EVALUATING GRAPH GENERATIVE MODELS WITH GRAPH KERNELS: WHAT STRUCTURAL CHARACTERISTICS ARE CAPTURED?

Anonymous authors
Paper under double-blind review

ABSTRACT

For a number of practical problems, it is important to measure the similarity between graphs. In particular, it is essential for assessing the quality of graph generative models. In evaluation measures for graph generative models, *graph kernels* are often used to measure the similarity of graphs. Recently, it has been shown that the choice of a graph kernel may drastically affect research outcomes in this area. Therefore, it is essential to choose a kernel that is suitable for the task at hand. In this paper, we propose a framework for comparing graph kernels in terms of which high-level structural properties they are sensitive to. For this, we choose several pairs of random graph models that are different in one particular property: heterogeneity of degree distribution, the presence of community structure, the presence or particular type of latent geometry, and others. Then, we design continuous transitions between these models and measure which graph kernel is sensitive to the corresponding change. We show that using such diverse graph modifications is crucial for evaluation: many kernels can successfully capture some properties and fail on others. One of our conclusions is that simple and long-known Shortest Path and Graphlet kernels are able to successfully capture all graph properties that we consider in this work.

1 INTRODUCTION

Many real-world objects can be represented as graphs: social and citation networks, molecules, the Internet, transportation networks, and so on. For a number of practical problems, it is important to measure the similarity or distance between graphs: it can be helpful for graph classification, graph clustering, or evaluating graph generative models.

A number of *graph kernels* have been proposed in the literature to evaluate the similarity between graphs (Kriege et al., 2020; Nikolentzos et al., 2021). Graph kernels are often based on some graph statistics like node degrees, shortest path lengths, small subgraph counts, and so on. Thus, different kernels capture different graph properties. However, it is not obvious *which* properties are captured by a given kernel. This makes it challenging to choose a suitable kernel for a particular problem.

In this paper, we are particularly motivated by the problem of evaluating the performance of graph generative models. To evaluate such a model, one needs a measure that compares a generated set of graphs with a reference set. A standard approach is to first convert all graphs to their vector representations or to compute similarities for pairs of graphs and then to compare the obtained representations or similarity sets. If similarity is measured via a graph kernel, then maximum mean discrepancy (MMD) is typically used in the second step of this procedure. The problem of choosing the right measure for graph generative model evaluation has gained significant attention recently (O'Bray et al., 2022; Thompson et al., 2022; Shirzad et al., 2022). In particular, it was shown that the choice of a kernel or even a particular parameter for a given kernel may drastically affect the outcome of the model comparison (O'Bray et al., 2022).

We propose a framework for comparing graph kernels in terms of their sensitivities to different graph properties. For this, we choose several pairs of random graph models that are different in a particular graph property — the presence of community structure, degree distribution, latent geometry, etc. Then, we design continuous transitions between the models: in each transition, the presence of a given property gradually increases. To evaluate whether a particular kernel is sensitive to a particular type of change, we generate several sets of graphs at different points of the transition and check whether the kernel is able to distinguish these different sets. As a result, we provide a detailed comparison of graph kernels that can help practitioners in choosing the best one for their application. Our framework aligns well with how the performance of graph generative models is evaluated and thus can help in the development of better measures for this task.

One of our main observations is that many kernels are sensitive to some of the properties while being insensitive to others. For example, our experiments show that the popular Weisfeiler-Leman kernel is insensitive to geometry, which makes it insuitable for tasks like molecular modeling, where the nodes (atoms) have spatial locations. This shows the importance of using different structural properties for assessing which characteristics a kernel is sensitive to. Among the best-performing kernels are the long-known Shortest Path and Graphlet kernels. Another kernel that performs well on all the properties is a more complicated Pyramid Match kernel.

We hope that our investigation will be helpful for researchers and practitioners for choosing suitable kernels for evaluating graph generative models.

2 PRELIMINARIES

In this paper, we analyze graph kernels and focus on undirected graphs $G = (V, E)$ with no node/edge attributes since most of the known kernels can be applied to such graphs.

2.1 MEASURING GRAPH SIMILARITY

Various approaches can be used to measure similarity or dissimilarity between graphs. Two major research directions are concentrated on *graph distances* and *graph kernels*.

Graph distances measure dissimilarity between graphs and are supposed to satisfy the axioms of a metric space. However, the positivity axiom is usually violated since the distance between two different graphs can be equal to zero. Indeed, if we guarantee that $D(G, G') = 0$ if and only if G and G' are isomorphic, then computing such distance is at least as hard as graph isomorphism testing, which is infeasible for most applications.

In turn, a *graph kernel* is a symmetric, positive semidefinite function defined on the space of graphs. This function can be expressed as an inner product in some Hilbert space. A survey of many graph kernels can be found in, e.g., Kriege et al. (2020); Nikolentzos et al. (2021).

In this paper, we conduct a comparative analysis of graph kernels. Let us note that any kernel $K(\cdot,\cdot)$ can be transformed to a distance measure (up to the positivity axiom) as, e.g., $D(G_1, G_2) = \sqrt{K(G_1, G_1) + K(G_2, G_2) - 2K(G_1, G_2)}$. Similarly, there are multiple ways to transform a graph distance to a graph kernel.

A typical approach to define a kernel or distance for graphs is to describe each graph based on its characteristics: degree distribution, substructure counts, shortest path lengths, spectral properties, and so on. The obtained descriptors can then be used to obtain a graph kernel or graph distance. Some kernels are defined for graphs with discrete node labels. However, they can be applied to unlabeled graphs as well. For this, we assume that all node labels are equal.

Let us now define two popular and long-known graph kernels. In Appendix A, we define other graph kernels used in our study.

The Shortest Path (SP) kernel was introduced in Borgwardt & Kriegel (2005). Each shortest path is described by the following triplet: its length and labels of starting and ending nodes. Then, the graph is represented by a vector f_G, where each coordinate is the frequency of a particular triplet in a graph. Then, the kernel is a scalar product of f_G and $f_{G'}$. In our setting, where each node has the same label, the SP kernel compares the shortest-path histograms. Thus, the vector f_G can be interpreted as a signature of the topology described by the network.

The Graphlet kernel was proposed in Pržulj (2007). *Graphlets* are small connected subgraphs of a graph. Consider graphlets of size k. We assign an index to each graphlet and let f_G be a vector such that its i-th entry is equal to the frequency of occurrence of i-th graphlet in G. Then, the kernel can be computed as the scalar product of f_G and $f_{G'}$.

2.2 GRAPH KERNELS FOR EVALUATING GRAPH GENERATIVE MODELS

One important application of graph kernels is measuring the performance of graph generative models. In this section, we briefly review this research area and relevant literature.

To evaluate a graph generative model, one usually needs to compare a set of graphs produced by the model with some reference set of graphs (not available to the model during training). Typically, the comparison consists of two steps. First, each graph is described as a point in a vector space, or for each pair of graphs their similarity is defined. Then, two sets of points or two sets of similarity values are compared. Arguably the most widely-used approach is to define a kernel $K(\cdot,\cdot)$ for measuring the similarity of graphs and then use *maximum mean discrepancy* (MMD) to measure the distance between two sets of graphs $\mathcal{G}_1$ and $\mathcal{G}_2$:

$$\mathrm{MMD}^2(\mathcal{G}_1,\mathcal{G}_2) = \frac{1}{|\mathcal{G}_1|^2}\sum_{G_1,G_2\in\mathcal{G}_1} K(G_1,G_2)+\frac{1}{|\mathcal{G}_2|^2}\sum_{G_1,G_2\in\mathcal{G}_2} K(G_1,G_2)-\frac{2}{|\mathcal{G}_1||\mathcal{G}_2|}\sum_{G_1\in\mathcal{G}_1}\sum_{G_2\in\mathcal{G}_2} K(G_1,G_2).$$

If $K(G_1,G_2)$ is a kernel, then it can be expressed as $K(G_1,G_2) = \langle f(G_1), f(G_2)\rangle$ for some vector representation $f(G)$. Then, $\mathrm{MMD} = \|\mu_1 - \mu_2\|^2$, where $\mu_i = \frac{1}{|\mathcal{G}_i|}\sum_{G\in\mathcal{G}_i} f(G)$. Thus, MMD is indeed a distance. Besides MMD, there are other approaches to compare distributions: Fréchet Distance (Heusel et al., 2017), Improved Precision & Recall (Kynkäänniemi et al., 2019), Density & Coverage (Naeem et al., 2020). Overall, there are many ways to compare two distributions of graphs.

O'Bray et al. (2022) compare some of the options and, similarly to our work, consider graphs without node features and labels. The authors compare MMD-based measures. To define a kernel, O'Bray et al. (2022) describe each graph by a vector via simple structural characteristics: a histogram of the degree distribution, a histogram of local clustering coefficients, or eigenvalues of the Laplacian. Then, different transformations for obtaining a kernel are considered: the first Wasserstein distance (EMD), total variation distance (TV), and the radial basis function kernel (RBF). Each transformation has a parameter that needs to be specified. The important conclusion of the paper is that for different kernels and their hyperparameters, the outcome of the models' comparison may drastically differ. This supports the importance of the problem we consider in this paper. Another contribution of O'Bray et al. (2022) is their approach to comparing measures: it is proposed to make perturbations to a given set of graphs and measure whether MMD correlates with the degree of change. The considered modifications are the following: random edge insertions, random edge deletions, random rewiring operations, and random node additions. The key distinction of our paper is that we design specific graph distribution changes that target particular graph properties and allow us to give a much more detailed understanding of what can be captured by a particular kernel.

Thompson et al. (2022) also compares evaluation measures for graph generative models and assumes that graphs may have features of nodes and edges. They consider two types of measures: classic (as in O'Bray et al. (2022)) and based on neural representations. Thompson et al. (2022) suggest adopting the widely used measures from image generation literature (Fréchet Distance, Improved Precision & Recall, Density & Coverage). But for graphs (in contrast to images), this would require training a neural network for each dataset. Instead, the authors suggest using representations obtained via a randomly initialized GIN model; see Appendix A for the details. In the experiments, Thompson et al. (2022) measure the rank correlation between the degree of perturbation and the measure and show that the proposed randomly initialized GNNs work well. Several aspects are tested: fidelity (whether a measure can detect random graphs added to a set of graphs or detect randomly rewired edges); diversity (a measure should be sensitive to mode dropping and mode collapse); sensitivity to node and edge features; sample efficiency (the minimum number of samples to discriminate a set of random graphs from real samples). However, in terms of graph structure, only random rewiring and the Erdős–Rényi random graph model are considered. Finally, Shirzad et al. (2022) propose replacing a randomly initialized GIN model with a contrastively trained GNN model, which makes the obtained evaluation measure dataset-dependent.

3 COMPARING GRAPH KERNELS

The main question that we aim to answer in this paper is 'What structural characteristics are captured by what kernel?'. With 'captured', we mean that the kernel can distinguish two sets of graphs if they differ in that particular aspect. In this section, we describe the setup that we propose to answer this question.

To measure the sensitivity of a kernel to a graph characteristic, we will consider a sequence of graph generators where, at each step, the considered graph characteristic is more present. To that end, we consider graph generators that include a *step* parameter θ which we vary between 0 and 1, such that the characteristic is absent at $\theta = 0$, and strongly present at $\theta = 1$. We describe below which graph generators we consider and how we interpolate between them.

In all of our generators, we keep the number of nodes n constant, and we also preserve the expected number of edges m. We use the Erdős-Rényi (ER) random graph model as *baseline* generator, where each of the considered graph characteristics is absent. In the Erdős-Rényi model, all edges are added independently with probability $p = m/\binom{n}{2}$.

3.1 STRUCTURAL GRAPH CHARACTERISTICS

In this section, we discuss structural graph characteristics that we consider in our research and introduce the generative models that are used for each of these characteristics.

Density Density is the simplest graph characteristic that can be modeled by varying the probability p in the basic Erdős-Rényi model.

Degree heterogeneity The degree distribution of an ER graph differs from what is observed in real-world networks: ER graphs have Binomially distributed degrees, leading to a variance that is lower than the mean degree. However, many real-world networks have a much more *heterogeneous* degree distribution, where the variance is often many times larger than the mean degree. Many networks even seem to have *power-law* degree distributions, leading to many hubs and a high degree variance (Barabási & Bonabeau, 2003). Several generative models incorporate degree heterogeneity by prescribing an (expected) degree sequence, as can be done with the Configuration Model and the Chung-Lu model (Chung & Lu, 2002; Van der Hofstad, 2016). We use the Chung-Lu model because it is a generalization of ER and is simple to work with. The input to this model is the vector of the expected degrees $(w_1, \ldots, w_n)^T$. Given the expected degrees, an edge between two nodes u and v is added with probability $\frac{w_u w_v}{\sum_i w_i}$ independently of all other edges. We sample the prescribed degrees from a Pareto distribution with power-law exponent $\gamma \geq 2$ and scale parameter chosen to ensure that the expected number of edges equals m. In the limit $\gamma \to \infty$, this is equivalent to the ER model, while small finite values lead to degree sequences with high variance.

Clustering In many real-world networks, nodes with common neighbors are more likely to be connected to each other (Watts & Strogatz, 1998; Van der Hofstad, 2016). This phenomenon is often referred to as *clustering* (Holland & Leinhardt, 1971; Peixoto, 2022), and it results in an abundance of triangles, which is often quantified by the *clustering coefficient*. Social networks and many other real graphs are well-known for having a high degree of clustering (Holland & Leinhardt, 1971; McPherson et al., 2001).

There are three main mechanisms that are used to explain and model the high level of clustering: *triadic closure*, *community structure*, and *latent geometry*. In our analysis, we treat these as three different graph characteristics.

Triadic closure The simplest way to incorporate the over-representation of triangles in a network is to assume that two nodes have an increased likelihood of forming an edge between them whenever they share common neighbors. We model this similarly to Peixoto (2022), by generating the graph in two phases. First, we generate an ER graph with some edge probability p_1. In the second phase, we go over each node pair that is at distance 2 and place an edge with probability p_2. To ensure that the expected number of edges is equal to m, we choose $p_1 < m/\binom{n}{2}$ so that the first stage leads to less than m edges (in expectation). Then, we set p_2 to

$$p_2 = \frac{\frac{m}{\binom{n}{2}} - p_1}{(1-p_1)(1-(1-p_1^2)^{n-2})}. \tag{1}$$

With these parameters, the expected number of edges in the obtained graph equals m, see the derivation in Appendix B.1. We refer to this generative model as the *triadic* model. Taking $p_1 = m/\binom{n}{2}$ leads to the ER model, while lower values of p_1 lead to higher global clustering coefficients.

Community structure Many real-world networks contain groups of nodes that are more densely connected to each other than to the rest of the network. In network science, these groups are referred to as *communities* (Fortunato, 2010), and they often have a natural interpretation, like friend groups in social networks or subject areas in citation networks. In addition, the presence of community structure can explain the clustering that is observed in real networks: the presence of densely connected groups leads to an increased number of triangles.

The simplest generative model for community structure is the *Planted Partition* (PP) model (Holland et al., 1983), where we are given a partition of the network nodes into communities, and node pairs of the same community connect with probability p_{in}, while node pairs of different communities connect with probability p_{out}. We consider two communities of size $n/2$ each, and we parameterize p_{in}, p_{out} as

$$p_{\text{in}}(\lambda) = \frac{4m\lambda}{n^2(1+\lambda) - 2\lambda n}, \quad p_{\text{out}}(\lambda) = \frac{4m}{n^2(1+\lambda) - 2\lambda n}, \tag{2}$$

so that the expected number of edges equals m, while $p_{\text{in}}/p_{\text{out}} = \lambda$. This parametrization of the PP model reduces to the ER model for $\lambda = 1$, and the (expected) global clustering coefficient increases monotonously with λ.

Latent geometry Community structure explains clustering by assuming a certain (finite) set of types, and that nodes of the same type have a higher likelihood of connecting than nodes of different type. The *Random Geometric* (RG) model generalizes this notion by considering a continuous type space and assuming that nodes whose types are *similar* to each other have a larger probability of connecting to each other. In citation networks, for example, papers tend to cite papers on related topics. This continuous type space can be thought of as some *feature space*, and the (dis)similarity may be quantified by some distance measure in this space. However, often one only has access to the network connections, and not the positions in the feature space. In such cases, we say that the network has a *latent geometry*.

The simplest way to model this, is by assigning to each node a coordinate in some *latent space*, and connecting two nodes if their distance is lower than some threshold. This model is referred to as a *random geometric graph* (Penrose, 2003). We consider a two-dimensional *torus* geometry, where each node is assigned a coordinate in $[0,1)^2$ uniformly at random, and two nodes are connected whenever their distance is below some threshold r. We choose r to ensure that the network has m edges in expectation, see Appendix B.2 (for $h = 1$).

Dimensionality Whenever a network has latent geometry, its connections depend highly on the characteristics of that geometry. For example, networks with hyperbolic geometry tend to have a high level of degree heterogeneity (Krioukov et al., 2010). In addition, the dimension of the latent space heavily affects both the local properties (e.g., clustering) and global characteristics (e.g., diameter). In general, the clustering coefficient decreases with dimension.

To model varying dimensionality, we use the random geometric graph model: we consider a torus with width 1 and height $h \in (0,1]$. For $h = 1$, this corresponds to the standard two-dimensional torus, while the limit $h \downarrow 0$ results in a one-dimensional torus, i.e., a circle. Note that the radius r that leads to m edges in expectation, depends on h. The desired value of $r(h,m)$ is derived in Appendix B.2.

Complementarity In some particular types of networks, such as protein-protein interaction networks and economic networks (Talaga & Nowak, 2022; Mattsson et al., 2021), it has been observed that the clustering coefficient is significantly lower than that of an ER graph with the same number of edges. Instead, these graphs have a large number of quadrangles (cycles of length four), leading to a locally bipartite structure (Estrada, 2006). This phenomenon is usually explained by *complementarity* (Talaga & Nowak, 2022): in these networks, nodes do not connect if they are similar to each other, but if they differ in some specific way. In economic networks, for example, companies that produce a certain product will trade with companies that are in need of that particular product. This leads 'similar' nodes to have many common neighbors, but rarely a direct connection. In turn, dissimilar nodes have a high likelihood of connecting.

There are several random graph models for complementarity: one option is *disassortative* community structure, e.g., by a PP model with $p_{\text{in}} < p_{out}$, but there are also ways to model complementarity using latent geometry. We model complementarity by a latent spherical geometry (Talaga & Nowak, 2022). This has the nice property that each point in the latent space has a unique *antipodal* point at maximum distance. We assign each node to a point on the hypersphere (with radius 1) uniformly at random and connect two nodes if their distance exceeds $\pi - r$ for some value of r, chosen to ensure m edges in expectation. We refer to this generator as the *Spherical Complementarity* (SC) model.

3.2 INTERPOLATING GRAPH CHARACTERISTICS

For each graph characteristic, we define an interpolation between two graph generators, so that the strength of that characteristic changes monotonously along the transition. We parametrize each of these interpolations by a step parameter $\theta \in [0,1]$, so that $\theta = 0$ leads to the 'left' generator, while $\theta = 1$ leads to the 'right' generator. We make use of the fact that most of the generating models introduced in Section 3.1 are generalizations of the ER model. In these cases, we simply parameterize a transition away from the ER model. The ER model is chosen with $p = m/\binom{n}{2}$ so that the expected number of edges equals m. For the geometric models used for latent geometry and complementarity, we will take a different approach to interpolate between generators.

Density (ER(p)) We use the ER model with the edge probability $p(\theta + 1)$. Thus, when θ changes from 0 to 1, the edge probability increases from p to $2p$.

Heterogeneity (ER↔CL) We use the Chung-Lu (CL) model with weights drawn from a Pareto distribution with power-law exponent $1 + 1/\theta$, and scale chosen such that the expected number of edges equals m. Note that $\theta = 0$ leads to a power-law exponent ∞, which leads all the weights to be constant so that the corresponding CL model is equivalent to the ER model. The values $\theta > 0.5$ give weight distributions with infinite variance, which leads to a high variance in the degree distribution.

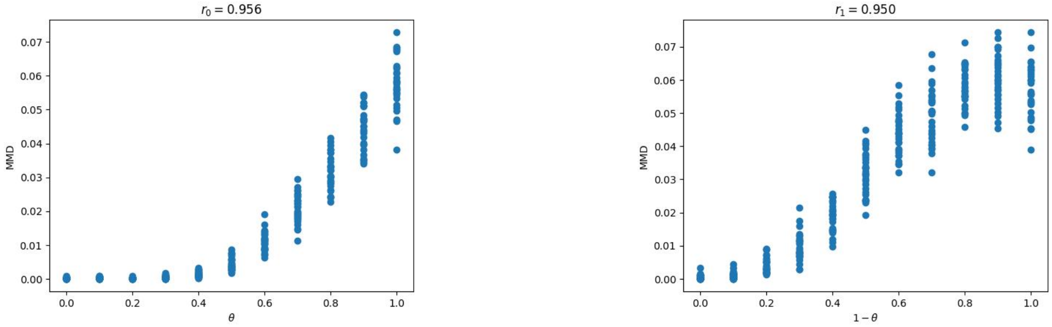
Triadic closure (ER↔Triadic) We use the Triadic model with $p_1(\theta) = (1 - \frac{\theta}{2}) \cdot p$, and p_2 as prescribed by (1). Hence, $\theta = 0$ leads to $p_2 = 0$, so that ER is recovered, while $\theta = 1$ leads to $p_2 \approx 0.33$, so that roughly a third of the shortest paths of length 2 will be shortcut in the second phase.

Communities (ER↔PP) We use the PP model with p_{in}, p_{out} as given in (2) for $\lambda = 1 + 5.6\theta$, so that $\theta = 0$ leads to $p_{in} = p_{\text{out}} = p$, while $\theta = 1$ leads to every vertex having (in expectation) one neighbor outside its community.

Latent geometry (ER↔Torus) For this transition, we take the *mixture* of an ER graph and a RG torus graph: we generate a graph from each generator and denote their adjacency matrices by $A^{(ER)}$ and $A^{(RG)}$. We construct the adjacency matrix A as follows: for each node pair with indices $1 \le i < j \le n$ we draw an independent Bernoulli random variable B_{ij} with success probability θ and set $A_{ij} = A_{ij}^{(ER)} + B_{ij}(A_{ij}^{(RG)} - A_{ij}^{(ER)})$. Hence, $\theta = 0$ leads to $A = A^{(ER)}$, while $\theta = 1$ leads to $A = A^{(RG)}$. We complete the adjacency matrix symmetrically: $A_{ij} = A_{ji}$ for $i > j$ and $A_{ii} = 0$.

Dimensionality (Torus↔Circle) To interpolate between a two- and one-dimensional torus, we consider a torus of height $h = 1 - \theta$ and connection radius $r(h, m)$ as derived in Appendix B.2. This way, $\theta = 0$ leads to a torus on $[0, 1)^2$ while $h \downarrow 0$ leads to a one-dimensional torus (a circle).

Complementarity (ER↔SC) For this transition, we take the same approach as for the interpolation ER↔Torus: we construct a graph by taking a mixture of graphs generated by the ER and SC models, such that $\theta = 0$ results in an ER graph, while $\theta = 1$ results in a SC graph.



(a) Comparison with $\theta^* = 0$.

(b) Comparison with $\theta^* = 1$.

Figure 1: Scatter plots of the MMD values of the SP kernel for the ER↔CL interpolation.

3.3 MEASURING THE SENSITIVITY OF A KERNEL

We now explain how we quantify the sensitivity of a kernel w.r.t. an interpolation between graph generators. We consider a discretization $\Theta \subset [0,1]$ of the interpolation such that $\{0,1\} \subset \Theta$. For $\theta \in \Theta$, let $\mathcal{G}_\theta$ denote a set of g graphs sampled independently from the interpolation generator at step θ. Furthermore, let $\mathrm{MMD}(\mathcal{G}_{\theta_1}, \mathcal{G}_{\theta_2}; K)$ denote the MMD value between $\mathcal{G}_{\theta_1}$ and $\mathcal{G}_{\theta_2}$ w.r.t. the kernel K. We compare each $\theta \in \Theta$ to each endpoint $\theta^* \in \{0,1\}$. For each endpoint $\theta^* \in \{0,1\}$, we quantify the sensitivity of the kernel K w.r.t. the interpolation as the *Spearman correlation* between $\mathrm{MMD}(\mathcal{G}_{\theta^*}, \mathcal{G}_\theta; K)$ and $|\theta - \theta^*|$. If this value is close to 1, it means that the MMD values tend to increase when transitioning θ away from θ^*. But if this value is close to zero, it indicates that there is no clear monotone relation between θ and $\mathrm{MMD}(\mathcal{G}_{\theta_1}, \mathcal{G}_{\theta_2}; K)$. This leads to two different correlation coefficients r_0 and r_1, corresponding to the different values of θ^*. See Figure 1 for an illustration.

To ensure the independence of the different MMD values that are used to compute a correlation coefficient, we sample two different sets of graphs for each MMD value. Between each pair of θ's that we compare, we compute ℓ MMD values. This results in sampling $\ell \cdot g$ graphs for each $\theta \in \Theta \setminus \{0,1\}$, and $(1 + |\Theta|) \cdot \ell \cdot g$ graphs for each $\theta^* \in \{0,1\}$.

4 EXPERIMENTS

We follow the framework described in the previous section. In our experiments, we consider graphs with $n = 50$ nodes and (in expectation) $m = 190$ edges. We discretize the interpolation interval $[0,1]$ by $\Theta = \{0.0, 0.1, \ldots, 1.0\}$. Thus, we will have $|\Theta| = 11$ steps in our interpolation. We consider sets of $g = 100$ graphs and compute $\ell = 30$ different MMD values for each pair of interpolation steps that we compare.

In our experiments, we use the following kernels and representations:

- Shortest Path (SP) kernel (Borgwardt & Kriegel, 2005);
- Weisfeiler-Leman (WL) kernel (Shervashidze et al., 2011) with $l = 5$;
- Weisfeiler-Lehman optimal assignment (WL-OA) kernel (Kriege et al., 2016) with $l = 5$;
- Graphlet kernel (Pržulj, 2007) with $k = 3$ and $k = 4$ (referred to as Graphlet-3 and Graphlet-3, respectively);
- Neighborhood Subgraph Pairwise Distance kernel (NSPDK) (Costa & De Grave, 2010) with $r = 3$, $d = 4$;
- Pyramid Match (PM) kernel (Nikolentzos et al., 2017) with Pyramid histogram level 4 and the dimension of the hypercube 6;
- NetLSD graph representations (Tsitsulin et al., 2018) with heat diffusion process;
- Random GIN (RandGIN) representations (Thompson et al., 2022).

	Density ER(p)	Heterogeneity (ER↔CL)	Triadic closure (ER↔Triadic)	Communities (ER↔PP)	Geometry (ER↔Torus)	Dimensionality (Torus↔Circle)	Complementarity (ER↔SC)
SP	0.994	0.953	0.950	**0.959**	**0.957**	**0.985**	**0.929**
WL	**0.996**	**0.993**	0.984	0.387	0.150	0.468	0.097
WL-OA	**0.996**	**0.993**	**0.986**	0.408	0.601	0.549	0.428
Graphlet-3	**0.996**	0.988	**0.996**	**0.979**	**0.972**	0.721	**0.972**
Graphlet-4	**0.996**	**0.991**	**0.996**	**0.973**	**0.980**	**0.843**	**0.970**
NSPDK	0.365	0.956	0.953	0.373	0.854	0.589	0.580
PM	0.981	0.990	0.966	0.950	0.922	0.826	0.923
NetLSD	**0.996**	0.939	0.919	0.948	0.956	**0.950**	0.794
RandGIN	0.938	0.947	0.893	0.132	0.527	0.067	0.285

Table 1: The sensitivity of the kernels to various structural characteristics. Sensitivity is measured via the Spearman correlation as described in Section 3.3. For each interpolation and kernel, we measure two sensitivity values r_0 and r_1 corresponding to comparisons with different endpoints and report their average $(r_0 + r_1)/2$. The top three results are colored.

For most graph kernels, we use the GraKeL python library (Siglidis et al., 2020). Following Nikolentzos et al. (2021), we normalize the kernel values as $K(G_1, G_2)/\sqrt{K(G_1, G_1)K(G_2, G_2)}$. For NetLSD and RandGIN we use the implementation provided by the authors with the default parameters. In both cases, we use cosine similarity to convert graph representations to kernel values. Thus, for all the obtained kernels we have $K(G, G) = 1$. Our code and experiments are available through this (anonymized) repository.[1]

The results are shown in Table 1. One of our main conclusions is that most kernels perform well on some of the interpolations while performing badly on others. This shows the importance of using different interpolations for assessing which characteristics a kernel is sensitive to.

We note that some transitions are easier to detect by all the kernels: for heterogeneity and triadic closure, all kernels show relatively good and stable performance. In contrast, for communities, geometry, dimensionality, and complementarity the difference in performance between different kernels is huge: some kernels have correlations larger than 0.97, while others may have near-zero performance, which suggests that these kernels are completely insensitive to these characteristics. Finally, while most kernels perform well on the density transition, this turns out to be challenging for NSPDK. Let us now go over each kernel individually and summarize the results.

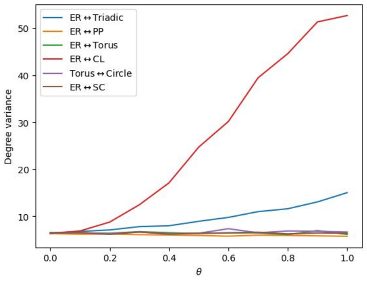


Figure 2: Degree variances for each of the interpolations.

The Weisfeiler-Leman kernel and the related WL-OA kernel are considered to be powerful graph kernels. However, in our experiments, we observe that they only perform well on interpolations where the degree variance changes monotonously. For instance, WL-OA is the best kernel for heterogeneity, which is expected as the first step of the WL procedure is based on node degrees. Also, both WL and WL-OA have high performance on triadic closure. In contrast, for all the remaining interpolations, the Spearman correlation coefficient is quite small. Especially hard for the WL kernel are the interpolations corresponding to geometry and complementarity. This can partially be explained by Figure 2, which shows that the interpolations of heterogeneity and triadic closure significantly affect the degree variance, while the degree distributions in all other interpolations remain quite homogeneous.Finally, let us note that WL-OA performs at least as well as WL on each of the transitions.

The Shortest Path kernel has quite stable performance. Importantly, it is in the top-3 for the most difficult transitions — communities, geometry, dimensionality, complementarity — with correlation values always exceeding 0.9. Interestingly, it is not among the best kernels for simple transitions (heterogeneity, density, and triadic closure), while still having good performance. Notably, SP is the best for dimensionality, which is arguably the most subtle transition. This can be explained by the fact that shortest path lengths capture both local and global information of the graphs, making this

---

[1] Available at `https://anonymous.4open.science/r/graph-kernels-BC69/`

kernel sensitive to non-trivial transitions. Figure 3 in Appendix shows the shortest path distribution for different graph generators. It can be clearly seen that all models are distinguishable. In particular, the most difficult dimensionality is easy for the SP kernel: for Circle, the shortest paths can be significantly longer than for Torus.

The Graphlet kernel is another good option in terms of the overall performance: it is among the best-performing kernels for all interpolations but dimensionality, where the difference with the SP kernel is noticeable. The poorer performance for dimensionality may be explained by the fact that the graphlet kernel is not able to capture the global properties of graphs which are affected when we vary the dimension (e.g., its diameter). Figure 4 additionally illustrates the difficulty of the dimensionality transition: it turns out that for small values of θ (when the height h is not too small), the distribution of graphlets does not change much, and thus the MMD values are close to zero up to $\theta = 0.7$. We also see that increasing the kernel sizes from 3 to 4 significantly improves the performance for dimensionality since this change makes the considered neighborhoods larger. For all other interpolations, increasing graphlet sizes does not lead to noticeable improvements.

The Neighborhood Subgraph Pairwise Distance kernel turns out to be insensitive to community structure and density. Other interpolations that are hard to detect for this kernel are dimensionality and complementarity. We hypothesize that such poor performance for some of the transitions can be explained by a particular graph invariant used to compare two rooted subgraphs (as the exact graph isomorphism is infeasible). Unfortunately, this graph invariant also makes this kernel harder to theoretically analyze or intuitively explain. Let us note that NSPDK has the advantage that it can be used for graphs with node attributes. However, when graphs do not have node labels or attributes, we do not advise using this kernel.

The Pyramid Match kernel is another kernel that seems to be sensitive to all the characteristics. Similarly to the graphlet kernel, PM does not perform so well on dimensionality. Let us also note that PM is dominated by Graphlet-4 for each interpolation. On the other hand, this kernel is scalable and thus it is a good option for larger graphs where the Graphlet kernel cannot be applied.

The NetLSD representations are also sensitive to most of the properties. Interestingly, it is the second-best on dimensionality, which is arguably the most challenging interpolation. However, the performance on complementarity is quite poor.

Random GIN representations were proposed in Thompson et al. (2022) in the context of evaluating graph generative models. We note that there are some differences in our setup. First, our graphs do not have node features, while being able to process such features is one of the main advantages of RandGIN. Second, Thompson et al. (2022) use GIN representations not for a kernel but to compute such measures as precision, recall, or Fréchet Distance. Nevertheless, our results show that when used within our framework, the kernel based on RandGIN is not sensitive to such interpolations as communities, geometry, dimensionality, and complementarity. This is similar to the results of the WL kernel and can be explained by the low diversity of degrees in such models.

5 CONCLUSION

In this paper, we propose a framework for comparing graph kernels in terms of the high-level structural graph properties that they are sensitive to. For this, we carefully design an experimental setup based on interpolations between random graph generators differing in one particular structural property. We test several popular kernels and graph representations using the proposed setup. We show that simple and long-known kernels often have better sensitivity to particular characteristics compared to more advanced approaches. We also demonstrate that the wide diversity of the considered structural properties is critical for a thorough evaluation.

The results shown in Table 1 are useful when deciding which graph kernel is most suitable when evaluating generative models in a particular application. For example, our experiments show that the widely-used Weisfeiler-Lehman kernel is insensitive to geometry. In turn, geometry can be especially relevant, e.g., for the application of molecular modeling because atoms have spatial locations and using the WL kernel is not a good option in this case.

REFERENCES

Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific american*, 288(5):60–69, 2003.

Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pp. 8–pp. IEEE, 2005.

Fan Chung and Linyuan Lu. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics*, 6(2):125–145, 2002.

Fabrizio Costa and Kurt De Grave. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 255–262. Omnipress; Madison, WI, USA, 2010.

Ernesto Estrada. Protein bipartivity and essentiality in the yeast protein- protein interaction network. *Journal of proteome research*, 5(9):2177–2184, 2006.

Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Paul W Holland and Samuel Leinhardt. Transitivity in structural models of small groups. *Comparative group studies*, 2(2):107–124, 1971.

Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

Nils M Kriege, Pierre-Louis Giscard, and Richard Wilson. On valid optimal assignment kernels and applications to graph classification. *Advances in neural information processing systems*, 29, 2016.

Nils M Kriege, Fredrik D Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 5(1):1–42, 2020.

Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

Carolina ES Mattsson, Frank W Takes, Eelke M Heemskerk, Cees Diks, Gert Buiten, Albert Faber, and Peter MA Sloot. Functional structure in production networks. *Frontiers in big Data*, 4: 666712, 2021.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International Conference on Machine Learning*, pp. 7176–7185. PMLR, 2020.

Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. Matching node embeddings for graph similarity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. Graph kernels: A survey. *Journal of Artificial Intelligence Research*, 72:943–1027, 2021.

Leslie O'Bray, Max Horn, Bastian Rieck, and Karsten Borgwardt. Evaluation metrics for graph generative models: Problems, pitfalls, and practical solutions. In *International Conference on Learning Representations*, 2022.

Tiago P Peixoto. Disentangling homophily, community structure, and triadic closure in networks. *Physical Review X*, 12(1):011004, 2022.

Mathew Penrose. *Random geometric graphs*, volume 5. OUP Oxford, 2003.

Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Hamed Shirzad, Kaveh Hassani, and Danica J Sutherland. Evaluating graph generative models with contrastively learned features. *arXiv preprint arXiv:2206.06234*, 2022.

Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skianis, and Michalis Vazirgiannis. Grakel: A graph kernel library in python. *Journal of Machine Learning Research*, 21(54):1–5, 2020.

Szymon Talaga and Andrzej Nowak. Structural measures of similarity and complementarity in complex networks. *Scientific Reports*, 12(1):16580, 2022.

Rylee Thompson, Boris Knyazev, Elahe Ghalebi, Jungtaek Kim, and Graham W Taylor. On evaluation metrics for graph generative models. In *International Conference on Learning Representations*, 2022.

Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander Bronstein, and Emmanuel Müller. NetLSD: hearing the shape of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2347–2356, 2018.

Remco Van der Hofstad. *Random graphs and complex networks*, volume 43. Cambridge university press, 2016.

Duncan J Watts and Steven H Strogatz. Collective dynamics of ‘small-world’networks. *nature*, 393 (6684):440–442, 1998.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

A DEFINITIONS OF GRAPH KERNELS

Weisfeiler-Leman (WL) subtree kernel was proposed in Shervashidze et al. (2011) and is based on the WL color refinement procedure. This procedure works as follows. Initially, all nodes have their labels (or one fixed label for unlabelled graphs). At each iteration, a node's label is replaced by another label identifying a multiset of labels of its neighbors. The procedure stops when it converges. Based on that, the WL kernel for l iterations is

$$\mathrm{WL}(G_1, G_2) = \sum_{i=0}^{l} \sum_{u \in V(G_1)} \sum_{v \in V(G_2)} \mathbb{1}\{\mathrm{wl}^i(u) = \mathrm{wl}^i(v)\},$$

where $\mathrm{wl}^i(u)$ is the label at i-th iteration of the WL procedure.

Weisfeiler-Lehman optimal assignment (WL-OA) kernel was introduced by (Kriege et al., 2016) and it improves the WL kernel by finding the optimal matching of nodes. Formally,

$$\text{WL-OA}(G_1, G_2) = \max_{B \in \mathcal{B}(V_1, V_2)} \sum_{v_1, v_2 \in B} k(v_1, v_2)\,,$$

where $\mathcal{B}(V_1, V_2)$ is the set of all bijections between the node sets and

$$k(v_1, v_2) = \sum_{i=0}^{l} \mathbb{1}\{\mathrm{wl}^i(v_1) = \mathrm{wl}^i(v_2)\}\,.$$

Since WL-OA requires a bijections between the nodes, it can be applied only to graphs of the same size.

The Neighborhood Subgraph Pairwise Distance (NSPDK) kernel proposed in Costa & De Grave (2010) considers pairs of rooted subgraphs of radius $r' \leq r$ whose roots are located at distance $d' \leq d$ from each other. A kernel $k_{r',d'}(G_1, G_2)$ counts the number of such pairs of rooted subgraphs in the first graph that are identical to pairs in the second graph. Then,

$$\mathrm{NSPD}(G_1, G_2) = \sum_{r'=0}^{r} \sum_{d'=0}^{d} \frac{k_{r',d'}(G_1, G_2)}{k_{r',d'}(G_1, G_1) k_{r',d'}(G_2, G_2)}\,.$$

To make the computation of this kernel feasible, graph invariants can be employed to encode each rooted subgraph. Then, these invariants can be compared instead of graph isomorphism checking.

NetLSD treats a graph as a dynamic system and simulates heat and wave diffusion processes on nodes and edges of a given graph, followed by measuring system conditions at fixed timestamps (Tsitsulin et al., 2018). More formally, let λ_j be the j-th smallest eigenvalue of the normalized Laplacian of a graph G. For a timestamp t, we define the *heat trace* h_t and *wave trace* w_t of a graph G as follows:

$$h_t = \sum_j e^{-t\lambda_j}, \;\; w_t = \sum_j e^{-it\lambda_j}\,. \tag{3}$$

Here $t > 0$ for the heat trace and $t \in [0, 2\pi)$ for the wave trace.

Then, the *heat trace signature* and *wave trace signature* of G are defined as the sequences of the corresponding traces at different timestamps, i.e., $h(G) = \{h_t\}_{t \in \mathcal{T}_h}$ and $w(G) = \{w_t\}_{t \in \mathcal{T}_w}$. As in the original article, we use 250 log-spaced time stamps between 10^{-2} and 10^2 for $\mathcal{T}_h$ and 250 equally-spaced time stamps between 0 and 2π for $\mathcal{T}_w$, respectively.

Finally, the NetLSD distance (heat or wave) between two graphs G and G' can be computed as any distance measure between the corresponding signatures. In our comparison, we use NetLSD representations to obtain a graph kernel. For this, we measure the cosine similarity between the graph representations.

Random GIN was proposed by Thompson et al. (2022). This graph method makes use of a randomly initialized Graph Isomorphism Network (GIN) (Xu et al., 2019). To obtain a graph representation, a readout function is applied to aggregate node representations at each GIN layer and then the obtained vectors are concatenated.

Pyramid Match kernel proposed in Nikolentzos et al. (2017) first embeds the vertices of each graph into a low-dimensional vector space using the eigenvectors of the d largest (in magnitude) eigenvalues of the graph's adjacency matrix. Since the signs of these eigenvectors are arbitrary, it replaces all their components by their absolute values. Each vertex is thus a point in the d-dimensional unit hypercube. To find an approximate correspondence between the sets of vertices of two graphs, the kernel maps these points to multi-resolution histograms, and compares the emerging histograms with a weighted histogram intersection function, see Nikolentzos et al. (2017) for more details.

B DERIVATIONS FOR GRAPH GENERATORS

B.1 TRIADIC MODEL

Let p_1, p_2 be the connection probabilities of the first and second phases respectively. We compute the number of edges that will be placed in the second phase. To this end, we first compute the number of node pairs that are at distance 2 after phase 1. For a node pair i, j to be at distance 2, it may not have an edge between it (w.p. $1 - p_1$), but it must have at least one common neighbor. A vertex $k \neq i, j$, is a common neighbor of i and j with probability p_1^2. By De Morgan's law, the probability of having at least one common neighbor is $1 - (1 - p_1^2)^{n-2}$, so that the probability of two nodes being at distance 2 after the first phase, is

$$(1 - p_1) \cdot \left(1 - (1 - p_1^2)^{n-2}\right).$$

Each of these pairs will lead to an edge in phase 2 with probability p_2. Therefore, the total number of edges will be

$$\binom{n}{2} \left(p_1 + (1 - p_1) \cdot \left(1 - (1 - p_1^2)^{n-2}\right) \cdot p_2\right).$$

This is equal to m whenever

$$p_2 = \frac{\frac{m}{\binom{n}{2}} - p_1}{(1 - p_1)(1 - (1 - p_1^2)^{n-2})}.$$

B.2 RANDOM GEOMETRIC MODELS

For a two-dimensional torus with width 1 and height $h \leq 1$, we find the radius r that leads to m edges in expectation. Two nodes are connected when their distance is smaller than r, i.e., whenever the second node is inside a circle with radius r around the first node. If $h \geq 2r$, then the surface area of this circle is simply πr^2, while the total area of the torus is h. Therefore, the expected edge density is πr^2. Solving $\binom{n}{2} \cdot \frac{\pi}{h} r^2 = m$ leads to

$$r = \sqrt{\frac{h \cdot m}{\pi \cdot \binom{n}{2}}},$$

which is valid as long as $2r \leq h$, i.e., whenever $h \geq \frac{4m}{\pi \cdot \binom{n}{2}}$. Otherwise, the circle overlaps itself, which leads to a smaller surface area, given by the following integral

$$A(h, r) = \int_{-h/2}^{h/2} \int_{-\sqrt{r^2-y^2}}^{\sqrt{r^2-y^2}} 1 dx dy = 2r^2 \int_{-\frac{h}{2r}}^{\frac{h}{2r}} \sqrt{1 - u^2} du = r^2 h \sqrt{1 - \left(\frac{h}{2r}\right)^2} + 2r^2 \arcsin\left(\frac{h}{2r}\right).$$

To obtain m edges in expectation, we need to choose r such that $A(h, r)/h = m/\binom{n}{2}$. After substituting $z = \frac{h}{2r}$, we get

$$\frac{m}{\binom{n}{2}} = \frac{h}{2} \left(\sqrt{\frac{1}{z^2} - 1} + \frac{1}{z^2} \arcsin(z)\right) = \frac{h}{2} f(z).$$

We then use Newton-Raphson iteration to solve $f(z) = \frac{2m}{h\binom{n}{2}}$. Let us denote the obtained value by $z = f^{-1}\left(\frac{2m}{h\binom{n}{2}}\right)$. Finally, we take the radius $r = \frac{h}{2z}$. In summary, we choose the radius as

$$r(h,m) = \begin{cases} \sqrt{\frac{h\cdot m}{\pi\cdot\binom{n}{2}}} & \text{if } h \geq \frac{4m}{\pi\cdot\binom{n}{2}}, \\ \frac{h}{2f^{-1}\left(\frac{2m}{h\binom{n}{2}}\right)} & \text{else.} \end{cases} \tag{4}$$

The one-dimensional torus (the circle) is approximated as $h \downarrow 0$. For a circle, we need $r = \frac{m}{2\binom{n}{2}}$ to obtain m edges in expectation. In the remainder, we show that $f^{-1}\left(\frac{2m}{h\binom{n}{2}}\right) \sim \frac{h}{m}\binom{n}{2}$ as $h \downarrow 0$, so that indeed $r(h,m) \to \frac{m}{2\binom{n}{2}}$: first, note that $f(z) \sim 2/z$ as $z \downarrow 0$, so that

$$f\left(\frac{h}{m}\binom{n}{2}\right) = \frac{2m}{h\binom{n}{2}} + o(h^{-1}) \Rightarrow f^{-1}\left(\frac{2m}{h\binom{n}{2}}\right) = f^{-1}\left(f\left(\frac{h}{m}\binom{n}{2}\right) + o(h^{-1})\right).$$

Next, we take the Taylor expansion of f^{-1} around $f\left(\frac{h}{m}\binom{n}{2}\right)$:

$$f^{-1}\left(f\left(\frac{h}{m}\binom{n}{2}\right) + o(h^{-1})\right) = f^{-1}\left(f\left(\frac{h}{m}\binom{n}{2}\right)\right) + \frac{o(h^{-1})}{f'\left(f^{-1}\left(f\left(\frac{h}{m}\binom{n}{2}\right)\right)\right)} = \frac{h}{m}\binom{n}{2} + \frac{o(h^{-1})}{f'\left(\frac{h}{m}\binom{n}{2}\right)}.$$

Finally, the derivative is given by $f'(z) = -2\frac{\arcsin(z)}{z^3} \sim z^{-2}$, so that the second term is $o(h)$. In conclusion, we have

$$f^{-1}\left(\frac{2m}{h\binom{n}{2}}\right) = \frac{h}{m}\binom{n}{2} + o(h),$$

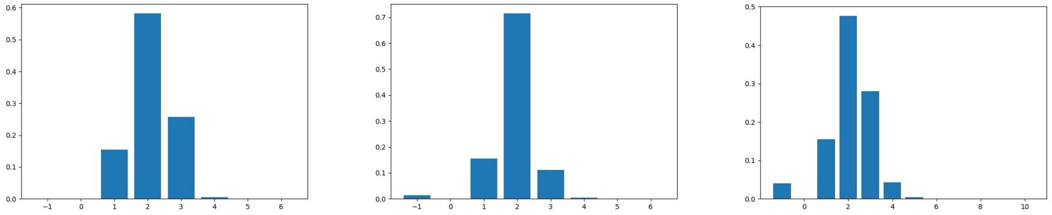
as required.

C LIMITATIONS

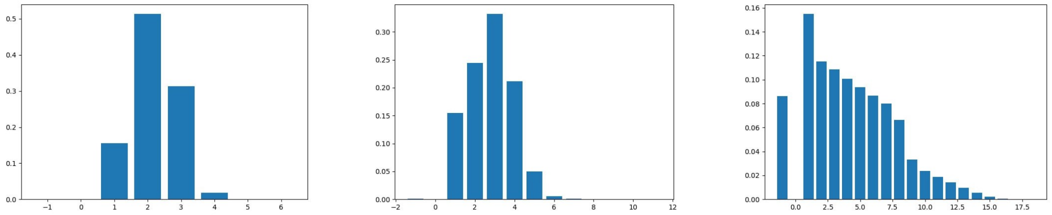
In this study, we model each of the properties in the most straightforward way, aiming to minimally affect other graph characteristics. However, for some graph properties, it is challenging (or even impossible) to vary a property in isolation from everything else. Thus, our transformations designed to vary a particular property may affect other characteristics as well.

Another limitation of our work is that we only consider graphs without attributes. Our general framework easily extends to attributed graphs, but the main challenge would be to design meaningful graph generators for this scenario: there are many ways to combine standard graph generators with different attribute distributions.

Finally, our study considers relatively small graphs. Our main motivation is that evaluating graph generative models using graph kernels is typically applied to such small networks. For example, in molecular modeling, the graphs consist of a relatively small number of atoms. Other applications where small graphs are relevant include modeling ego networks in sociology. Additionally, we note that not all kernels are scalable: for instance, the graphlet kernel has a prohibitively high computational cost on large networks. We consider smaller graphs so that we can include all popular graph kernels in our experiments.

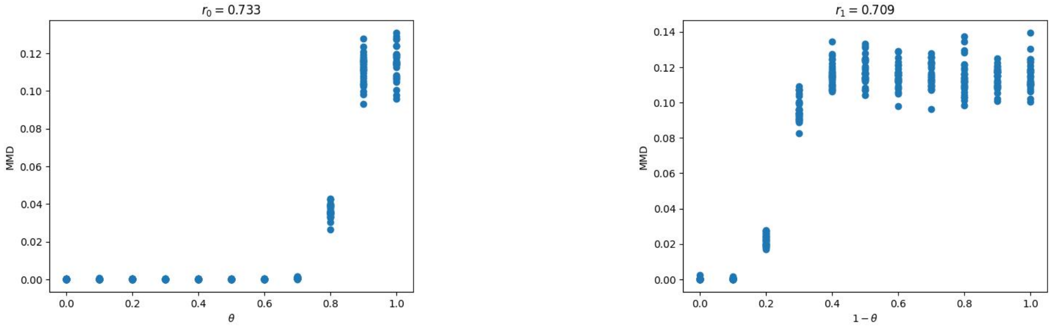


(a) Erdős–Rényi (b) Chung-Lu (c) Triadic



(d) PPM (e) Torus (f) Circle

Figure 3: The distribution of shortest path lengths for different graph generators, -1 corresponds to disconnected node pairs.



(a) Comparison with $\theta^* = 0$. (b) Comparison with $\theta^* = 1$.

Figure 4: Scatter plots of the MMD values of the Graphlet-3 kernel for the torus↔circle interpolation.