

Quantifying randomness in complex graph sets using pairwise graph distances

Bram Mornie^{1*}, Didier Colle¹, Pieter Audenaert¹, Mario Pickavet¹

¹IDLab, Department of Information Technology, Ghent University –
imec, Technologiepark-Zwijnaarde, Ghent, 9052, Belgium.

*Corresponding author(s). E-mail(s): bram.mornie@ugent.be;
Contributing authors: didier.colle@ugent.be;
pieter.audenaert@ugent.be; mario.pickavet@ugent.be;

Abstract

While simple random graph models often have a strong theoretical basis, graphs with complex constraints can only be generated by heuristic algorithms. Using these methods, there is no guarantee that the generated graphs are sufficiently random. However, this is important knowledge in many applications of random graphs, such as creating realistic and diverse synthetic datasets. To address this problem, we propose a randomness measure based on pairwise graph distances, and we present four new feature-based graph distance measures tailored to graphs with bounded frequencies of small subgraphs (graphlets). Three of the distances use features derived from graphlet frequencies, while the fourth is derived from the joint degree distribution and therefore much easier to compute. We evaluate these distances in a series of experiments on synthetic and real networks. Our experimental results show that two graphlet-based distances do not reliably show good results and, in particular, do not reproduce the expected trends in experiments on measuring randomness. However, our novel Radial Graphlet Distribution Distance is effective, and comparable in performance to state-of-the-art methods. These findings highlight the importance of selecting an appropriate graph distance. Finally, we show that our easy-to-compute Joint Degree Distance is a viable alternative to graphlet-based distances, especially for measuring randomness in sets of very large networks.

Keywords: complex networks, graph distances, network motifs, graphlet analysis, random graphs

1 Introduction

Graphs are used extensively to represent complex data in many domains, such as biology, social sciences, transport and logistics, and so on. As part of the analysis of real-world networks, a common task is to generate realistic random graphs. A wide variety of random graph models exists, most of which were designed to reproduce one or several structural properties typically observed in certain real networks: long-tailed degree distribution [1], small-worldness [2], community structure [3], etc. Random graphs are particularly useful when evaluating algorithms for solving network problems [4, 5], or they can serve as a null model to identify significant, non-random structures in real-world networks [6, 7]. In both applications, it is important that certain properties are indeed reproduced by the random graph model.

On the other hand, the set of generated graphs must also be structurally diverse. Assume a graph generation method is meant to produce graphs with, for example, a fixed degree distribution. This constraint already lowers the diversity of possible graph structures, but this effect is unavoidable and not undesirable. It allows to analyze whether other properties of real-world networks are significant, or merely a direct consequence of the specific degree distribution. However, the generation method itself might introduce additional, unwanted constraints. Suppose that our generated graphs with fixed degree distribution also all have a small diameter, but that this small diameter is purely a side-effect of the specific generation method. Then the large diameter of a real network with the same degree distribution would be considered significant. This inaccurate conclusion is entirely the result of the biased graph generator. For some simple graph ensembles, such as graphs with a fixed degree sequence, there exist generation methods that can be proven to sample uniformly from the ensemble [8, 9]. However, the exact approach quickly becomes unfeasible when considering more complex ensembles.

Therefore, in this paper, we address the challenging problem of how to measure *randomness* in a set of graphs. We specifically consider graphs with constraints on the degree distribution and *graphlet frequencies*. *Graphlets* or *motifs* are small, connected subgraphs that were originally introduced as the basic building blocks of larger networks [7, 10]. The frequency of occurrence of each graphlet topology can serve as a fingerprint of the network structure. Graphlets have found widespread use in network biology [11–14] and social network analysis [15, 16], among other fields. Random graphs with prescribed graphlet frequencies can serve as network models and null models that preserve local structural patterns. This enables statistical analyses of network properties beyond degree-based constraints while accounting for higher-order network structure. However, generating unbiased graphs with fixed frequencies of graphlets with three or more nodes is a very difficult task. While there exist some Markov Chain Monte Carlo methods that can obtain correct samples by rewiring the original network [17–19], the convergence time of the Markov Chain is prohibitively long. Other generators that consider graphlets, rely on heuristic algorithms to obtain graphs with graphlet frequencies close to a target network [7, 20, 21]. Machine learning models are an alternative, some of which have been shown to reproduce graphlet frequencies in training graphs quite well [22, 23]. Both approaches have in common that theoretically analyzing whether the generation algorithms are unbiased, is not an

option. For this reason, and given the importance of graphlet statistics in numerous applications, graphs with constrained graphlet frequencies provide a quintessential scenario in which a randomness measure computed directly from the generated graphs is required.

A randomness measure derived from pairwise graph distances is a promising candidate. Many graph distances have been proposed in literature [24], but we find that most of them do not satisfy all properties that we believe a distance measure should have for the task considered here. These are (i) a clear interpretation, (ii) efficiently computable for many pairs of graphs, and (iii) suitable for graphs with similar degree distributions and graphlet frequencies. Moreover, we consider only unlabelled, undirected graphs, which rules out all distances that require known node-correspondence (e.g. [25, 26]). Therefore, we propose four new distance measures, each based on how the small structures shared between graphs (i.e. nodes of a certain degree and graphlet instances), are connected to form the entire graph.

The remainder of this paper is organized as follows. The next section provides a brief overview of related work on measuring randomness in sets of graphs and graph distances. Then we describe our new graph distances and show how to compute a measure of randomness from them. In the experimental section, we first compare our graph distances to each other and to some popular distance measures from literature. Then we apply our randomness measure to evaluate several graph generation methods. We conclude by summarizing the results of our work.

2 Related work

Randomness. Most works on measuring randomness or complexity of graphs consider the *intrinsic* randomness of a single graph. That is not the aim of our paper. Instead, we want to measure how much randomness is present in a set of graphs, without drawing conclusions about how complex any individual graph in the set is. This was also the topic of a recent paper by Velikonitvsev et al. [27]. Just as in the current work, a randomness measure is derived from pairwise graph distances. However, their paper focuses on generating diverse graphs, while little attention is paid to which graph distance should be used in the randomness measure. In that sense, our work fulfills a complementary role.

A few more theoretical measures related to randomness have been proposed in literature as well. Bianconi introduced the entropy of ensembles of random graphs [28], which is computed from the total number of graphs in the ensemble. Estimates of entropy are given for some simple ensembles based on methods from statistical mechanics, but the same computation methods cannot be applied to graphs with more complicated constraints (e.g. constraints on graphlet frequencies). Moreover, we consider sets of graphs generated by heuristic algorithms, which do not necessarily correspond to well-defined ensembles. Fujita et al. study correlations between random graphs [29], which the authors link to correlations between the parameters of the corresponding random graph model (e.g. the probability of an edge in the Erdős-Rényi model [30]). They tested several graph features and found that the spectral radius of the graph spectrum is most strongly correlated to the parameters

of the models. While nothing prevents us from using spectral radii to identify differences and similarities between graphs, the link to the parameters of the generation algorithm cannot be made. Therefore, this would be a rather arbitrary choice.

Graph distances. The literature on graph distances is abundant [24, 31]. We provide a brief overview of different approaches followed to define graph distances, without striving for completeness.

Perhaps the most straightforward distance measure for graphs is the *graph edit distance* [32]. It is defined as the minimum number of elementary operations (e.g. edge additions or removals) needed to transform one graph into another. The main downside of the graph edit distance is that computing it exactly requires finding an optimal alignment between the two graphs, which is an NP-hard problem. Moreover, even in networks without edge attributes, some edges may be considered more important than others. For instance, assume a network consists of two dense clusters, with very few edges connecting the clusters. Removing one of the edges between the two clusters is likely to have a bigger impact than removing an edge within a cluster. Nevertheless, both operations result in a network with an edit distance of one compared to the original network.

Many distance measures, including the ones proposed in this work, are computed from a feature vector description of the graphs [33, 34]. These features can be derived from node degrees, node centrality, shortest paths, subgraph counts, etc. Such measures are generally not metrics, because the feature vectors of two graphs can be the same even though the graphs are not isomorphic. This is inevitable if you want to avoid distance measures that are at least as hard to compute as graph isomorphism testing, and it is rarely an issue in practice. Features derived from graphlet counts have proven to result in effective distance measures. Popular examples of such distances are *Relative Graphlet Frequency Distance* [11], *Graphlet Degree Distribution Agreement* [12], *Graphlet Correlation Distance* [35] and *GRAFENE* [36]. Two feature-based distance measures, *NetSimile* [33] and *Graphlet Correlation Distance*, will be discussed in more detail in the experimental section.

Another class of distance measures is based on the spectrum of a matrix representation of graphs, usually the adjacency or Laplacian matrix [37–39]. Spectral distances could be classified as feature-based as well, but they are usually considered separately. Graph spectra carry information about the structure of the graph and are therefore suitable for comparing networks. The main drawback of spectral distances is their high sensitivity to small changes in the graph’s structure. Although this may be an advantage in some applications, it becomes problematic when considering real networks subject to noise.

3 Methods

In this section, we propose four new distance measures specifically designed to compare graphs with similar degree sequences and/or graphlet frequencies. Then, we discuss the actual randomness measure. We start by introducing some definitions and notations that will be used throughout the manuscript.

Let $G = (V, E)$ be an undirected, unlabelled graph consisting of a vertex set V with cardinality $|V| = n$, and an edge set E with cardinality $|E| = m$. The degree of node $v \in V$ is denoted by k_v . Graphlets are connected, induced subgraphs of a few (usually ≤ 5) nodes. All graphlets with up to five nodes are illustrated in Fig. 1. The nodes in a certain graphlet topology can have different roles. For example, we can make a distinction between the central node in graphlet G_1 , and the two ‘leaf’ nodes. The leaf nodes can be mapped to each other by an automorphism and are therefore in the same *automorphism orbit*, or *orbit* in short. The frequency of graphlet G_i in a network G , i.e. the number of distinct induced subgraphs of G isomorphic to G_i , is denoted by $C_i^g(G)$. Likewise, the number of times a node v is touched by orbit j , referred to as the j^{th} *graphlet degree* of v [12], is denoted by $C_j^o(v)$. We use the term *graph distance* for any function or algorithm that receives two graphs G and H as input and returns a non-negative real number $d(G, H)$, that measures how different G and H are.

3.1 Graph distance based on node degrees

Our first distance measure is based on the *joint degree matrix* [40]. The joint degree matrix J of a simple, undirected graph with maximum degree $k_{\max}$ is a symmetric $k_{\max} \times k_{\max}$ matrix where entry J_{kl} counts the number of edges between nodes of degree k and nodes of degree l . To the best of our knowledge, the joint degree matrix has not been used to define a graph distance before. Nevertheless, we believe it provides an excellent basis for measuring distance between graphs with similar degree distributions. Our reasoning is as follows. If we assume two non-isomorphic graphs G and H have the exact same degree distribution, then the difference between G and H must lie in other structural properties. Unless there is some motivation from the application field from which G and H originate, there is no reason to favor any specific property over another. However, since we know that the node degrees are the same,

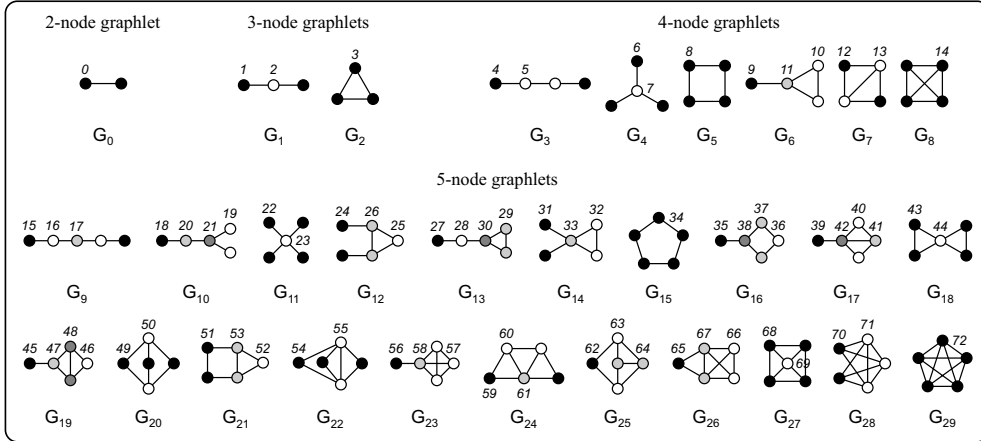


Fig. 1 All graphlets of up to five nodes. Automorphism orbits are labelled from 0 to 72. Within each graphlet, nodes belonging to the same orbit are of the same colour

we argue that the most logical approach is to consider how nodes of certain degrees are connected to each other to form the graph. This information is captured in the joint degree matrix.

We define the *Joint Degree Distance* (JDD) between graphs G and H as the Euclidean distance between the upper triangular parts of their respective joint degree matrices:

$$d_{\text{JD}}(G, H) = \left[\sum_{k=1}^K \sum_{l=k}^K \left(\frac{J_{kl}(G)}{\sum_{i=1}^K \sum_{j=i}^K J_{ij}(G)} - \frac{J_{kl}(H)}{\sum_{i=1}^K \sum_{j=i}^K J_{ij}(H)} \right)^2 \right]^{\frac{1}{2}}, \quad (1)$$

where $K = \max(k_{\max}(G), k_{\max}(H))$ and rows and columns of zeros are added to $J(G)$ or $J(H)$ in case $k_{\max}(G) \neq k_{\max}(H)$. We also divide the joint degree matrices by the number of edges in the graph to reduce the dependency on graph size. However, the JDD is still not perfect in that sense. As an example, consider the distance between graphlets G_4 and G_{11} . These graphs have the same star topology, and, therefore, the distance between them should be small. But it is easy to see that this is not the case. In G_4 , all edges lie between nodes of degree one and a node of degree three. This means only element $(1, 3)$ in the upper triangle part of $J(G_4)$ is different from zero. In G_{11} , the same holds for element $(1, 4)$ instead. The result is that the JDD between G_4 and G_{11} is effectively maximal. We have experimented with alternative versions of the JDD, in which rows and columns of zeros were removed from the joint degree matrix or the degrees were sorted based on frequency of occurrence, but no version consistently outperformed the base definition.

3.2 Graph distances based on graphlets

Next, we present several distance measures based on graphlets. Our starting point is the same as before: since we assume the graphs we compare have similar graphlet frequencies, we look at how graphlet instances ‘are connected’ (overlap) to form the graph. The three distances discussed below measure the degree of overlap in different ways. In the following definitions, we use N_g for the number of graphlets used in the distance measure (e.g., $N_g = 8$ if all three- and four-node graphlets are used).

3.2.1 Reduced Node Number Distance

The idea behind the *Reduced Node Number Distance* (RNND) is to measure whether instances of the same graphlet tend to cluster together, or are spread across the graph. The frequency of graphlet G_i in G provides an upper bound for the number of nodes in G that can touch an instance of G_i . This upper bound is given by $n_i C_i^g(G)$, where n_i is the number of nodes in G_i , and is only reached if separate instances of G_i do not have any nodes in common. However, in large networks, a single node may be part of tens or even hundreds of different graphlet instances. This distinction is measured by a new feature, which we call the *Reduced Node Number* (RNN) and from which the RNND is derived.

As an illustration, consider a graph with the topology of G_{23} . This graph contains four instances of graphlet G_2 . Forming four completely separate instances of G_2 would

require 12 nodes, but due to the specific graph structure, only four nodes are needed to form all four instances (the node of degree one does not touch a G_2). We say that the RNN of G_2 in this graph is $\frac{4}{12} = \frac{1}{3}$.

In general, for a graphlet G_i in graph G , we define $RNN_i(G)$ as

$$RNN_i(G) = \frac{|\{v \in V_G | C_i^g(v) \geq 1\}|}{n_i C_i^g(G)}, \quad (2)$$

where $C_i^g(v) \geq 1$ indicates that node v is part of at least one instance of G_i . If a graphlet does not appear at all in the graph, its RNN is zero. The RNN is hence a value between zero and one. The maximum is reached for a graphlet with purely non-overlapping instances, while low values indicate strong clustering. Our RNN measure is calculated in a straightforward way from the vectors formed by the RNNs of all considered graphlets:

$$d_{\text{RNN}}(G, H) = \left[\sum_{i=1}^{N_g} (RNN_i(G) - RNN_i(H))^2 \right]^{\frac{1}{2}}. \quad (3)$$

The RNN is sensitive to the degree to which instances of the same graphlet type tend to overlap, but it does not contain any information about how different graphlet types are connected. Furthermore, it does not measure how far apart non-overlapping graphlet instances or clusters of instances typically are. The subsequent distances were designed to overcome these limitations.

3.2.2 Joint Graphlet Distance

Whereas the RNN only considers overlapping instances of the same graphlet type, the *Joint Graphlet Distance* (JGD) measures overlap between any combination of graphlet types. We first introduce the *joint graphlet matrix* JG , from which the JGD is derived. We use this term because we believe this concept is the most natural extension of the joint degree matrix to graphlets. The joint graphlet matrix is an $N_g \times N_g$ matrix where entry JG_{ij} counts how many times, on average, a node touches graphlet G_j , for each time it touches graphlet G_i :

$$JG_{ij}(G) = \frac{1}{C_j^g(G)} \frac{\sum_{v \in V} C_i^g(v) C_j^g(v)}{\sum_{v \in V} C_i^g(v)}. \quad (4)$$

Note that $C_i^g(v)$ is the number of times node v is touched by graphlet G_i . If G_i or G_j does not appear in G , $JG_{ij}(G)$ is zero by definition. We found that making a distinction between automorphism orbits does not in general lead to better results. The division by $C_j^g(G)$ only serves as a normalization of the matrix elements. The combination of terms in the second factor is best explained by example. Consider a small graph with the topology of G_8 . In terms of three-node graphlets, this graph contains two wedges (G_1), with the leftmost and central node (according to the orientation in Fig. 1) touching both wedges, and one triangle (G_2). Ignoring the normalization

factor, matrix element $JG_{1,2}$ is equal to $\frac{4}{6}$. The interpretation is straightforward: for each wedge, two of the three nodes touch the triangle. The matrix is not symmetric, as $JG_{2,1} = \frac{4}{3} \neq JG_{1,2}$. The value of this element also makes perfect sense, as two of the three nodes within the triangle touch one wedge, while the third node touches both wedges.

The JGD is once again defined as the Euclidean distance between the joint graphlet matrices:

$$d_{\text{JG}}(G, H) = \left[\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (JG_{ij}(G) - JG_{ij}(H))^2 \right]^{\frac{1}{2}}. \quad (5)$$

3.2.3 Radial Graphlet Distribution Distance

Our final distance measure is derived from another new graph feature, which we call *Radial Graphlet Distribution* (RGD), and improves upon the RNND by taking into account the full distribution of graphlet instances across the graph. The RGD of orbit i shows how instances of orbit i are distributed with respect to a fixed node touched at least once by orbit i , averaged over all relevant nodes. The exact form and the name of this measure are inspired by radial distribution functions in physics and chemistry, which describe density variations as a function of distance from a reference particle. Similarly, RGD describes variations in orbit “density” as a function of hop distance from a reference node.

Algorithm 1 describes how this feature is computed. We assume the input graph is connected here, although it is also possible to include nodes at infinite distance in the RGD vector. Element d in the rgd vector of a single node n_1 contains which fraction of nodes touched by orbit i , is at hop distance d from n_1 . If a certain orbit is homogeneously distributed over the graph, its RGD will be a broad distribution over $[0, D]$, with D the diameter of the graph. On the other hand, if a generally sparse graph contains a single cluster of high density, the RGD of, for example, orbit 14 will contain a peak at low hop distance.

We define the Radial Graphlet Distribution Distance (RGDD) as

$$d_{\text{RGD}}(G, H) = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \left[\sum_{r=0}^{D_{\max}} (RGD_i^r(G) - RGD_i^r(H))^2 \right]^{\frac{1}{2}}, \quad (6)$$

where $\mathcal{N}$ is a set of orbits and $D_{\max}$ is the maximum hop distance in $RGD_i(G)$ and $RGD_i(H)$. Through the use of orbit statistics and hop distances, the RGDD combines locally determined node roles with a global view of the graph.

In [35], Yaveroğlu et al. noted that orbit counts are not all independent. Instead, orbit counts from large graphlets can be linked to orbit counts from smaller ones through combinatorial equations. These redundancies introduce noise, and may result in less sensitive distance measures. Based on the redundancy equations, Yaveroğlu et al. selected a set of 11 non-redundant orbits for up to four node graphlets. They found that their Graphlet Correlation Distance (GCD) performs best when only the non-redundant orbits are included, instead of all 15. This version even outperformed

Algorithm 1 Radial Graphlet Distribution

Input: Connected graph $G = (V, E)$, orbit index i

Output: RGD of orbit i

```
1:  $\mathbf{C}_i^o \leftarrow$  dictionary of orbit counts  $C_i^o(v), \forall v \in V$ 
2:  $total \leftarrow \sum_{v \in V} \mathbf{C}_i^o[v]$ 
3:  $V_i \leftarrow$  all nodes  $v \in V$  for which  $\mathbf{C}_i^o[v] \neq 0$ 
4:  $RGDs \leftarrow$  empty list
5: for  $n_1 \in V_i$  do
6:    $rgd \leftarrow$  empty vector of size  $\text{diameter}(G)+1$ 
7:   for  $n_2 \in V_i$  do
8:      $d \leftarrow$  hop distance between  $n_1$  and  $n_2$ 
9:      $rgd[d] \leftarrow rgd[d] + \mathbf{C}_i^o[n_2]/total$ 
10:  end for
11:  Store  $rgd$  in  $RGDs$ 
12: end for
13:  $RGD \leftarrow \text{avg}(RGDs)$ 
14: return  $RGD$ 
```

the GCD with all 73 orbits for up to five-node graphlets. We have compared a version of RGDD that uses their set of non-redundant orbits to versions with all 15 or 73 orbits in, respectively, all four-node and five-node graphlets. We also concluded that the version with non-redundant orbits generally performs best in classification experiments (Section 4.3), although the differences are small. Concretely, we use the set $\mathcal{N} = \{0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11\}$ in the experiments in the next section, where the numbers refer to the labels in Fig. 1. Other sets of non-redundant orbits are possible, but, as in [35], we observed that the choice of orbits does not have a notable impact.

3.3 Computational complexity

The JDD is very easy to compute. Node degrees of each node can be determined in $O(m + n)$ time, while filling the joint degree matrix takes only $O(m)$. For the graphlet-based distances, computing graphlet counts of all size $\leq g$ graphlets requires $O(nk_{\max}^{g-1})$ time. The additional operations in the RNND and JGD take $O(nN_g)$ time. Unless sparse graphs (small $k_{\max}$) and very large graphlets (large N_g) are considered, counting graphlets is the most time-consuming step. The RGDD additionally requires the shortest path lengths between all pairs of nodes. This can be computed in $O(nm + n^2)$ time by performing a breadth-first search from every node, since we consider unweighted graphs. Once all shortest path lengths are known, the radial graphlet distribution can be computed in $O(n^2)$ time (see Algorithm 1). Therefore, the total complexity of the RGDD is $O(nm + n^2 + nk_{\max}^{g-1})$. The first two terms might be dominating on sparse graphs, although graphlet counting will often still be the slowest step in practice.

3.4 Measuring randomness

We define randomness of a set of graphs as the degree to which the set is similar to graphs uniformly sampled from the full set of admissible graphs. As an example, if we would consider graphs with a fixed number of nodes and edges, graphs obtained from the Erdős-Rényi model [30] would be maximally random. In contrast, a (multi)set containing only isomorphic graphs would show no randomness at all. The downside of using any measure derived solely from pairwise graph distances, is that we cannot measure randomness as such directly. What is actually measured, is better referred to as *diversity*, the degree to which the set covers the space of admissible graphs. These are two distinct concepts. However, we argue that high diversity is generally accompanied by high randomness in the application we envision: the evaluation of random graph generators. Low randomness would be the result of an unwanted bias introduced by, e.g., a heuristic generation method. It is highly unlikely that this bias would simultaneously cause more diversity. Instead, the bias would lead to some subspace of the full space of admissible graphs being sampled exclusively, and any acceptable graph distance would therefore have to measure low diversity. It is, of course, possible to construct a random graph generator that artificially increases diversity by introducing additional bias, resulting in less randomness. While our randomness measure would lead to incorrect conclusions in this case, maximizing diversity (at the expense of randomness) would clearly be an objective then, and there would be no reason to evaluate the true randomness anymore.

The best way to measure randomness based on graph distances would be to compare the full distance distribution to a reference distribution, obtained from an unbiased model. However, this reference distribution is only available for graphs with very basic constraints, and we must resort to measures that do not require a reference. Therefore, to quantify the amount of randomness in a set $\mathcal{S}$ containing N graphs, we use the average pairwise distance:

$$\text{Randomness}(\mathcal{S}; d) = \frac{2}{N(N-1)} \sum_{\substack{G, H \in \mathcal{S} \\ G \neq H}} d(G, H). \quad (7)$$

While some advanced measures have been shown to perform better in specific applications [41, 42], the average is non-parametric and relatively easy to compute. Other popular, suitable options are derived from maximum or minimum distances (either in the whole set or between one graph and all others) [27], but we want to avoid measures that could be dominated by outliers.

In recent work, Mironov and Prokhorenkova formulated three properties a diversity (randomness) measure should have: monotonicity, uniqueness and continuity [43]. The average does not have the uniqueness property. However, no popular measure satisfies all three properties. While the authors present two new measures that do have all desired properties, computing these measures is NP-hard.

4 Results

In the following, we compare our distance measures to each other and to two existing graph distances in several experiments. Finally, we evaluate three graph generative models in terms of randomness. For the graphlet-based distance measures, we consider graphlets of three and four nodes. Going beyond four-node graphlets increases the computational cost of the distance measures significantly, while their performance does not improve notably. This is because larger graphlets may also introduce noise, as mentioned in the previous section.

Data. For several experiments, we use graphs generated from a target network. The target networks are a combination of synthetic and real-world networks:

- **BA:** A Barabási–Albert random graph [1] with 100 nodes and 300 edges. We used a small ($n = 10$, $m = 30$) Erdős–Rényi random graph as the seed.
- **GEO:** A random geometric graph [44] with 200 nodes and 500 edges. Nodes were placed at random in a three-dimensional unit cube.
- **E. coli:** A network of 682 interactions between 341 proteins in *Escherichia coli*, downloaded from the IntAct database [45].
- **H. pylori:** A network of 1465 interactions between 714 proteins in *Helicobacter pylori*, downloaded from the MINT database [46].
- **Windsurfers:** A network of interpersonal contacts between windsurfers in Moreno Beach, California, during the fall of 1986 [47]. The network contains 336 interactions between 43 people. It was obtained from KONECT [48].
- **NetScience:** A network of co-authorships between scientists in the field of network science [49]. The network contains 914 collaborations between 379 scientists. It was obtained from NetworkRepository [50].

Baseline graph distances. We compare our distance measures to two graph distances from literature. We provide a brief explanation here, and refer the reader to the appropriate references for more information.

- *Graphlet Correlation Distance* [35]: The GCD considers correlations between the frequencies of different orbits. As mentioned previously, some orbits are redundant, and it was found that the best performing version of GCD uses only 11 orbits. For an n -node graph, the orbit counts for all nodes are combined in an $n \times 11$ matrix. Spearman’s correlation coefficient is computed between all column pairs to obtain the 11×11 *Graphlet Correlation Matrix*. The distance between two graphs is finally defined as the Euclidean distance between the upper triangular parts of their Graphlet Correlation Matrices. Spearman’s correlation coefficient can be computed in $O(n \log(n))$ time. Therefore, the time complexity is also dominated by the complexity of counting graphlets. The GCD has been shown to outperform many other distance measures, including several graphlet-based ones. However, to the best of our knowledge, it has never been tested in the specific scenario where graphs have similar graphlet frequencies.

- *NetSimile* [33]: NetSimile was in the first place proposed as a graph descriptor, but a distance is readily obtained from these descriptors. We will refer to the distance measure, rather than the descriptor, as NetSimile. First, a combination of seven local and egonet-based features (e.g. degree, local clustering coefficient, size of egonet, etc.) is computed for each node. Then, a feature vector for the entire graph is obtained by computing the median, mean, standard deviation, skewness, and kurtosis of the node-specific features. The authors propose to use the Canberra distance between these feature vectors as a graph distance. The local, node-specific features can be computed in $O(nk_{\max}^2)$ time. Computing the median requires $O(n\log(n))$ time, while the other aggregate statistics take only $O(n)$. We include NetSimile in our experiments because it is a prototypical feature-based method, that does not use graphlets.

The experiments have been performed on a machine with a 10-core (1.8 GHz) Intel Core i7-1365U CPU and 16 GB of RAM, running Windows 11 Pro. All distances have been implemented in Python.

4.1 Watts-Strogatz random graphs

In our first experiment, we consider the Watts-Strogatz (WS) random graph model [2]. This model has the interesting property that it contains a parameter p with which the degree of disorder in the graph can be tuned. A WS graph with $p = 0$ has the structure of a ring lattice, where every node is connected to its $\frac{2m}{n}$ nearest neighbours. For $p > 0$, this parameter indicates the probability of rewiring each edge in the graph. As p increases, the rewired graph retains fewer and fewer structural properties of the regular ring lattice. Complete disorder is obtained for $p = 1$. Of course, starting from any other highly regular graph, we could construct models that are equally useful for this experiment, but the WS model is well established in the graph modeling community.

Since p reflects the degree of randomness in a WS graph, the ability of a distance measure to distinguish clearly between graphs with varying p values provides an initial indication of its suitability for capturing randomness. Therefore, we compared WS graphs with $n = 1000$, $m = 10,000$, and 20 p values distributed log-uniformly over $[10^{-4}, 1]$, to a WS graph of the same size and with $p = 0$. The results are shown in Fig. 2. Two important properties of WS graphs are the clustering coefficient C and the average shortest path length L . Both are often studied as a function of p [2]. Hence, the quantities $C(0) - C(p)$ and $L(0) - L(p)$ form two ‘natural’ distances for WS graphs. We also show a rescaled version of these in Fig. 2 as reference curves for our other distance measures.

The first thing to note is that the shape of the JDD curve is very similar to the curves of the clustering coefficient and shortest path length. The RGDD shows a more gradual increase over the entire p interval, although there is a slight decrease near $p = 1$. Intuitively, we expect the distance to be a monotonically increasing function of p . The distances based on the clustering coefficient and shortest path length confirm that our intuition is correct. Since RNND and JGD show the exact opposite trend over most of the p interval, these distances are less useful, at least in this specific context.

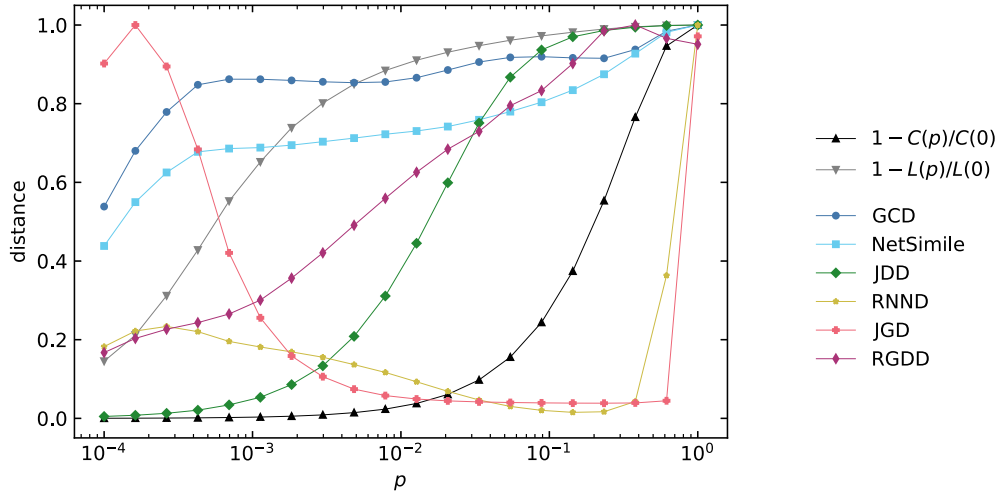


Fig. 2 Distance from WS graphs ($n = 1000$, $m = 10,000$) with varying p to the WS graph of the same size with $p = 0$. Curves are rescaled such that the maximum distance is one. C is the local clustering coefficient averaged over all nodes, and L is the average shortest path length. Each point is an average over 1000 graphs

Interestingly, NetSimile and GCD result in similar curves, despite being based on very different features. Finally, note that the GCD does not vary much over a large interval and decreases slightly between $p = 10^{-3}$ and $p = 10^{-2}$, indicating that GCD is perhaps also not perfect for measuring randomness.

4.2 dk -random graphs

The dk -series, introduced by Mahadevan et al. [51], is a series of probability distributions designed to systematically study the properties of a network. The dk -distribution of a graph specifies all degree correlations within subgraphs of size d . By definition, the $0k$ -‘distribution’ is the average degree of the graph. The $1k$ -distribution is the well-known degree distribution, and the $2k$ -distribution is the joint degree distribution. For $d \geq 3$, dk -distributions contain the frequencies of subgraphs of size d composed of nodes of degrees $k_1, k_2, k_3, \dots$, for all combinations of node degrees. Note that the set of $3k$ -random graphs belonging to a graph G (i.e. all graphs with the same $3k$ -distribution as G), is therefore only a small subset of the graphs with the same frequencies of three-node graphlets as G .

The dk -series is inclusive, meaning that the $(d+1)k$ -distribution contains the same information about a graph as the dk -distribution, plus extra. This is easy to see for the average degree ($0k$), which is completely fixed by the degree distribution ($1k$). A direct consequence of inclusiveness is that the set of dk -random graphs always contains more randomness than the set of $(d+1)k$ -random graphs, because $(d+1)k$ -random graphs have strictly stronger constraints. Therefore, dk -random graphs have the useful property that we know in advance that the randomness should be a decreasing function of d . For this reason, they form an excellent test case for our randomness

and distance measures. To the best of our knowledge, the only available algorithms to uniformly sample dk -random graphs with $d > 2$ are the Markov Chain Monte Carlo methods by Tabourier et al. [17, 19] and Tao [18]. However, the computation time required to obtain just one unbiased sample in a strongly constrained graph ensemble is impractically long. Therefore, we restrict our experiment to $0k$ -, $1k$ -, and $2k$ -random graphs.

We generated 500 dk -random graphs, with $d \in \{0, 1, 2\}$, for each of the target networks discussed at the beginning of this section. This was done by randomizing the original network with a suitable edge swapping method, for which we used the RandNetGen software provided by Colomer-de-Simón et al. [52]. Then, we measured the randomness with the six considered distance measures. The results are shown in Fig. 3.

At first glance, the JDD again shows nice results, clearly distinguishing between $0k$ -, $1k$ - and $2k$ -random graphs. However, there is a caveat. The JDD between graphs with shared $2k$ -distribution is trivially equal to zero, because all those graphs have the same joint degree matrix. This means that the JDD does not provide any useful information for dk -random graphs with $d \geq 2$.

Overall, we see many shared trends with the results of the experiment on WS graphs. Randomness measured with the RNND and JGD does not result in the correct decreasing curve on several networks, and produces randomness values greater than one in multiple cases. Since $0k$ -random graphs, with Randomness = 1 by definition, are supposed to be maximally random, these results indicate that RNND and JGD are not suited for measuring randomness. The GCD, NetSimile, and RGDD generally produce

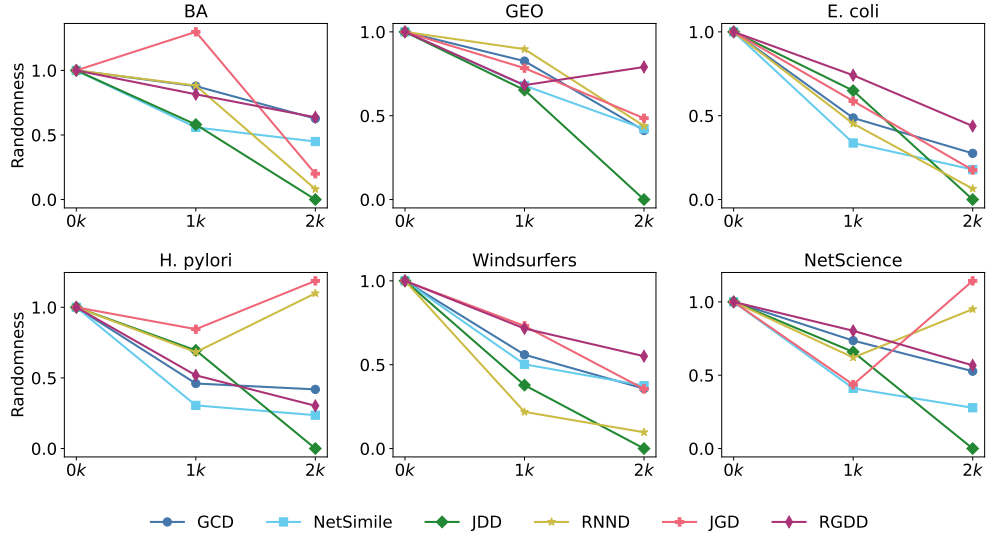


Fig. 3 Randomness in a set of 500 dk -random graphs, measured using our six graph distances. Data from the same distance measure is rescaled such that $0k$ -random graphs have Randomness = 1. The $0k$ -, $1k$ - and $2k$ -random graphs have the same average degree, same degree distribution, and same joint degree distribution as the target network, respectively

the expected outcome. Remarkably, the RGDD measures a higher randomness in $2k$ -random graphs compared to $1k$ -random graphs in the experiment on the geometric network. We have repeated the experiment for several random geometric graphs and consistently reproduced this result. It is unclear why the RGDD fails in this specific case.

4.3 Classification of generated graphs

As our last test, we consider a typical experiment for evaluating the quality of graph distance measures. An effective graph distance should assign a larger distance to a pair of graphs generated by two different models, than to a pair generated by the same model. As such, the graph distance can identify graphs coming from the same model. Even though this experiment does not have a direct link to randomness, it still provides an objective evaluation of the general performance of our graph distances.

For this purpose, we use the Precision-Recall curve [53]. Graph pairs generated by the same model define the *true* set, while graph pairs coming from different models define the *false* set. Given a distance $d(G, H)$ and a threshold ϵ , the graphs G and H are classified as *positive* (i.e. coming from the same model) if $d(G, H) < \epsilon$, otherwise this is a *negative* pair. *True positive* and *true negative* pairs are correctly classified as originating from the same and different models, respectively. Likewise, *false positive* and *false negative* pairs are incorrectly classified. Based on these definitions, Precision and Recall are given by $P = TP / (TP + FP)$ and $R = TP / (TP + FN)$, where TP , FP , and FN are the number of *true positive*, *false positive*, and *false negative* pairs. The Precision-Recall curve plots the Precision versus the Recall for the threshold ϵ varied between the minimum and maximum distance over all graph pairs, where each point on the curve corresponds to one value of ϵ . The *Area Under the Precision-Recall curve* (AUPR) provides a quantitative measure of the classification. It is a value between 0 and 1, where $AUPR = 1$ indicates a perfect classification.

Most graph classification experiments combine models that generate very different graphs (e.g. ER, BA, random geometric, ...) [35]. However, it is more suitable in this work to perform the experiment on multiple models that generate graphs with similar degree distribution and graphlet frequencies, as our graph distances were tailored to these types of graphs. The classification task will obviously be more challenging, because all models are intended to generate graphs with some shared structure. But since the models do not sample uniformly from the space of admissible graphs, there will likely still be smaller differences between graphs coming from the same model, than between graphs coming from different models. We will see below that this is indeed the case.

In our experiment, we have included two heuristic algorithms and one machine learning model. The heuristic methods were proposed in a previous work [21], where we also compared them to the same machine learning model in terms of computation speed and quality of generated graphs (i.e. how well the degree distribution and graphlet frequencies match those of the target network). We give a very short explanation of each model below.

- GRAIP: The GRAPhlet-based Incremental generator for Probabilistic networks [21], GRAIP in short, derives bounds on the degree distribution and graphlet counts from the uncertainty on edges in a probabilistic network, and incrementally generates graphs of which the properties lie within these bounds.
- SwapCon: This is an extension of the well-known edge swapping method [7, 20] to probabilistic networks [21]. The configuration model [54] is used to generate an initial graph with the correct degree distribution, again obtained from a probabilistic network, and pairs of edges are swapped until the graphlet counts are close enough to the target.
- GraphGen: This is a graph generative machine learning model proposed by Goyal et al. [22]. The authors showed that GraphGen is successful in reproducing the orbit counts in training graphs. Although the model was designed for labelled graphs, it still works well for unlabelled graphs. Moreover, node degrees can be used as labels to facilitate training.

Both GRAIP and SwapCon use probabilistic networks as input, and GraphGen requires a training set of sufficient size. The *E. coli* and *H. pylori* networks already contain uncertainty information (which we ignored in the previous experiments). We manually added a uniform edge existence probability of 0.7 to the Windsurfers and

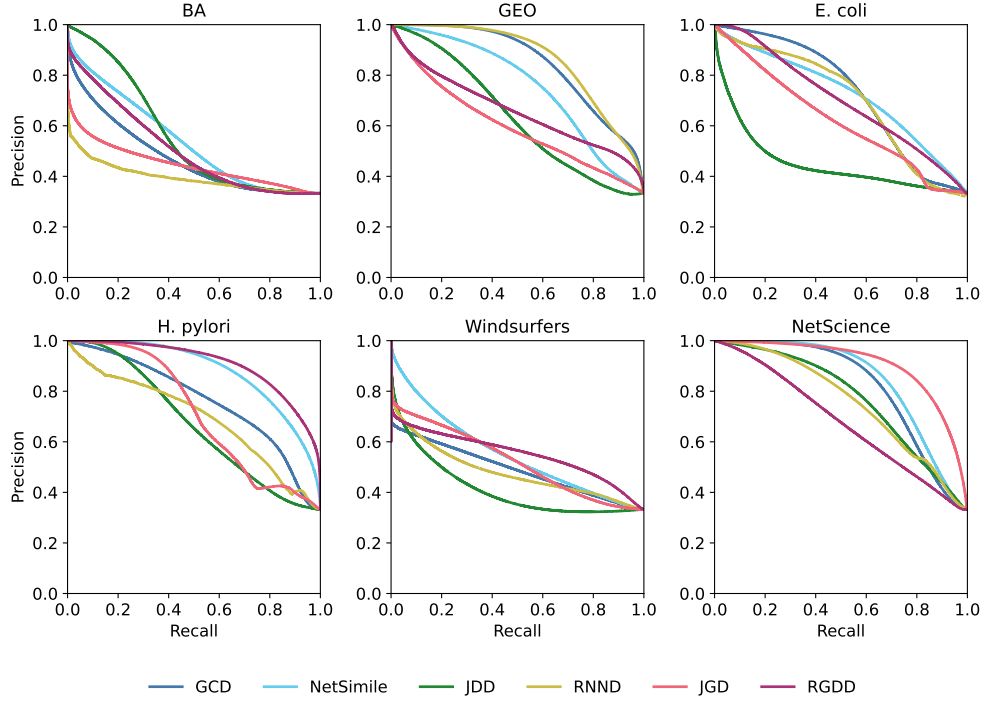


Fig. 4 Precision-Recall curves obtained by classifying 3000 graphs generated by three different models with six distance measures

NetScience networks. Property bounds for GRAIP and SwapCon were obtained by sampling 10,000 graphs from the probabilistic networks. Those sampled graphs were also used as training sets for GraphGen. For the synthetic networks (BA and GEO), we do not add uncertainty, but instead generate 10,000 random graphs with the same parameters and calculate the average properties. Finally, we generated 1000 graphs per target network with each model. Hence, we included a total of 3000 graphs in each classification experiment.

The Precision-Recall curves obtained by our four new distance measures, GCD, and NetSimile, are shown in Fig. 4. The accompanying AUPR values are compared in Fig. 5. First, we see that all distance measures perform better than a fully random classifier ($\text{AUPR} = 0.33$), even when all graphs have a similar structure. There is no distance that is clearly better or worse than all others in every classification experiment. The GCD, NetSimile and RGDD show good performance overall. NetSimile, in particular, is among the best distance measures in all six experiments. The combination of node- and egonet-based features is clearly effective in a classification task, and even seems more reliable than graphlet-based features. We also note that, despite its simplicity, the JDD still outperformed all other distances in the BA experiment.

Interestingly, each graph distance obtained the highest AUPR score in one out of six experiments. This indicates how difficult it is to rank graph distance measures in terms of ‘quality’. Even in a shared experimental setting, the ranking can change entirely depending on the exact topology of the considered networks.

4.4 Evaluation of randomness

We evaluate the models from the previous experiment in terms of randomness. We use the same sets of 1000 graphs per model. As distance measure, we use the RGDD.

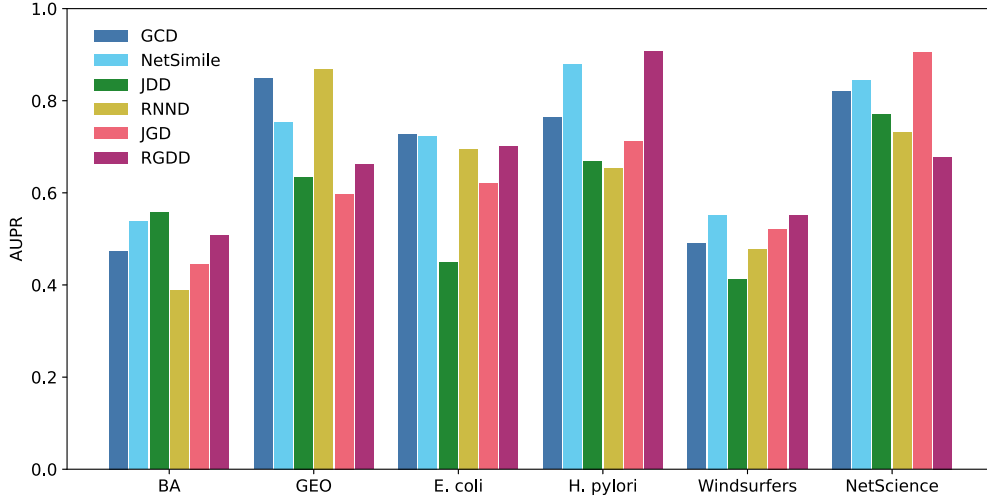


Fig. 5 Comparison of the Area Under the Precision-Recall curve (AUPR) quality metric obtained by the six distance measures, in each of the classification experiments

The JDD, GCD and NetSimile result in the same general trends. The RNND and JGD do not reproduce these trends, but we already saw that these distances cannot be relied upon for measuring randomness. The results are shown in Fig. 6.

We see that GraphGen and SwapCon generally produce graph sets with higher randomness compared to GRAIP. One might have expected low randomness in GraphGen graphs, because this model does not focus on any specific property, but rather tries to reproduce the general structure of training graphs. However, GraphGen does not impose any strict constraints on graph properties. As a result, we also saw much more variety in size in graphs generated by GraphGen. The only exception is the windsurfers network, where the randomness is notably smaller. This is likely because GraphGen simply works ‘better’ (in terms of reproducing topologies in the training set) on smaller networks, which we also observed in previous work [21].

Randomization through edge swapping is often criticized because it does not sample the space of admissible graphs uniformly, and may not be able to escape certain structures in the initial graph within reasonable time [55]. Despite that, we see that the SwapCon method still produces highly random sets of graphs. It is important to note, however, that SwapCon does not use edge swaps to randomize a network, but rather to bring graphlet frequencies of a random graph closer to those found in the target network. Traditionally, the initial random graph is obtained by randomizing the real network [7], but this approach is less suitable for probabilistic networks. Instead, the stub matching method is used to generate a completely new initial graph. This approach likely benefits randomness as well, although we did not have randomness in mind when designing it.

GRAIP results in the lowest randomness in all six experiments. There are two reasons for this. First of all, GRAIP imposes the strongest constraints on generated graphs, with strict bounds on graphlet frequencies. Second, GRAIP generates graphs

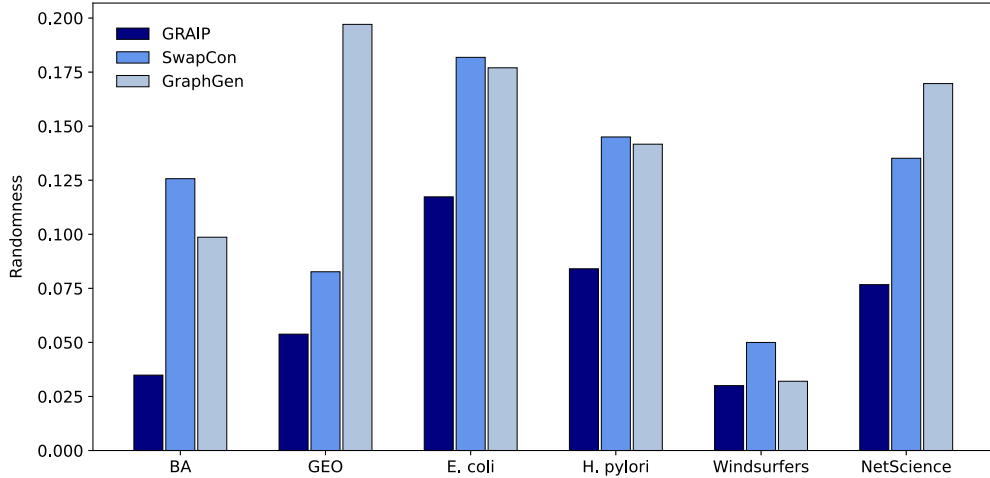


Fig. 6 Randomness in a set of 1000 graphs generated by GRAIP, SwapCon and GraphGen, measured using the Radial Graphlet Distribution Distance

incrementally by making small adjustments to a seed graph in each step. We terminated this incremental generation process as soon as we obtained a graph with graphlet frequencies within the bounds imposed by the target network. This approach works well for obtaining high-quality graphs quickly, but it is probably not optimal if high randomness is also required. This is something to keep in mind for practitioners using GRAIP.

4.5 Runtime analysis

Finally, we report practical computation times of the proposed distances and analyze the scaling behavior of their running times. This is included to provide empirical context for the theoretical complexity discussed in the Methods section. Note that the implementations are not optimized for computational speed.

All runtime measurements were obtained by computing the distance between pairs of Erdős-Rényi (ER) random graphs, in two complementary settings. First, we fix the number of edges per node at five ($m = 5n$) and increase the number of nodes n . Second, we fix the number of nodes at $n = 200$ and increase the number of edges. The mean and standard deviation over 100 independent runs are shown in Fig. 7.

The observed trends are consistent with the theoretical analysis. The RNND and JGD exhibit nearly identical running times, as graphlet counting constitutes the dominant computational cost for both methods. The RGDD is more expensive to compute on low density graphs, although graphlet counting likewise becomes the dominant cost as graph density increases. In contrast, the JDD has substantially lower absolute running times and exhibits a weaker dependence on graph density.

5 Conclusion

To generate graphs with complex constraints, such as bounds on graphlet frequencies, one must resort to heuristic algorithms or machine learning methods. These methods

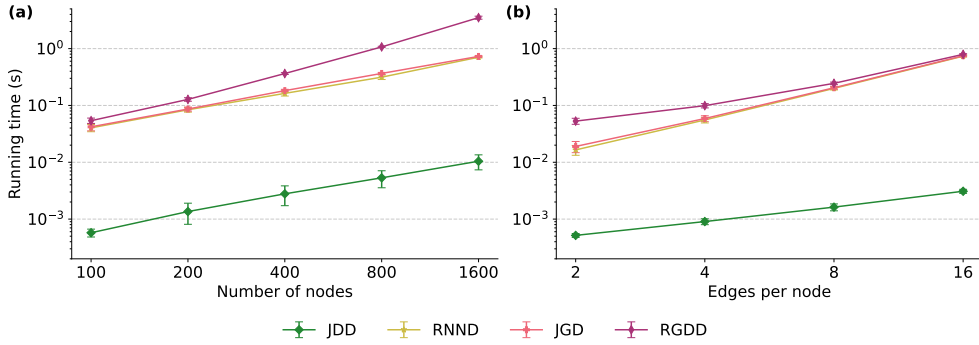


Fig. 7 Runtime comparison of the proposed graph distances. **(a)** Scaling with number of nodes at fixed number of edges per node ($m = 5n$). **(b)** Scaling with edge density for fixed $n = 200$. Points show mean runtime over 100 runs, where a run is a distance computation between a pair of ER graphs. Error bars indicate the standard deviation.

do not sample the space of admissible graphs uniformly. Yet the question of how random the generated graphs are, is relevant in several applications. For that reason, we have studied the challenging problem of quantifying randomness in a set of graphs. We have proposed to use the average pairwise graph distance as a measure of randomness, and we have presented four novel graph distances. We have then evaluated these distances in a series of experiments. Our Reduced Node Number Distance and Joint Graphlet Distance do not seem particularly useful. Neither managed to consistently recover the correct trend in experiments where it is known in advance which set of graphs is most random. The Radial Graphlet Distribution Distance showed much better results, and is comparable to the popular and highly effective Graphlet Correlation Distance in classification experiments. The Joint Degree Distance is much easier to compute than graphlet-based distances, but still has excellent performance in experiments on randomness. It is also not significantly worse than the other distances for graph classification, and even outperformed its competitors in one experiment. Its overall decent performance, combined with the ease of use, makes it a useful distance for very large networks. Although our work primarily focused on graphs with bounded node degrees and graphlet statistics, we expect comparable results in other scenarios.

Our experiments also indicate that the optimal graph distance can depend heavily on the target network(s), even for a specific task like measuring randomness. In fact, for many applications it is impossible to unambiguously define optimality in the first place. There will often be more than one feasible and meaningful distance measure. We have provided several experiments to test if a distance is feasible for measuring randomness. We encourage practitioners to test different distance measures on the type of networks being studied, and to consider using a combination of multiple distances. We also emphasize that measuring randomness should be just one aspect of evaluating graphs generated by heuristic methods. It is at least as important to check the quality of the generated graphs, although this is beyond the scope of the current work. On a final note, each proposed graph distance is based on a graph feature that could serve as a fingerprint of the network. Therefore, these features might have additional applications beyond measuring distance, for example in machine learning. Exploring these applications is a possible direction for future work.

Declarations

Funding. This work was partly supported by Ghent University: the BOF project “BioGraph” [BOF.24Y.2019.0010.01], and the BOF-BAF projects [bof/baf/4y/2024/01/806 and BOF/STA/202009/039].

Conflict of interest. The authors have no competing interests to declare that are relevant to the content of this article.

Author contribution. B.M. conducted the experiments and wrote the manuscript. B.M., D.C., P.A. and M.P. conceived the research, analyzed the results and reviewed the manuscript.

Data availability. A Python implementation of the methods proposed in this work is publicly available at <https://github.com/bmornie/Randomness-graphs>. The

datasets used in the experiment are also provided, or they can be downloaded from the sources cited in the main text.

References

- [1] Barabási AL, Albert R (1999) Emergence of scaling in random networks. *Science* 286(5439):509–512. <https://doi.org/10.1126/science.286.5439.509>
- [2] Watts DJ, Strogatz SH (1998) Collective dynamics of ‘small-world’ networks. *Nature* 393(6684):440–442. <https://doi.org/10.1038/30918>
- [3] Abbe E (2018) Community detection and stochastic block models: Recent developments. *Journal of Machine Learning Research* 18(177):1–86. URL <http://jmlr.org/papers/v18/16-480.html>
- [4] Pratapa A, Jalihal AP, Law JN, et al (2020) Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nature Methods* 17(2):147–154. <https://doi.org/10.1038/s41592-019-0690-6>
- [5] Pezoulas VC, Zaridis DI, Mylona E, et al (2024) Synthetic data generation methods in healthcare: A review on open-source tools and methods. *Computational and structural biotechnology journal* 23:2892–2910. <https://doi.org/10.1016/j.csbj.2024.07.005>
- [6] Newman ME (2003) The structure and function of complex networks. *SIAM review* 45(2):167–256. <https://doi.org/10.1137/S003614450342480>
- [7] Milo R, Shen-Orr S, Itzkovitz S, et al (2002) Network motifs: simple building blocks of complex networks. *Science* 298(5594):824–827. <https://doi.org/10.1126/science.298.5594.824>
- [8] Del Genio CI, Kim H, Toroczkai Z, et al (2010) Efficient and exact sampling of simple graphs with given arbitrary degree sequence. *PLoS One* 5(4):1–7. <https://doi.org/10.1371/journal.pone.0010012>
- [9] Kim H, Del Genio CI, Bassler KE, et al (2012) Constructing and sampling directed graphs with given degree sequences. *New Journal of Physics* 14(2):023012. <https://doi.org/10.1088/1367-2630/14/2/023012>
- [10] Shen-Orr S, Milo R, Mangan S, et al (2002) Network motifs in the transcriptional regulation network of *Escherichia coli*. *Nature Genetics* 31(1):64–68. <https://doi.org/10.1038/ng881>
- [11] Pržulj N, Corneil DG, Jurisica I (2004) Modeling interactome: scale-free or geometric? *Bioinformatics* 20(18):3508–3515. <https://doi.org/10.1093/bioinformatics/bth436>

- [12] Pržulj N (2007) Biological network comparison using graphlet degree distribution. *Bioinformatics* 23(2):e177–e183. <https://doi.org/10.1093/bioinformatics/btl301>
- [13] Wang P, Lü J, Yu X (2014) Identification of important nodes in directed biological networks: A network motif approach. *PLoS One* 9(8):1–15. <https://doi.org/10.1371/journal.pone.0106132>
- [14] Defoort J, Van de Peer Y, Vermeirssen V (2018) Function, dynamics and evolution of network motif modules in integrated gene regulatory networks of worm and plant. *Nucleic Acids Research* 46(13):6480–6503. <https://doi.org/10.1093/nar/gky468>
- [15] Topirceanu A, Duma A, Udrescu M (2016) Uncovering the fingerprint of online social networks using a network motif based approach. *Computer Communications* 73:167–175. <https://doi.org/10.1016/j.comcom.2015.07.002>, online Social Networks
- [16] Zhang X, Xu L, Xu Z (2022) Influence maximization based on network motifs in mobile social networks. *IEEE Transactions on Network Science and Engineering* 9(4):2353–2363. <https://doi.org/10.1109/TNSE.2022.3163203>
- [17] Tabourier L, Roth C, Cointet JP (2011) Generating constrained random graphs using multiple edge switches. *Journal of Experimental Algorithmics (JEA)* 16:1.1–1.15. <https://doi.org/10.1145/1963190.2063515>
- [18] Tao T (2016) An improved MCMC algorithm for generating random graphs from constrained distributions. *Network Science* 4(1):117–139. <https://doi.org/10.1017/nws.2015.35>
- [19] Tabourier L, Karadayi J (2024) Probabilistic k-swap method for uniform graph generation beyond the configuration model. *Journal of Complex Networks* 12(1):cnae002. <https://doi.org/10.1093/comnet/cnae002>
- [20] Silva ME, Paredes P, Ribeiro P (2017) Network motifs detection using random networks with prescribed subgraph frequencies. In: *Complex Networks VIII*. Springer International Publishing, pp 17–29, https://doi.org/10.1007/978-3-319-54241-6_2
- [21] Mornie B, Colle D, Audenaert P, et al (2025) Generating random graphs with prescribed graphlet frequency bounds derived from probabilistic networks. *PLoS One* 20(8):e0328639. <https://doi.org/10.1371/journal.pone.0328639>
- [22] Goyal N, Jain HV, Ranu S (2020) GraphGen: A scalable approach to domain-agnostic labeled graph generation. In: *Proceedings of The Web Conference 2020*, pp 1253–1263, <https://doi.org/10.1145/3366423.3380201>

- [23] Gamage A, Chien E, Peng J, et al (2020) Multi-motifgan (mmgan): Motif-targeted graph generation and prediction. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp 4182–4186, <https://doi.org/10.1109/ICASSP40776.2020.9053451>
- [24] Tantardini M, Ieva F, Tajoli L, et al (2019) Comparing methods for comparing networks. *Scientific Reports* 9(1):17557. <https://doi.org/10.1038/s41598-019-53708-y>
- [25] Koutra D, Vogelstein JT, Faloutsos C (2013) DeltaCon: A principled massive-graph similarity function. In: *Proceedings of the 2013 SIAM International Conference on Data Mining (SDM)*, pp 162–170, <https://doi.org/10.1137/1.9781611972832.18>
- [26] Liu Q, Dong Z, Wang E (2018) Cut based method for comparing complex networks. *Scientific Reports* 8(1):5134. <https://doi.org/10.1038/s41598-018-21532-5>
- [27] Velikonitvsev F, Mironov M, Prokhorenkova L (2024) Challenges of generating structurally diverse graphs. In: *Advances in Neural Information Processing Systems*, vol 37. Curran Associates, Inc., pp 57993–58022
- [28] Bianconi G (2007) The entropy of randomized network ensembles. *Europhysics Letters* 81(2):28005. <https://doi.org/10.1209/0295-5075/81/28005>
- [29] Fujita A, Takahashi DY, Balardin JB, et al (2017) Correlation between graphs with an application to brain network analysis. *Computational Statistics & Data Analysis* 109:76–92. <https://doi.org/10.1016/j.csda.2016.11.016>
- [30] Erdős P, Rényi A (1959) On random graphs I. *Publicationes Mathematicae Debrecen* 6:290–297. <https://doi.org/10.5486/pmd.1959.6.3-4.12>
- [31] Hartle H, Klein B, McCabe S, et al (2020) Network comparison and the within-ensemble graph distance. *Proceedings of the Royal Society A* 476(2243):20190744. <https://doi.org/10.1098/rspa.2019.0744>
- [32] Gao X, Xiao B, Tao D, et al (2010) A survey of graph edit distance. *Pattern Analysis and Applications* 13:113–129. <https://doi.org/10.1007/s10044-008-0141-y>
- [33] Berlingerio M, Koutra D, Eliassi-Rad T, et al (2012) NetSimile: A scalable approach to size-independent network similarity. *arXiv preprint arXiv:12092684*
- [34] Roy M, Schmid S, Tredan G (2014) Modeling and measuring graph similarity: the case for centrality distance. In: *Proceedings of the 10th ACM International Workshop on Foundations of Mobile Computing*. Association for Computing Machinery, New York, NY, USA, p 47–52, <https://doi.org/10.1145/2634274.2634277>

- [35] Yaveroğlu ÖN, Malod-Dognin N, Davis D, et al (2014) Revealing the hidden language of complex networks. *Scientific Reports* 4(1):4547. <https://doi.org/10.1038/srep04547>
- [36] Faisal FE, Newaz K, Chaney JL, et al (2017) GRAFENE: Graphlet-based alignment-free network approach integrates 3d structural and sequence (residue order) data to improve protein structural comparison. *Scientific Reports* 7(1):14890. <https://doi.org/10.1038/s41598-017-14411-y>
- [37] Wilson RC, Zhu P (2008) A study of graph spectra for comparing graphs and trees. *Pattern Recognition* 41(9):2833–2841. <https://doi.org/10.1016/j.patcog.2008.03.011>
- [38] Gera R, Alonso L, Crawford B, et al (2018) Identifying network structure similarity using spectral graph theory. *Applied Network Science* 3:1–15. <https://doi.org/10.1007/s41109-017-0042-3>
- [39] Tsitsulin A, Mottin D, Karras P, et al (2018) Netlsd: Hearing the shape of a graph. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery, New York, NY, USA, p 2347–2356, <https://doi.org/10.1145/3219819.3219991>
- [40] Amanatidis Y, Green B, Mihail M (2008) Graphic realizations of joint-degree matrices. Unpublished manuscript
- [41] Xie Y, Xu Z, Ma J, et al (2023) How much space has been explored? measuring the chemical space covered by databases and machine-generated molecules. In: *The Eleventh International Conference on Learning Representations*
- [42] Dan Friedman D, Dieng AB (2023) The vendi score: A diversity evaluation metric for machine learning. *Transactions on machine learning research*
- [43] Mironov M, Prokhorenkova L (2025) Measuring diversity: Axioms and challenges. *arXiv preprint arXiv:241014556*
- [44] Penrose M (2003) *Random geometric graphs*, vol 5. OUP Oxford
- [45] Del Toro N, Shrivastava A, Ragueneau E, et al (2022) The IntAct database: efficient access to fine-grained molecular interaction data. *Nucleic Acids Research* 50(D1):D648–D653. <https://doi.org/10.1093/nar/gkab1006>
- [46] Licata L, Briganti L, Peluso D, et al (2012) MINT, the molecular interaction database: 2012 update. *Nucleic Acids Research* 40(D1):D857–D861. <https://doi.org/10.1093/nar/gkr930>
- [47] Freeman LC, Freeman SC, Michaelson AG (1988) On human social intelligence. *J Soc and Biol Struct* 11(4):415–425

- [48] Kunegis J (2013) KONECT – The Koblenz Network Collection. In: WWW '13 Companion: Proceedings of the 22nd International Conference on World Wide Web, pp 1343–1350, URL <http://dl.acm.org/citation.cfm?id=2488173>
- [49] Newman MEJ (2006) Finding community structure in networks using the eigenvectors of matrices. *Phys Rev E* 74:036104. <https://doi.org/10.1103/PhysRevE.74.036104>
- [50] Rossi R, Ahmed N (2015) The network data repository with interactive graph analytics and visualization. In: Proceedings of the AAAI conference on artificial intelligence
- [51] Mahadevan P, Krioukov D, Fall K, et al (2006) Systematic topology analysis and generation using degree correlations. *SIGCOMM Computer Communication Review* 36(4):135–146. <https://doi.org/10.1145/1151659.1159930>
- [52] Orsini C, Dankulov MM, Colomer-de Simón P, et al (2015) Quantifying randomness in real networks. *Nature communications* 6(1):8627. <https://doi.org/10.1038/ncomms9627>
- [53] Davis J, Goadrich M (2006) The relationship between precision-recall and roc curves. In: Proceedings of the 23rd International Conference on Machine Learning. Association for Computing Machinery, New York, NY, USA, p 233–240, <https://doi.org/10.1145/1143844.1143874>
- [54] Bollobás B (1998) Random graphs. Springer, https://doi.org/10.1007/978-1-4612-0619-4_7
- [55] Artzy-Randrup Y, Stone L (2005) Generating uniformly distributed random networks. *Physical Review E* 72:056708. <https://doi.org/10.1103/PhysRevE.72.056708>