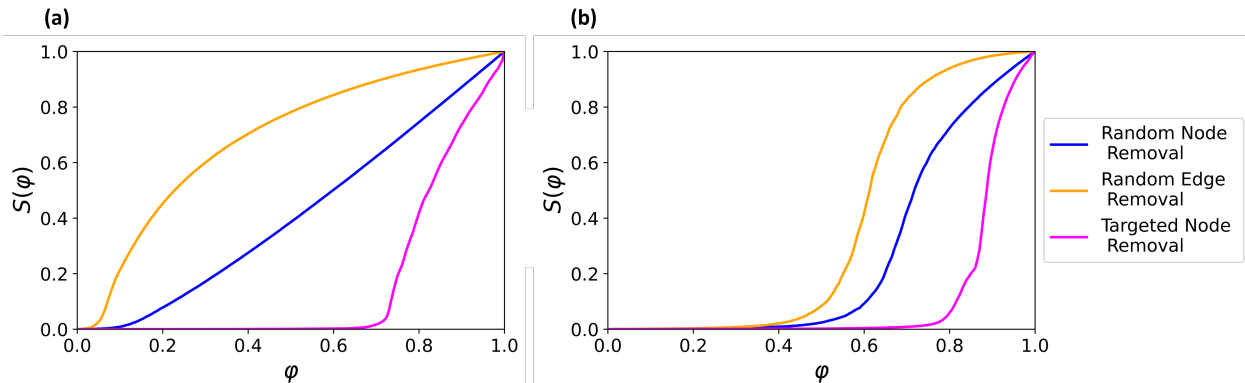

Figure 1: LCC size $S(\varphi)$ against ratio of active nodes or edges φ for **(a)** the lastfm social network [19] and **(b)** a power grid in the western United States [20]. Robustness curves are given for random node removal, random edge removal, and targeted node removal.

Information about the robustness of a network may be condensed into a simple, single metric using the LCC curve. The robustness measure proposed by Schneider et al [21] measures the area underneath the curve of the size of the LCC plotted against the ratio of nodes or edges removed from the network. The robustness measure ρ is given by

$$\rho = \int_0^1 S(\varphi)d\varphi. \tag{27}$$

For random and targeted node removal, the robustness ρ is between 0 and 0.5, and for edge removal ρ is in the range of 0 to 1. This robustness measure captures the information provided by the entire LCC curve, allowing one to draw conclusions about how a network behaves throughout the entire process of node or edge removal.

2.4 Measuring Prediction Inaccuracy

In this paper we consider how accurate the percolation predictions made using generating function models are compared to "real" results obtained via simulation. One method for doing this is to measure the coefficient of determination, R^2, between robustness ρ values obtained via simulation and via generating function predictions. However, this method only works for a dataset of many networks. If we want to assess how well a single network is predicted by generating function models, we require an alternative method. To assess prediction inaccuracy for an individual network, we consider the area between the predicted LCC curve and the real LCC curve, originally developed as a method of assessing prediction inaccuracy in [15]. Prediction inaccuracy is given by

$$\Delta = \int_0^1 |S_{real} - S_{predict}|d\varphi, \tag{28}$$

which ranges between 0 and 0.5 for random and targeted node removal, and 0 and 1 for random edge removal. Higher values indicate more inaccurate predictions. These Δ values may then be compared against a variety of network properties in order to determine the instances in which generating function models fail to accurately predict network robustness. We consider the cutoff for a prediction being highly inaccurate to be $\Delta \geq 0.05$, which is 10% of maximum inaccuracy for random and targeted node removal and 5% of maximum inaccuracy for random bond removal.

6

2.5 Modularity

In our results, we find that one network measure which indicates how well generating function models predict percolation is the modularity of a network's maximum modularity partitioning. Modularity is measured by dividing up a network into several different communities, and then summing over the difference between the number of edges any two nodes v and w in a community share and the expected number of edges between them [22]. This may be written as

$$Q = \frac{1}{2m} \Big(\sum_{vw} A_{vw} - \frac{k_v k_w}{2m} \Big) \delta(c_v, c_w), \tag{29}$$

where m is the number of edges, A_{vw} is the adjacency matrix value for nodes v, w which takes the value 1 if there is an edge between them and zero otherwise, k_i is the degree of the ith node, c_i is the community to which node i belongs and $\delta(c_v, c_w)$ takes a value of 1 if nodes v and w are in the same community, and 0 otherwise. Modularity takes values from $-\frac{1}{2}$ to 1, with lower values for networks with less modular and more random structure, and higher values for networks with highly modular and non-random structure.

Several algorithms exist for computing maximum modularity. They divide a given network into communities such that the modularity Q is maximized [23, 24], with the most recent innovation being the Leiden algorithm [25], with computational time scaling almost linearly with number of nodes. In our results, we use the Leiden algorithm in order to partition real networks and measure the modularity Q of these partitions.

2.6 Mixing Time

Another network measure which we relate to the quality of generating function predictions is mixing time. The mixing time of a network may be estimated analytically by using spectral methods on the transition matrix for a simple random walk. A simple random walker on a network will move from some node to any one of its neighbours with uniform probability, and so a network's transition probability matrix P for this kind of walk is given by

$$P = D^{-1}A, \tag{30}$$

where D is the network's diagonal matrix, which has each node's degree value on the diagonal and 0 values elsewhere, and A is the network's adjacency matrix, where for simple undirected networks each entry a_{ij} is 1 if there is an edge between nodes i and j, and 0 otherwise. It has been established that the reciprocal of the spectral gap between the two largest eigenvalues of the transition matrix P give an estimate of mixing time [26]. The mixing time measures how long it takes for a random walk to reach a stationary state - in other words, the expected length of time before the trajectory of the random walker is unaffected by its initial starting point. The largest eigenvalue of P is simply $\lambda_1 = 1$, so the mixing time is estimated as simply

$$t_{mix} = \frac{1}{1 - \lambda_2}, \tag{31}$$

where λ_2 is the second largest eigenvalue of the transition matrix P. λ_2 can be found fairly rapidly on sparse matrices (and therefore sparse networks) using the implicitly restarted lanczos method [27] implemented in computational packages such as scipy.sparse.linalg [28], and so it is easy to quickly estimate mixing time on the real networks we consider. Mixing time has previously been related to the small world-ness of networks [29] and can be used to indicate when predictions of certain network properties will be inaccurate [30–32], and we use it in the following for a similar purpose.

3 Results

3.1 Predicting Structural Robustness with Generating Function and Message Passing Models

To assess how accurate the predictions made by generating function models are, we can compare predicted robustness from these models against real robustness obtained from repeated Newman-Ziff simulations. For this, we use a dataset of 100 different real-world networks, including social, biological, ecological technological and infrastructural networks. A table of these networks is provided in the Supplementary

Materials. Some real world networks are recorded as having separate, disconnected components, and so for consistency we only consider the largest connected component in the initial data. This means that all of the networks we consider are initially connected. We obtain real robustness values for these 100 networks via simulation and for each generating function method we calculate predicted robustness. We then measure the coefficient of determination, R^2, between the real and predicted robustness values. Since the real robustness values are calculated by simulating node or edge removal, we repeat the simulations between 10 and 200 times depending on the size of the network. From these repeated tests, we are able to get average real robustness values and the error in robustness. These results for the three generating function models described in Section 2.1 are presented in Figure 2.
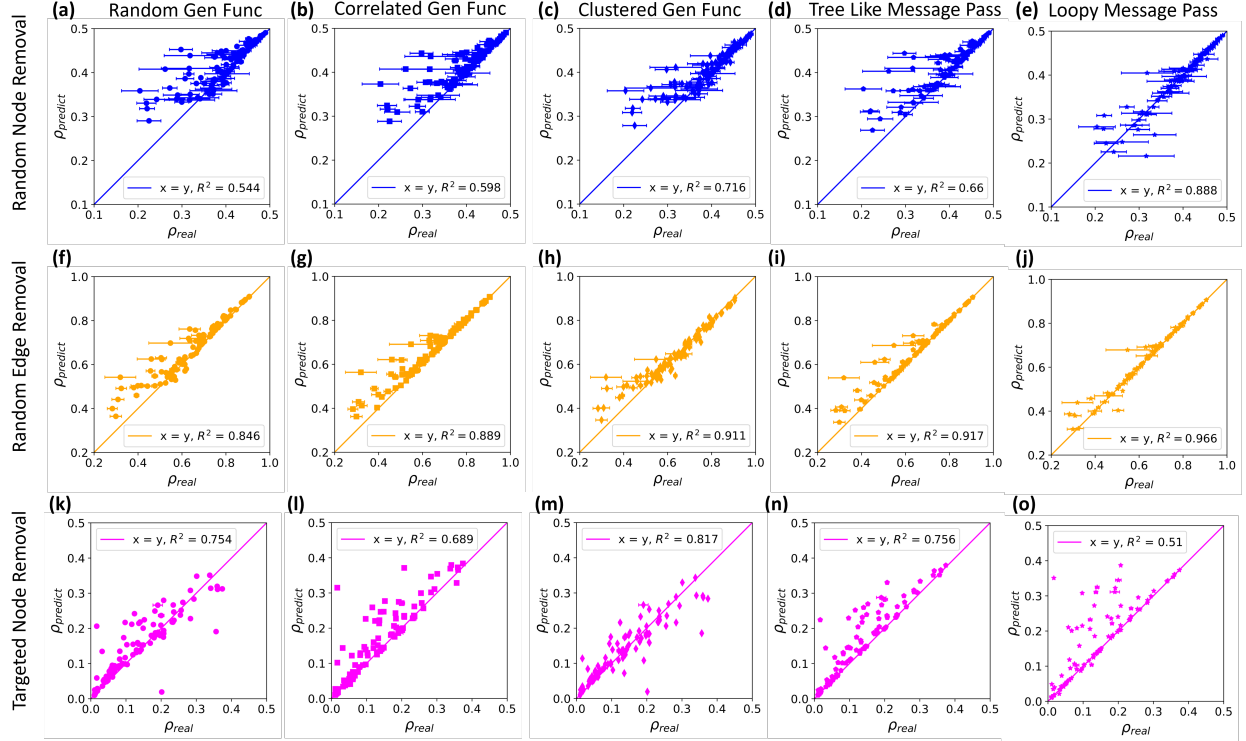

Figure 2: Comparisons of real and predicted robustness values for generating function models. Plots in blue are for random node removal, plots in orange are for random edge removal, and plots in purple are for targeted node removal. **(a)** **(f)** and **(k)** are for generating function predictions assuming random structure, **(b)**, **(g)** and **(l)** are for generating function predictions including correlation information, **(c)**, **(h)** and **(m)** are for generating function predictions including clustering information, **(d)**, **(i)** and **(n)** are predictions for the tree like message passing method and **(e)**, **(j)** and **(o)** are predictions for the loopy message passing method. The error bars for all plots represent the error in robustness from simulations.

For random node and edge removal, predicted values are generally closer to real values for more robust networks. For random and targeted node removal, there are some significant discrepancies between predicted and real robustness values for all models, but for edge removal, the predictions are typically more accurate. It is also notable that for almost all models and removal scenarios, the robustness predictions mostly over estimate robustness when the predictions are inaccurate.

We can see from Figure 2 that while the generating function models are generally indicative of robustness, there can still be significant discrepancies between real and predicted robustness values, especially for random and targeted node removal. Clearly, there are elements of network structure which are relevant to robustness that are ignored by these models. The tree like message passing model only outperforms generating function models for random edge removal by a small margin, indicating that this model is not much of an improvement upon generating function models in terms of producing accurate predictions. The loopy message passing model outperforms all others for random node and random edge removal significantly, but is the worst model for predicting targeted node removal.

Additionally, it should be noted that the message passing models are more computationally expensive

than the generating function models. For the generating function models, there is only one self consistency equation to solve for each active ratio of nodes or edges. For correlated structure, there are $n_{p \neq 0}$ equations, where $n_{p \neq 0}$ is the number of non-zero entries in the degree distribution. In comparison, the tree like message passing method has $2E$ self consistency equations, where E is the number of edges in the network, and for the loopy model there are estimated to be $N\log N$ equations for sparse networks [16], where N is the number of nodes in the network. As the authors of the loopy model note, this can make the loopy network passing method on par with directly simulating percolation in terms of computational expense, and so message passing methods may not be favourable for limited computational resources or larger networks. For our dataset of 100 networks, the median value of $\frac{2E}{n_{p \neq 0}} = 319$, and the median value of $\frac{N\log N}{n_{p \neq 0}} = 614$, so message passing methods are considerably more computationally expensive than generating function models. Furthermore, it is possible for $n_{p \neq 0}$ to remain quite small on very large but sparse networks, making generating function models more suited to efficiently predicting robustness of very large sparse networks. Data on values of $n_{p \neq 0}$, $2E$ and $N\log N$ for our dataset is given in the Supplementary Materials. With this in mind, it is worth attempting to refine generating function models further for applications on large networks or where computational resources are scarce.

3.2 Introducing the Tree Factor and a New Generating Function Model

In order to improve upon the predictions of existing generating function models, we can alter them such that they do not assume locally tree like structure. The generating function model which uses transitivity [13] does this to a certain extent, and several message passing models [15, 16] include more precise information about the local neighbourhoods of nodes such that they do not assume local tree likeness. Such an approach has even been taken for calculating the mean first passage times of random walks on networks [31], and so our new model follows a similar logic to these other methods for incorporating realistic local neighbourhood information.

Here we adapt the generating function model for correlated structure to include information about how locally tree-like the network is. Let us consider a network for which no nodes or edges have been removed, but may be made up of disconnected components. Starting from some node A of degree k_A, the node is not connected to the LCC with probability $(u_A)^{k_A}$, where u_A is the average probability that a neighbour of A is not connected to the LCC. To find u_A, we want to know the probability that each of A's neighbours B, C, D are not in the LCC.

In the original generating function model, we would assume that each neighbour B, C, D of A has unique second neighbours, i.e. B, C, D do not share any neighbours besides A, and to get u_A we would average over values of $(u_B)^{k_B-1}, (u_C)^{k_C-1}, (u_D)^{k_D-1}$. However, this is not typically the case on real networks. To bring the model closer to reality, we could adjust each of the exponents k_B-1, k_C-1, k_D-1 such that they reflect how B, C, D connects A to its second neighbours. In order to do this, we count each second neighbour only once, divide up second neighbours between each first neighbour they are connected to, and adjust the degree of each first neighbour accordingly when calculating the probability that a given first neighbour is not in the LCC. For node removal, we count nodes that are distance two (i.e. shortest path length two) away from the origin node A as second neighbours, and for edge removal we count nodes that are two non-backtracking steps away from the origin node A as second neighbours. An example node neighbourhood is shown in Figure 3.

For the generating function model, we must average over degree values, and so cannot individually adjust the remaining degree of every first neighbour in every local neighbourhood. For a given origin node, we can get an average adjustment factor, which we call the tree factor, T, by dividing the number of second neighbours by the total remaining degrees of the first neighbours. This may be written as

$$T = \frac{n_2}{\sum k'}, \tag{32}$$

where n_2 is the number of second neighbours, and k' is the remaining degree of some first neighbour. In our example neighbourhood from Figure 3, node A has a tree factor of $T_A = \frac{1}{2}$ when treating nodes at distance two as second neighbours, and a tree factor of $T_A = \frac{3}{4}$ when treating nodes at two non-backtracking steps away as second neighbours. Using A's tree factor T_A, we could calculate u_A as the average of $(u_B)^{(k_B-1)T_A}, (u_C)^{(k_C-1)T_A}, (u_D)^{(k_D-1)T_A}$

Expanding this over an entire network, we can define the degree-averaged tree factor for the average node of some degree k, given by

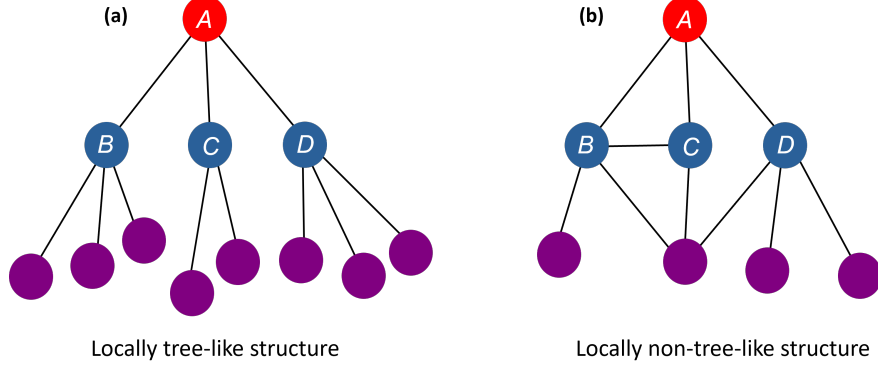$$T_k = \frac{\langle n_2 \rangle_k}{\langle \sum k' \rangle_k}, \tag{33}$$

9

Figure 3: Example local neighbourhoods of some node A. Neighbourhood **(a)** exhibits locally tree-like structure, whereas **(b)** is not locally tree-like. To provide an example of how degree values are adjusted, node B connects to 3 unique nodes in **(a)**, whereas in **(b)** node B is counted as connecting to $\frac{4}{3}$ nodes at distance two from A, and $\frac{7}{3}$ nodes at two non-backtracking steps away from A.

where we divide the average number of second neighbours by the average total remaining degree of first neighbours for some starting node of degree k.

Using the degree-averaged tree factor, we can scale up our calculation for u_A across an entire network for any given node of degree k, and once we reintroduce considerations of node and edge removal this leads us to a new expression for u_k, similar to 12, where we use the degree averaged tree factor T_k to allow for averaging over the entire network.

$$u_k = 1 - \varphi + \varphi \sum_{j=0}^{\infty} \pi(j|k)(u_j)^{(j-1)T_k}. \tag{34}$$

And for targeted node removal we can adjust Equation (15) in a similar manner

$$f_1(u_k) = \varphi_k \sum_{j=0}^{\infty} \pi(j|k)\varphi_j(u_j)^{(j-1)T_k}. \tag{35}$$

Using Equation (34) in conjunction with Equations (13) and (14), and using Equation (35) with Equations (10) and (9), we may make new predictions for network robustness. As before, this new tree factor generating function model requires solving $n_{p \neq 0}$ self consistency equations in order to find the value of $S(\varphi)$ for some φ. The predictions of network robustness made by this model are given in Figure 4.
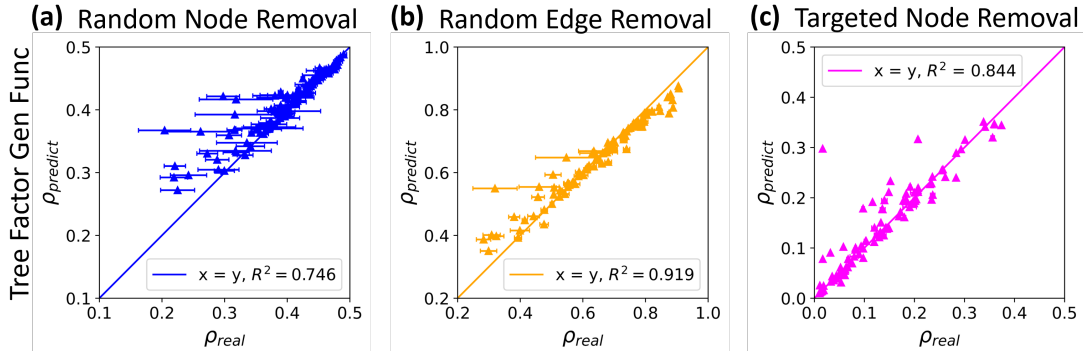

Figure 4: Comparison between predicted robustness using tree factor generating function model and real robustness for various real networks. **(a)** is for robustness against random node removal, **(b)** is for random edge removal, and **(c)** is for targeted node removal.

Overall, this method of prediction outperforms the previously discussed generating function models and the tree like message passing model for all types of node and edge removal. However, there are still

some outlier networks which negatively affect the overall prediction accuracy. Specifically, the tree factor generating function model substantially overestimates the robustness of some networks. In the following we identify the properties of these outliers which explain why they are poorly predicted.

3.3 Limitations of mean-field models

In the Supplementary Materials, we consider two possible sources of inaccuracy: network size, and lack of local network information. We conclude that both of these are insufficient explanations for the prediction outliers we see in Figure 4. However, we are still missing information which cannot be easily captured using a mean field theory approach. One possibility is that networks are inaccurately predicted if they exhibit very different structural properties in different areas of the network, and so the statistical assumptions of the generating function models break down for such networks. In order to identify such networks, we may measure how well they may be subdivided into smaller local communities, and measure the quality of these divisions by calculating their modularity.

We partition real world networks from our dataset and calculate their modularity for partitions found using modularity maximising methods described in Section 2.5. Prediction inaccuracy Δ is also calculated for each individual network using the method from Section 2.4. These modularity and prediction inaccuracy values are compared in Figure 5 for the tree structure generating function model predictions. Modularity is compared against prediction inaccuracy for random node removal, random edge removal, and targeted node removal.
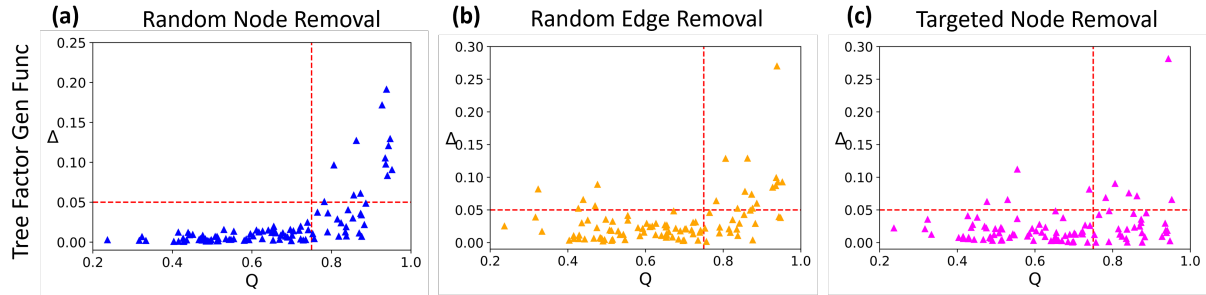

Figure 5: Prediction inaccuracy against modularity. **(a)** is for robustness against random node removal, **(b)** is for random edge removal, and **(c)** is for targeted node removal. Dashed lines are included to mark the points at which $Q = 0.75$ and $\Delta = 0.05$.

We can see that low modularity networks are generally well predicted by the tree factor generating function model, whereas poorly predicted networks tend to have high modularity, and this is true for all methods of removal. We would expect that the outliers in Figure 4 are primarily highly modular networks based on these results. While this allows us to identify a group of networks in which we will find poor predictions, this does not perfectly isolate the poorly predicted networks from the well predicted networks.

Given this consideration of modularity and its impact on robustness predictions, it is important to note that generating function models which incorporate the modular structure of networks also exist [33–35]. The relevant generating function equations and resulting equations for network LCC size are given in the Supplementary Materials. Using the generating function model which incorporates modular structure, we find that it does not produce robustness predictions which are more accurate than the tree factor model. Additionally, we establish that prediction inaccuracy does not stem from any sort of heterogeneity between the internal structure of different modules. These results are all provided in the Supplementary Materials. If the internal structures of the modules are not the primary source of prediction inaccuracy, then we are led to consider the structure which exists between different modules on highly modular networks.

3.4 Modular Dispersal

In order to understand the structure which exists between modules in networks, we constructed modular pseudographs where modules are represented as nodes, and edges exist between nodes only if there is a direct connection between the modules they represent. An example of this is given in Figure 6, where

Zachary's karate club network [36] is partitioned using the Leiden algorithm and a modular pseudograph is drawn based on these modules.
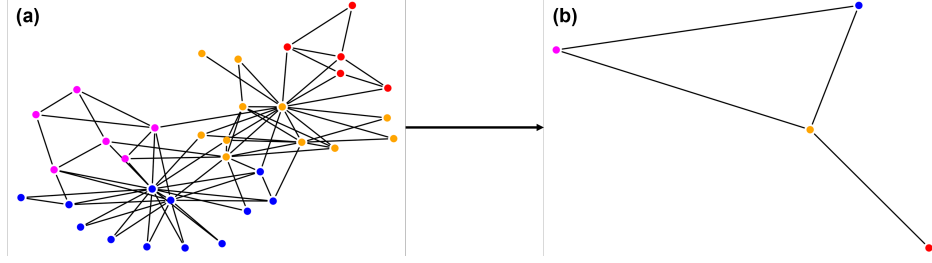

Figure 6: Reduction of the Zachary karate club network **(a)** into a modular pseudograph **(b)**.

Applying this to highly modular networks, we can examine the interconnection between their modules. Two examples are given in Figure 7, where Figure 7**(c)** depicts the pseudograph of a network which is well predicted by the tree factor generating model, and Figure 7**(a)** is the pseudograph of a poorly predicted network.
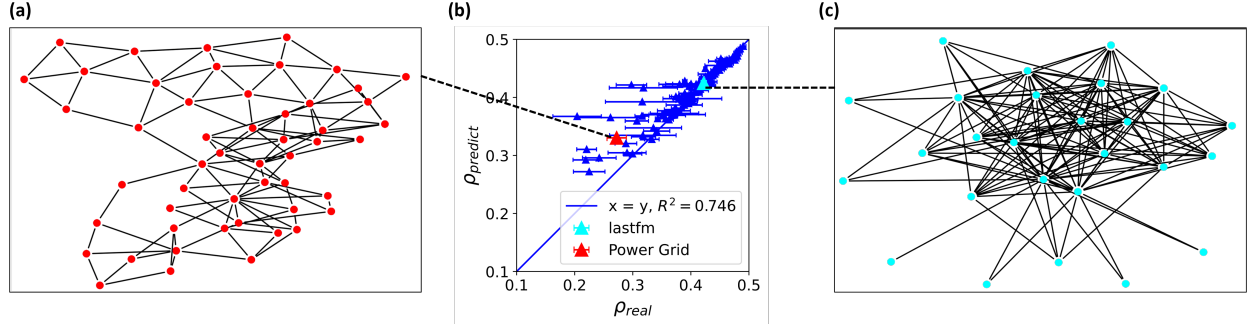

Figure 7: Two modular pseudographs and a comparison of real and predicted robustness values against random node removal using tree factor predictions. **(a)** is the pseudograph of a power grid in the western United States [20], where the original network has $Q = 0.935$. **(b)** is a replication of Figure 4**(a)**, with the lastfm and power grid networks highlighted. **(c)** is the pseudograph for the lastfm social network [19], with the original network having $Q = 0.819$.

We can see that the modules of the pseudograph in Figure 7**(c)** are generally closely connected to one another, whereas in Figure 7**(a)** they are substantially more dispersed, and so reaching one module from another may require passing through many other modules. This dispersal may be measured by the average shortest path length on the pseudograph, and we define this dispersal D as

$$D = \frac{\sum_{i,j}^{N} dist(v_i, v_j)}{N(N-1)}, \tag{36}$$

where N is the number of modules and $dist(v_i, v_j)$ is the shortest distance between modules v_i and v_j. $N(N-1)$ gives the total number of paths in the pseudograph, but this only holds true if the pseudograph is connected. For all networks we consider this is true, as we start by considering the largest connected component in the initial network data. Note that the dispersal D is calculated by treating modules as single nodes with no interior structure, and so the size of the modules themselves does not influence dispersal, only the connections between them. In Figure 8 modular dispersal D is compared to prediction inaccuracy for networks with $Q > 0.75$, which make up 32 networks out of our dataset of 100.

Networks with more dispersed modules, as measured by pseudograph average shortest path length, tend to be more inaccurately predicted by the tree factor generating function model, although some well predicted networks are classed as highly dispersed, particularly for targeted node removal. This allows us to identify and isolate a subset of networks for which many predictions will be highly inaccurate, and to understand the structural characteristics of such networks.
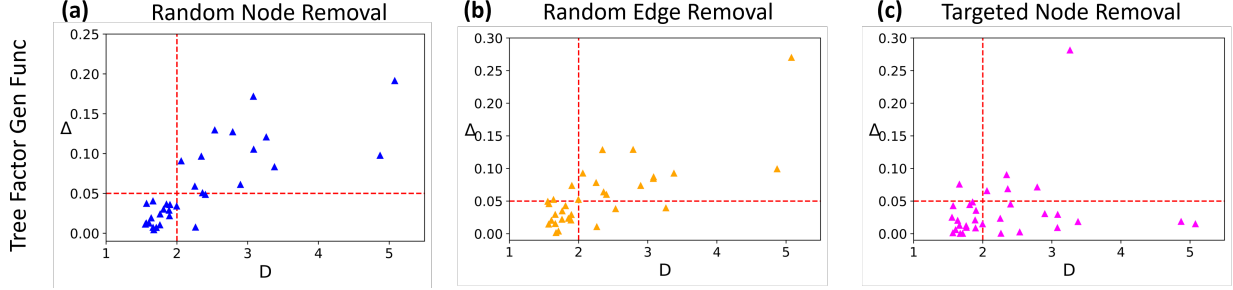
12

Figure 8: Prediction inaccuracy for highly modular networks against modular dispersal. **(a)** is for robustness against random node removal, **(b)** is for random edge removal, and **(c)** is for targeted node removal. Dashed lines are included to mark the points at which $D = 2$ and $\Delta = 0.05$.

However, this approach has some drawbacks. Firstly, modularity itself is not a property of a network, but instead a measure of how well a given partition divides up a network. Modularity maximisation is only one approach to understanding network community structure, and recent research has established that there is no one "ideal" community detection method [37]. At best, we might conclude that networks we call "highly modular" have strong indications of community structure, but the limitations of modularity maximising methods mean that this is not certain, and the communities found using this method may not be "true" communities. Secondly, identifying highly modular, highly dispersed networks is a two step process, so clearly a single heuristic would be preferable for identifying poorly predicted networks. Therefore, we must consider other properties which may be possessed by highly modular, highly dispersed networks.

3.5 High Mixing Time Networks

In the Supplementary Materials, we investigate the possibility that poorly predicted networks are lattice-like and non-small world, using Watts-Strogatz models for reference, but find that this is not the case when using metrics such as transitivity and average shortest path length to measure lattice-likeness and small worldness. However, this is only one approach to understanding small worldness. While networks with small average shortest path lengths have small distances between nodes, this does not necessarily mean there are many paths between nodes, and this is suggested by the presence of highly modular, highly dispersed networks with relatively low average shortest path lengths. On such networks, we might expect that a random walk process may take a long time to reach nodes, as the walker may get stuck within modules. In order to measure how well a random walker may move around a network, we may measure the mixing time t_{mix} described in Section 2.6, which previous research has indicated [29] is a better indicator of the traversability of a network than metrics such as average shortest path length. We evaluate mixing time t_{mix} on our dataset of real networks and compare with prediction inaccuracy Δ, and these results are shown in Figure 9.

We see that high mixing time networks tend to be more inaccurately predicted for random node and random edge removal for both the tree factor generating function model and the loopy message passing model. If we consider all networks with $t_{mix} > 550$ to be "high mixing time", we find that 13 of 14 networks with $\Delta > 0.05$ for random node removal are classed as high mixing time networks. Similarly, if we consider networks with $Q > 0.75$ and $D > 2$ to be highly modular, highly dispersed networks, we find that 13 out of 15 highly modular, highly dispersed networks are also classed as high mixing time networks.

Interestingly, this result is similar to results found by Maier et al. [30–32], where Maier and Brockmann [30] developed a method for predicting the average time to visit all nodes on a network (cover time) and applied this to real networks, but found that this method would break down on networks with sufficiently large mixing times. Similarly, in work by Maier et al. [31], cover time was predicted on artificially constructed networks by relating it to the average time to visit one node starting from any other (mean first passage time), which is itself related to mixing time, and these predictions also broke down for conditions which resulted in high mixing times.

To demonstrate the usefulness of identifying high mixing time networks, we can replot Figure 4 but exclude networks with $t_{mix} > 550$. This gives Figure 10. Similar results are provided in the Supplementary Materials, where robustness comparison plots are given for other generating function
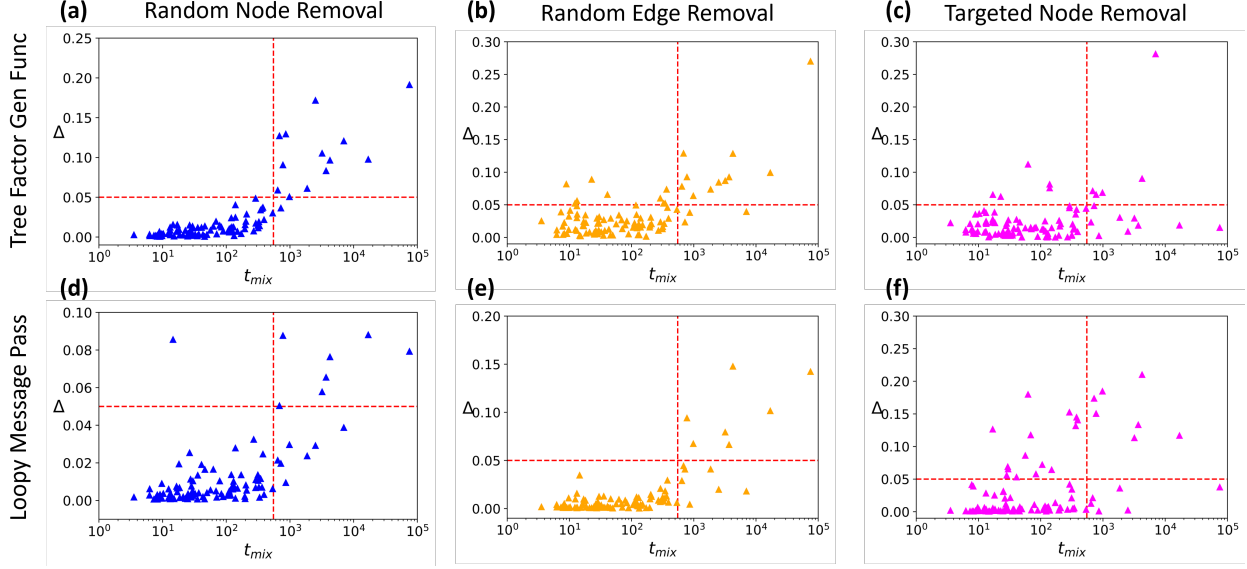
Figure 9: Prediction inaccuracy on real networks against mixing time t_{mix} plotted on a log scale. **(a)**, **(b)** and **(c)** are for the tree factor generating function model, and **(d)**, **(e)** and **(f)** are for the message passing model. For random node and random edge removal, inaccuracy tends to be higher for high mixing time, and for targeted node removal there is no clear relationship. Dashed lines are included to mark the points at which $t_{mix} = 550$ and $\Delta = 0.05$.

model methods while excluding high mixing time networks. For all generating function models and methods of node or edge removal, we see a significant improvement in overall prediction quality once high mixing time networks are excluded.

By excluding high mixing time networks, the overall accuracy of predicted robustness values (as measured by the coefficient of determination) is substantially improved. The number of excluded networks is 14 out of 100, which is a sizeable portion, but nevertheless this result indicates that the tree factor generating function model predicts robustness against random node removal, random edge removal and targeted node removal quite accurately for a significant majority of networks. When high mixing time networks are excluded, the tree factor model outperforms all other generating function models with the exception of the correlated structure generating function model for random edge removal, which exhibits $R^2 = 0.971$. The tree factor also outperforms the tree like message passing model for random node and targeted node removal, but not random edge removal. As before, the loopy message passing model is the most accurate model for random node and random edge removal, but the least accurate for targeted node removal.

The robustness comparison data for other models when excluding high mixing time networks is provided in the Supplementary Materials.

4 Conclusion

We have shown that existing generating function models have limited effectiveness when attempting to predict the robustness of real-world networks as they necessarily ignore certain aspects of network structure. The tree like message passing model is only a marginal improvement on these models for random edge removal, and while the loopy message passing model is the best for random node and random edge removal predictions, the message passing models are considerably more computationally expensive than generating function methods. In order improve upon exisitng generating function models, we have introduced a method to measure the local tree likeness of networks.

We defined a new quantity, the tree factor T. This is a local property which measures the ratio of a node's second neighbours to the possible maximum second neighbours it could have. Averaging this quantity for each degree value in a network, we can create a new generating function model which accounts for how locally tree like the neighbourhoods of nodes are and produces predictions of network robustness that are more accurate than those made by other generating function models and the tree
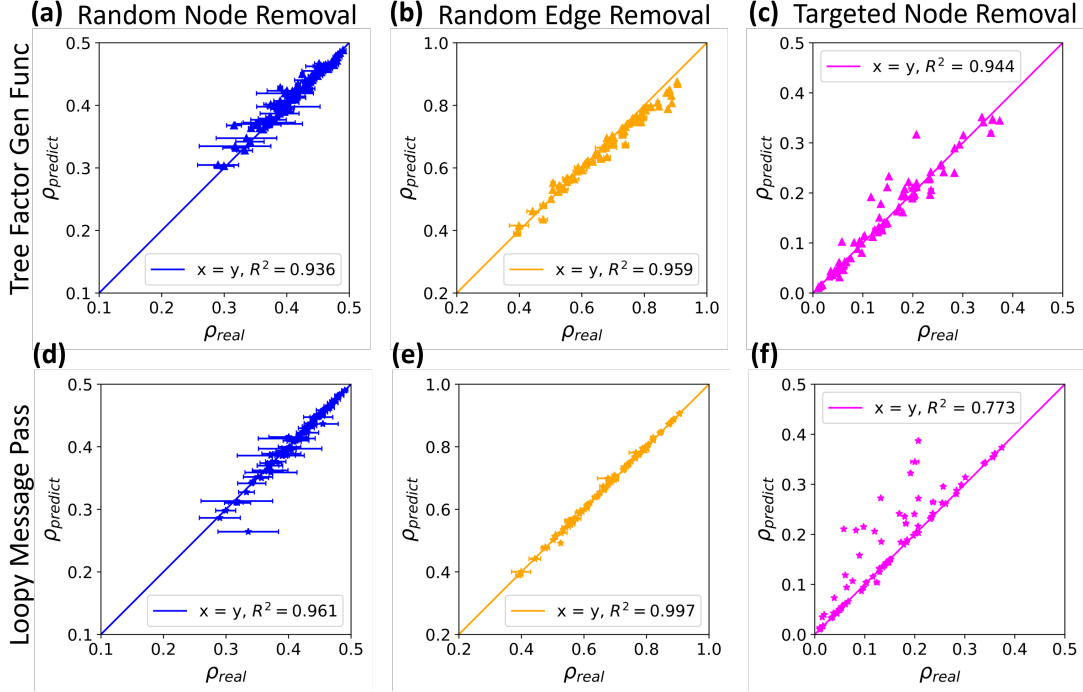
14

Figure 10: Comparison between predicted robustness using tree factor generating function model and real robustness for various real networks, excluding high mixing time networks. **(a)**, **(b)** and **(c)** are for the tree factor generating function model, and **(d)**, **(e)** and **(f)** are for the message passing model.

like message passing model. It remains to be seen whether the inclusion of the tree factor may improve other network models which assume locally tree like structure.

We go on to consider other possible structural properties which can cause percolation predictions to be inaccurate. Our results rule out the possibility that network size is a significant influence on prediction inaccuracy, which points against the assumption that generating function models fail to make accurate predictions due to the finite-ness of real networks. Similarly, we find that inaccuracies do not appear to stem from either missing local information or differences between the structure of modules for highly modular networks. Instead, we demonstrate that the discrepancies in accuracy are most pronounced on networks that exhibit community structure in the form of high maximum modularity and have dispersed modules. This is likely because such networks are not statistically similar throughout their entire structures in the manner assumed by the generating function model, with modules that are far away from the "core" of the network being more vulnerable to disconnection than is modelled by existing generating function models.

However, it is a well known problem in community detection that it is difficult to definitively identify community structure on networks using any one community detection method, and so at best we can only say that poorly predicted networks have strong indications of dispersed communities. Alternatively, we find that poorly predicted networks exhibit high mixing times for simple random walks, but interestingly this does not appear to be due to a lack of small world characteristics. Mixing time is a singular, easy to assess metric, making it a useful diagnostic tool for determining when one might expect percolation prediction methods to give accurate results. This also relates to other results [30–32] where high mixing time is indicative of the conditions under which models of network behaviour break down, and as such examining the general unpredictability of high mixing time networks may be worth future study.

When high mixing time networks are excluded from our dataset, then the predictions of robustness given by the tree factor generating function model are generally quite accurate, and for the loopy message passing model they are nearly perfectly accurate for random edge removal. While message passing methods produce more accurate predictions for random node and random edge removal than generating function models when excluding high mixing time networks, they require solving often several hundred times more self consistency equations in order to predict LCC size, making them much more computationally expensive. Depending on how much one may be able to tolerate a certain amount of prediction

15

inaccuracy and the computational resources available to them, it may be more suitable to use generating function models than message passing models in some contexts.

Clearly a useful future direction when seeking to improve percolation prediction methods is to incorporate information either about how dispersed highly modular networks are, or about structures which give rise to high mixing times. Additionally, such networks' robustness values tend to be substantially overestimated by percolation prediction methods, indicating that high mixing times and high modularity with module dispersal make a network considerably more vulnerable to node or edge removal than it might otherwise be without modular dispersal, indicating that reducing mixing time or making modules closer to one another could substantially improve network robustness.

Funding

References

[1] G. Koh, P. Porras, B. Aranda, H. Hermjakob, and S. Orchard, "Analyzing protein–protein interaction networks," *Journal of Proteome Research*, vol. 11, no. 4, pp. 2014–2031, 2012.

[2] A. Majeed and I. Rauf, "Graph theory: A comprehensive survey about graph theory applications in computer science and social networks," *Inventions*, vol. 5, no. 1, 2020.

[3] G. Caldarelli, S. Battiston, D. Garlaschelli, and M. Catanzaro, *Emergence of Complexity in Financial Networks*, pp. 399–423. Springer Berlin Heidelberg, 2004.

[4] F. Wenli, L. Zhigang, H. Z. Ping, and M. Shengwei, "Cascading failure model in power grids using the complex network theory," *Iet Generation Transmission & Distribution*, vol. 10, pp. 3940–3949, 2016.

[5] J. Montoya, S. Pimm, and R. Solé, "Ecological networks and their fragility," *Nature*, vol. 442, pp. 259–264, 2006.

[6] M. Li, R. Liu, L. Lü, M. Hu, S. Xu, and Y. Zhang, "Percolation on complex networks: Theory and application," *Physics Reports*, vol. 907, pp. 1–68, 2021.

[7] M. Newman, "Spread of epidemic disease on networks," *Phys. Rev. E*, vol. 66, p. 016128, Jul 2002.

[8] D. Li, B. Fu, Y. Wang, G. Lu, Y. Berezin, H. Stanley, and S. Havlin, "Percolation transition in dynamical traffic network with evolving critical bottlenecks," *Proceedings of the National Academy of Sciences*, vol. 112, pp. 669 – 672, 2014.

[9] R. Cohen, K. Erez, D. ben Avraham, and S. Havlin, "Breakdown of the internet under intentional attack," *Phys. Rev. Letters*, vol. 86, pp. 3682–3685, Apr 2001.

[10] M. Newman and R. Ziff, "A fast Monte Carlo algorithm for site or bond percolation," *Phys. Rev. E*, vol. 64, p. 016706, 2001.

[11] M. Newman, S. Strogatz, and D. Watts, "Random graphs with arbitrary degree distributions and their applications," *Phys. Rev. E*, vol. 64, p. 026118, Jul 2001.

[12] A. Vazquez and Y. Moreno, "Resilience to damage of graphs with degree correlations," *Phys. Rev. E*, vol. 67, p. 015101, 02 2003.

[13] Y. Berchenko, Y. Artzy-Randrup, M. Teicher, and L. Stone, "Emergence and size of the giant component in clustered random graphs with a given degree distribution.," *Phys. Rev. Letters*, vol. 102 13, p. 138701, 2009.

[14] B. Karrer, M. Newman, and L. Zdeborová, "Percolation on sparse networks," *Phys. Rev. Letters*, vol. 113, p. 208702, Nov 2014.

[15] F. Radicchi and C. Castellano, "Beyond the locally treelike approximation for percolation on real networks," *Phys. Rev. E*, vol. 93, 03 2016.

[16] G. Cantwell and M. Newman, "Message passing on networks with loops," *Proceedings of the National Academy of Sciences*, vol. 116, no. 47, pp. 23398–23403, 2019.

[17] M. Newman, "Chapter 13: Random graphs with general degree distributions," in *Networks: An Introduction*, pp. 428–485, Oxford University Press, 03 2010.

[18] M. Molloy and B. Reed, "A Critical Point For Random Graphs With A Given Degree Sequence," *Random Structures and Algorithms*, vol. 6, pp. 161–180, 1995.

[19] B. Rozemberczki and R. Sarkar, "Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, (New York, NY, USA), p. 1325–1334, Association for Computing Machinery, 2020.

[20] I. S. Duff, R. G. Grimes, and J. G. Lewis, "Users' guide for the harwell-boeing sparse matrix collection (release 1)," *Rutherford Appleton Laboratory Technical Reports*, 1992.

[21] C. Schneider, A. Moreira, J. Andrade, S. Havlin, and H. Herrmann, "Mitigation of malicious attacks on networks," *Proceedings of the National Academy of Sciences*, vol. 108, no. 10, pp. 3838–3841, 2011.

[22] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, p. 026113, 03 2004.

[23] A. Clauset, M. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, no. 6, p. 066111, 2004.

[24] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics*, vol. 2008, no. 10, p. P10008, 2008.

[25] V. Traag, L. Waltman, and N. Van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.

[26] B. Mohar, *Some applications of Laplace eigenvalues of graphs*, pp. 225–275. Dordrecht: Springer Netherlands, 1997.

[27] D. Calvetti, L. Reichel, and D. C. Sorensen, "An implicitly restarted lanczos method for large symmetric eigenvalue problems," *Electronic Transactions on Numerical Analysis*, vol. 2, no. 1, p. 21, 1994.

[28] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[29] B. F. Maier, "Generalization of the small-world effect on a model approaching the erdős–rényi random graph," *Scientific reports*, vol. 9, no. 1, pp. 1–9, 2019.

[30] B. F. Maier and D. Brockmann, "Cover time for random walks on arbitrary complex networks," *Physical Review E*, vol. 96, no. 4, p. 042307, 2017.

[31] B. F. Maier, C. Huepe, and D. Brockmann, "Modular hierarchical and power-law small-world networks bear structural optima for minimal first passage times and cover time," *Journal of Complex Networks*, vol. 7, no. 6, pp. 865–895, 2019.

[32] B. F. Maier, *Spreading Processes in Human Systems*. PhD thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät, 2020.

[33] E. Leicht and R. D'Souza, "Percolation on interacting networks," *arXiv preprint arXiv:0907.0894*, 2009.

[34] S. Shai, D. Kenett, Y. Kenett, M. Faust, S. Dobson, and S. Havlin, "Critical tipping point distinguishing two types of transitions in modular network structures," *Phys. Rev. E*, vol. 92, p. 062805, Dec 2015.

[35] G. Dong, J. Fan, L. Shekhtman, S. Shai, R. Du, L. Tian, X. Chen, H. Stanley, and S. Havlin, "Resilience of networks with community structure behaves as if under an external field," *Proceedings of the National Academy of Sciences*, vol. 115, no. 27, pp. 6911–6915, 2018.

[36] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, no. 4, pp. 452–473, 1977.

[37] L. Peel, D. Larremore, and A. Clauset, "The ground truth about metadata and community detection in networks," *Science Advances*, vol. 3, 08 2016.

Supplementary Materials: Improving mean-field network percolation models with neighbourhood information

Chris Jones [ID]^{1,2} and Karoline Wiesner [ID]^2

^{1}School of Mathematics, University of Bristol, Bristol, UK
^{2}Institute of Physics and Astronomy, University of Potsdam, Potsdam, Germany

1   Network Data

In Table 1, we provide data for 100 different real world networks used in this paper. The format for each entry in the table is the name of the network and a citation for its source. References for each network's original source is given where possible. All network data was accessed via either The KONECT Project [1] or via Network Repository [2].

Network Name	$n_{p \neq 0}$	$2E$	$N \mathrm{Log} N$	Reference
abiword	29	3438	5561.439	[3]
air-traffic	27	4816	8143.029	[1]
arabidopsis-interactions	75	21442	43208.15	[4]
amazon-links	25	7772	13948.81	[5]
asoiaf	51	5646	9741.357	[6]
bible-names	79	18118	35847.19	[1]
bio-CE-CX	40	4050	6695.511	[7]
bio-CE-GN	185	107360	253896.3	[7]
bio-CE-GT	46	6362	11141.66	[7]
bio-CE-HT	22	5376	9218.306	[7]
bio-CE-LC	24	2600	4048.126	[7]
bio-CE-PG	254	94618	221166.9	[7]
bio-DM-CX	222	153426	374733	[7]
bio-dmela-large	93	51138	112700.8	[8]
bio-DM-HT	27	9124	16693.07	[7]
bio-fly-brain	82	17810	35171.49	[9]
bio-HS-HT	99	27300	56444.56	[7]
bio-HS-LC	151	78954	181449.9	[7]
bio-human-protein-large	71	12836	24435.89	[10]
bio-mouse-brain	32	3072	4894.297	[9]
bio-SC-CC	162	69758	158440.1	[7]
bio-SC-GT	171	67964	153980.9	[7]
bio-SC-HT	269	126046	302478.8	[7]
bio-SC-LC	106	40896	88144.19	[7]
bio-yeast-large	27	3896	6408.119	[11]
ca-CondMat	122	182572	452815.5	[12]
ca-Erdos992	56	14856	28752.84	[13]
ca-GrQc	65	26844	55403.56	[12]
chess	152	111558	264753.4	[1]

cit-citeseer	31	7336	13074.33	[14]
cit-cora	37	10138	18780.25	[15]
cit-DBLP	122	99126	232706.1	[16]
cit-iui	28	2316	3547.775	[17]
cond-mat-newman	69	89238	207456.9	[18]
crime	20	2946	4666.763	[1]
deezer_europe	93	185504	460729.2	[19]
djavax	54	8816	16063.82	[20]
dnc-corecipient	124	20768	41705.93	[1]
econ-mahindas	71	15026	29118.99	[2]
econ-poli	39	5334	9137.203	[2]
econ-poli-large	90	34936	74103.5	[2]
email-EU	232	108794	257601.1	[12]
escorts	121	78032	179131.9	[21]
euroroad	9	2610	4065.872	[22]
fb-pages-company	106	104252	245881.2	[23]
fb-pages-government	219	178858	442805.8	[23]
fb-pages-politician	144	83412	192690	[23]
fb-pages-public-figure	184	134076	323546.9	[23]
fb-pages-sport	162	173622	428722.6	[23]
fb-pages-tvshow	85	34478	73033.23	[23]
Franz1	7	10222	18954.17	[2]
ia-email-univ	48	10902	20367.53	[24]
ia-fb-messages	65	12902	24575.9	[25]
ia-reality	74	15360	29839.57	[26]
lastfm_asia	98	55612	123573.7	[19]
mammalia-voles-bhp-trapping	35	9124	16693.07	[27]
mammalia-voles-kcs-trapping	32	7152	12706.95	[27]
mammalia-voles-plj-trapping	30	4900	8303.457	[27]
mammalia-voles-rob-trapping	29	7796	13997.1	[27]
openflights	124	31290	65621.01	[28]
p2p-Gnutella04	65	79988	184052.2	[29]
p2p-Gnutella05	68	63674	143359.9	[29]
p2p-Gnutella06	67	63050	141820.1	[29]
p2p-Gnutella08	76	41552	89701.67	[29]
p2p-Gnutella09	73	52016	114828.1	[29]
p2p-Gnutella24	52	130718	314723.5	[29]
p2p-Gnutella25	46	109386	259131.7	[29]
phonecalls.edgelist	42	105682	249566.5	[30]
polblogs	144	33428	70584.57	[31]
power-bcspwr10	13	16542	32402.12	[32]
power-US-Grid	16	13188	25183.47	[33]
proteins_stelzl	45	6212	10846.78	[34]
proteins_vidal	50	12014	22698.4	[35]
rt-alwefaq	66	14118	27168.28	[36]
rt-bahrain	67	15954	31124.97	[36]
rt-ksa	55	16214	31689.13	[36]
rt-pol	163	96106	224970.7	[37]
small	25	6414	11244.06	[2]
soc-advogato	171	78748	180931.8	[38]
soc-anybeat	185	98264	230496.1	[39]

soc-gplus.edges	130	78364	179966.3	[2]
soc-hamsterster	111	32194	67715.97	[1]
soc-sign-bitcoinalpha	113	28240	58595.67	[40]
soc-sign-bitcoinotc	131	42978	93094.99	[40]
soc-wiki-Vote	43	5828	10095.52	[41]
tech-as-caida2007	158	106762	252352.6	[42]
tech-internet-as	205	170246	419660.4	[43]
tech-pgp	83	48632	106647.4	[44]
tech-routers-rf	62	13264	25345.15	[45]
tech-WHOIS	242	113886	270789	[46]
uc-forum	78	14038	26997.01	[47]
uk	3	13674	26218.98	[48]
unicode-language	32	2490	3853.486	[1]
vtk	31	2714	4250.911	[3]
web-edu	47	12948	24673.53	[49]
web-EPA	73	17794	35136.42	[50]
web-indochina-2004	100	95212	222684.8	[51]
web-spam	174	74750	170900.2	[52]
web-webbase-2001	111	51186	112817	[51]
wikiquote-be	36	4788	8089.623	[1]

Table 1: Real world network data names, number of non-zero entries in degree distributions, two times number of edges, number of nodes times logarithm of number of nodes and references.

2   Network Size, Local Information and Prediction Inaccuracy

Here we consider the possibility that a significant source of inaccuracy in predictions made by generating function models comes from the fact that the models assume the network in question is infinite, whereas in reality we are considering finite networks. This critique is raised in the review of percolation prediction methods of [53]. However, we lack a well reasoned understanding of how much finite-ness has an adverse affect on prediction accuracy when using generating functions. Therefore, we calculate prediction inaccuracies for varying sizes of randomly configured networks.

In Figure 11, we plot prediction inaccuracy against nodes in a network for randomly configured networks with degree distributions generated by sampling either power-law or log-normal distributions. These distributions are used since they are representative of the degree distributions found on real networks [54, 55]. For all predictions, we use generating functions assuming random structure, i.e. without correlations, clustering or tree factor information. Separate networks are generated 20 times for each size, with averages of inaccuracies being taken. The power-law distribution we sample from is of the form

$$p(k) = \frac{k^{-\alpha}}{\sum_{n=0}^{\infty}(n + k_{min})^{-\alpha}}, \tag{37}$$

where α is an exponent term, k_{min} is the minimum degree and $\sum_{n=0}^{\infty}(n + k_{min})^{-\alpha}$ is a normalisation term. For all tests, $\alpha = 2.45$ and $k_{min} = 1$. The log-normal distribution we sample from is of the form

$$p(k) = \frac{1}{k\sigma\sqrt{2\pi}}\exp\Big(-\frac{(\ln(k) - \mu)^2}{2\sigma^2}\Big), \tag{38}$$

where μ and σ are the mean and standard deviation parameters respectively for the normal distribution that the log-normal distribution is based on. For all tests, $\mu = 0.55$ and $\sigma = 0.2$.

Let us consider the cutoff for a prediction being highly inaccurate to be $\Delta \geq 0.05$, where $\Delta = \int |S_{real} - S_{predict}| d\varphi$, which is 10% of maximum inaccuracy for random and targeted node removal and 5% of maximum
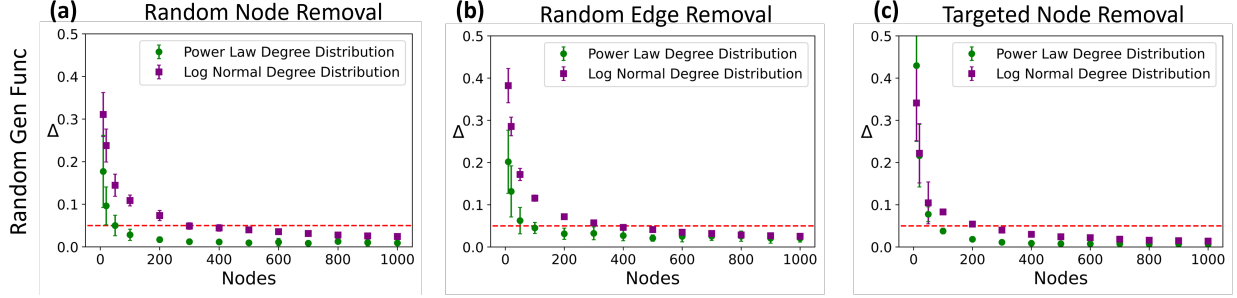
Figure 11: Prediction inaccuracy against network size for randomly configured networks. Points in green are for power law graphs, and points in purple are for log normal graphs. **(a)** plots prediction inaccuracy for random node removal, **(b)** is for random bond removal, and **(c)** is for targeted node removal. For all predictions, random structure is assumed.

inaccuracy for random bond removal. From Figure 11, we can see that for all distributions and removal methods prediction inaccuracy is under this threshold for networks above 400 nodes in size. For 1000 nodes, the largest inaccuracy is approximately 0.025. In our dataset of 100 networks, the smallest network is 453 nodes large, and 87 networks exceed 1000 nodes in size. Therefore, we would not expect that network size plays a significant role in producing inaccurate predictions for the networks we consider.

We confirm this by comparing network size against prediction inaccuracy for tree factor predictions of robustness against random node removal for our 100 network dataset. This is provided in Figure 12, alongside comparisons between inaccuracy and "global tree factor" and average degree. The global tree factor is given by

$$T_G = \frac{\langle n_2 \rangle}{\langle \sum k' \rangle} = \frac{\langle n_2 \rangle}{\langle k^2 \rangle - \langle k \rangle}, \tag{39}$$

where $\langle n_2 \rangle$ is the real average number of second neighbours, and $\langle k^2 \rangle - \langle k \rangle$ is the theoretical average nunber of second neighbours, derived in [56].
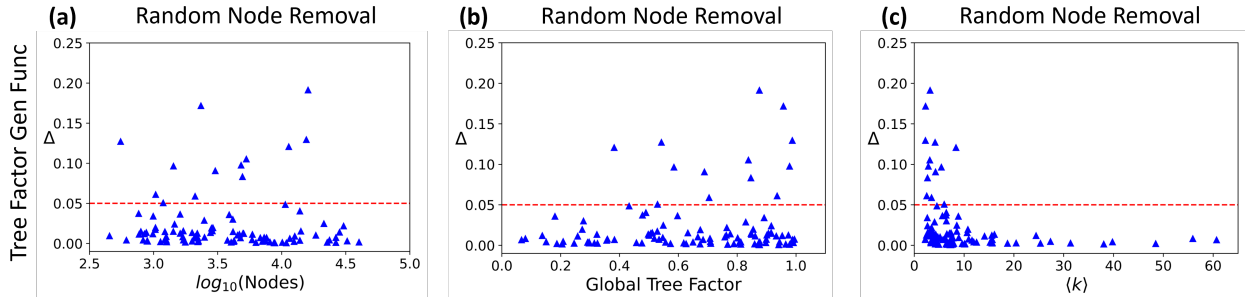

Figure 12: Prediction inaccuracy against various network properties. In **(a)**, prediction inaccuracy is compared to the logarithm of number of nodes, in **(b)** inaccuracy is compared to the global tree factor, and in **(c)** prediction inaccuracy is compared to expected degree. These results are all for tree factor predictions of random node removal.

In Figure 12**(a)**, prediction inaccuracy is compared to the logarithm of nodes. We can see that network size has little bearing on prediction inaccuracy, since we only consider networks larger than 400 nodes in size, which the results in Figure 11 also suggest would not produce especially inaccurate predictions due to their size.

We compare the global tree factor against prediction inaccuracy in Figure 12**(b)**, but again find little indication that this property impacts prediction inaccuracy. The global tree factor acts as an indicator of the average local structure one is likely to find in a network, similar to global measures of degree-degree correlation and clustering. With the lack of a clear relationship between prediction inaccuracy and global

4

tree factor, we may infer that we have captured local network structure as best as possible with the tree factor method.

Finally, Figure 12(c) shows that for more dense networks, percolation predictions are more accurate, whereas for sparse networks predictions range from being very inaccurate to very accurate. This is likely explained by the fact that network configuration plays a smaller role in determining robustness for dense networks as opposed to sparse networks. While this result allows us to identify networks with similar properties which may be inaccurately predicted by the generating function model, this does not identify information that is currently missing from existing models, nor does it allow us to isolate only those networks for which predictions are inaccurate.

3 Generating Functions for Modular Networks

When predicting percolation on modular networks, it is possible to use a generating function formalism originally developed in order to describe percolation on interacting networks [57] which has subsequently been extended to modular networks [58, 59].

The generating function analogous to the probability distribution generating function of Equation (1) for some module i is

$$g_i = (1 - r_i)g_{ii}(u_{ii}) + r_i g_{ii}(u_{ii}) \prod_{j \neq i}^{m} g_{ij}(u_{ij}), \tag{40}$$

where r_i is the fraction of nodes in i with connections to other modules, m is the number of modules, u_{ii} and u_{ij} are the probabilities that an edge within the module and an edge leading out of the module respectively is not connected to the LCC, and $g_{ii}(u_{ii})$ and $g_{ij}(u_{ij})$ are the generating functions for connections within the module and connections leading out of the module respectively.

In order to determine values for u_{ii} and u_{ij}, it is necessary to define branching generating functions analogous to Equation (2) and define new self consistency relations. These branching functions are given by

$$g_i^{(ii)} = (1 - r_i) \sum q_i u_{ii}^{k_i - 1} + r_i \sum q_i u_{ii}^{k_i - 1} \prod_{j \neq i}^{m} g_{ij}(u_{ij}), \tag{41}$$

$$g_i^{(ij)} = g_{ii}(u_{ii}) \sum q_{ij} u_{ij}^{k_{ij} - 1} \prod_{j \neq i}^{m-1} g_{ij}(u_{ij}), \tag{42}$$

where q_i and q_{ij} are remaining degree distributions for module i and for interconnections between i and j respectively. Using these generating functions, self consistency relations for u_{ii} and u_{ij} may be defined as

$$u_{ii} = 1 - g_i^{(ii)}[1 - \varphi u_{ii}, 1 - \varphi u_{ij}], \tag{43}$$

$$u_{ij} = 1 - g_i^{(ij)}[1 - \varphi u_{ii}, 1 - \varphi u_{ij}]. \tag{44}$$

With these self consistency relations, it is then possible to calculate the the proportion of nodes in module i which belong to the LCC as the network undergoes random node removal as

$$S_i^{node} = \varphi(1 - g_i[1 - \varphi u_{ii}, 1 - \varphi u_{ij}]). \tag{45}$$

For random edge removal, the expression for the proportion of nodes in module i in the LCC is very similar, simply removing the initial φ term to give

$$S_i^{edge} = (1 - g_i[1 - \varphi u_{ii}, 1 - \varphi u_{ij}]). \tag{46}$$

Finally, for either node or edge removal, the size of the LCC is given by summing over values for the relevant S_i values for different modules, weighted by module size

$$S = \sum_i^m \frac{N_i}{N} S_i, \tag{47}$$

where N_i is the number of nodes in module i and N is the total number of nodes in the network.

Due to the fact that this method separates the degree values of nodes into their intra and inter module degrees, it is not possible to straightforwardly adapt this method in order to make predictions of robustness against targeted attacks.

With this model we are able to make comparisons between real and predicted robustness, similar to those made in Figures 1 and 3, and these are given in Figure 13.
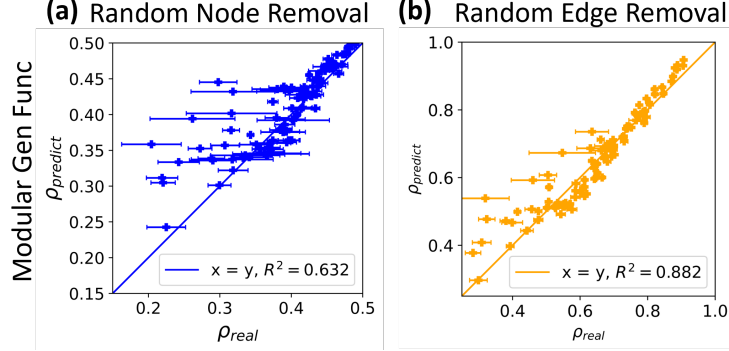
Figure 13: Comparison between predicted robustness using modular generating function model and real robustness for various real networks. **(a)** considers robustness for random node removal, and **(b)** is for random edge removal. This model cannot be directly adapted in order to generate predictions for targeted attacks.

Despite the inclusion of information about modular network structure in this model, we can see that it still produces many inaccurate predictions, and is generally less accurate than the tree factor model as measured by the coefficient of determination. One possible reason for continued discrepancies in prediction accuracy for the modular generating function model is that it is not the modularity of a network alone which impacts how accurate robustness predictions are, but that there is some aspect of the structure within modules which also influences robustness and prediction accuracy.

To test this, we measure the heterogeneity of module structures for highly modular ($Q > 0.75$) networks. We measure the differences between the structure of modules in three different ways. Firstly, we examine how well the degree distributions of each module in a network matches up to the global degree distribution of the entire network. This is calculated using the K-L divergence [60] and may be written as

$$D(p_{net}|p_{mod}) = \sum_k p_{net}(k) \log\left(\frac{p_{net}(k)}{p_{mod}(k)}\right), \tag{48}$$

where $p_{net}(k)$ is the network degree distribution and $p_{mod}(k)$ is the module degree distribution. Averaging over this divergence for every module then gives a measure of how significantly modules differ from the overall network.

Secondly, we can look at the size of different modules, and measure whether modules tend to be similar in size or span a range of sizes. We may calculate the normalised module size as the number of nodes in the module divided by the number of nodes in the network. Taking the standard deviation of this normalised module size across the whole network then provides a measure of module size heterogeneity.

Finally, we may measure the module average tree factor for each module. This is calculated analogously to the global tree factor from Equation (39), where the average real number of second neighbours in the module is divided by the average theoretical number of second neighbours. As before, taking the standard deviation of this module average tree factor gives a measure of the structural heterogeneity between different modules. Each of these three measures is plotted against prediction inaccuracy using the tree factor method for predicting random node removal for networks with $Q > 0.75$ in Figure 14.
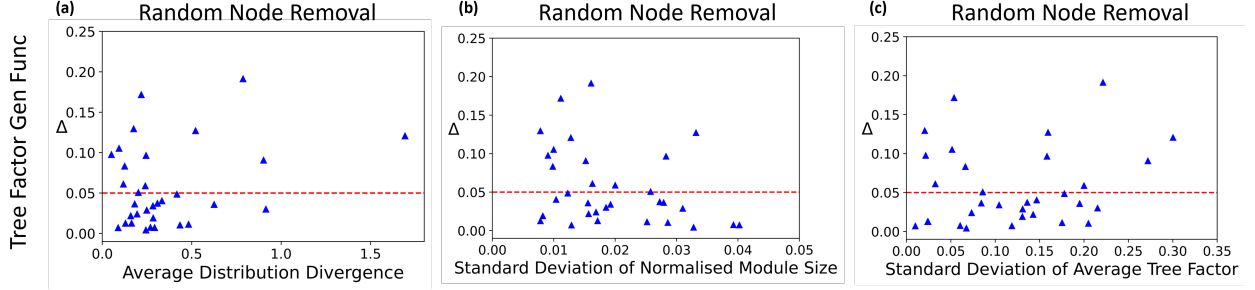
Figure 14: Measures of module heterogeneity of highly modular networks against prediction inaccuracy. In **(a)**, prediction inaccuracy is compared to the average K-L divergence between module degree distribution and network degree distribution. In **(b)**, inaccuracy is compared to the standard deviation of normalised module size, and in **(c)** prediction inaccuracy is compared the standard deviation of the module average tree factor. These results are all for tree factor predictions of random node removal.

From the results in Figure 14, we can see that none of our measures of module heterogeneity correlate with prediction inaccuracy, indicating that the source of inaccuracy is not due to any sort of difference between the internal structures of different modules.

4 Prediction Inaccuracy of Loopy Message Passing Model

In this section, we examine how prediction inaccuracy for the loopy message passing model compares to modularity and modular dispersion. In Figure 15, we provide comparisons for prediction inaccuracy Δ and modularity Q.
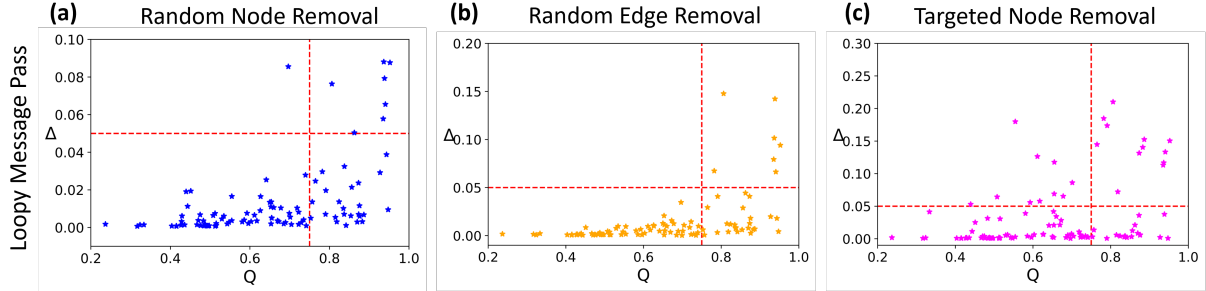

Figure 15: Prediction inaccuracy against modularity. **(a)** is for robustness against random node removal, **(b)** is for random edge removal, and **(c)** is for targeted node removal. Dashed lines are included to mark the points at which $Q = 0.75$ and $\Delta = 0.05$.

We can see that, similarly to the generating function model, the loopy message passing model tends to be more inaccurate for higher modularity networks for random node and random edge removal, although there is not a clear relationship between inaccuracy and modularity for targeted node removal. In Figure 16, we compare prediction inaccuracy Δ and modular dispersal D.

Once again, we see that more dispersed networks tend to be more poorly predicted for random node and random edge removal, with the relationship being less clear for targeted node removal. For the loopy message passing model, prediction inaccuracy tends to be lower for random node and random edge removal as compared to generating function models, and so more networks which are classed as "well predicted" (i.e. those for which $\Delta \leq 0.05$) are classed as being highly modular and highly dispersed.
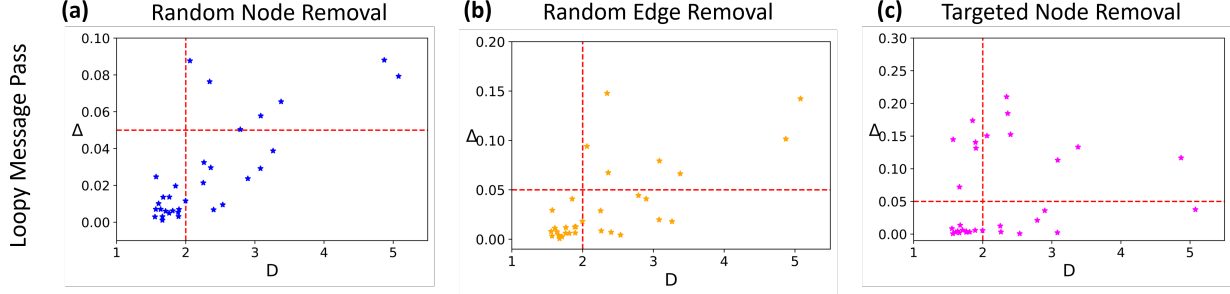
Figure 16: Prediction inaccuracy for highly modular networks against modular dispersal. **(a)** is for robustness against random node removal, **(b)** is for random edge removal, and **(c)** is for targeted node removal. Dashed lines are included to mark the points at which $D = 2$ and $\Delta = 0.05$.

5 Small World-ness

In the main text, we establish that highly modular, highly dispersed networks are poorly predicted by generating function models, and identify the need for a single metric which indicates when predictions are likely to be unreliable. Clearly highly modular, highly dispersed networks exhibit certain structural non-randomness, and one may also expect that because their modules are dispersed, they are typically further away from being "small worlds" than other networks. Therefore, assessing the "small world-ness" of a network may provide an alternative approach to identifying networks poorly predicted by generating function models.

First, we consider a model of a small world network in order to systematically evaluate how small world-ness relates to prediction inaccuracy. The model we use is the Watts-Strogatz small world network [33], which generates a network in a ring lattice and rewires edges with some rewiring probability. The initial lattice configuration ensures a high clustering coefficient, and the rewiring brings the average shortest path length of the network close to that of an equivalent randomly configured network. These properties - relatively high clustering and relatively low average shortest path length - are taken to be indicative of small world-ness. A related metric ω measures a network's small world-ness along these lines [61], with ω given by

$$\omega = \frac{L_{rand}}{L} - \frac{C}{C_{latt}}, \tag{49}$$

where L and L_{rand} are the average shortest path lengths of the network and its equivalent randomly configured network respectively, and C and C_{latt} are the transitivity values of the real network and its equivalent ring lattice network respectively. Values from ω typically range from -1, which indicates lattice-likeness, to 1, which indicates randomness in configuration. Values around 0 are taken to be indicative of small world properties.

This metric of small world-ness is not infallible, for example a network with a high average shortest path length and low clustering coefficient would be erroneously identified as small world, but it is possible to also consider the $\frac{L_{rand}}{L}$ and $\frac{C}{C_{latt}}$ components separately as measures of randomness and lattice-likeness. As reference values for L_{rand} and C_{latt}, we use approximations sourced from [62], which are

$$L_{rand} = \frac{\ln N}{\ln \langle k \rangle}, \tag{50}$$

$$C_{latt} = \frac{3(\langle k \rangle - 2)}{4(\langle k \rangle - 1)}. \tag{51}$$

For the Watts-Strogatz model, we consider networks with 2000 nodes, average degree of 6 and rewiring probabilities from 0 to 1 in increments of 0.05. For these networks, prediction inaccuracy for random node removal is compared against the small world-ness measure ω, the randomness measure $\frac{L_{rand}}{L}$ and the lattice likeness measure $\frac{C}{C_{latt}}$. These results are given in Figure 17.

These results indicate that, on Watts-Strogatz networks, generating function model predictions are poor when networks are lattice like, and improve when networks exhibit small world and random properties.
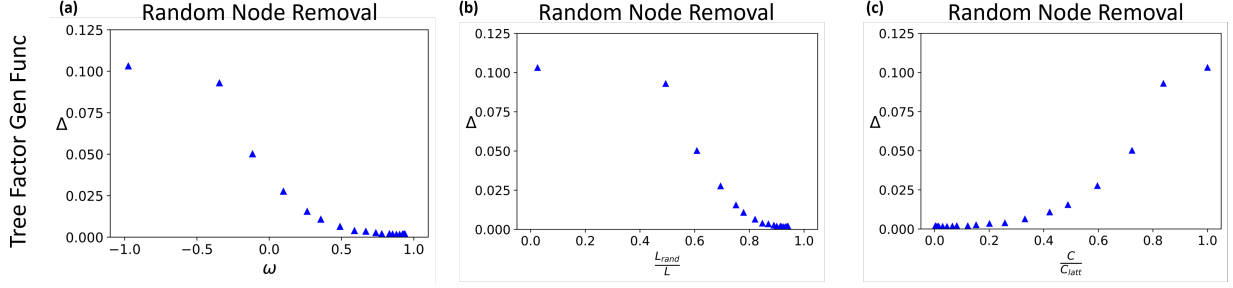
8

Figure 17: Prediction inaccuracy for random node removal on Watts-Strogatz networks against measures of (a) small world-ness ω, (b) randomness $\frac{L_{rand}}{L}$ and (c) lattice likeness $\frac{C}{C_{latt}}$. As rewiring probability goes from 0 to 1, small world-ness and randomness increase, whereas lattice likeness and prediction inaccuracy decrease.

However, when we apply measures of small world-ness, randomness and lattice-likeness to our dataset of real networks, we see that none of them clearly discriminate between poorly predicted and well predicted networks, and this is shown for random node removal in Figure 18.
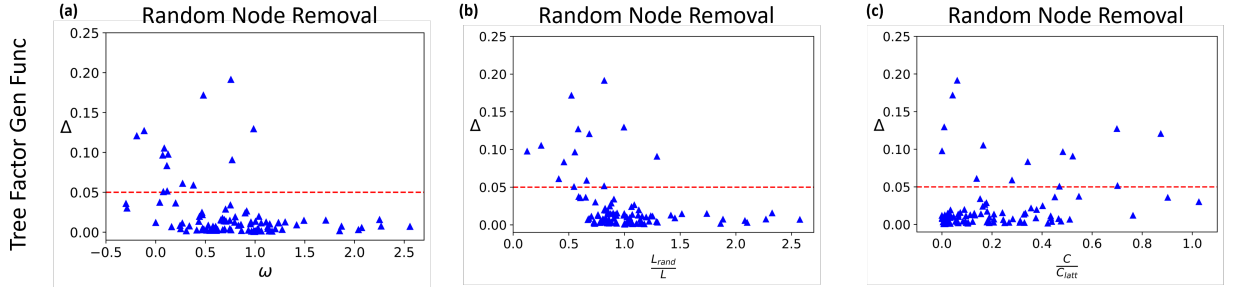

Figure 18: Prediction inaccuracy for random node removal on real world networks against measures of (a) small world-ness ω, (b) randomness $\frac{L_{rand}}{L}$ and (c) lattice likeness $\frac{C}{C_{latt}}$. Dashed lines are included to mark the point at which prediction inaccuracy = 0.05.

One insight we gain from these results is that it is possible for poorly predicted networks to have average shortest path lengths that are close to those of equivalent random networks, and yet our prior results suggest that they tend to possess modular structures with relatively large distances between modules. From this we may conclude that on a poorly predicted network, any two nodes may not necessarily be far away from each other, but they may be only tenuously connected if they are in different communities. This leads us to consider the mixing time of a random walk as a possible measure for identifying networks which are poorly predicted.
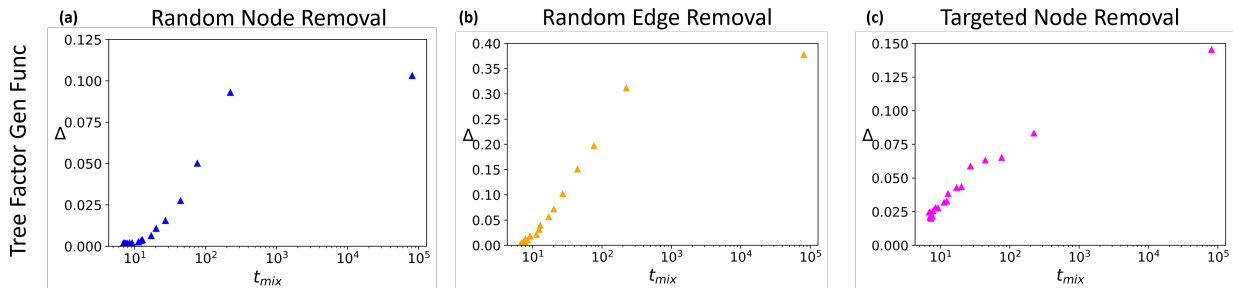

Figure 19: Prediction inaccuracy for (a) random node removal, (b) random edge removal and (c) targeted node removal on Watts-Strogatz networks against mixing time t_{mix}. As rewiring probability goes from 0 to 1, mixing time and prediction inaccuracy decrease.

From Figure 19, we can see that as mixing time decreases, prediction inaccuracy also decreases. In the

9

main text, we also find that high mixing time networks are poorly predicted. While these results on the Watts-Strogatz model gives us a sense of how predictions become more accurate as mixing time decreases, it is worth noting that real networks can exhibit much higher mixing times and remain well predicted compared to Watts-Strogatz networks.

6 Robustness Predictions Without High Mixing Time Networks

In this section, we provide robustness data for generating functions which assume random, correlated, clustered and modular structure alongside the tree like and loopy message passing methods when high mixing time networks are excluded from the dataset. This data is provided in Figure 20.
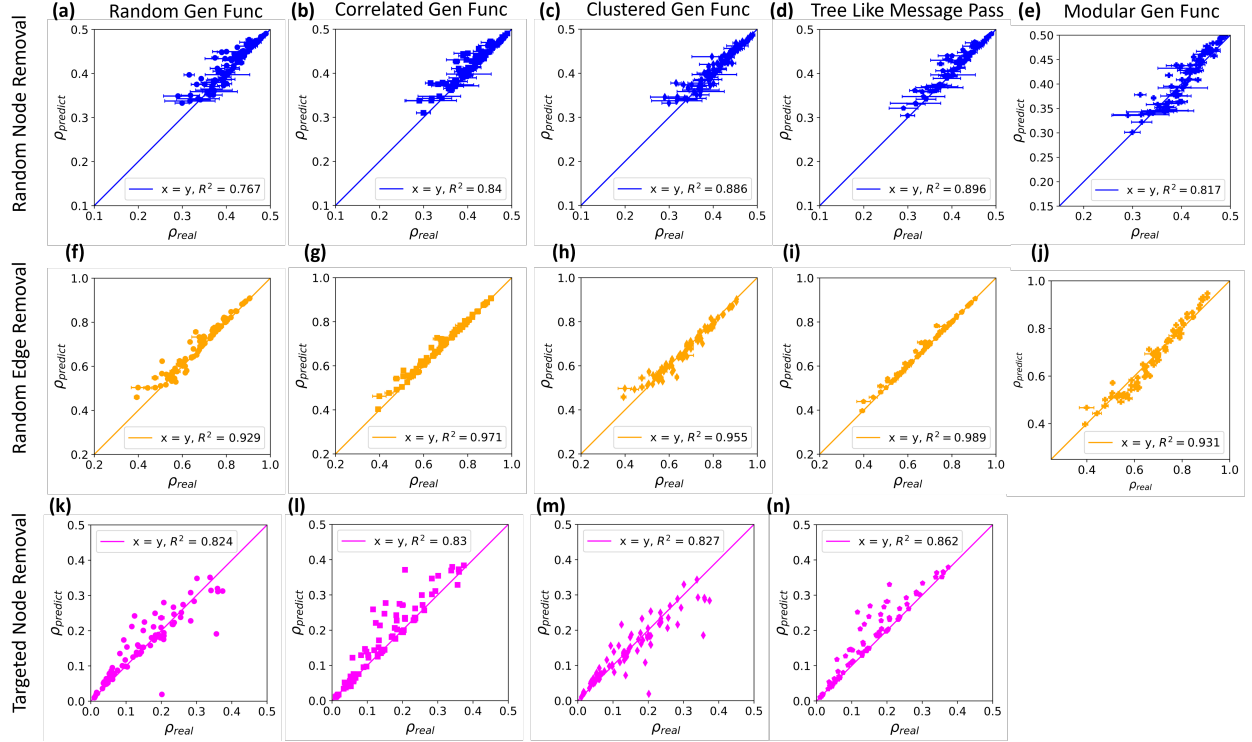

Figure 20: Comparisons of real and predicted robustness values. Plots in blue are for random node removal, plots in orange are for random edge removal and plots in purple targeted node removal. Predictions in **(a)**, **(f)** and **(k)** use the random structure generating function model, **(b)**, **(g)** and **(l)** use the correlation structure generating function model, **(c)**, **(h)** and **(m)** use the clustering structure generating function model, **(d)**, **(i)** and **(n)** use the tree like message passing model, and **(e)** and **(j)** use the modular structure generating function model. Note that there are no predictions for robustness against targeted attacks using the modular generating function model, as the model cannot be adapted to make such predictions.

For all methods and removal strategies, the overall quality of predictions as measured by the coefficient of determination is significantly improved. With the exclusion of high mixing time networks, all models shown here under perform the tree factor generating function model in terms of R^2 for all methods of removal with the exception of the correlations generating function model and tree like message passing model for random edge removal. All models shown here under perform the loopy message passing model for random node removal and random edge removal, but outperform the loopy message passing model for targeted node removal.

References

[1] J. Kunegis, "Konect: The koblenz network collection," in *Proceedings of the 22nd International Conference on World Wide Web*, p. 1343–1350, Association for Computing Machinery, 2013. http://konect.cc/, accessed 16/05/22.

[2] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015. https://networkrepository.com, accessed 16/05/22.

[3] C. R. Myers, "Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs," *Phys. Rev. E*, vol. 68, p. 046116, Oct 2003.

[4] M. Dreze, A.-R. Carvunis, B. Charloteaux, M. Galli, S. J. Pevzner, M. Tasan, Y.-Y. Ahn, P. Balumuri, A.-L. Barabási, V. Bautista, P. Braun, D. Byrdsong, H. Chen, J. D. Chesnut, M. E. Cusick, J. L. Dangl, C. de los Reyes, A. Dricot, M. Duarte, J. R. Ecker, C. Fan, L. Gai, F. Gebreab, G. Ghoshal, P. Gilles, B. J. Gutierrez, T. Hao, D. E. Hill, C. J. Kim, R. C. Kim, C. Lurin, A. MacWilliams, U. Matrubutham, T. Milenkovic, J. Mirchandani, D. Monachello, J. Moore, M. S. Mukhtar, E. Olivares, S. Patnaik, M. M. Poulin, N. Przulj, R. Quan, S. Rabello, G. Ramaswamy, P. Reichert, E. A. Rietman, T. Rolland, V. Romero, F. P. Roth, B. Santhanam, R. J. Schmitz, P. Shinn, W. Spooner, J. Stein, G. M. Swamilingiah, S. Tam, J. Vandenhaute, M. Vidal, S. Waaijers, D. Ware, E. M. Weiner, S. Wu, and J. Yazaki, "Evidence for network evolution in an arabidopsis interactome map," *Science*, vol. 333, no. 6042, pp. 601–607, 2011.

[5] L. Šubelj and M. Bajec, "Ubiquitousness of link-density and link-pattern communities in real-world networks," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 85, 04 2011.

[6] "Character interaction networks for george r. r. martin's "a song of ice and fire" saga." `https://github.com/mathbeveridge/asoiaf`. Accessed: 20/07/22.

[7] A. Cho, J. Shin, S. Hwang, C. Kim, H. Shim, H. Kim, H. Kim, and I. Lee, "WormNet v3: a network-assisted hypothesis-generating server for Caenorhabditis elegans," *Nucleic Acids Research*, vol. 42, pp. W76–W82, 05 2014.

[8] R. Singh, J. Xu, and B. Berger, "Global alignment of multiple protein interaction networks with application to functional orthology detection," *Proceedings of the National Academy of Sciences*, vol. 105, no. 35, pp. 12763–12768, 2008.

[9] K. Amunts, C. Lepage, L. Borgeat, H. Mohlberg, T. Dickscheid, M.-E. Rousseau, S. Bludau, P.-L. Bazin, L. Lewis, A.-M. Oros-Peusquens, N. Shah, T. Lippert, K. Zilles, and A. Evans, "Bigbrain: An ultrahigh-resolution 3d human brain model," *Science (New York, N.Y.)*, vol. 340, pp. 1472–1475, 06 2013.

[10] R. M. Ewing, P. Chu, F. Elisma, H. Li, P. J. Taylor, S. Climie, L. D. B. McBroom-Cerajewski, M. D. Robinson, L. O'Connor, M. Li, R. Taylor, M. Dharsee, Y. Ho, A. M. Heilbut, L. Moore, S. Zhang, O. I. Ornatsky, Y. V. Bukhman, M. Ethier, Y. Sheng, J. Vasilescu, M. Abu-Farha, J.-P. Lambert, H. Duewel, I. I. Stewart, B. L. Kuehl, K. A. Hogue, K. Colwill, K. Gladwish, B. Muskat, R. Kinach, S. L. Adams, M. F. Moran, G. B. Morin, T. Topaloglou, and D. Figeys, "Large-scale mapping of human protein–protein interactions by mass spectrometry," *Molecular Systems Biology*, vol. 3, pp. 89 – 89, 2007.

[11] H. Jeong, S. Mason, A.-L. Barabasi, and Z. Oltvai, "Lethality and centrality in protein networks," *Nature*, vol. 411, pp. 41–2, 06 2001.

[12] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discov. Data*, vol. 1, mar 2007.

[13] V. Batagelj and A. Mrvar, "Some analyses of erdos collaboration graph," *Social Networks*, vol. 22, no. 2, pp. 173–186, 2000.

[14] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in *Proceedings of the Third ACM Conference on Digital Libraries*, (New York, NY, USA), p. 89–98, Association for Computing Machinery, 1998.

[15] L. Šubelj and M. Bajec, "Model of complex networks based on citation dynamics," *WWW 2013 Companion - Proceedings of the 22nd International Conference on World Wide Web*, 03 2013.

[16] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," in *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, (New York, NY, USA), Association for Computing Machinery, 2012.

[17] N. Blagus and M. Bajec, "The network of collaboration: Informatica and uporabna informatika," *Uporabna Informatika*, vol. 23, no. 1, pp. 22–31, 2015.

[18] M. Newman, "The structure of scientific collaboration networks," *Proceedings of the national academy of sciences*, vol. 98, no. 2, pp. 404–409, 2001.

[19] B. Rozemberczki and R. Sarkar, "Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, (New York, NY, USA), p. 1325–1334, Association for Computing Machinery, 2020.

[20] L. Subelj and M. Bajec, "Unfolding communities in large complex networks: Combining defensive and offensive label propagation for core extraction," *Phys. Rev. E*, vol. 83, p. 036103, Mar 2011.

[21] L. E. C. Rocha, F. Liljeros, and P. Holme, "Simulated epidemics in an empirical spatiotemporal network of 50,185 sexual contacts," *PLOS Computational Biology*, vol. 7, 03 2011.

[22] L. Šubelj and M. Bajec, "Robust network community detection using balanced propagation," *The European Physical Journal B*, vol. 81, 06 2011.

[23] B. Rozemberczki, R. Davies, R. Sarkar, and C. Sutton, "Gemsec: Graph embedding with self clustering," in *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, (New York, NY, USA), p. 65–72, Association for Computing Machinery, 2019.

[24] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas, "Self-similar community structure in a network of human interactions," *Phys. Rev. E*, vol. 68, p. 065103, Dec 2003.

[25] T. Opsahl and P. Panzarasa, "Clustering in weighted networks," *Social Networks*, vol. 31, no. 2, pp. 155–163, 2009.

[26] N. Eagle and A. (Sandy) Pentland, "Reality mining: Sensing complex social systems," *Personal Ubiquitous Comput.*, vol. 10, p. 255–268, mar 2006.

[27] S. Davis, B. Abbasi, S. Shah, S. Telfer, and M. Begon, "Spatial analyses of wildlife contact networks," *Journal of the Royal Society, Interface / the Royal Society*, vol. 12, 01 2015.

[28] V. Colizza, R. Pastor-Satorras, and A. Vespignani, "Reaction–diffusion processes and metapopulation models in heterogeneous networks," *Nature Physics*, vol. 3, pp. 276–282, 2007.

[29] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design," *IEEE Internet Computing Journal*, vol. 6, 10 2002.

[30] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[31] A. Veenstra, "The two blogospheres: Political blog use, participation, and sophistication during the 2008 u.s. election season," *Open Journal of Political Science*, vol. 04, pp. 278–290, 01 2014.

[32] I. S. Duff, R. G. Grimes, and J. G. Lewis, "Users' guide for the harwell-boeing sparse matrix collection (release 1)," *Rutherford Appleton Laboratory Technical Reports*, 1992.

[33] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.

[34] U. Stelzl, U. Worm, M. Lalowski, C. Haenig, F. H. Brembeck, H. Goehler, M. Stroedicke, M. Zenkner, A. Schoenherr, S. Koeppen, J. Timm, S. Mintzlaff, C. Abraham, N. Bock, S. Kietzmann, A. Goedde, E. Toksöz, A. Droege, S. Krobitsch, B. Korn, W. Birchmeier, H. Lehrach, and E. E. Wanker, "A human protein-protein interaction network: A resource for annotating the proteome," *Cell*, vol. 122, no. 6, pp. 957–968, 2005.

[35] J.-F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, G. F. Berriz, F. D. Gibbons, M. Dreze, N. Ayivi-Guedehoussou, N. Klitgord, C. Simon, M. Boxem, S. Milstein, J. Rosenberg, D. Goldberg, L. V. Zhang, S. L. Wong, G. Franklin, S. Li, J. S. Albala, J. Lim, C. Fraughton, E. Llamosas, S. Cevik, C. C. Bex, P. Lamesch, R. S. Sikorski, J. Vandenhaute, H. Y. Zoghbi, A. Smolyar, S. A. Bosak, R. Sequerra, L. A. Doucette-Stamm, M. E. Cusick, D. E. Hill, F. P. Roth, and M. Vidal, "Towards a proteome-scale map of the human protein–protein interaction network," *Nature*, vol. 437, pp. 1173–1178, 2005.

[36] R. A. Rossi, D. F. Gleich, A. H. Gebremedhin, and M. M. A. Patwary, "Fast maximum clique algorithms for large graphs," in *Proceedings of the 23rd International Conference on World Wide Web*, (New York, NY, USA), p. 365–366, Association for Computing Machinery, 2014.

[37] M. D. Conover, J. Ratkiewicz, M. R. Francisco, B. Gonçalves, F. Menczer, and A. Flammini, "Political polarization on twitter," in *ICWSM*, 2011.

[38] P. Massa, M. Salvetti, and D. Tomasoni, "Bowling alone and trust decline in social network sites," in *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, pp. 658–663, 2009.

[39] M. Fire, R. Puzis, and Y. Elovici, *Link Prediction in Highly Fractional Data Sets*, pp. 283–300. New York, NY: Springer New York, 2013.

[40] S. Kumar, F. Spezzano, V. S. Subrahmanian, and C. Faloutsos, "Edge weight prediction in weighted signed networks," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 221–230, 2016.

[41] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, (New York, NY, USA), p. 1361–1370, Association for Computing Machinery, 2010.

[42] R. A. Rossi, S. Fahmy, and N. Talukder, "A multi-level approach for evaluating internet topology generators," *2013 IFIP Networking Conference*, pp. 1–9, 2013.

[43] R. Rossi, S. Fahmy, and N. Talukder, "A multi-level approach for evaluating internet topology generators," in *2013 IFIP Networking Conference*, pp. 1–9, 2013.

[44] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas, "Models of social networks based on social distance attachment," *Phys. Rev. E*, vol. 70, p. 056122, Nov 2004.

[45] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 2–16, 2004.

[46] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, k. c. claffy, and A. Vahdat, "The internet as-level topology: Three data sources and one definitive metric," *SIGCOMM Comput. Commun. Rev.*, vol. 36, p. 17–26, jan 2006.

[47] T. Opsahl, "Triadic closure in two-mode networks: Redefining the global and local clustering coefficients," *Social Networks*, vol. 35, no. 2, pp. 159–167, 2013.

[48] D. A. Bader, H. Meyerhenke, P. Sanders, C. Schulz, A. Kappes, and D. Wagner, "Benchmarking for graph clustering and partitioning," in *Encyclopedia of Social Network Analysis and Mining*, 2014.

[49] D. F. Gleich, L. Zhukov, and P. Berkhin, "Fast parallel pagerank: A linear system approach," *Yahoo! Technical Report*, 2004.

[50] W. De Nooy, A. Mrvar, and V. Batagelj, *Exploratory social network analysis with Pajek*, vol. 27. Cambridge University Press, 2011.

[51] P. Boldi, B. Codenotti, M. Santini, and S. Vigna, "Ubicrawler: a scalable fully distributed web crawler," *Software: Practice and Experience*, vol. 34, no. 8, pp. 711–726, 2004.

[52] C. Castillo, K. Chellapilla, and L. Denoyer, "Web spam challenge 2008," in *AIRWeb 2008*, 2008.

[53] M. Li, R. Liu, L. Lü, M. Hu, S. Xu, and Y. Zhang, "Percolation on complex networks: Theory and application," *Physics Reports*, vol. 907, pp. 1–68, 2021.

[54] A. Barabási and E. Bonabeau, "Scale-free networks," *Scientific American*, vol. 288, no. 5, pp. 60–69, 2003.

[55] A. Broido and A. Clauset, "Scale-free networks are rare," *Nature Communications*, vol. 10, p. 1017, 2019.

[56] M. Newman, "Chapter 13: Random graphs with general degree distributions," in *Networks: An Introduction*, pp. 428–485, Oxford University Press, 03 2010.

[57] E. Leicht and R. D'Souza, "Percolation on interacting networks," *arXiv preprint arXiv:0907.0894*, 2009.

[58] S. Shai, D. Kenett, Y. Kenett, M. Faust, S. Dobson, and S. Havlin, "Critical tipping point distinguishing two types of transitions in modular network structures," *Phys. Rev. E*, vol. 92, p. 062805, Dec 2015.

[59] G. Dong, J. Fan, L. Shekhtman, S. Shai, R. Du, L. Tian, X. Chen, H. Stanley, and S. Havlin, "Resilience of networks with community structure behaves as if under an external field," *Proceedings of the National Academy of Sciences*, vol. 115, no. 27, pp. 6911–6915, 2018.

[60] J. M. Joyce, *Kullback-Leibler Divergence*, pp. 720–722. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[61] Q. Telesford, K. Joyce, S. Hayasaka, J. Burdette, and P. Laurienti, "The ubiquity of small-world networks," *Brain connectivity*, vol. 1, pp. 367–75, 12 2011.

[62] Z. P. NEAL, "How small is it? comparing indices of small worldliness," *Network Science*, vol. 5, no. 1, p. 30–44, 2017.