Authors' Response to Reviews of

Structify-Net: Random Graph generation with controlled size and customized structure

Cazabet, Citraro and Rossetti
PCI Network Science,

RC: ***Reviewers' Comment***, AR: Authors' Response, ☐ Manuscript Text

1. Introduction to the Answer letter

AR: We thank the recommender and both reviewers for their time, attentive reading and useful recommendations for improvements. In this letter, we answer point by point to all comments made, and point the way we improved the article to take them into account. We also join a document highlighting all the differences made between the original submission and the new revision.

2. Recommender comments

RC: ***I think this preprint shows potential and would like to recommend a revised version of it. The current version falls a bit short. The reviewers make some very nice suggestions to this end. In particular, I'd like to emphasize the point about connecting better with previous literature. It seems there are a number of models that could be considered as part of the current version (e.g., see Peter Hoff's work on latent space models), so not pointing this out seems like a missed opportunity.***

AR: We understand the comments made by the recommender, and modified the article in that direction, as detailed in answers to reviewers. In particular, we added a whole new section concerned with related work. We want to stress, nevertheless, that the literature on related topics is enormous, which is precisely a consequence of the genericity of our approach, and we think that writing an exhaustive overview of statistical network models is beyond the scope of this article (and has not been attempted yet...).

RC: ***Providing a summary of features/benefits over other packages that involve network generation would also help promote the proposed software.***

AR: We added a section listing other packages we could found for random network generation. We are willing to add any other relevant package pointed to us.

3. Reviewer 1

RC: ***For the most part, the method and experiments are clearly presented and easy to follow.***

AR: We thank the reviewer for this comment

RC: ***However, I do think the authors could beef up their literature review and motivational examples in order to better place their method in context with previous work.***

AR: We added a whole section on related works, to better point the similarities and differencies with existing methods and softwares. Keeping this discussion short was a choice to make the article easy to read, since it is not intended in particular for the statistical network model audience, but more generally for the network science community, rather as a tool, or a practical solution to a common problem, rather than a theoretical contribution. We agree nevertheless than adding context can always be useful, so we made our best to add something easy to read yet placing the method in the literature. We also added a motivational example on benchmark for network tasks, and clarified in various places that the paper was not intended as a theoretical contribution in statistical network models.

RC: ***I also think it would be nice if the authors could clarify how their method can be used in conjunction with real data given its limitations for statistical inference.***

AR: We added a subsection untilted *Working With Node Attributes*, giving concrete examples of application on real data, but without diving into the details. Again, it was a choice to keep the article concise, to restrain the article to one contribution. Extending on one possible application such as network properties comparison, graph distances or diffusion properties require to add a lot of details on choices made, experimental design, etc., and previous reviewers found that the paper had too many ideas in a single paper with these elements. We plan to develop these ideas in later publications, using Structufy-Net as a brick to develop more theoretical questions. But of course, clarifying how a user can use the library is the purpose of this paper, so we are willing to add more details if something stays unclear.

RC: ***The proposed method has many similarities to a graphon model, and I think the paper could be improved with some discussion of the graphon literature to demonstrate why the proposed method should be preferred. Perhaps one advantage of the proposed method is that it has a more easily controlled noise level?***

AR: We now mention the similarities with graphons in the related work section. It is true that there is a similarity, the presence of a bivariate function, although the graphon match directly to a probability, and not to a rank. However, as we understand it, we do not claim that our random graph model should be preferred over graphons. It would be like claiming, for instance, that the LFR benchmark should be preferred over SBM. We understand our proposition as an answer to a practical problem, how to generate graphs with a controlled expected size and diverse structures. We do not claim that the statistical model behind it is not particularly clever or close to real structures, it is just convenient for the objective task. Indeed, controlling the noise level is useful when generating random graphs, and this idea is not present in graphons, because graphons have been proposed with very different objectives in mind. We tried to dicuss this in the new related work section.

RC: ***On a similar note, the graphs being generated are specific instances of inhomogeneous random graphs, so it could be worth discussing that literature further to identify any similar previous work and how the proposed method improves upon it.***

AR: We are not certain to understand this remark. We tried to introduce various models of inhomogeneous random graphs, additionnaly to the parallels already made in the zoo with SBM, latent space models, etc, in the new related work section. Again, this literature is enormous and we are not sure what exactly would be useful to point to the reader.

RC: ***I think I generally understand the authors' use of the rational Bezier curve as a means of interpolating between the two extremes of complete equality and inequality in the distribution of the probabilities over the edge pairs while fixing m. But how did they decide on this family of curves for interpolation? Is it the most "natural" in some sense? Or does its derivative have a particularly nice form? I'm curious to know because I hadn't heard of these functions before reading the paper.***

AR: We chose the Bézier as an interpolation because it seemed to us a simple, practical answer to the problem. We

tried to clarified that point in the new version when introducing it. In practice, we know the two extremities, when the network is fully deterministic, and when the network is fully random. When plotting these two extremes as in figure 2a), the Bézier solution seems like a simple, practical and logical solution. However, we are fully open to other propositions, it could make perfectly sense to have an alternative interpolation function, and it would be little work to add it to the software. We just do not have any particular idea of what other function we could use to interpolate smoothly between these two extremums.

RC: ***In the motivation section, the authors highlight diffusion as a particular process for which understanding the role of network structure is important, but there are many other example applications they could mention (e.g. synchronization, percolation). By singling out diffusion it appears like the proposed method has a more limited set of applications than it does, and I think the motivation would be strengthened by adding some additional examples.***

AR: We added another example application (Benchmark for network taks), and mention in the diffusion section the other examples provided by the reviewer (synchronization, percolation). Our problem, in some sense, is that we see so many potential usages, that it is difficult to know what to say. Indeed, the answer is the same as simply asking "what is a null/reference model useful for" ? The answer, basically, is "network science" :)

RC: ***The authors do a good job of pointing out the limitations of their method at the end of the paper, in particular the scalability and the independence assumption they make. I would also add here the challenges resulting from the poor compatibility of this method with parameter estimation techniques, necessitating the comparison of ad hoc summary statistics as discussed in the last paragraph.***

AR: We added this parameter estimation difficulty in the discussion of the limit, and we totally agree with it. We are not sure however to understand the comment on the summary statistics: in principle, it seems to us that the random graph generators produced by this framework are equivalent in nature to existing ones such as block models: after picking a rank function and an epsilon, we have a fix probability for each node pair, so we can compute a probability to generate a given graph, and thus in theory, perform a likelihood maximization. Of course, if epsilon and the rank function are considered free parameters, we would end up with a trivial solution (much as with SBM), and it would thus be required to add new constraints, such as a fix epsilon, and constraints on the families of rank functions acceptable. But we totally agree that there is no reason to think that doing this would be computationally tractable, or even a good idea.

RC: ***I think it would be interesting to see how the detectability of community structure varies with epsilon for block-structured rank matrices. Does this have a clean phase transition like in the standard detectability setting? Not a suggestion for publication, just a thought for future studies.***

AR: Thank you, that is a question more to look at :)

RC: ***A very minor point that I found confusing was that the authors said epsilon "controls how strongly the random graph is driven by the community structure." I think they should change the wording to something like "planted structure" or "rank structure", since their method can represent much more than community structure.***

AR: We thank the reviewer for pointing this, it was indeed a writing mistake

RC: ***Another very minor point for presentation is that I found the in-text citation format a bit confusing. There should be brackets or parentheses or something around the in-text citations to separate them from the other text.***

AR: We are not sure to understand this comment, in principle we have parenthese around the in-text citation,

unless we missed some of them (we found 2 occurences at least). This format is imposed by the journal, and we are not used to it either.

RC: ***I enjoyed reading this paper, and I hope the authors find my suggestions helpful.***

AR: Thank you for the comment and indeed we agree that these comments helped improve the paper.

4. Reviewer 2

RC: ***The theoretical contribution of the paper is not clear enough. The paper does not explain the intuition behind the general generative process that it proposes (the two steps on page 3), and most importantly what the rank function may represent or may be interpreted.***

AR: We are not sure to see how to explain the intuition better. The rank function should not be interpreted beyond a ranking of node pairs. Maybe the confusion arised from the fact that many statistical models such as the SBM are used for inferential tasks, while this is not the objetive of the proposed framework. Instead, this article focus on the generation of random graphs with planted structures, a pracical problem in network science. We tried to make this point clearer in several locations in the article, as detailed in answers to other comments and to the other reviewer.

RC: ***Right now, it is not clear why one would use this framework instead of directly using specific and already-known models.***

AR: We added a related work section, in which we tried to clarify the relation with some of existing specific, already-known models. The objective of our framework is clearly not to generate blocks or spatial structure, for which perfectly fine models already exist, but rather to allow the generation of other, more exotic structure, as we tried to demonstrate in the structure Zoo.

RC: ***Linked to this, it is not clear what the Structify-net package adds to the existing software tools we already have. In particular, the graph generator functions in NetworkX seem to already do a fine job in helping users to generate the graphs they are interested in. The authors should show what their tool compares to already existing solutions and why it is a useful addition.***

AR: We added a section listing related frameworks for network generation, including networkx. Indeed, networkx, igraph or graph-tool are valuable softwares to generate random graphs having a block structure, and some simple spatial structures. However, the point of our proposed framework is to allow users to generate random graphs with **different** structures, the ones they might be interested in, not limited to blocks or spatial ones. We tried to clarify this in the related work section, in the introduction to the zoo, and in the introduction.

RC: ***The application on replicating the Watts-Strogatz experiment does not seem to be the best way to demonstrate the value of the framework, because it is very specific. A better way could be to compare the results of the graph generator to several classic graph generators (either theoretically, looking at graph characteristics that are produced, such as clustering, diameter, etc., or comparing software capabilities, such as computing time).***

AR: We agree that the Watts-Strogatz experiment is quite simple and limited. But we used it specifically as a toy model, to illustrate the possibility on simple cases without entering in many unrelated nitpicky details. As the zackary karate club, it is not really useful in itself, but illustrates the usage. Comparing with classic graph generators would be something difficult without making a lot of assumptions, because the principle of our proposition is to have a very large power of expression. Its main interest is that the users can generate the type

of structures they want. The softaware is not designed to generate graphs of a particular structure that could be compared with one flavor of SBM or latent spaces, but to allow users to create their own structures. In some sense, it is a *meta-generator*, a generator of network random graph generators. We nevertheless agree that we could do much more by comparing network properties on various networks. In fact, if the reviewer check the documentation of the library (https://structify-net.readthedocs.io/en/latest/Tutorial/Tutorial.html), they can see that we already performed these type of experiments. The problem was that when we wrote a first paper including both a presentation of the method and examples of applications on real networks comparing with other models such as fitter SBM, the preivous proof readers said that the papers was becoming utterly complex, and that the contribution was becoming unclear. It therefore seems necessary to us to first introduce the framework and its associated software in a paper, and then in a later work, to actually use this tool as a brick to do something useful.

RC: ***Page 2, there are many fields where the Erdös-Rényi model is definitely not the most commonly used. Maybe the most known, or elementary?***

AR: We agree with the reviewer that the claim is not very meaningful, and rephrased our sentence to say that ER is the simplest random graph in the sense that it preserves no other property than the number of edges.

RC: ***Page 4, "P(r) = p, p in [0,1]" reads as: the probability function is a constant p.***

AR: Thank you, we corrected this error.

RC: ***Page 4, what is the rationale behind using Bézier curves (accent missing)? What do the endpoints and control point represent?***

AR: We corrected the accent error, thank you. About the rationale of using Bézier, we added a paragraph to explain it. This choice seemed a practical solution to a problem which is simple to understand when looking at figure 2,a): independently of the choice of the interpolation function, we know what the two extreme functions are those for $\epsilon = 0$ and $\epsilon = 1$. The Bézier function is just a practical way to interpolate between the two without introducing additional parameters, and visually coherent with intuition. We are however open to any other idea for a smooth interpolation between those two natural extremes.

RC: ***The sub-sections on block structures page 6 are straightforward and could be shortened.***

AR: We would prefer to keep them. The article is not written for an audience expert in statistical models, but rather a generic network science approach, and large fractions of this audience is not familiar with SBM approaches.

RC: ***In comparison, the sections on star structure and core-preiphery page 7 could be better developed. Are the formulations for the rank functions proposed the only ones possible? How were they derived?***

AR: The formulations are only examples, and we clarified that further in the introduction of the Zoo chapter, in the description on these models, and introduction. We tried to better explain throughout the article that the model we proposed is intended for the users to create their own structures, and that the Zoo is only here to provide a few examples of what can be done. We also developed the description of those 2 in particular.

RC: ***Why are fractal models useful? (see first minor comment)***

AR: They are only useful to exemplify the flexibility of the model (we now clarify this in the introduction of the zoo section). We do think that some of these models have interesting properties, in particular, having both a hierarchical organization and a bloc structure intertweened (and not simply as a bloc hierarchy) is something that sounds worth investigating. But studying the properties of these networks is not the focus of this article.

RC: ***Page 8, "hierarchical structure" in networks has been defined in many ways and by many other papers,***

and hierarchy in networks is a vibrant topic in social network analysis...

AR: We totally agree and are aware of that. Here we talk about one definition of one particular notion, widely known in network science (it has its own wikipedia page, "Hierarchical Network Model"). We think that the problem was that we were not clear enough about that, and we rewrote the section to make it clearer.

RC: ***Check typos: page 2,"heterogeneityBarthélémy", page 4, "Section describes"...***

AR: Thank you, we corrected these ones and tried our best to correct most of them.

Structify-Net: Random Graph generation with controlled size and customized structure

Rémy Cazabet[1], Salvatore Citraro[2], Giulio Rossetti[2]

[1] Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205, F-69622 Villeurbanne, France
[2] Institute of Information Science and Technologies "A. Faedo" (ISTI), National Research Council (CNR), Italy

Correspondence: remy.cazabet@univ-lyon1.fr

Abstract

Network structure is often considered one of the most important features of a network, and various models exist to generate graphs having one of the most studied types of structures, such as blocks/communities or spatial structures. In this article, we introduce a framework for the generation of random graphs with a controlled size —number of nodes, edges— and a customizable structure, beyond blocks and spatial ones, based on node-pair rank and a tunable probability function allowing to control the amount of randomness. We introduce a *structure zoo* —a collection of original network structures— and conduct experiments on the small-world properties of networks generated by those structures. Finally, we introduce an implementation as a Python library named *Structify-net*.

Keywords: Network Generation, Random Graphs, Network Structure, Python Library

~~Introduction~~

1 Introduction

The structure of networks has long been one of the most studied research questions in network science. In this article, we introduce a method to generate networks of a chosen size, organized according to a structure that can be expressed as an arbitrary ranking function for node pairs. This process thus allows the generation of classic structures such as communities, blocks, and spatial organizations, but also more exotic ones. We subsequently show an application of this framework to study network properties, by extending the classic small-world experiment by (Watts and Strogatz, 1998). A python library (Cazabet, 2023) allowing reproduction of the results and generating networks with custom structures is also introduced.

Context

Generating networks of a chosen size and respecting some constraints is a key topic in network science. It is used in various tasks, for instance, to study network properties (Wang and Chen, 2003), as null models (e.g., Durak et al., 2013), to study the impact on diffusion processes (e.g., Ódor et al., 2021), as benchmarks (e.g., Lancichinetti et al., 2008), etc. In this article, we focus more particularly on a class of ~~statistical network~~ random graph models, in which the probability of observing edges between each pair of nodes is independent of the probability ~~to observe~~ of observing edges between other pairs. This class of models is commonly used to generate various structures, from the homogeneous Erdős–Rényi (ER) generator, to configuration models preserving node degrees, block structures (Abbe, 2017), spatial structures (Cazabet et al., 2017; Waxman, 1988), etc. See section 7 for an overview of related works.

The originality of our work is to propose a generic framework to generate many different network structures while allowing to set:

- The number of nodes n;
- The number of edges m (equivalently, the density);
- A parameter $\epsilon \in [0, 1]$ controlling the strength of the structure bias, i.e., the network is fully determined by the structure definition when $\epsilon = 0$, and increase in randomness with ϵ, becoming an ER network for $\epsilon = 1$.

~~Motivational examples~~

The main advantage of our proposition compared with existing frameworks such as SBM or latent space models is 1) the simplicity for the user to design their own structure logic, by providing their own ranking function (see Section 2), 2) to control the *strenght* of the structure bias using a single numerical parameter. Structify-Net is not intended to be used in parameter inference tasks, but only for the generation of null models, reference models, and random graphs with controlled properties in general.

1.1 Motivational examples

Generating multiple random networks with common properties, either fixed beforehand or preserved from an observed network, is needed in most domains of network science. In this section, we list three motivational examples of usages of such models, for which Structify-net provides a simple practical solution. These examples are in no way exhaustive since random networks are used in many other contexts in network science.

Reference model

When one is interested in studying a network property, such as the transitivity or the homophily of a node attribute, one usually needs a random graph model as a reference. The ~~most commonly used~~ simplest one of them is the Erdős–Rényi (ER) random graph model, in which only the nodes and the expected number of edges are preserved; but in ~~some~~ most cases, one would like to compare with ~~another reference model~~other reference models. For instance, when studying the transitivity, one might wonder if the observed transitivity of a network is significantly higher than the transitivity of a similar graph having a spatial structure, a block structure, or a strong degree of heterogeneity, etc. To explore those hypotheses, one needs a generative model to generate random networks having the same number of nodes and edges as the observed network, but with a particular structure.

~~Influence of structure on diffusion~~Benchmark for network tasks

~~Diffusion~~ A generative model can also be used as a benchmark to test algorithms for complex network tasks developed for capturing fundamental patterns of networks and their functions. A common task where synthetic networks are used to evaluate the performances of an algorithm is community detection, namely the task of identifying —in its general intuition— well-connected and/or well-separated groups of nodes within a network (Fortunato and Hric, 2016). Generators with planted communities are used to estimate the agreement between their ground-truth structure and the communities captured by an algorithm, as for instance the LFR benchmark (Lancichinetti et al., 2008). To test the robustness of a wide variety of algorithms defined for different purposes, one needs generators able to model a wide range of planted structures capturing the properties one intends an algorithm to handle. These properties can range from density to homophily, plus any preferred combination of structural properties leading to clique-, grid-, and star-based structures, among others (Yamaguchi et al., 2020).

Influence of structure on dynamical processes

Dynamical processes on networks, such as diffusion or synchronization, have been studied for a long time~~, for instance in the context of the~~ . Typical examples would be the diffusion of pandemics ~~, or disinformation~~ or polarization on social media. The structure and properties of the network are well-known to have various effects on ~~the diffusion : faster diffusion due to degree heterogeneity Barthélemy et al., 2004, the influence of community structure Peng et al., 2020, or clustering Zhuang et al., 2017~~ these processes. For instance, diffusion speed depends on degree heterogeneity (Barthélemy et al., 2004), community structure (Peng et al., 2020) and clustering (Zhuang et al., 2017), etc. In order to experiment with which factors might control the speed and scale of a particular diffusion process, one should compare such processes with random networks that 1)are comparable in terms of size, i.e., number of nodes and edges, and 2)differ in their structure.

~~Method~~

2 Method

Structify-Net principle is to create probabilistic random graph generators in two steps:

1. A rank function R sets an order among the node pairs, from **most likely to appear** to **less likely to appear**;
2. A probability function P assigns to each pair of nodes a probability to be connected by an edge, based on its rank.

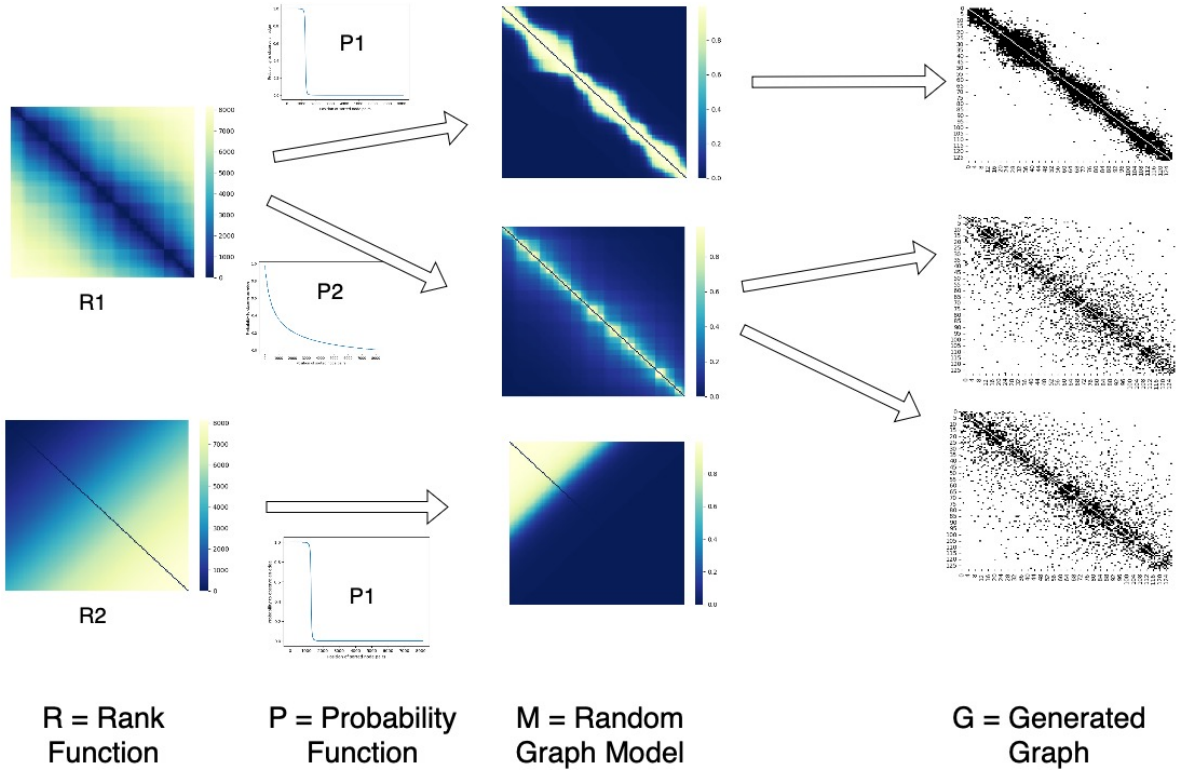


Figure 1. Generating networks using the Structify-Net approach. A rank function defines an ordering between node pairs. A probability function is used to assign an edge probability to each node pair based on their rank in the ordering. This gives a Random graph model, that can be used to generate graph instances. Note how the same Rank function $R1$ can give 2 Random Graph Models using different Probability ~~function~~ functions $P1$ and $P2$, how the same Probability function $P1$ is used for two different Rank functions $R1$ and $R2$, and how multiple graphs can be generated from a single Random Graph Model.

P allows to control the expected number of edges $\widehat{m}$. The function P is independent from the graph structure represented by R; and reciprocally R is independent from the expected number of edges $\widehat{m}$ or the function P.

~~Rank function~~

2.1 Rank function

The principle of Structify-Net generator is to describe a network structure by an edge-pair ranking function. More formally, we define $R(u, v) = r$ the function assigning a value r to each undirected node pair, such as $r \in [1, \frac{n*(n-1)}{2}]$ corresponds to the rank of the node pair (u, v), and $r(u, v) < r(u2, v2)$ means that it is more likely to observe an edge between the pair (u, v) than between the pair $(u2, v2)$. This function can be expressed directly in that form, or be trivially derived from a function R' assigning a cost to each node pair, coupled with a sorting function, ranking pairs by increasing or decreasing values of $R'(u, v)$. In practice, in that case, we also add an infinitesimal random value ι to each cost, in order to avoid ties.

An intuitive example of such a cost function is for the *spatial* structure: given a position vector W_u for each node u (provided to mimic real data, or generated in fully synthetic network generation), the tendency to observe edges can be a function of the distance, e.g., $R'(u, v) = ||W_u, W_v||$. By sorting node pairs by increasing distance, we obtain a spatial structure such that the closer the nodes, the higher their tendency to be connected.

Section 3 describes in more detail various types of network structures that can be represented this way.

~~Probability Function~~

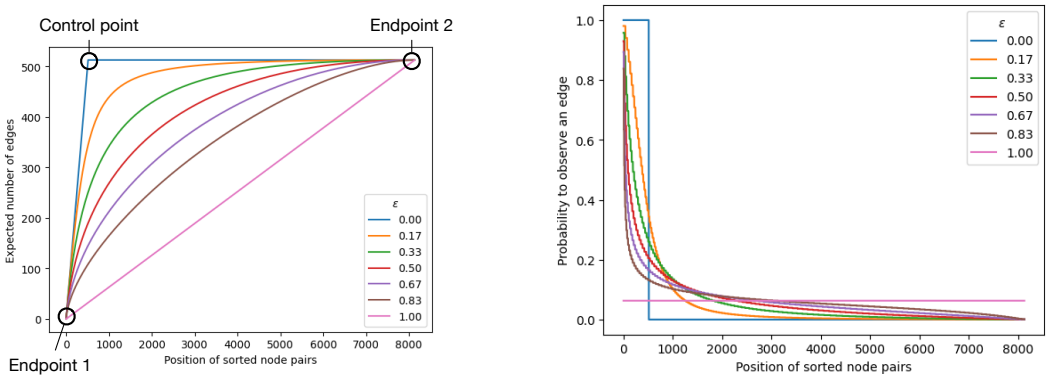
2.2 Probability Function

To go from a ranking of node pairs to a random network generator, we use a rank probability function ~~$P(r) = p, p \in [0, 1]$~~ assigning a probability to each rank, i.e., $P(r) \in [0, 1]$. The only constraint to this function is that it must be non-increasing, so that a node pair of lower rank is at least as likely to be connected by an edge than a node pair of higher rank.

The probability function controls the expected number of edges:

$$\widehat{m} = \sum_{r}^{L} P(r)$$

with $L = \frac{n(n-1)}{2}$



(a) ~~Bezier~~ Bézier interpolation of the number of edges encountered at a given rank

(b) The corresponding probability function, i.e., probability of observing an edge at a given rank

Figure 2. Example of probability functions for various values of ϵ. In this example, we set $m = 128, n = 512$

~~Bezier~~ Bézier Interpolated Probability Function

We propose a family of probability functions controlled by 1) the target expected number of edges m, 2) a parameter ϵ, which controls how strongly is the random graph driven by the ~~community~~ planted structure. The family is defined as follows: at one extreme ($\epsilon = 1$), the probability of observing an edge is independent of the rank, i.e., $P(r) = m/L$, as in an ER random graph. Conversely, at the other extreme $\epsilon = 0$, the m edges connect the m pairs of nodes of lower rank:

$$P(r) = \begin{cases} 0 & \text{if } r \leq m \\ 1 & \text{otherwise} \end{cases}$$

To interpolate between the two, we use a rational ~~Bezier~~ Bézier parametric curve, as illustrated in Fig. 2. The ~~Bezier~~ Bézier curve is defined by **two endpoints**, corresponding to the two points shared by both cumulative distributions: the points (0,0) and (L,m). The **control point** of the curve is (m,m). A weight b allows controlling how close the curve is to each of the two extremes. If $b = 0$, the curve corresponds to $\epsilon = 0$, and $\epsilon(\lim_{b\to\infty}) = 1$. For convenience, we thus ~~simply~~ rescale the given parameter ϵ into b as follows:

$$b = \frac{\log(0.5)}{\log(1 - \epsilon)}$$

By convention, if $\epsilon = 0$, $b = b^{max}$ and if $\epsilon = 1$,$b = 0$, with b^{max} a large integer constant.

The function giving the probability of observing an edge between node pairs given their rank is defined by the derivative of the parametric ~~Bezier~~ Bézier curve (See Fig. 2).

The choice of the Bézier parametric curve arises as a natural solution to the problem as introduced in Fig. 2a: the curve for $\epsilon = 0$ and $\epsilon = 1$ are independent of the interpolation method, the chosen family function should thus propose a smooth interpolation between the two. The Bézier curve answers this problem in a convenient way, although other functions could be used.

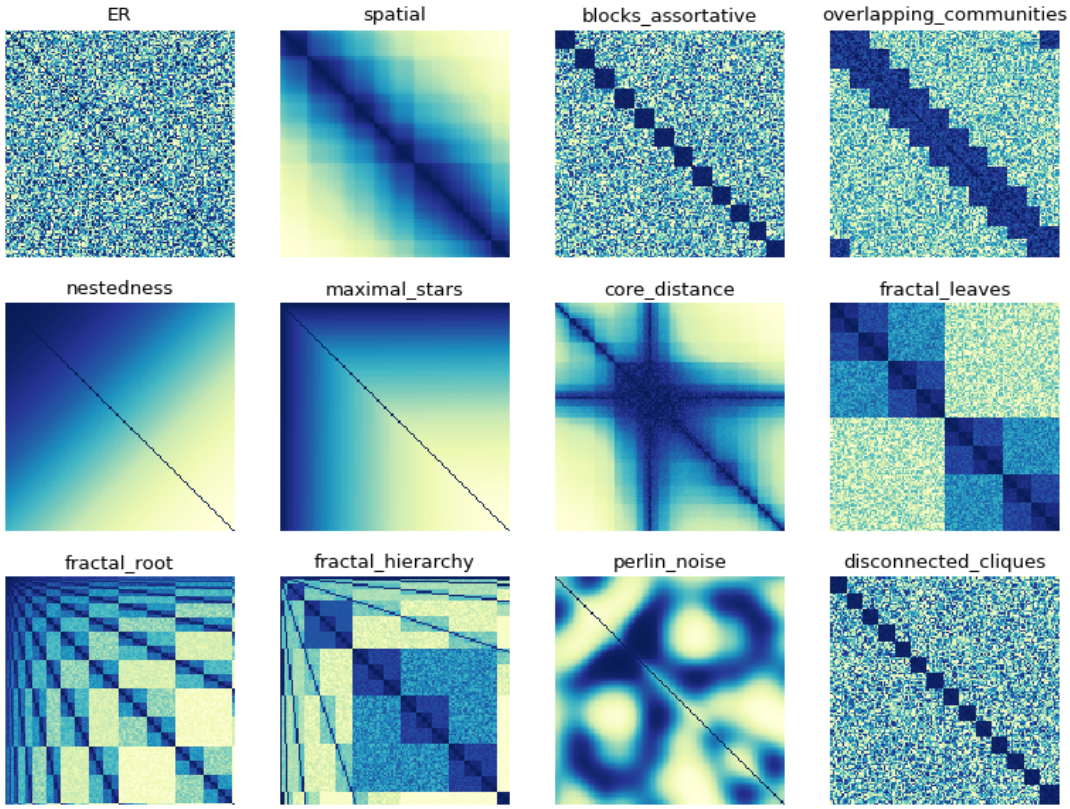


Figure 3. The Structure Zoo. Matrix of node-pairs ranks for networks with 128 nodes. Darker colors correspond to lower ranks. For *Disconnected cliques*, we set $m = 128 * 8$. When involving spatial or clique positions, nodes are ordered according to this value.

~~Structure Zoo~~

3 Structure Zoo

To illustrate the expression power of the Structify-Net rank generation approach, we propose a collection of structures, available in the Python library under the name of Structure Zoo. This collection contains both classic structures widely found in the literature, together with original ones. Fig. 3 introduces matrix representations of the node-pair ranks of all structures in the zoo. The structures in the Zoo are only to be taken as examples, chosen arbitrarily among a few well-known structure types, and a few original ones. The main interest of Structify-Net is for the users to be able to specify their own structure with their own rank functions. The Zoo only represents a set of toy examples.

~~Spatial structure~~

3.1 Spatial structure

Spatial structures are commonly found in the literature. Several versions of random graphs spatial models exist, for instance, the Waxman Graph (Waxman, 1988). More complex versions exist such as the Gravity

model (Cazabet et al., 2017). A simple spatial structure can be easily implemented as a Rank model by using a cost Function,

$$R'(u,v) = d(W_u, W_v)$$

with $d(u,v)$ a notion of distance, such as Euclidean or Haversine distance. W is a matrix such as W_i is a vector representing the position of the node in a d dimensional space. Positions can come from observed data, or be generated. In Fig. 3, we attributed to nodes random positions in a 1-dimensional space.

~~**Assortative block structure**~~

3.2 Assortative block structure

Community structure is one of the best-known types of organization of networks. A simple way to implement such a structure as a random graph generator is to use the stochastic Block Model (SBM), with a constraint of having an assortative structure, ~~i.e~~i.e., edges are more likely to be present between nodes affiliated to the same community than to nodes affiliated to different ones. A simple way to implement this as a rank model is using the following cost function:

$$R'(u,v) = \begin{cases} 0, & \text{if } B_u = B_v \\ 1, & \text{otherwise} \end{cases}$$

With B the block affiliation vector, such as B_i identifies the block to which node i is affiliated with. Of course, many variants are possible, for instance, to take into account the size of blocks/communities.

~~**Overlapping Assortative Block structure**~~

3.3 Overlapping Assortative Block structure

A variant of the block structure allowing nodes to have multiple affiliations. There are numerous ways to model this situation. In the example provided here, we keep the same threshold cost function as for the non-overlapping case, extending it to multiple affiliations, i.e., we use the following cost function:

$$R'(u,v) = \begin{cases} 0, & \text{if } (B_u \cap B_v) \neq \emptyset \\ 1, & \text{otherwise} \end{cases}$$

with B_u the set of blocks to which node u belongs. In Fig. 3, each node belongs to exactly two communities ~~,~~ and the affiliations are chosen such as each community has half of its nodes shared with another community c1 and the other half shared with another community c2.

~~**Block Structure: Disconnected Cliques**~~

3.4 Block Structure: Disconnected Cliques

Communities are often understood as sets of nodes that are strongly connected to each other and more weakly connected to the rest of the graph. A special case of extreme community structure can be set up by having only cliques, without links between them — disconnected cliques. Keeping the same threshold cost function as for assortative blocks, we can find the value for community sizes such as ~~we obtain~~ obtaining the densest possible disconnected subgraphs for $\epsilon = 0$, for a fixed m. Given the average degree $\widehat{k} = \frac{m}{2n}$, we want cliques to be of size $n_c = \lceil \widehat{k} \rceil$. Because n is not necessarily a multiple of n_c, we set the number of communities to $\lfloor \frac{n}{n_c} \rfloor$, and group the remaining nodes in an additional community. The already defined assortative block structure is then applied as usual with those blocks.

~~Nested structure~~

3.5 Nested structure

Nested network structures are well-known in some fields, such as ecology and economics (Alves et al., 2019; Mariani et al., 2019). A nested structure is a type of hierarchical structure, in which the properties/links of each entity are subsets of the properties/links of entities at a higher hierarchical level. We implement this as follows:

$$R'(u, v) = u + v$$

Where u, v are consecutive node indices in [1,n]~~.~~, and $u < v$

~~Star structure~~

3.6 Star structure

Hubs are known to play an important role in many real networks. The hub-and-spoke structure is frequent both in human-designed infrastructure and in natural systems, forming patterns also known as *stars*. ~~We can~~ One way to obtain such a structure ~~with~~ is by using the following rank function:

$$R'(u, v) = u \times n + v$$

Where u, v are consecutive node indices in [1,n], and $u < v$.

~~Core Periphery~~

As seen in Fig. 3, this rank function ranks first all the pairs of nodes including the node of ID 0, then all the pairs of nodes containing node ID 1 and another node with a larger ID, and so on and so forth. The structure therefore tends to create stars with nodes of low IDs in the center.

3.7 Core Periphery

Core periphery structure is another well-known type of organization for complex systems. This organization is often modeled using blocks, one block being the dense core, another block, internally sparse, representing the periphery, and the density between the two blocks is set at an intermediate value. To illustrate the flexibility of the Rank approach, we propose a *soft-core* alternative, the *coreness* dissolving progressively into a periphery. To do so, we consider nodes embedded into a latent space, as for the spatial structure — random 1d positions in our example. The node-pair rank score is computed as the inverse of the product of 3 distances: the distances from both nodes to the center, and the distance between the two nodes. As a consequence, when two nodes belong to the center, they are very likely to be connected; two nodes far from the center are unlikely to be connected unless they are extremely close to each other.

$$R'(u, v) = d(W_u, W_v)d(W_u, \mathbf{0})d(W_v, \mathbf{0})$$

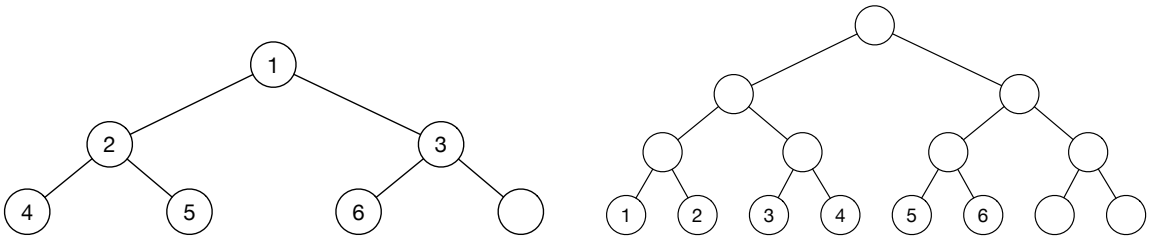
~~With~~ with $\mathbf{0}$ the vector corresponding to the center of the ~~location considered as~~ space, i.e., the *core* of the ~~space~~generated network.

~~Perlin noise~~

Note that this is only an example of a rank function implementing a soft core, and one could imagine many variations of it.

3.8 Perlin noise

Perline noise (Perlin, 2002) is a type of gradient noise frequently used in computer graphics to create images with a realistic feel, such as textures and landscapes. We use it to generate an adjacency matrix, from the upper triangle of a 2d image of size (in pixels) $n \times n$. The R' cost function is the black intensity of the pixel. In practice, Perlin noise tends to create continuous shapes of lower and higher values, with smooth transitions between the two (see Fig. 3) for an example. Such a structure can be interpreted as a fuzzy version of a non-assortative SBM; with stronger relations between some groups of nodes and some other groups of nodes. Perlin noise has a parameter, called *octaves*, allowing ~~to add~~ the addition of smaller-scale structures on top of each other.



(a) Fractal Root tree embedding **(b)** Fractal Leaves tree embedding

Figure 4. Two methods to create fractal structures by embedding nodes into complete binary trees. In the example, we embed 6 nodes. In the simpler case, the probability ~~to observe~~ of observing a graph in the resulting graph is proportional to the distance between the nodes in the tree.

~~Fractal structures~~

3.9 Fractal structures

To better illustrate the expression power of the Structify-Net structure definition, we propose three variations of what we call *fractal structures*. The principle is to embed the nodes into a complete binary tree and to compute the rank scores based on distances on that binary tree. The purpose is to introduce heterogeneity among nodes, which can be used to create specific structures.

~~Fractal root~~

~~In the fractal root structure, we embed nodes of the graph in a complete binary tree of size n, and define the cost function as the distance between nodes in the embedding tree (Fig. 4), $R'(u,v) = d^T(u,v)$, with d^T the geodesic —shortest path— distance between the nodes in the embedding tree.~~

Fractal leaves

In the fractal leaves structure, we create a complete binary tree such ~~as~~ that the number of leaves is n (Fig. 4). We embed nodes of the network on the leaves and use the distance between them in the graph as the cost function. This creates ~~a kind~~ (see 3) a sort of *Matryoshka doll*, hierarchical block structure, in which —considering that edge probability decreases with distance without reaching zero— small dense blocks are contained into larger, sparser blocks, recursively.

Fractal ~~hierarchy~~root

In ~~a seminal paper (Ravasz and Barabási, 2003), hierarchical structure in networks has been defined as having a~~ the fractal root structure, we embed nodes of the graph in a complete binary tree of size n, and define the cost function as the distance between nodes in the embedding tree (Fig. 4), $R'(u,v) = d^T(u,v)$, with d^T the geodesic —shortest path— distance between the nodes in the embedding tree. This structure has

similarities with the previous one, but introduces a particular role for some of the nodes: the root and nodes close to the role now occupy a central, pivotal role, since they are on the shortest paths between nodes on their rights and on their lefts. The network has both a hierarchy of blocks and a sort of central core composed of the nodes close to the root of the tree.

Fractal hierarchy

The fractal principle and custom rank function can be used to create random networks with particular properties of interest. For instance, it has been pointed out in a seminal article(Ravasz and Barabási, 2003) that most real networks have a negative correlation between nodes' individual clustering coefficient and their degrees~~. This definition requires networks having~~ , while most network models do not reproduce this correlation. In the original article, a network model called the "hierarchical network model" has been introduced to generate networks with these properties. To reproduce the so-called *hirarchical property* —note that many other notions of hierarchical networks exist and that this is only the one introduced in (Ravasz and Barabási, 2003 — networks must have 1) heterogeneous degrees, 2) a high average clustering coefficient, and 3) a controlled relation between degrees and clustering coefficient. In the original article, such networks were created through an iterative deterministic algorithm, replacing graph parts with predefined subgraphs until reaching the target size. Instead, we propose here a rank-score approach, embedding nodes in a complete tree as in the fractal root embedding. However, we propose to use a ternary tree instead of a binary one, to increase local clustering. We then choose ~~score functions~~ a score function such as: 1) leaves tend to have high clustering and low degree, and 2) root and other nodes in the higher levels tend to have high degrees and low clustering coefficients. The principle is thus to have a high probability ~~to observe~~ of observing edges 1) between groups of nodes at the bottom of the tree if they have close common ancestors and 2) between nodes at the top of the tree and nodes at the bottom of the tree. The purpose of this example is to show that, by designing an appropriate rank function, one can obtain random graph generators such that the generated graphs have a property of interest.

The rank-score is ~~then~~ thus defined as follows:

$$R'(u,v) = \begin{cases} D(T_u, T_v), & \text{if } ANCESTOR(T_u, T_v) \\ S(T_u, T_v), & \text{otherwise} \end{cases}$$

With T_u, the position of node u in the embedding tree, $ANCESTOR$ a function such as $ANCESTOR(u,v) = TRUE$ if u is an ancestor of v in the embedding tree. Functions D and S are scores capturing respectively a Descendent and Sibling similarity. We use:

$$D(u,v) = min(h(T_u), h(T_v)) + h(T) - (max(h(T_u), h(T_v))$$

with $h(T)$ the global height of the tree and $h(u)$ the height of node u, such as $h(u) = 0$ if u is a leaf, and $h(u) = h(T)$ if u is the root of the tree. This function ranks first pairs of nodes that are far away in terms of tree levels, with a value of 0 between the root and the leaves.

$$S(u,v) = \begin{cases} (d(T_u, T_v) - 2) + h(T_u), & \text{if } h(T_u) == h(T_v) \\ d(T_u, T_v) + h(T), & \text{otherwise} \end{cases}$$

with $d(u,v)$ the shortest path distance in the tree.

~~**Discussion on the structure zoo**~~

3.10 Discussion on the structure zoo

Structures introduced in this structure zoo are only a few examples of the infinite number of possibilities for structures that can be defined by cost functions. We stress that once such a cost function has been chosen, we are able to generate graphs following them, with a chosen number of nodes and edges.

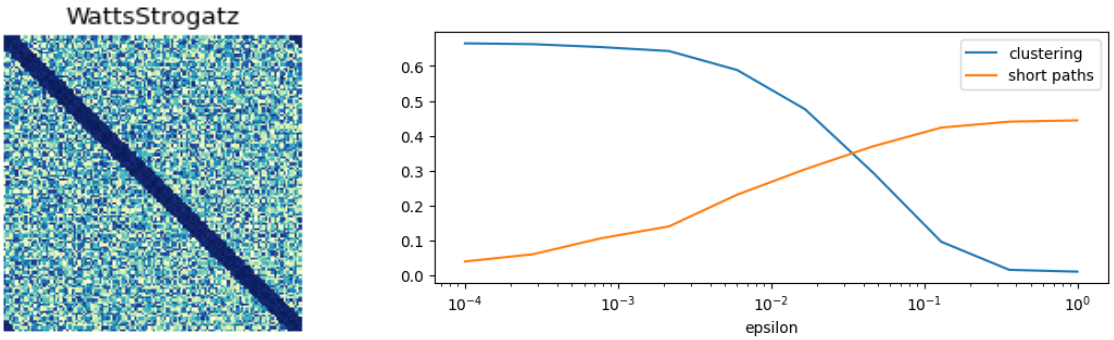
The structures we introduced allow the generation of synthetic networks without prior data, but one can perfectly define a cost function defined on node attributes, e.g., take a real network in which nodes are located in space, belong to known groups, and have other characteristic attributes, and define a structure by using a cost function taking all these attributes into account. Fig. 3 proposes a representation as matrices of all these structures on a network of 128 nodes and 1048 edges.

~~Application: Swall World Structures~~

4 Application: Swall World Structures

One of the most famous properties of network structure is the so-called *small world* property. Introduced in (Watts and Strogatz, 1998), it characterizes a network as being a small world if it has both 1) a high clustering coefficient — significantly larger than in an ER random graph, 2) a short average distance — of the same order of magnitude as in an ER random graph. This property, considered ubiquitous in real networks, has been reproduced in (Watts and Strogatz, 1998) by progressively adding randomness to a regular network, built such as the n nodes are ordered in a circle, and each node is connected to its $\widehat{k}/2$ neighbors in both directions. The small world property emerges because, when we rewire edges at random, the average distance decreases faster than the clustering coefficient — both being large in the regular network and low in the ER random graph.

We conduct an experiment to observe how other structures behave in ~~term~~ terms of small-worldness when submitted to a similar experiment, i.e., starting with an archetypal structure, and adding noise progressively.



(a) The proposed rank matrix. It has similarity with, e.g., the spatial one in Fig. 3

(b) The small-world profile. As expected, the short path index increases significantly while the clustering coefficient remains close to the original value when increasing randomness

Figure 5. Replicating the Watts-Strogatz experiment

~~Reproducing the Watts-Strogatz experiment~~

4.1 Reproducing the Watts-Strogatz experiment

Watts-Strogatz rank model

To mimic the original small-world experiment, we define a rank-based structure using a cost function, parameterized by the number of nodes n and the desired average node degree k.

$$R' = \begin{cases} 0, & \text{if } (v-u) \mod (n-k/2) < k/2 \\ 1, & \text{otherwise} \end{cases}$$

with u, v node indices taken from $[0, .., n-1]$. The corresponding rank matrix is shown if Fig. 5a

Scoring functions

In the original article, clustering coefficients and average distances were expressed as a fraction of the value obtained for the regular structure. We cannot reuse this approach for multiple structures, having different starting values. Instead, for the clustering, we directly use the average clustering coefficient score, $CC(g) \in [0, 1]$.

For the average distance, we propose a scaled value defined as:

$$\widehat{\delta}(G) = \begin{cases} 0, & \text{if } \frac{|\mathcal{G}(G)|}{n} \leq 0.9 \\ \frac{1}{1+\max(0,\widehat{d}(\mathcal{G}(G))-2)}, & \text{otherwise} \end{cases}$$

with $\mathcal{G}(G)$ the largest connected component of graph G, and $\widehat{d}(G)$ the average shortest path distance between nodes of graph G.

The property of this score is that $\widehat{\delta} \in [0, 1]$, with $\widehat{\delta} = 1$ if every node can reach any other node in two hops or less (e.g., a full star structure), and $\hat{\delta}$ decreases quickly as the average distance $\widehat{d}$ increases.

Watts-Strogatz experiment replication

We use our setting to replicate an experiment similar in nature to the one in the original article. We used the same parameters, i.e., $n = 1000, k = 10$. We progressively add randomness, from a deterministic network to an ER random graph by varying parameter ϵ of the probability function. Fig 5b shows results coherent with the original article: after adding some randomness, the short path index has increased significantly, while the clustering coefficient still remains close to its value for the deterministic network.

~~Small-World profiles for other structures~~

4.2 Small-World profiles for other structures

We can apply the same process to the other structures defined in our structure zoo, with the same number of nodes and edges. In Fig. 6, we observe a wide variety of behaviors.

- Fractal-hierarchy and maximal-star structures display a *super-small-world* behavior, having both short paths and high clustering coefficients. This can be easily understood: their hierarchical nature creates a giant hub maximizing reachability. Fractal-hierarchy is designed such as most nodes of low degree have a high clustering coefficient, due to a local structure. On the contrary, in maximal stars, most nodes are connected only to a few hubs, but since those hubs are connected to each other, they also have a high clustering coefficient.

- Nested, overlapping communities, and Perlin noise seems, on the contrary, to be *anti-small-world*, with both a low clustering coefficient and long paths. Again, this is due to different factors. For instance, the nestedness and Perlin noise concentrate so many edge probability between a small subset of nodes, that many nodes are disconnected, leading to the absence of a giant component —thus to an infinite average distance. The overlapping community, instead, is created in a way that makes it look like the original Watts-Strogatz circular structure, as can be observed in Fig. 3. Its low clustering coefficient comes from 1)many nodes not having a degree of at least 2, or 2)structures being too large compared to the number of edges, so that the probability ~~to form~~ of forming triangles is low.

- Other structures tend to follow a pattern roughly similar to the original article, with more or less pronounced profiles. In some cases, the short distance score remains at zero until reaching a certain amount of noise, due to the absence of a giant component.

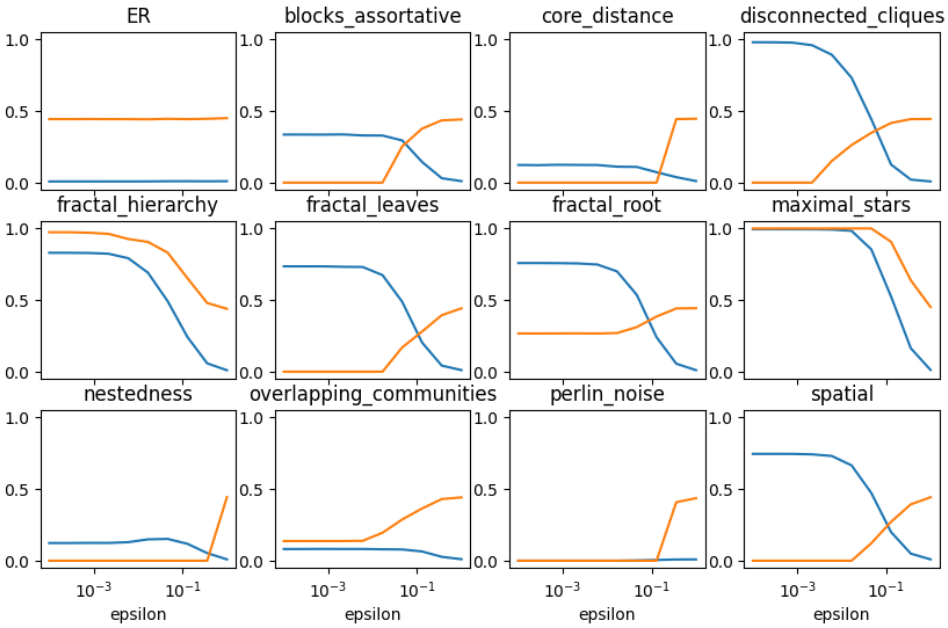


Figure 6. Small-World profiles of networks generated by generators of the *structure zoo*. **Blue** is clustering, **orange** is the short path index (lower values correspond to longer paths). We observe *super-small-world*, e.g., fractal-hierarchy, *anti-small-world*, e.g., nestedness, profiles similar to the Watts-Strogatz network, e.g., spatial or blocks assortative.

~~Python library~~

5 Working With Node Attributes

In the previous section, we have seen an experiment in random networks without metadata/attributes, i.e., with interchangeable nodes. However, as mentioned in the model definition, it is possible to use any node information in the rank function. In this section, we provide two simple motivational examples.

Let's imagine working with a dataset such as a global air transport network. Nodes correspond to airports and are identified by their name, location, and country. Edges might correspond to having at least one direct flight between two airports. To study a property of such a network, for instance, its clustering, average shortest path, or properties of a virus diffusion on that network, one would usually compare that property with its value in networks generated by a reference null model, such as ER or Configuration model. Another possibility offered by Structify-net is to use the node attributes. For instance, we can consider only the airport positions, and rank node pairs according to the distance between them in the dataset, which corresponds to using the *Spatial structure* defined in the Zoo, using metadata instead of random positions. Alternatively, we could use the country information as metadata to the *Assortative Block Structure* of the Zoo. Of course, it could be possible to use already-existing methods such as block models or spatial models to do the same. Structify-Net simply offers a convenient way to do it, for any model, just by providing the appropriate function.

But it is of course possible to go far beyond simple blocks or latent space, and to propose a custom ranking function based on the case study. For instance, one can use machine learning to design a ranking function retaining complex properties from an observed graph. In the airport example, we could use a classification algorithm such as logistic regression or a decision tree to learn how likely it is to observe an edge between two nodes given their properties. From the observed network, we would extract a set of training examples $\{distance, sameCountry \in \{0,1\}, edge \in \{0,1\}\}$, and train a classifier, which can then assign a class probability to each node-pair. This probability would not however be usable directly to create null models preserving the number of edges. Instead, we can use this probability as a rank function, and

generate random graphs with a chosen number of edges using Structify-Net. The structure produced will preserve some of the properties related to attributes of the original graph —probably, a higher tendency of airports to be connected if they are located in the same country and if they are close in space.

6 Python library

An important aspect of such a generator is to allow other researchers to use it for their own needs, whether it be to generate networks according to a structure described in the structure zoo, or to define their own. We thus release with this paper a pip installable python library Cazabet, 2023, together with its documentation[1]. For convenience, the library is compatible with Networkx (Hagberg et al., 2008). Obtaining a rank model corresponding to one of those defined in the zoo, such as the nested structure, is as simple as calling it:

```
import structify_net.zoo as zoo
n=128
rank_model = zoo.sort_nestedness(n)
```

Generating a network as a Networkx object from it is straightforward:

```
import structify_net.zoo as zoo
n=128
m=512
generator = zoo.sort_nestedness(n).get_generator(epsilon=0.5,m=m)
g_generated = generator.generate()
```

One can also define a custom structure by providing a rank-score function:

```
import structify_net as stn
n=128
m=512

def R_nestedness(u,v,_):
    return u+v
rank_nested = stn.Rank_model(n,R_nestedness)
g = rank_nested.generate_graph(epsilon=0.1,m=m)
```

The library allows easy plotting of the rank-score matrices and node-pair probability matrices, and more generally reproduces all the content of the current article.

~~Discussion~~

~~This~~

7 Related Works

Many works can be found in the literature on the generation of random graphs. A complete survey is beyond the scope of this paper; we will nevertheless briefly introduce in this section the most common random graph models and existing software for random graph generation.

We can make a distinction between generative models that are designed for model parameter inference, and those that are not. Models designed for inference are usually controlled by a limited number of parameters, and have some appropriate statistical properties allowing to infer parameter values to match a specific observed graph or series of graphs(e.g., SBM or latent-space models). On the contrary, generative-only models are designed to generate graphs with some specific properties in order to use them in downstream tasks (e.g., LFR or Waxman graphs). Structify-Net, although defined as an edge-independent random graph model, rather belongs to the second category, since it is not designed for parameter inference.

[1] https://structify-net.readthedocs.io/en/latest/

7.1 Common Random Network Models

The simplest way to generate random graphs is certainly the Erdos-Renyi (ER) random graph model, that we already introduced. ER random graphs are fully homogeneous and do not have any particular structure; but a controlled expected size. Configuration models, in particular the Chung-Lu version, allow the generation of graphs of controlled size without mesoscopic organization but preserving the nodes' degrees.

Block structures

Stochastic Block Models (SBM) define random graph models with block structures. In their simpler form, they are defined by two sets of attributes: a vector defining the block to which each node belongs, and a matrix defining the number of edges between each pair of blocks. They exist in various flavors, the canonical version being edge-independent (Snijders and Nowicki, 1997), while microcanonical versions (Peixoto, 2017) are not. These models also exist with or without node degree preservation. The literature on SBM mostly focuses on inference, but SBM can naturally be used to generate random networks, either by choosing model parameters or by using those obtained after inference. A popular way to set manually the parameters for custom structure generation is to fix a number of blocks and a number of nodes in each block, then to choose an internal edge probability p_{in} and an external probability p_{out}. Usually, $p_{in} > p_{out}$, thus defining assortative blocks. Arbitrary block structures, with different block sizes or non-assortative structures can be defined by setting the parameters accordingly.

Multiple variants of block models exist, such as overlapping SBM(Latouche et al., 2009) or hierarchical ones (Schaub et al., 2023). Block models can also be used to generate core-periphery structures, typically by setting one core block and one or several peripheral blocks. This structure however cannot generate other types of possible core-periphery structures, such as a continuous change between core and periphery.

A popular random graph generator with community structure is the LFR Benchmark(Lancichinetti et al., 2008). Not designed for inference, it allows the generation of networks with realistic properties with a limited number of parameters, thus in a more convenient way than with manually initialized SBM. A more recent variant solving some of the problems of LFR is the ABCD random graph generator(Kamiński et al., 2021).

Latent space structure

Various models exist to generate random graphs in which nodes are embedded into a space, the probability of observing an edge depending on the distance between them. Among popular examples, we can cite Random Geometric Graphs (RGG, Dall and Christensen, 2002), in which nodes are connected if their distance is below a parameter, and Waxman Graphs (Waxman, 1988), in which edge probability decreases exponentially with the distance. The gravity model(Wojahn, 2001) is an alternative in which the probability of observing an edge depends both on nodes' degrees and on a deterrence function controlling the influence of distance on edge probability. The parameters, in particular the deterrence functino, can also be inferred to fit a given observed network (Cazabet et al., 2017). Latent spaces are not limited to geographical space, and models have been proposed for the inference of social spaces, for instance (Hoff et al., 2002).

A model related both to SBM and to spatial models is the Random Dot Product Graph (RDPG) model (Young and Scheinerman, 2007). Nodes are characterized by a vector defining their positions in a latent space, and the probability of observing an edge between nodes is given as the dot product between their vectors.

Some authors consider instead that networks are better represented in hyperbolic space, leading to the proposition of Hyperbolic random graph generators (Aldecoa et al., 2015).

Homophily

Other generators model edge probabilities depending on the nodes' attributes. They allow to analyze the interplay between similarities in structure (e.g., common friends in social networks) and similarities in node attributes (Asikainen et al., 2020), or to investigate mechanisms of non-structural closures such as the

formation of links between nodes having similar attributes that do not share common neighbors, as a base of node homophily (Murase et al., 2019).

Generic random graph models

A difference between models introduced until then in this section and Structify-Net is the restriction to a single type of network structure. Since Structify-Net accepts any node-pair ranking function as input, it allows the generation of block, spatial, but also other types of structures such as core-periphery, nestedness, etc.

Another family of highly expressive random graph models is the Exponential Random Graph Model family (ERGM, Lusher et al., 2013). ERGMs define the probability of observing a given graph G as $P(G) = \frac{exp(\Theta \cdot X(G))}{c(\Theta)}$, with Θ a vector of network parameters, $X(G)$ network characteristics, including for instance the number of triangles or node properties, and $c(\Theta)$ a normalizing constant ensuring that the sum of $P(G)$ for all G is equal to 1. ERGMs are mostly used in the context of model inference, and allow in theory to model non-independent edges, e.g., taking into account a triangle closure propension. However, due to the computational complexity, this approach is limited to small graphs.

Finally, an approach sharing some similarities with our framework is the *graphon* (Glasscock, 2015), contraction of *graph function*, first introduced in (Lovász, 2006). A *graphon* can be defined (Orbanz and Roy, 2014) as a bivariate function $W : [0, 1]^2 \rightarrow [0, 1]$. That function returns an edge probability for each pair of nodes, based on a node latent variable. Graphons were first introduced mostly as theoretical objects, in the context of sequences of large, dense graphs. More recently, works have focused on the inference of this nonparametric model, as smooth-graphons (Sischka and Kauermann, 2022a) or combined with an SBM approach (Orbanz and Roy, 2014; Sisch). While graphons share the principle of using a function to characterize the network structure with our approaches, they are part of a very different literature. Graphons are more generic, so much so that SBM, spatial, and nearly all latent-variable-based statistical models can be considered a special case of graphons. The literature on the topic focuses on inference problems, and notions such as node-pair ranking or randomness parameters are not present.

The Structify-net framework is aimed to play a different role compared with all methods introduced in this section. ERGM and Graphons are families of models, designed for model inference rather than network generation. They are so general that they do not offer much help to define a particular structure, and are used in general in a restricted context, for instance with block-approximations for graphons, or imposed number of triangles for ERGMs. SBM, gravity models and configuration models are on the contrary more specific than structify-net, focusing on a single type of network structure. Furthermore, they are often used in the context of model inference. On the contrary, other models such as LFR benchmark or Waxman graphs are designed, as Structify-Net, to generate networks with controlled properties, but they also focus on one specific type of structure. Our contribution thus occupies an original position in the scientific landscape on random graphs: it is designed for the generation of random graphs and not the inference task, it is more flexible than LFR or SBM, and offers a more convenient way to generate graphs of controlled properties compared with ERGM or Graphons.

7.2 Software

Several easy-to-use libraries propose to generate networks with blocks, following the Stochastic Block Model approach. Among the most popular, we can cite networkx (Hagberg et al., 2008) and iGraph (Csardi, Nepusz, et al., 2006), which include an SBM generation function allowing to define blocks of arbitrary sizes, arbitrary probabilities of observing edges between them, and then generate a graph accordingly. More advanced functions are proposed in the graph-tool (Peixoto, 2014) library, allowing to generate *microcanonical*, degree-preserving versions, and several other variants of block structures.

The same libraries offer, under the name of *geometric models*, some possibilities for spatially constrained network structures. Most of these methods, however, do not allow setting the number of edges, since they

instead require setting a threshold below which edges exist, or using an a priori function (Waxman random graph). Only the *k-nearest neighbors* method allows one to choose the number of edges among multiples of the number of nodes, but it is a deterministic generator. These libraries also contain other types of network generators with non-random structures, such as lattices, or networks defined by a process such as the forest-fire model.

Another notable network generator is the LFR benchmark, which is implemented in networkx in a simplified version, or available as a standalone code to have access to all its settings. Software to work with graphons is much more scarce; we found an R library implementing graphon inference and graphon random graph generation [2]; a recent python code exists also, although not in the form of a documented library [3]. These libraries however have nothing to see, in term of usage of capabilities, with Structify-Net. They are designed for completely different purposes. The reference library for working with ERGM is the R package ergm(Hunter et al., 2008)[4], focusing on model inference.

8 Discussion

This article introduced a new method to generate random networks with a customizable network structure, and a target number of nodes and edges, while controlling the amount of randomness. To the best of our knowledge, this is the first random network generator allowing to do so. We think that having such a generator opens doors to new research directions in network science, for studying the properties of networks with some particular structures —as we have done in the experimental section, or as a reference model for observed graphs, to name a few.

Moreover, one of the main strengths of the generator is its ability to control situations where a process/rule of the structural organization (expressed by a pair-node rank) can be mixed with "unknown" random processes (expressed by ϵ); thus, among the observed edges, some of them strictly follow the structural constraints imposed by the rank, and some of them can go beyond the explanation of such constraints. The possibility to analyze this mix —between edges driven by the organization and randomness— is quite important, especially given that network behaviors like small-worldness or homophily can be better explained when randomness is added to a rule of connection (Talaga and Nowak, 2020).

Regarding the analysis of such network behaviors, another strength of the generator is the possibility to exploit the properties of nodes when defining a rank, thus including elements representing, in principle, physical position, political opinions on a spectrum, gender, or even degree. We focused here on the distance between vectors of nodes' positions in a d dimensional space for building a spatial structure (cf. "Structure Zoo", 2.1). We also focused on the affiliation to the same group for building an assortative block structure (cf. "Structure Zoo", 2.2). Similarities between such structures lead us to acknowledge the significance of incorporating node metadata/attributes to generalize a wide variety of network behaviors. We focused here on analyzing the small-world property, but the same can be applied to other behaviors. In principle, behaviors like homophily (McPherson et al., 2001), could be described just as a particular case of either a spatial organization (if attributes are numerical) or of an assortative block structure (if attributes are categorical).

~~Limits and future work~~

8.1 Limits and future work

The main limit of the current work is scalability: node-pair ranks and probability matrices are dense matrices, which can be memory-demanding for large graphs. The generation process also requires an independent random draw for each node-pair. These limits could be overcome in future work. Another limit is that network

[2] https://cran.r-project.org/web/packages/graphon/index.html
[3] https://github.com/BenjaminSischka/GraphonPy
[4] https://gvegayon.github.io/appliedsnar/the-ergm-package.html

structures in which probabilities of observing an edge between a pair of nodes are not independent of adjacent ones —for instance, to generate random networks of a specific size and a specific clustering coefficient— cannot be expressed by a rank-structure. Another limit compared with some other random graph models such as SBM or RDPG is the absence of an efficient inferential solution. In particular, without any constraint on the node-pair order, an inferential approach would always find a trivial uninformative solution in which all connected node-pairs are ranked first. Note however that, since it is possible to compute the probability to obtain a given graph for a set of parameters, it could be possible in theory to use maximal-likelihood inference on a subset of the models, e.g., by fixing the amount of randomness ϵ, and constraining the domain of acceptable rank functions.

~~Finally, in~~ In future work, we plan to compare the properties of real-world networks with those of the synthetic ones generated from structures such as those of the zoo. Having such a variety of possible structures, we expect to be able to characterize real networks, by observing similarities and differences with the synthetic ones, e.g., a real network might have a clustering coefficient and an average distance compatible with the Watts-Strogatz network, but differ in degree heterogeneity and robustness, while another synthetic network might have more similar properties in all those aspects. In particular, we will investigate the role of randomness, to test the original idea of the Watts and Strogatz small-world definition, i.e., that randomness is at the source of complex networks' properties.

Fundings

This work is supported by the European Union – Horizon 2020 Program under the scheme "INFRAIA-01-2018-2019 – Integrating Activities for Advanced Communities", Grant Agreement n.871042, "SoBigData++: European Integrated Infrastructure for Social Mining and Big Data Analytics" (http://www.sobigdata.eu).

This project was partly founded by BITUNAM grant ANR-18-CE23-0004.

Conflict of interest disclosure

The authors declare that they comply with the PCI rule of having no financial conflicts of interest in relation to the content of the article.

Data, script, code, and supplementary information availability

Scripts and codes are available online as a documented, pip-installable Python library.

- GitHub (https://github.com/Yquetzal/structify_net
- DOI: https://doi.org/10.5281/zenodo.7966895
- Documentation: https://structify-net.readthedocs.io/en/latest/

.

References

Abbe E (2017). Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research* 18, 6446–6531.

Aldecoa R, C Orsini, and D Krioukov (2015). Hyperbolic graph generator. *Computer Physics Communications* 196, 492–496.

Alves LG, G Mangioni, I Cingolani, FA Rodrigues, P Panzarasa, and Y Moreno (2019). The nested structural organization of the worldwide trade multi-layer network. *Scientific reports* 9, 1–14.

Asikainen A, G Iñiguez, J Ureña-Carrión, K Kaski, and M Kivelä (2020). Cumulative effects of triadic closure and homophily in social networks. *Science Advances* 6, eaax7310.

Barthélemy M, A Barrat, R Pastor-Satorras, and A Vespignani (2004). Velocity and hierarchical spread of epidemic outbreaks in scale-free networks. *Physical review letters* 92, 178701.

Cazabet R (2023). Structify-net. https://github.com/Yquetzal/structify_net/.

Cazabet R, P Borgnat, and P Jensen (2017). Enhancing space-aware community detection using degree constrained spatial null model. In: *Complex Networks VIII: Proceedings of the 8th Conference on Complex Networks CompleNet 2017 8*. Springer, pp. 47–55.

Csardi G, T Nepusz, et al. (2006). The igraph software package for complex network research. *InterJournal, complex systems* 1695, 1–9.

Dall J and M Christensen (2002). Random geometric graphs. *Physical review E* 66, 016121.

Durak N, TG Kolda, A Pinar, and C Seshadhri (2013). A scalable null model for directed graphs matching all degree distributions: In, out, and reciprocal. In: *2013 IEEE 2nd Network Science Workshop (NSW)*. IEEE, pp. 23–30.

Fortunato S and D Hric (2016). Community detection in networks: A user guide. *Physics reports* 659, 1–44.

Glasscock D (2015). What is... a graphon. *Notices of the AMS* 62, 46–48.

Hagberg A, P Swart, and D S Chult (2008). *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Hoff PD, AE Raftery, and MS Handcock (2002). Latent space approaches to social network analysis. *Journal of the american Statistical association* 97, 1090–1098.

Hunter DR, MS Handcock, CT Butts, SM Goodreau, and M Morris (2008). ergm: A package to fit, simulate and diagnose exponential-family models for networks. *Journal of statistical software* 24, nihpa54860.

Kamiński B, P Prałat, and F Théberge (2021). Artificial benchmark for community detection (abcd)—fast random graph model with community structure. *Network Science* 9, 153–178.

Lancichinetti A, S Fortunato, and F Radicchi (2008). Benchmark graphs for testing community detection algorithms. *Physical review E* 78, 046110.

Latouche P, E Birmelé, and C Ambroise (2009). Overlapping stochastic block models. *arXiv preprint arXiv:0910.2098*.

Lovász L (2006). The rank of connection matrices and the dimension of graph algebras. *European Journal of Combinatorics* 27, 962–970.

Lusher D, J Koskinen, and G Robins (2013). *Exponential random graph models for social networks: Theory, methods, and applications*. Cambridge University Press.

Mariani MS, ZM Ren, J Bascompte, and CJ Tessone (2019). Nestedness in complex networks: observation, emergence, and implications. *Physics Reports* 813, 1–90.

McPherson M, L Smith-Lovin, and JM Cook (2001). Birds of a feather: Homophily in social networks. *Annual review of sociology*.

Murase Y, HH Jo, J Török, J Kertész, and K Kaski (2019). Structural transition in social networks: The role of homophily. *Scientific reports* 9, 4310.

Ódor G, D Czifra, J Komjáthy, L Lovász, and M Karsai (2021). Switchover phenomenon induced by epidemic seeding on geometric networks. *Proceedings of the National Academy of Sciences* 118, e2112607118.

Orbanz P and DM Roy (2014). Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE transactions on pattern analysis and machine intelligence* 37, 437–461.

Peixoto TP (2017). Nonparametric Bayesian inference of the microcanonical stochastic block model. *Physical Review E* 95, 012317.

— (2014). The graph-tool python library. *figshare*. https://doi.org/10.6084/m9.figshare.1164194.

Peng H, A Nematzadeh, DM Romero, and E Ferrara (2020). Network modularity controls the speed of information diffusion. *Physical Review E* 102, 052316.

Perlin K (2002). Improving noise. In: *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pp. 681–682.

Ravasz E and AL Barabási (2003). Hierarchical organization in complex networks. *Physical review E* 67, 026112.
Schaub MT, J Li, and L Peel (2023). Hierarchical community structure in networks. *Physical Review E* 107, 054305.
Sischka B and G Kauermann (2022a). EM-based smooth graphon estimation using MCMC and spline-based approaches. *Social Networks* 68, 279–295.
— (2022b). Stochastic Block Smooth Graphon Model. *arXiv preprint arXiv:2203.13304*.
Snijders TA and K Nowicki (1997). Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of classification* 14, 75–100.
Talaga S and A Nowak (2020). Homophily as a Process Generating Social Networks: Insights from Social Distance Attachment Model. *Journal of Artificial Societies and Social Simulation* 23, 6. ISSN: 1460-7425.
Wang XF and G Chen (2003). Complex networks: small-world, scale-free and beyond. *IEEE circuits and systems magazine* 3, 6–20.
Watts DJ and SH Strogatz (1998). Collective dynamics of 'small-world'networks. *nature* 393, 440–442.
Waxman BM (1988). Routing of multipoint connections. *IEEE journal on selected areas in communications* 6, 1617–1622.
Wojahn OW (2001). Airline network structure and the gravity model. *Transportation Research Part E: Logistics and Transportation Review* 37, 267–279.
Yamaguchi H, Y Ogawa, S Maekawa, Y Sasaki, and M Onizuka (2020). Controlling internal structure of communities on graph generator. In: *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, pp. 937–940.
Young SJ and ER Scheinerman (2007). Random dot product graph models for social networks. In: *International Workshop on Algorithms and Models for the Web-Graph*. Springer, pp. 138–149.
Zhuang Y, A Arenas, and O Yağan (2017). Clustering determines the dynamics of complex contagions in multiplex networks. *Physical Review E* 95, 012312.