What Would a Graph Look Like in This Layout? A Machine Learning Approach to Large Graph Visualization

Oh-Hyun Kwon, *Student Member, IEEE*, Tarik Crnovrsanin, and Kwan-Liu Ma, *Fellow, IEEE*

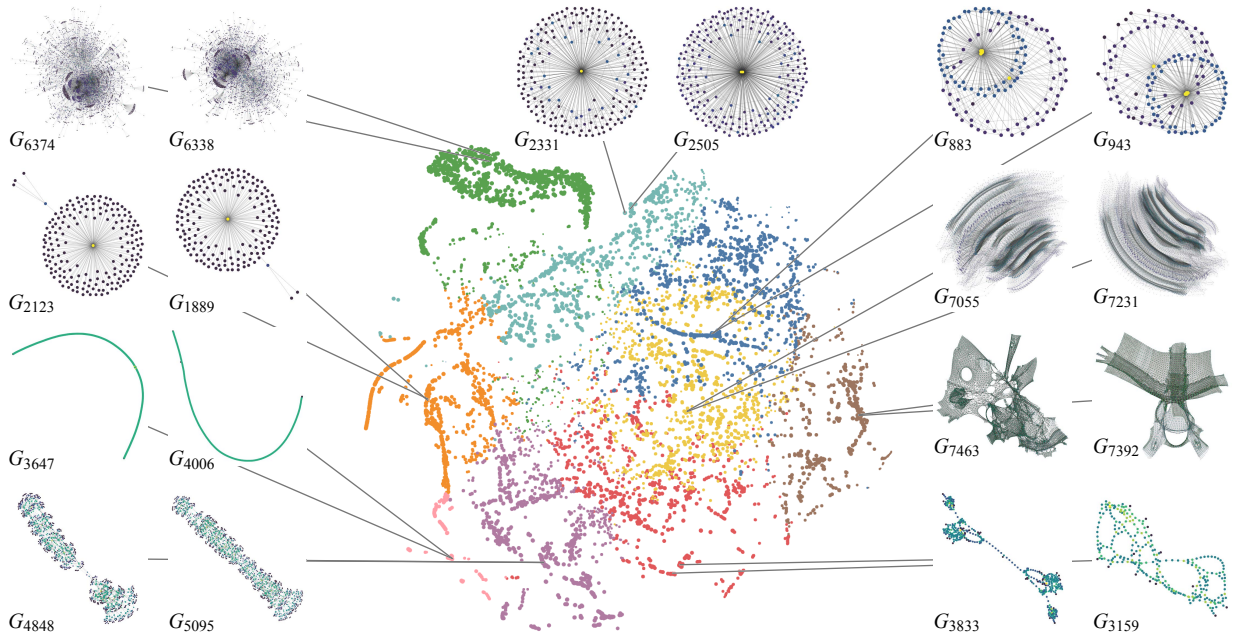


Fig. 1. A projection of topological similarities between 8,263 graphs measured by our RW-LOG-LAPLACIAN kernel. Based on the topological similarity, our approach shows what a graph would look like in different layouts and estimates their corresponding aesthetic metrics. We clustered the graphs based on their topological similarities for the purpose of the user study. The two graphs in each pair are the most topologically similar, but not isomorphic, to each other. The projection is computed with t-SNE [83], and the highlighted graphs are visualized with FM^3 layout [37]. An interactive plot is available in the supplementary materials [1].

Abstract—Using different methods for laying out a graph can lead to very different visual appearances, with which the viewer perceives different information. Selecting a "good" layout method is thus important for visualizing a graph. The selection can be highly subjective and dependent on the given task. A common approach to selecting a good layout is to use aesthetic criteria and visual inspection. However, fully calculating various layouts and their associated aesthetic metrics is computationally expensive. In this paper, we present a machine learning approach to large graph visualization based on computing the topological similarity of graphs using graph kernels. For a given graph, our approach can show what the graph would look like in different layouts and estimate their corresponding aesthetic metrics. An important contribution of our work is the development of a new framework to design graph kernels. Our experimental study shows that our estimation calculation is considerably faster than computing the actual layouts and their aesthetic metrics. Also, our graph kernels outperform the state-of-the-art ones in both time and accuracy. In addition, we conducted a user study to demonstrate that the topological similarity computed with our graph kernel matches perceptual similarity assessed by human users.

Index Terms—Graph visualization, graph layout, aesthetics, machine learning, graph kernel, graphlet

1 INTRODUCTION

Graphs are popularly used to represent complex systems, such as social networks, power grids, and biological networks. Visualizing a graph can help us better understand the structure of the data. Many graph visualization methods have been introduced [36, 43, 86], with the most popular and intuitive method being the node-link diagram.

Over the last five decades, a multitude of methods have been developed to lay out a node-link diagram. A graph's layout results can be greatly different depending on which layout method is used. Because the layout of a graph significantly influences the user's understanding of the graph [36, 46, 56, 57], it is important to find a "good" layout that can effectively depict the structure of the graph. Defining a good layout can be highly subjective and dependent on the given task. A suitable starting point for finding a good layout is to use both the aesthetic criteria, such as *reducing edge crossings*, and the user's visual inspection.

When the graph is large, computing several graph layouts and selecting one through visual inspection and/or aesthetic metrics is, unfortunately, not a practical solution. The amount of time it would take to compute these various layouts and aesthetic metrics is tremendous. For a graph with millions of vertices, a single layout can take hours or days to calculate. In addition, we often must consider multiple aesthetic metrics to evaluate a single graph layout, since there is no consensus on which criteria are the most effective or preferable [26]. As graph data is commonly used in data analysis tasks and is expected to grow

• *All authors are with the University of California, Davis. E-mail: kw@ucdavis.edu, tecrnovr@ucdavis.edu, ma@cs.ucdavis.edu.*

in size at even higher rates, alternative solutions are needed.

One possible solution is to quickly estimate aesthetic metrics and show what a graph would look like through predictive methods. In the field of machine learning, several methods have been used to predict the properties of graphs, such as the classes of graphs. One prominent approach to predicting such properties is to use a graph kernel. Graph kernel methods enable us to apply various kernelized machine learning techniques, such as the Support Vector Machine (SVM) [18], on graphs.

In this paper, we present a machine learning approach that can show what a graph would look like in different layouts and estimate their corresponding aesthetic metrics. The fundamental assumption of our approach is the following: given the same layout method, if the graphs have similar topological structures, then they will have similar resulting layouts. Under this assumption, we introduce new graph kernels to measure the topological similarities between graphs. Then, we apply machine learning techniques to show what a new input graph would look like in different layouts and estimate their corresponding aesthetic metrics. To the best of our knowledge, this is the first time graph kernels have been utilized in the field of graph visualization.

The primary contributions of this work include:

- A fast and accurate method to show what a graph would look like in different layouts.
- A fast and accurate method to estimate graph layout aesthetic metrics.
- A framework for designing graph kernels based on graphlets.
- A demonstration of the effectiveness of graph kernels as an approach to large graph visualization.

We evaluate our methods in two ways. First, we compare 13 graph kernels, which include two state-of-the-art ones, based on accuracy and computation time for estimating aesthetic metrics. The results show that our estimations of aesthetic metrics are highly accurate and fast. Our graph kernels outperform existing kernels in both time and accuracy. Second, we conduct a user study to demonstrate that the topological similarity computed with our graph kernel matches the perceptual similarity assessed by human users.

2 Background

In this section, we present the notations and definitions used in this paper and introduce graph kernels.

2.1 Notations and Definitions

Let $G = (V, E)$ be a graph, where $V = \{v_1, \ldots, v_n\}$ is a set of n vertices (or nodes), and $E = \{e_1, \ldots, e_m \mid e = (v_i, v_j),\ v_i, v_j \in V\}$ is a set of m edges (or links). An edge $e = (v_i, v_j)$ is said to be *incident* to vertex v_i and vertex v_j. An edge that connects a vertex to itself $e = (v_i, v_i)$ is called a *self loop*. Two or more edges that are incident to the same two vertices are called *multiple edges*. A graph is considered *simple* if it contains no self loops or multiple edges. An *undirected* graph is one where $(v_i, v_j) \in E \Leftrightarrow (v_j, v_i) \in E$. A graph is called *unlabeled* if there is no distinction between vertices other than their interconnectivity. In this paper, we consider simple, connected, undirected, and unlabeled graphs.

Given a graph G, a graph $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. A subgraph $G' = (V', E')$ is called an *induced (or vertex-induced)* subgraph of G if $E' = \{(v_i, v_j) \mid (v_i, v_j) \in E \text{ and } v_i, v_j \in V'\}$, that is, all edges in E, between two vertices $v_i, v_j \in V'$, are also present in E'. Two graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* if there exists a bijection $f : V \to V'$, called *isomorphism*, such that $(v_i, v_j) \in E \Leftrightarrow (f(v_i), f(v_j)) \in E'$ for all $v_i, v_j \in V$.

Suppose we have empirical data $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$, where the domain $\mathcal{X}$ is a nonempty set of *inputs* x_i and $\mathcal{Y}$ is a set of corresponding *targets* y_i. A kernel method predicts the target y of a new input x based on existing "similar" inputs and their outputs (x_i, y_i). A function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ that measures the similarity between two inputs is called a kernel function, or simply a kernel. A kernel function k is often defined as an inner product of two vectors in a *feature space* $\mathcal{H}$:

$$k(x, x') = \langle \phi(x), \phi(x') \rangle = \langle \mathbf{x}, \mathbf{x}' \rangle$$

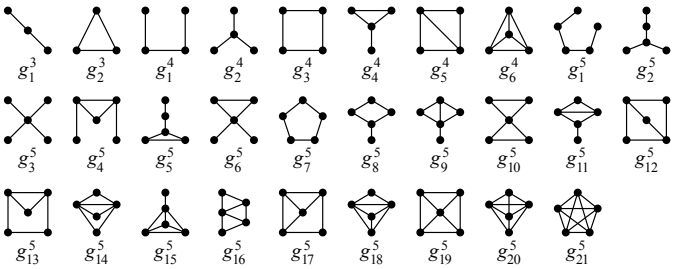


Fig. 2. All connected graphlets of 3, 4, or 5 vertices.

where $\phi : \mathcal{X} \mapsto \mathcal{H}$ is called a *feature map* which maps an input x to a *feature vector* $\mathbf{x}$ in $\mathcal{H}$.

2.2 Measuring Topological Similarities between Graphs

Based on our assumption, we need to measure the topological similarities between graphs. Depending on the discipline, this problem is called *graph matching*, *graph comparison*, or *network alignment*. In the last five decades, numerous approaches have been proposed to this problem [22, 29]. Each approach measures the similarity between graphs based on different aspects, such as isomorphism relation [48, 79, 92], graph edit distance [11], or graph measures [4, 19]. However, many of these traditional approaches are either computationally expensive, not expressive enough to capture the topological features, or difficult to adapt to different problems [64].

Graph kernels have been recently introduced for measuring pairwise similarities between graphs in the field of machine learning. They allow us to apply many different kernelized machine learning techniques, e.g. SVM [18], on graph problems, including graph classification problems found in bioinformatics and chemoinformatics.

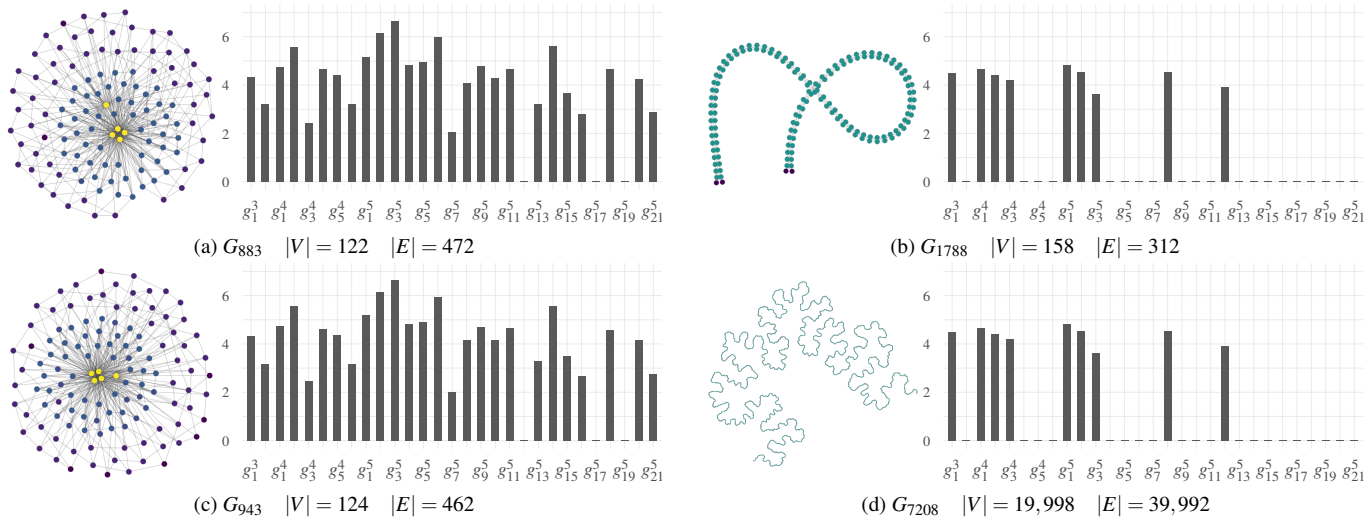
A graph kernel can be considered to be an instance of an R-convolution kernel [42]. It measures the similarity between two graphs based on the recursively decomposed substructures of said graphs. The measure of similarity varies with each graph kernel based on different types of substructures in graphs. These substructures include walks [35, 50, 85], shortest paths [8], subtrees [72, 74, 75], cycles [44], and graphlets [76]. Selecting a graph kernel is challenging as many kernels are available. To exacerbate the problem, there is no theoretical justification as to why one graph kernel works better than another for a given problem [76].

Many graph kernels often have similar limitations as previously mentioned approaches. They do not scale well to large graphs (time complexity of $O(|V|^3)$ or higher) or do not work well for unlabeled graphs. To overcome this problem, a graph kernel based on sampling a fixed number of graphlets has been introduced to be accurate and efficient on large graphs [64, 76].

Graphlets are small, induced, and non-isomorphic subgraph patterns in a graph [67] (Fig. 2). Graphlet frequencies (Fig. 3) have been used to characterize biological networks [66, 67], identify disease genes [59], and analyze social network structures [82]. Depending on the definition, the relative frequencies are called *graphlet frequency distribution*, *graphlet degree distribution*, or *graphlet concentrations*. While these works often have different definitions, the fundamental idea is to count the individual graphlets and compare their relative frequencies of occurrence between graphs. A graph kernel based on graphlet frequencies, called a graphlet kernel, was first proposed by Shervashidze et al. [76]. The main idea is to use a graphlet frequency vector as the feature vector of a graph, then compute the similarity between graphs by defining the inner product of the feature vectors.

3 Approach

Our approach is a supervised learning of the relationship between topological features of existing graphs and their various layout results. This also includes the layouts' aesthetic metrics. Like many supervised learning methods, our approach requires empirical data (training data) of existing graphs, their layout results, and corresponding aesthetic metrics. This generally takes a considerable amount of time, but can



(a) G_{883} $|V| = 122$ $|E| = 472$ (b) G_{1788} $|V| = 158$ $|E| = 312$

(c) G_{943} $|V| = 124$ $|E| = 462$ (d) G_{7208} $|V| = 19,998$ $|E| = 39,992$

Fig. 3. Examples of graphlet frequencies. The x-axis represents connected graphlets of size $k \in \{3,4,5\}$ and the y-axis represents the weighted frequency of each graphlet. Four graphs are drawn with sfdp layouts [45]. If two graphs have similar graphlet frequencies, i.e., high topological similarity, they tend to have similar layout results (a and c). If not, the layout results look different (a and b). However, in rare instances, two graphs can have similar graphlet frequencies (b and d), but vary in graph size, which might lead to different looking layouts.

be considered a preprocessing step as it is only ever done once. The benefit of machine learning is that as we add more graphs and their layout results, the performance generally improves.

In this section, we introduce:

1. A framework for designing better graphlet kernels
2. A process of using graph kernels to determine what a graph would look like in different layouts
3. A method to estimate the aesthetic metrics without calculating actual layouts

3.1 Graphlet Kernel Design Framework

One of the key challenges of our approach is choosing a graph kernel. While many graph kernels are available, we focus on sampling based graphlet kernels because they are computationally efficient and are designed for unlabeled graphs. To improve the performance of a graphlet kernel, we introduce a framework for designing graphlet kernels. Our framework consists of three steps:

1. Sampling graphlet frequencies
2. Scaling graphlet frequency vectors
3. Defining an inner product between two graphlet frequency vectors

For each step, we discuss several possible design choices. Our framework defines several new types of graphlet kernels.

Existing studies related to graphlets and graphlet kernels are scattered throughout the literature, including machine learning and network science. Each of the studies focuses on certain aspects of a graphlet kernel. Our framework unifies the related works for graphlet kernels.

3.1.1 Sampling Graphlet Frequencies

One of the challenges for constructing a graphlet kernel is computing the graphlet frequencies. Exhaustive enumeration of graphlets with k vertices in a graph G is $O(|V|^k)$, which is prohibitively expensive, even for graphs with a few hundred or more vertices. Thus, sampling approaches have been introduced to obtain the graphlet frequencies in a short amount of time with an acceptable error.

Random vertex sampling (RV): Most existing works on graphlet kernels have sampled graphlets using a random vertex sampling method [9]. To sample a graphlet of size k, this method randomly chooses k vertices in V and induces the graphlet based on their adjacency in G. This step is repeated until the number of sampled graphlets is sufficient. Since this sampling method randomly chooses k vertices without considering their interconnectivity in G, it has several limitations. As many real-world networks are sparse [4], $|E| \ll O(|V|^2)$, most randomly sampled graphlets are disconnected. Consequently, the frequency of disconnected graphlets would be much higher than connected ones. If disconnected graphlets lack discriminating traits between graphs, and they outnumber the informative graphlets, comparing graphs becomes increasingly difficult. While we could only sample connected graphlets by excluding disconnected ones, this requires a tremendous number of sampling iterations to sample a sufficient number of connected graphlets. Since there is no lower bound on the number of iterations for sampling certain amounts of connected graphlets, using RV to sample only connected graphlets would lead to undesirable computation times.

Random walk sampling (RW): There are other methods to sample graphlets based on random walks, such as Metropolis-Hasting random walk [71], subgraph random walk [88], and expanded Markov chain [13]. However, they have not been used for designing graphlet kernels in the machine learning community. Unlike RV sampling, they sample graphlets by traversing the structure of a given graph. That is, they search for the next sample vertices within the neighbors of the currently sampled vertices. Thus, they are able to sample connected graphlets very well.

3.1.2 Scaling Graphlet Frequency Vector

A graphlet frequency vector $\mathbf{x}$ is defined such that each component x_i corresponds to the relative frequency of a graphlet g_i. In essence, the graphlet frequency vector of a graph is the feature vector of the graph.

Linear scale (LIN): Many existing graphlet kernels use linear scaling, often called *graphlet concentration*, which is the percentage of each graphlet in the graph. As several works use weighted counts w_i of each graphlet g_i, this scaling can be defined as:

$$x_i = \frac{w_i}{\sum w_i}$$

Logarithmic scale (LOG): Similar to the vertex degree distribution, the distribution of graphlet frequency often exhibits a power-law distribution. This again can cause a significant problem if graphlets that lack discriminating traits between graphs outnumber the informative graphlets. Thus, several studies [67, 71] used a logarithmic scale of the graphlet frequency vector to solve this problem. While the exact definitions of these methods differ, we generalize it using the following definition:

$$x_i = \log\left(\frac{w_i + w_b}{\sum(w_i + w_b)}\right)$$

where $w_b > 0$ is a base weight to prevent $\log 0$.

3.1.3 Defining Inner Product

Several kernel functions can be used to define the inner product in a feature space $\mathcal{H}$.

Cosine similarity (COS): Most existing graphlet kernels use the dot product of two graphlet frequency vectors in Euclidean space, then normalize the kernel matrix. This is equivalent to the cosine similarity of two vectors, which is the L_2-normalized dot product of two vectors:

$$\langle \mathbf{x}, \mathbf{x}' \rangle = \frac{\mathbf{x} \cdot \mathbf{x}'^{\mathsf{T}}}{\|\mathbf{x}\| \, \|\mathbf{x}'\|}$$

Gaussian radial basis function kernel (RBF): This kernel is popularly used in various kernelized machine learning techniques:

$$\langle \mathbf{x}, \mathbf{x}' \rangle = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2} \right)$$

where σ is a free parameter.

Laplacian kernel (LAPLACIAN): Laplacian kernel is a variant of RBF kernel:

$$\langle \mathbf{x}, \mathbf{x}' \rangle = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_1}{\sigma} \right)$$

where $\|x - x'\|_1$ is the L_1 distance, or Manhattan distance, of the two vectors.

3.2 What Would a Graph Look Like in This Layout? (WGL)

Graph kernels provide us with the topological similarities between graphs. Using these pairwise similarities, we design a nearest-neighbor based method, similar to *k*-nearest neighbors, to show what a graph would look like in different layouts. Given a new input graph G_{input}, we find the *k* most topologically similar graphs and show their existing layout results to the users. Thus, if our assumption is true, users can expect the layout results of new input graphs by looking at the layout results of topologically similar graphs.

While graph kernels are able to find topologically similar graphs, many of them do not explicitly take the size of graphs into account. In rare cases, it is possible to find topologically similar graphs that vary in size. For instance, Fig. 3b and Fig. 3d have similar graphlet frequencies, yet have different layout results. To prevent this, we add some constraints when we find similar graphs, such as only counting a graph that has more than half and less than double the number of vertices in the input graph.

As a result, for the new input graph G_{input}, we find *k* most similar graphs as follows:

1. Compute the similarity between existing graphs and G_{input}
2. Remove the graphs that do not satisfy a set of constraints
3. Select the *k* most similar graphs

After we have obtained the *k* most similar graphs to the input graph G_{input}, we show their existing layout results to the user.

3.3 Estimating Aesthetic Metrics (EAM)

As the aesthetic metrics are continuous values, estimating the aesthetic metrics is a regression problem. There are several kernelized regression models, such as Support Vector Regression (SVR) [78], that can be used for estimating the aesthetic metrics based on the similarities between graphs obtained by graph kernels. Computing actual aesthetic metrics of a layout requires a calculation of the layout first. However, our approach is able to estimate the metrics without calculating actual layouts.

Training: To estimate the aesthetic metrics, we first need to train a regression model by:

1. Prepare the training data (layouts and their aesthetic metrics of existing graphs)
2. Compute a kernel matrix using a graph kernel (pairwise similarities between all graphs in the training data)
3. Train regression model

Estimation: To make an estimation of a new input graph, the following steps are required:

1. Compute similarity between the input graph and other graphs in the training data
2. Estimate value using the trained regression model

4 EVALUATION 1: ESTIMATING LAYOUT AESTHETIC METRICS

There are several questions we wanted to answer within this evaluation:

- Is our method able to accurately estimate the layout's aesthetic metrics without computing the layout?
- Is our method able to quickly obtain the estimations?
- Does our graph kernels, derived from our framework, outperform state-of-the-art graph kernels in terms of computation time and estimation accuracy?

We describe the experimental design, apparatus, implementation, and metrics used in the study. We also answer each of these questions in the results section.

4.1 Experimental Design

We perform 10-fold cross-validations to compare 13 different graph kernels in terms of their accuracy and computation times for estimating four aesthetic metrics on eight layout methods.

4.1.1 Datasets

We started by collecting around 3,700 graphs from [21], which includes, but not limited to, social networks, web document networks, and geometric meshes. Without loss of generality, graphs with multiple connected components were broken down into separate graphs (one connected component to one graph). After that, we removed any graph with less than 100 vertices, as there would be little benefit from a machine learning approach since most layout methods are fast enough for small graphs. This left us with a total of 8,263 graphs for our study. The graphs range from 100 vertices, and 100 edges up to 113 million vertices and 1.8 billion edges. More details about the graphs, such as characteristic measures and layout results, can be found in [1].

Not all graphs were used for each layout method, as some layout algorithms failed to compute the results within a reasonable amount of time (10 days) or ran out of memory. Exact numbers of graphs used for each layout are reported in [1].

4.1.2 Kernels

We compare a total of 13 graphlet kernels. Using our framework, 12 graphlet kernels are derived from a combination of 2 graphlet sampling methods (RV and RW) $\times$ 2 types of graphlet frequency vector scaling (LIN and LOG) $\times$ 3 inner products (COS, RBF, and LAPLACIAN). We denote a kernel derived from our framework by a combination of the above abbreviations. For example, RW-LOG-LAPLACIAN denotes a graphlet kernel which samples graphlets based on a random walk, uses a log scaled graphlet frequency vector, and computes the similarity using the Laplacian kernel function. We also compare with state-of-the-art graphlet kernels. The original graphlet kernel [76] can be constructed using our framework and is included in the 12 kernels (RV-LIN-COS). Lastly, the 13$^{\text{th}}$ graph kernel is a Deep Graphlet Kernel (DGK) [91].

We used a sampling method from Chen et al. [13] as the RW sampling. For all kernels, we sampled 10,000 graphlets of 3, 4, and 5 vertices for each graph. The graphlets of 3, 4, and 5 vertices are widely used due to computational costs. Also, the graphlets of 6 or more vertices are rare [71]. RW sampling considers only connected graphlets, as in [13], while RV sampling counts both connected and disconnected graphlets, as in [76, 91]. All kernel matrices are normalized such that the similarity between a graph and itself has a value of 1.

4.1.3 Layouts

The process of laying out a graph has been actively studied for over five decades. Several studies [23, 24, 36, 81, 86] provide a comprehensive review of these layout methods. While there are methods designed for specific purposes, such as orthogonal layout methods [52], in this work, we focus on two-dimensional layout methods that draw all edges in straight lines. Due to the volume of layout methods proposed, evaluating all the methods is impractical. For this evaluation, we used eight representative layout methods of five families based on groupings found in [36, 86]. Our selection process prioritized methods that have been popularly used, have a publicly available implementation, and have a proven track record over other state-of-the-art methods.

Force-directed methods: Force-directed methods are based on a physical model of attraction and repulsion. They are among the first layout methods to be developed and are some of the most commonly used layout methods today. In general, force-directed layout methods fall within two groups: spring-electrical [27, 30, 32] and energy-based approaches [20, 49]. We selected one from each group: Fruchterman-Reingold (FR) [32] from spring-electrical approaches and Kamada-Kawai (KK) [49] from energy-based approaches.

Dimension reduction based method: Dimension reduction techniques, including multidimensional scaling (MDS) or principal component analysis (PCA), can be used to lay out a graph using the graph-theoretic distance between node pairs. PivotMDS [10] and High-Dimensional Embedder (HDE) [41] work by assigning several vertices as the pivots, then constructing a matrix representation with graph-theoretic distances of all vertices from the pivots. Afterward, dimension reduction techniques are applied to the matrix. We selected HDE [41] in this family.

Spectral method: Spectral layout methods use the eigenvectors of a matrix, such as the distance matrix [16] or the Laplacian matrix [53] of the graph, as coordinates of the vertices. We selected the method by Koren [53] in this family.

Multi-Level methods: Multilevel layout methods are developed to reduce computation time. These multilevel methods hierarchically decompose the input graph into coarser graphs. Then, they lay out the coarsest graph and use the vertex position as the initial layout for the next finer graph. This process is repeated until the original graph is laid out. Several methods can be used to lay out the coarse graph, such as a force-directed method [34, 37, 40, 45, 87], a dimension reduction based method [17], or a spectral method [31, 54]. We selected sfdp [45] and FM^3 [37] in this family.

Clustering based methods: Clustering based methods are designed to emphasize graph clusters in a layout. When the graph size is large, users tend to ignore the number of edge crossings in favor of well-defined clusters [84]. We selected the Treemap based layout [60] and the Gosper curve based layout [61] from this family, which utilizing the hierarchical clustering of a graph to lay out the graph.

4.1.4 Aesthetic Metrics

Aesthetic criteria, e.g., minimizing the number of edge crossings, are used for improving the readability of a graph layout. Bennett et al. [7] reviewed various aesthetic criteria from a perceptual basis. Aesthetic metrics enable a quantitative comparison of the aesthetic quality of different layouts. While there are many aesthetic criteria and metrics available, many of them are informally defined or are defined only for specific types of layout methods. Many also do not have a normalized metric for comparing graphs of different sizes, or are too expensive to compute (e.g., symmetry metric in [68] is $O(n^7)$). In this evaluation, we chose four aesthetic metrics because they have a normalized form, are not defined only for specific types of layouts, and can be computed in a reasonable amount of time.

Crosslessness (m_c) [68]: *Minimizing the number of edge crossings* has been found as one of the most important aesthetic criteria in many studies [46, 52, 69, 70]. The crosslessness m_c is defined as

$$m_c = \begin{cases} 1 - \dfrac{c}{c_{\max}}, & \text{if } c_{\max} > 0 \\ 1, & \text{otherwise} \end{cases}$$

where c is the number of edge crossings and $c_{\max}$ is the approximated upper bound of the number of edge crossings, which is defined as

$$c_{\max} = \frac{|E|\,(|E|-1)}{2} - \frac{1}{2}\sum_{v \in V} (\deg(v)\,(\deg(v)-1))$$

Minimum angle metric (m_a) [68]: This metric quantifies the criteria of *maximizing the minimum angle between incident edges on a vertex*. It is defined as the average absolute deviation of minimum angles between the incident edges on a vertex and the ideal minimum angle $\theta(v)$ of the vertex:

$$m_a = 1 - \frac{1}{|V|}\sum_{v \in V} \left| \frac{\theta(v) - \theta_{\min}(v)}{\theta(v)} \right|, \;\; \theta(v) = \frac{360^\circ}{\deg(v)}$$

where $\theta_{\min}(v_i)$ is the minimum angle between the incident edges on the vertex.

Edge length variation (m_l) [38]: *Uniform edge lengths* have been found to be effective aesthetic criteria for measuring the quality of a layout in several studies [52]. The coefficient of variance of the edge length (l_{cv}) has been used to quantify this criterion [38]. Since the upper bound of the coefficient of variation of n values is $\sqrt{n-1}$ [51], we divide l_{cv} by $\sqrt{|E|-1}$ to normalize:

$$m_l = \frac{l_{\text{cv}}}{\sqrt{|E|-1}}, \;\; l_{\text{cv}} = \frac{l_\sigma}{l_\mu} = \sqrt{\frac{\sum_{e \in E}(l_e - l_\mu)^2}{|E| \cdot l_\mu^2}}$$

where l_σ is the standard deviation of the edge length and l_μ is the mean of the edge length.

Shape-based metric (m_s) [28]: Shape-based metric is a more recent aesthetic metric and was proposed for evaluating the layouts of large graphs. The shape-based metric m_s is defined by the mean Jaccard similarity (MJS) between the input graph G_{input} and the shape graph G_{S}:

$$m_s = \text{MJS}(G_{\text{input}}, G_{\text{S}}), \;\; \text{MJS}(G_1, G_2) = \frac{1}{|V|}\sum_{v \in V} \frac{|N_1(v) \cap N_2(v)|}{|N_1(v) \cup N_2(v)|}$$

where $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ are two graphs with the same vertex set and $N_i(u)$ is the set of neighbours of v in G_i. We use the Gabriel graph [33] as the shape graph.

4.2 Apparatus and Implementation

We perform 10-fold cross-validations of ε-SVR implemented by [12]. To remove random effects of the fold assignments, we repeat the whole experiment 10 times and report mean accuracy metrics (Table 1).

We obtained the implementation of DGK [91] from the authors. The implementation of each layout method was gathered from: sfdp [65], FM^3 [14], FR [65], KK [14], Spectral [39], and Treemap [60] and Gosper [61] are provided by the authors. Other kernels and layout methods were implemented by us. For crosslessness (m_c), a GPU-based massively parallel implementation was used. For other metrics, parallel CPU-based implementations written in C++ were used. The machine we used to generate the training data and to conduct the experiment has two Intel Xeon processors (E5-4669 v4) with 22 cores (2.20 GHz) each, and two NVIDIA Titan X (Pascal) GPUs.

4.3 Accuracy Metrics

Root-Mean-Square Error (RMSE) measures the difference between measured values (ground truth) and estimated values by a model. Given a set of measured values $\mathcal{Y} = \{y_1, \ldots, y_n\}$ and a set of estimated values $\tilde{\mathcal{Y}} = \{\tilde{y_1}, \ldots, \tilde{y_n}\}$, the RMSE is defined as:

$$\text{RMSE}(\mathcal{Y}, \tilde{\mathcal{Y}}) = \sqrt{\frac{1}{n}\sum_i (y_i - \tilde{y}_i)^2}$$

The coefficient of determination (R^2) shows how well a model "fits" the given data. The maximum R^2 score is 1.0 and it can have an arbitrary negative value. Formally, it indicates the proportion of the variance in the dependent variable that is predictable from the independent variable, which is defined as:

$$R^2(\mathcal{Y}, \tilde{\mathcal{Y}}) = 1 - \sum_i (y_i - \tilde{y}_i)^2 \Big/ \sum_i (y_i - y_\mu)^2$$

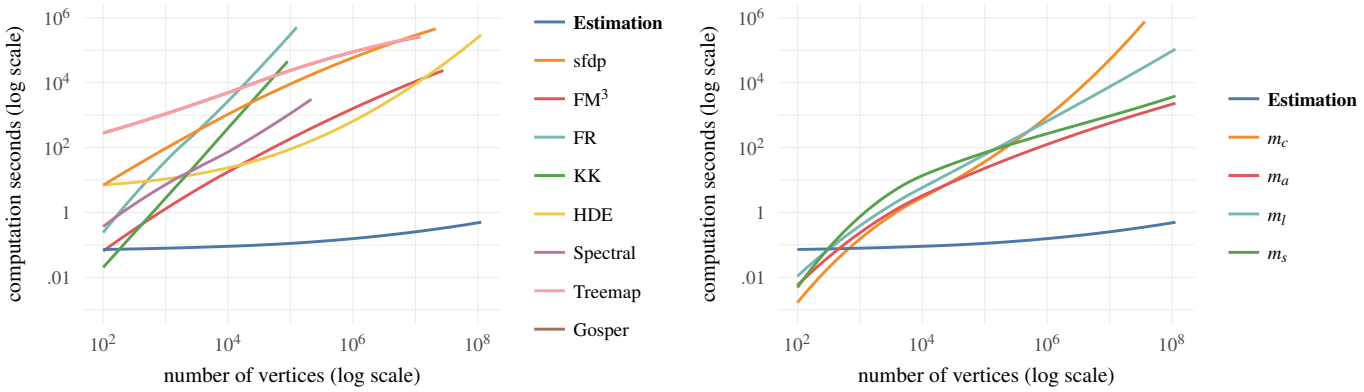
where y_μ is the mean of measured values y_i.

4.4 Results

We report the accuracy and computation time for estimating the aesthetic metrics.

Table 1. Estimation accuracy of the two most accurate kernels and the state-of-the-art kernels. We report Root-Mean-Square Error (RMSE) and the coefficient of determination (R^2) of estimation of four aesthetic metrics on eight layout methods.

Kernel		sfdp		FM^3		FR		KK		Spectral		HDE		Treemap		Gosper	
		RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2
Rank 1 RW-LOG-LAPLACIAN	m_c	**.0175**	**.9043**	.0468	.7319	**.0257**	**.8480**	**.0346**	**.8223**	**.1120**	**.6947**	**.0903**	**.8130**	**.0836**	**.6399**	**.0857**	**.6199**
	m_a	**.1011**	**.8965**	**.1041**	**.8919**	**.0982**	**.9004**	**.1024**	**.8876**	**.1153**	**.8793**	**.1152**	**.8666**	**.1053**	**.8552**	**.1071**	**.8580**
	m_l	**.0055**	**.9021**	**.0048**	**.8531**	**.0055**	**.9028**	**.0105**	**.4549**	**.0505**	**.6203**	**.0155**	**.5961**	**.0047**	**.8666**	**.0066**	**.8444**
	m_s	**.0514**	**.9060**	**.0474**	**.9325**	**.0417**	**.8533**	**.0485**	**.9084**	**.0534**	**.9031**	**.0486**	**.8942**	**.0112**	**.8429**	**.0323**	**.7495**
Rank 2 RW-LOG-RBF	m_c	.0176	.9036	**.0446**	**.7568**	.0279	.8218	.0350	.8182	.1138	.6845	.0917	.8072	.0841	.6356	.0882	.5976
	m_a	.1070	.8840	.1102	.8788	.1023	.8920	.1061	.8793	.1193	.8706	.1202	.8546	.1101	.8416	.1125	.8434
	m_l	.0062	.8793	.0050	.8412	.0059	.8874	.0106	.4497	.0519	.5992	.0167	.5291	.0052	.8417	.0073	.8127
	m_s	.0556	.8900	.0542	.9116	.0459	.8227	.0547	.8833	.0576	.8875	.0537	.8708	.0116	.8299	.0323	.7491
Rank 11 RV-LIN-COS [76]	m_c	.0387	.5312	.0771	.2716	.0577	.2364	.0783	.0916	.1533	.4280	.1770	.2827	.1324	.0978	.1336	.0763
	m_a	.2883	.1581	.2907	.1570	.2817	.1805	.2850	.1292	.3019	.1723	.2978	.1080	.2688	.0557	.2726	.0801
	m_l	.0168	.0972	.0121	.0561	.0169	.0895	.0138	.0609	.0812	.0200	.0239	.0403	.0116	.2026	.0156	.1378
	m_s	.1721	−.0552	.1904	−.0890	.0984	.1850	.1653	−.0656	.1777	−.0729	.1538	−.0606	.0246	.2373	.0628	.0521
Rank 12 DGK [91]	m_c	.0399	.5029	.0783	.2500	.0583	.2207	.0803	.0448	.1564	.4041	.1804	.2541	.1358	.0489	.1345	.0630
	m_a	.2891	.1536	.2924	.1467	.2837	.1690	.2862	.1217	.3052	.1537	.3003	.0930	.2716	.0357	.2754	.0612
	m_l	.0175	.0246	.0126	−.0134	.0177	.0047	.0140	.0294	.0811	.0203	.0243	.0018	.0128	.0029	.0185	−.3883
	m_s	.1756	−.0982	.1928	−.1171	.1077	.0236	.1676	−.0953	.1807	−.1094	.1550	−.0771	.0286	−.0846	.0682	−.1187



(a) Layout computation and estimation time.

(b) Aesthetic metric computation and estimation time.

Fig. 4. Computation time results in log scale. The plots show estimation times for our RW-LOG-LAPLICAN kernel, which has the highest accuracy. The plot on the left shows layout computation time while the plot on the right shows aesthetic metric computation time. As the number of vertices increases, the gap between our estimation and layout methods enlarges in both layout time and aesthetic metric computation time. Some layout methods could not be run on all graphs due to computation time and memory limitation of the implementation. Treemap and Gosper overlap in the layout plot because the majority of the computation time is spent on hierarchical clustering.

4.4.1 Estimation Accuracy

Due to space constraints, we only reported the results of the two most accurate kernels and state-of-the-art kernels [76, 91] in Table 1. The results shown are mean RMSE (lower is better) and mean R^2 (higher is better) from 10 trials of 10-fold cross-validations. The standard deviations of RMSE and R^2 are not shown because the values are negligible: all standard deviations of RMSE are lower than .0006, and all standard deviations of R^2 are lower than .0075. We ranked the kernels based on the mean RMSE of all estimations.

The most accurate kernel is our RW-LOG-LAPLACIAN kernel. Except for the crosslessness (m_c) of FM^3, our RW-LOG-LAPLACIAN kernel shows best estimation results in both RMSE and R^2 score for all four aesthetic metrics on all eight layout methods. The second most accurate kernel is our RW-LOG-RBF kernel and is the best for crosslessness (m_c) of FM^3.

Our RW-LOG-LAPLACIAN kernel (mean RMSE = .0557 and mean R^2 = .8169) shows an average of 2.46 times lower RMSE than existing kernels we tested. The original graphlet kernel [76] (mean RMSE = .1366 and mean R^2 = .1216), which is denoted as RV-LIN-COS, ranked 11$^{\text{th}}$. The DGK [91] (mean RMSE = .1388 and mean R^2 = .0540) ranked 12$^{\text{th}}$.

Within the kernels we derived, the kernels using RW sampling show higher accuracy (mean RMSE = .0836 and mean R^2 = .6247) than ones using RV sampling (mean RMSE = .1279 and mean R^2 = .2501). The kernels using LOG scaling show higher accuracy (mean RMSE = .0955 and mean R^2 = .5279) than ones using LIN (mean RMSE = .1160 and mean R^2 = .3469). The kernels using LAPLACIAN as the inner product show higher accuracy (mean RMSE = .0956 and mean R^2 = .5367) than ones using RBF (mean RMSE = .1047 and mean R^2 = .4393) and COS (mean RMSE = .1169 and mean R^2 = .3362).

4.4.2 Computation Time

Since some algorithms are implemented in parallel while others are not, we report CPU times. The estimation time is comprised of the computation steps required for estimation in Sect. 3.3. Our RW-LOG-LAPLACIAN kernel, which shows the best estimation accuracy, also shows the fastest computation time for estimation. On average, it takes .14093 seconds (SD = 1.9559) per graph to make the estimations. Fig. 4 shows the computation time for layouts and their aesthetic metrics.

Most of the time spent on estimation was devoted to sampling graphlets. The RW sampling [13] shows fastest computation time, with an average of .14089 seconds (SD = 1.9559) per graph. The graphlet sampling for DGK take longer than RW, with an average of 3.38 seconds (SD = 7.88) per graph. The RV sampling take the longest time, with an average of 6.81 seconds (SD = 7.04).

4.5 Discussion

Our graph kernels perform exceptionally well in both estimation accuracy and computation time. Specifically, RW-LOG-LAPLACIAN outperforms all other kernels in all metrics except crosslessness (m_c) on the FM^3. RW-LOG-RBF is the best performing kernel on FM^3's crosslessness (m_c) and is the second best kernel. Existing graph kernels are ranked in the bottom three of all kernel methods we tested.

A possible explanation for this is that certain types of graphlets are essential for accurate estimation. The kernels using RW sampling, which samples connected graphlets very efficiently, show a higher accuracy than other kernels. While disconnected graphlets are shown to be essential for classification problems in bioinformatics [76], for our problem, we suspect that connected graphlets are more important for accurate estimation.

Other sampling methods are not suitable for sampling connected graphlets. There are 49 possible graphlets, where the size of each graphlet is $k \in \{3,4,5\}$, and 29 of them are connected graphlets (Fig. 2). However, when we sample 10,000 graphlets per graph using RV sampling, on average only 1.913% ($SD = 6.271$) of sampled graphlets are connected graphlets. Furthermore, 35.77% of graphs have no connected graphlets in the samples even though all graphs are connected, making it impossible to effectively compare graphs.

The kernels using LOG scaling show more accurate estimations than the ones using LIN. We suspect this is because of the distribution of graphlets, which often exhibit a power-law distribution. Thus, when using LOG scaling, a regression model is less affected by the graphlets with overwhelming frequencies and becomes better at discriminating graphs with different structures.

Our estimation times are fast and scale well, as shown in Fig. 4. At around 200 vertices, our estimation times become faster than all other layout computation times. Our estimation times also outperform the computations of the four aesthetic metrics past 1,000 vertices. As the size of a graph increases, the differences become larger, to the point that our RW-LOG-LAPLACIAN takes several orders of magnitude less time than both layout computation and metric computation. Normally the layout has to be calculated in order to calculate aesthetic metrics. This is not the case for our estimation as the aesthetic metrics can be estimated without the layout result, leading to a considerable speed up.

It is interesting to see that each sampling method shows different computation times, even though we sampled the same amount of graphlets. A possible explanation for this discrepancy can be attributed to locality of reference. We stored the graphs using an adjacency list data structure in memory. Thus, RW sampling, which finds the next sample vertices within the neighbors of currently sampled vertices, tends to exhibit good locality. On the other hand, RV sampling chooses vertices randomly, thus it would show poor locality which leads to cache misses and worse performance. The sampling method of DGK is a variant of RV. After one graphlet is randomly sampled, its immediate neighbors are also sampled. This would have better locality than RV and could explain the better computation time.

In terms of training, except for DGK, all other kernels spend negligible times computing the kernel matrix, with an average of 5.99 seconds ($SD = 3.26$). However, computing the kernel matrix of DGK takes a considerable amount of time because it requires computation of language modeling (implemented by [73]). On average it take 182.96 seconds ($SD = 9.31$).

Since there are many parameters of each layout method, and most parameters are not discrete, it is impossible to test all combinations of parameter settings. To simplify the study, we only used default parameters for each layout method. It is possible to apply our approach to different predefined parameter settings on the same layout method. However, generating new training data for multiple settings can be time consuming.

5 EVALUATION 2: WHAT WOULD A GRAPH LOOK LIKE IN THIS LAYOUT?

In this section, we describe our user study that evaluates how well our WGL method (Sect. 3.2) is able to find graphs that users assess as being perceptually similar to the actual layout results.

5.1 Experimental Design

We designed a ranking experiment to compare topological similarity ranks (r_T) obtained by our WGL method and perceptual similarity ranks (r_P) assessed by humans. That is, if both our WGL method and participants' choices match, then we can conclude our WGL method is able to find perceptually similar graphs to the actual layout results.

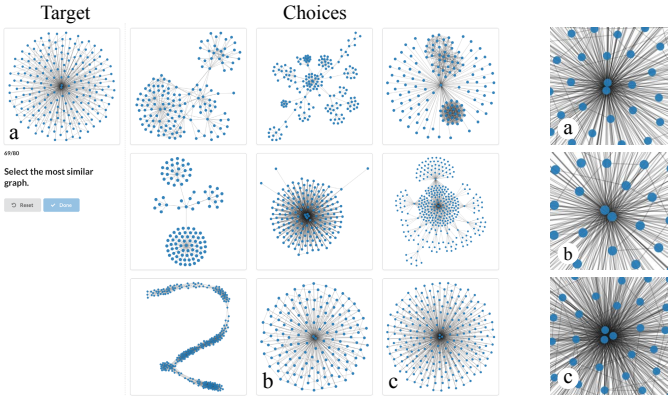


Fig. 5. A task from the user study. For each task, the participants were given one target graph and nine choice graphs. They were then asked to rank the three most similar graphs in order of decreasing similarity. The three images on the right show the central nodes of (a), (b), and (c).

5.1.1 Task

For each task, participants were given one target graph and nine choice graphs. An example of a task given to participants is shown in Fig. 5. They were asked to rank the three most similar graphs to the target graph in order of decreasing similarity. The instructions were given as such, "select the most similar graph." for the first choice and "select the next most similar graph." for the second and third choice. To avoid biases, we did not specify what is "similar" and let the participants decide for themselves. In each task, the same layout method was used for target and choice graphs. We did not notify this to the participants. To test our approach using graphs with different topological structures and different layout methods, we used 72 tasks comprised of nine graphs and eight layout methods.

While other task designs are possible, such as participants ranking all nine graphs or individually rating all nine choices, during our pilot study we found that these task designs are overwhelming to the participants. The participants found it very difficult to rank between dissimilar graphs. For example, in Fig. 5, selecting the fourth or fifth most similar graph is challenging as there is little similarity to the target graph after the first three choices. Also, the focus of our evaluation is to determine how well our WGL method is able to find similar graphs, not dissimilar graphs. Thus, we only need to see which graphs our participants perceive as similar in the task, which simultaneously reduces task load when compared to ranking all nine graphs.

5.1.2 Graphs

To gather an unbiased evaluation, the target graphs and their choice graphs must be carefully selected.

Selecting target graphs. To test our approach on graphs with different topological structures, we select nine target graphs as follows: We clustered the 8,263 graphs into nine groups using spectral clustering [77]. For each cluster, we select ten representative graphs which have the highest sum of topological similarity within the cluster (i.e., the ten nearest graphs to the center of each cluster). Then, we randomly select one graph from these ten representative graphs to be the target graph. Fig. 1 shows the selected nine target graphs in FM^3 layout [37].

Selecting choice graphs. To test our approach on graphs with different topological similarities obtained by graph kernels, we select nine choice graphs for each target graph as follows: We compute nine clusters of graphs based on the topological similarity between a graph and the target graph using Jenks natural breaks [47], which can be seen as one dimensional k-means clustering. We designate one of the nine choice graphs to represent what our approach would predict. This is done by selecting the graph with the highest similarity to the target graph from the cluster nearest to the target graph. For the other eight clusters, we randomly select one choice graph from a set of ten graphs that have the highest sum of similarity within the cluster (i.e., the ten nearest graphs to the center of each cluster). These can be considered as

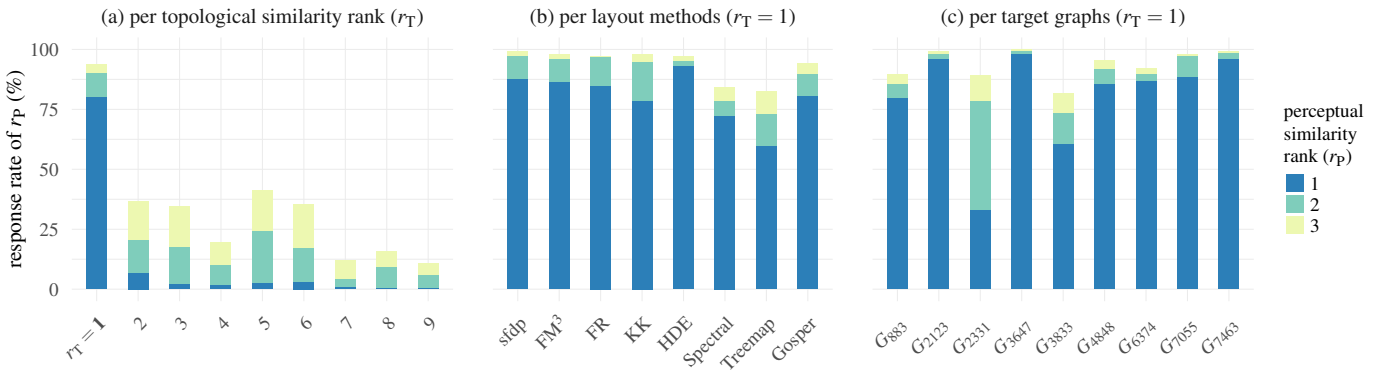


Fig. 6. Summary of the user study results. (a) response rate of perceptual similarity rank (r_P) for each topological similarity rank (r_T). The plot in the middle (b) shows the response rate on topological similarity rank of 1 per layout method while the plot on the right (c) shows per target graph.

Table 2. Descriptive statistics of perceptual similarity rank (r_P). μ: mean, SD: standard deviation, $\tilde{r}_P$: median, and IQR: interquartile range. Our predicted choices are ranked on average 1.35 by the participants.

	1	2	3	4	5	6	7	8	9
μ	**1.35**	3.36	3.45	3.68	3.31	3.45	3.82	3.74	3.83
SD	.82	.96	.84	.71	.91	.84	.53	.63	.53
$\tilde{r}_P$	**1**	4	4	4	4	4	4	4	4
IQR	0	1	1	0	1	1	0	0	0

(a) per topological similarity rank (r_T)

sfdp	FM^3	FR	KK	HDE	Spct.	Tree.	Gos.
1.16	1.19	1.21	1.29	1.14	1.65	1.84	1.35
.46	.55	.6	.63	.58	1.14	1.17	.81
1	1	1	1	1	1	1	1
0	0	0	0	0	1	2	0

(b) per layout methods ($r_T = 1$)

G_{883}	G_{2123}	G_{2331}	G_{3647}	G_{3833}	G_{4848}	G_{6374}	G_{7055}	G_{7463}
1.45	1.07	2	1.02	1.85	1.27	1.32	1.17	1.06
.98	.36	.94	.17	1.18	.74	.86	.53	.34
1	1	2	1	1	1	1	1	1
0	0	1	0	2	0	0	0	0

(c) per target graphs ($r_T = 1$)

representative graphs of each cluster. A list of the selected nine choice graphs for each target graph can be found in [1].

When we select a choice graph, we filter out graphs that have less than half or more than double the amount of vertices as the target graph. This can be considered as a constraint in our WGL method Sect. 3.2. To evaluate under the condition that isomorphic graphs to the target graph are not present in the training data, we also filter out graphs that have the same number of vertices and edges as the target graph.

The clustering approach based on topological similarity can be used for defining a topological classification of graphs. However, it would require further analysis of resultant clusters.

5.2 Apparatus

We used our RW-LOG-LAPLACIAN kernel to compute the topological similarity between graphs. Although other kernels can be used, we decided on one that shows the highest estimated accuracy in the first evaluation and provides the best chance to see if a kernel can find perceptually similar graphs.

The experiment was conducted on an iPad Pro which has a 12.9 inch display with 2,732×2,048 pixels. Each of the graphs were presented in 640×640 pixels. All vertices were drawn using the same blue color (Red: .122, Green: .467, Blue: .706, Alpha: .9) and edges were drawn using dark grey color (R: .1, G: .1, B: .1, A: .25), as shown in Fig. 5.

5.3 Procedure

Prior to the beginning of the experiment, the participants were asked several questions about demographic information, such as age and experience with graph visualization. To familiarize participants with the system, eight training tasks were given, at which time they were allowed to ask any questions to the moderator. Once the training was done, the moderator did not communicate with the participant. The 72 tasks were presented in randomized order to the participants. The nine choice graphs of each task were also presented in a randomized order. For each task, we asked the participants to briefly explain why he or she selected the particular graph (think aloud protocol).

5.4 Participants

We recruited 30 (9 females and 21 males) participants for our user study. The ages of the participants ranged from 18 to 36 years, with the mean age of 26.67 years (SD = 3.68). Most of the participants were from science and engineering backgrounds: 22 computer science, 2 electrical and computer engineering, 2 cognitive science, 1 animal science, 1 political science, and 1 literature. 28 participants indicated that they had seen a graph visualization (e.g., a node-link diagram), 21 had used one, and 16 had created one before. On average, each participant took 28.94 minutes ($SD = 11.49$) to complete all 72 tasks.

5.5 Results

For each task, the nine choices receive a topological similarity rank (r_T) from one to nine in order of decreasing topological similarity. We define predicted choice as the choice graph that our method ranks as the most topologically similar to the target graph ($r_T = 1$). Based on the responses by the participants, the three choices receive a perceptual similarity rank (r_P) from one to three, where one is the first chosen graph. Choices not selected by the participants are ranked $r_P = 4$. Results of our evaluation can be found in Fig. 6.

Overall, 80.46% of the time, participants' responses for rank one ($r_P = 1$) match our predicted choices ($r_T = 1$). 90.27% of participants' responses within rank one and rank two ($r_P = \{1,2\}$) contain the predicted choices, and 93.80% of responses within ranks one, two, and three ($r_P = \{1,2,3\}$) contain the predicted choice. A Friedman test (non-parametric alternative to one-way repeated measures ANOVA) shows a significant effect of the topological similarity rankings (r_T) on the perceptual similarity rankings (r_P) with $\chi^2(8) = 6343.9$, $p < .0001$. The mean r_P of the predicted choices is 1.35 ($SD = .82$ and $IQR = 0$), which is clearly different than other choices ($r_T > 1$) as shown in Table 2. Post-hoc analysis with Wilcoxon signed-rank tests using Bonferroni correction confirms that the predicted choices are ranked significantly higher ($p < .0001$) than all other choices.

To see the effect layout methods have on participants' responses, we break down the responses with a topological similarity rank of one ($r_T = 1$), or predicted choice, by each layout method. Except for Spectral and Treemap, the predicted choices are ranked in more than 78.52% (up to 93.33%) of participants' responses as being the most perceptually similar. In more than 94.44% (up to 99.26%) of participants' responses, the predicted choices are within the three most perceptually similar graphs ($r_P = \{1,2,3\}$), as shown in Fig. 6b. For Spectral, the predicted choices are ranked in 72.22% of participants' responses as being the most similar graph, and 84.07% of responses as being within the three most similar graphs. For Treemap, the predicted choices are ranked in 59.63% of participants' responses as the most similar graph, and 82.59% of responses as being within the three most similar graphs. A Friedman test shows a significant effect of layout method on perceptual similarity rankings (r_P) with $\chi^2(7) = 200.85$, $p < .0001$.

Except for Spectral and Treemap, the mean r_P of the predicted choices for each layout method is close to 1, from 1.16 to 1.35 ($SD = .46$–$.81$ and $IQR = 0$), as shown in Table 2b. The means r_P of Spectral and Treemap are 1.65 ($SD = 1.14$ and $IQR = 1$) and 1.84 ($SD = 1.17$ and $IQR = 2$), respectively. Post-hoc analysis shows that the predicted choices with Treemap are ranked significantly lower than the predicted choices with other layout methods ($p < .0001$) except for Spectral. The predicted choices with Spectral are also ranked by participants as being significantly lower than the predicted choices with other layout methods ($p < .05$) except for Gosper and Treemap.

We also break down the responses for the topological similarity rank of one ($r_T = 1$) by each target graph. Except for G_{2331} and G_{3833}, the predicted choices are ranked in more than 79.58% (up to 98.33%) of responses as being the most similar, and more than 92.08% (up to 99.99%) of responses as being within the three most similar graphs ($r_P = \{1,2,3\}$), as shown in Fig. 6c. For G_{2331}, the predicted choices are ranked in 32.92% of all responses as being the most similar graph, 78.33% of responses as being within the two most similar graphs, and 89.16% of responses as being within the three most similar graphs. For G_{3833}, the predicted choices are ranked in 60.42% of participants' responses as being the most similar graph, 73.33% of responses as being within the two most similar graphs, and 81.67% of responses as being within the three most similar graphs. A Friedman test shows a significant effect of target graphs on perceptual similarity rankings (r_P) with $\chi^2(8) = 511,\ p < .0001$. Except for G_{2331} and G_{3833}, the mean r_P of the predicted choices for each target graph is close to 1, from 1.06 to 1.45 ($SD = .34$–$.98$ and $IQR = 0$), as shown in Table 2b. The means r_P of G_{2331} and G_{3833} are 2 ($SD = .94$ and $IQR = 1$) and 1.85 ($SD = 1.18$ and $IQR = 2$), respectively. Post-hoc analysis shows that the predicted choices of G_{2331} and G_{3833} are ranked significantly lower than predicted choices of other target graphs ($p < .0001$).

5.6 Discussion

The results of the user study show that in more than 80% of participants' responses, the predicted choices are ranked as being the most perceptually similar graph to the target graphs. Also, more than 93% of the responses ranked the predicted choices as being within the three most perceptually similar graphs. Thus, we believe our WGL method is able to provide the expected layout results that are perceptually similar to the actual layout results.

When we analyze participants' responses for the predicted choices ($r_T = 1$) for each layout separately, we find that the predicted choices with Spectral and Treemap layouts are ranked lower than with other layouts. The common reasons given by participants for selecting choice graphs with Spectral layout were "line shape" and "number of vertices". We notice that the Spectral layouts have many nodes that overlap each other. Treemap, on the other hand, produces similar looking graphs due to its geometric constraints. This observation was mirrored by many participants who said "they all look similar" for the choices with Treemap layout. Common reasons for selecting choice graphs with Treemap layout were "edge density" and "overall edge direction".

It is interesting to see how people perceive certain structures as more important than others. For instance, when we look at the responses on target graphs separately, we notice that target graph G_{2331} has different response patterns. Target graph G_{2331} and its choice graph are shown in Fig. 5. Participants' responses for $r_P = 1$ are split between two choices, Fig. 5b and Fig. 5c. The common reasons why the participants ranked Fig. 5c as the most similar to Fig. 5a were "density", "shape", and "number of edges". On the other hand, the common reason why the participants ranked Fig. 5b as the most similar to Fig. 5a was "the number of central nodes". Our method also chose Fig. 5c as the most similar graph because of the general structure matching the target graph, but ranked Fig. 5b as being second most similar $r_T = 2$. In the case of target graph G_{2331}, the number of nodes in the center held more value to some participants than the overall structure.

In our user study, only one similar graph was chosen and shown by our system per target graph. In a real system, the user would be given several similarly looking graphs, including isomorphic graphs. Thus, the real system would show more layouts closer to the actual one.

6 Related Work

Only a handful of studies have used topological features for visualizing graphs. Perhaps this is why there is a scarcity of studies applying machine learning techniques to the process of visualizing a graph [25].

Niggemann and Stein [63] introduced a learning method to find an optimal layout method for a clustered graph. The method constructs a handcrafted feature vector of a cluster from a number of graph measures, including the number of vertices, diameter, and maximum vertex degree. Then, it attempts to find an optimal layout for each cluster. However, these features have been proved as not expressive enough to capture topological similarities in many graph kernel works [64].

Behrisch et al. [6] proposed a technique called Magnostics, where a graph is represented as a matrix view and image-based features are used to find similar matrices. One of the challenges of a matrix view is the vertex ordering. Depending on the ordering, even the same graph can be measured as a different graph from itself. Graph kernels do not suffer from the same problem since they measure the similarity using only the topology of the graph.

Several techniques have used machine learning approaches to improve the quality of a graph layout [25]. Some of these techniques used evolutionary algorithms for learning user preferences with a human-in-the-loop assessment [3, 5, 55, 80], while others have designed neural network algorithms to optimize a layout for certain aesthetic criteria [15, 58, 89]. One major limitation of these techniques is that models learned from one graph are not usable in other graphs. Since these techniques often require multiple computations of layouts and their aesthetic metrics, the learning process can be highly time-consuming. These techniques can benefit from our approach by quickly showing the expected layout results and estimating the aesthetic metrics.

Many empirical studies have been conducted to understand the relation between topological characteristics of graphs and layout methods. The main idea is to find the "best" way to lay out given graphs [36]. To achieve this, Archambault et al. [2] introduced a layout method which first recursively detects the topological features of subgraphs, such as whether a subgraph is a tree, cluster, or complete graph. Then, each subgraph is laid out using the suitable method according to its topological characteristics. A drawback of this method is that the feature detectors are limited to five classes. Our kernel can be utilized for heterogeneous feature detection with less computational cost.

A number of recent studies investigated sampling methods for large graph visualization. Wu et al. [90] evaluated a number of graph sampling methods in terms of resulting visualization. They found that different visual features were preserved when different sampling strategies were used. Nguyen et al. [62] proposed a new family of quality metrics for large graphs based on a sampled graph.

7 Conclusion

We have developed a machine learning approach using graph kernels for the purpose of showing what a graph would look like in different layouts and their corresponding aesthetic metrics. We have also introduced a framework for designing graphlet kernels, which allows us to derive several new ones. The estimations using our new kernels can be derived several orders of magnitude faster than computing the actual layouts and their aesthetic metrics. Also, our kernels outperform state-of-the-art kernels in both accuracy and computation time. The results of our user study show that the topological similarity computed with our kernel matches perceptual similarity assessed by human users.

In our work, we have only considered a subset of layout methods. A possible future direction is to include more layout methods with additional parameter settings for each method. Mechanical Turk could be used to conduct such an experiment at scale. Another possible future direction of this work is to introduce a new layout method which quickly predicts the actual layout instead of just showing the expected results of the input graph. We hope this paper opens a new area of study into using machine learning approaches for large graph visualization.

Acknowledgments

This research has been sponsored in part by the U.S. National Science Foundation through grants IIS-1320229 and IIS-1528203.

REFERENCES

[1] The Supplementary Materials. `http://graphvis.net/wgl`.

[2] D. Archambault, T. Munzner, and D. Auber. TopoLayout: Multilevel Graph Layout by Topological Features. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):305–317, 2007.

[3] B. Bach, A. Spritzer, E. Lutton, and J.-D. Fekete. Interactive Random Graph Generation with Evolutionary Algorithms. In *Proc. Graph Drawing*, pages 541–552, 2012.

[4] A.-L. Barabási. *Network Science*. Cambridge University Press, 2016.

[5] H. J. C. Barbosa and A. M. S. Barreto. An Interactive Genetic Algorithm with Co-evolution of Weights for Multiobjective Problems. In *Proc. Annual Conference on Genetic and Evolutionary Computation*, pages 203–210, 2001.

[6] M. Behrisch, B. Bach, M. Hund, M. Delz, L. V. Rden, J. D. Fekete, and T. Schreck. Magnostics: Image-Based Search of Interesting Matrix Views for Guided Network Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):31–40, 2017.

[7] C. Bennett, J. Ryall, L. Spalteholz, and A. Gooch. The Aesthetics of Graph Visualization. In *Proc. Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging*, Computational Aesthetics, pages 57–64, 2007.

[8] K. M. Borgwardt and H. P. Kriegel. Shortest-path Kernels on Graphs. In *Proc. IEEE International Conference on Data Mining*, pages 74–81, 2015.

[9] K. M. Borgwardt, T. Petri, S. V. N. Vishwanathan, and H.-P. Kriegel. An Efficient Sampling Scheme for Comparison of Large Graphs. In *Proc. Mining and Learning with Graphs*, 2007.

[10] U. Brandes and C. Pich. Eigensolver Methods for Progressive Multidimensional Scaling of Large Data. In *Proc. Graph Drawing*, pages 42–53, 2006.

[11] H. Bunke and G. Allermann. Inexact Graph Matching for Structural Pattern Recognition. *Pattern Recognition Letters*, 1(4):245–253, 1983.

[12] C.-C. Chang and C.-J. Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[13] X. Chen, Y. Li, P. Wang, and J. C. Lui. A General Framework for Estimating Graphlet Statistics via Random Walk. *Proc. VLDB Endowment*, 10(3):253–264, 2016.

[14] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel. The Open Graph Drawing Framework (OGDF). In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 17. CRC Press, 2013.

[15] A. Cimikowski and P. Shope. A Neural-Network Algorithm for a Graph Layout Problem. *IEEE Transactions on Neural Networks*, 7(2):341–345, 1996.

[16] A. Civril, M. Magdon-Ismail, and E. Bocek-Rivele. SDE: Graph Drawing Using Spectral Distance Embedding. In *Proc. Graph Drawing*, pages 512–513, 2005.

[17] J. D. Cohen. Drawing Graphs to Convey Proximity: An Incremental Arrangement Method. *ACM Transactions on Computer-Human Interaction*, 4(3):197–229, 1997.

[18] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[19] L. d. F. Costa, F. A. Rodrigues, G. Travieso, and P. R. Villas Boas. Characterization of Complex Networks: A Survey of Measurements. *Advances in Physics*, 56(1):167–242, 2007.

[20] R. Davidson and D. Harel. Drawing Graphs Nicely Using Simulated Annealing. *ACM Transactions on Graphics*, 15(4):301–331, 1996.

[21] T. A. Davis and Y. Hu. The University of Florida Sparse Matrix Collection. *ACM Transactions on Mathematical Software*, 38(1):1:1–1:25, 2011.

[22] M. Dehmer and F. Emmert-Streib, editors. *Quantitative Graph Theory: Mathematical Foundations and Applications*. Chapman and Hall/CRC Press, 2014.

[23] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for Drawing Graphs: An Annotated Bibliography. *Computational Geometry: Theory and Applications*, 4(5):235–282, 1994.

[24] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1998.

[25] R. dos Santos Vieira, H. A. D. do Nascimento, and W. B. da Silva. The Application of Machine Learning to Problems in Graph Drawing A Literature Review. In *Proc. International Conference on Information, Process, and Knowledge Management*, pages 112–118, 2015.

[26] C. Dunne and B. Shneiderman. Improving Graph Drawing Readability by Incorporating Readability Metrics: A Software Tool for Network Analysts. Technical Report HCIL-2009-13, University of Maryland, 2009.

[27] P. Eades. A Heuristic for Graph Drawing. *Congressus Numerantium*, 42:149–160, 1984.

[28] P. Eades, S.-H. Hong, A. Nguyen, and K. Klein. Shape-Based Quality Metrics for Large Graph Visualization. *Journal of Graph Algorithms and Applications*, 21(1):29–53, 2017.

[29] F. Emmert-Streib, M. Dehmer, and Y. Shi. Fifty Years of Graph Matching, Network Alignment and Network Comparison. *Information Sciences*, 346–347:180–197, 2016.

[30] A. Frick, A. Ludwig, and H. Mehldau. A Fast Adaptive Layout Algorithm for Undirected Graphs. In *Proc. Graph Drawing*, pages 388–403, 1994.

[31] Y. Frishman and A. Tal. Multi-Level Graph Layout on the GPU. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1310–1319, 2007.

[32] T. M. J. Fruchterman and E. M. Reingold. Graph Drawing by Force-directed Placement. *Software: Practice and Experience*, 21(11):1129–1164, 1984.

[33] K. R. Gabriel and R. R. Sokal. A New Statistical Approach to Geographic Variation Analysis. *Systematic Biology*, 18(3):259–278, 1969.

[34] P. Gajer and S. G. Kobourov. GRIP: Graph Drawing with Intelligent Placement. *Journal of Graph Algorithms and Applications*, 6(3):203–224, 2002.

[35] T. Gärtner, P. Flach, and S. Wrobel. On Graph Kernels: Hardness Results and Efficient Alternatives. In B. Schölkopf and M. K. Warmuth, editors, *Learning Theory and Kernel Machines*, pages 129–143. Springer Berlin Heidelberg, 2003.

[36] H. Gibson, J. Faith, and P. Vickers. A Survey of Two-dimensional Graph Layout Techniques for Information Visualization. *Information Visualization*, 12(3–4):324–357, 2013.

[37] S. Hachul and M. Jünger. Drawing Large Graphs with a Potential-Field-Based Multilevel Algorithm. In *Proc. Graph Drawing*, pages 285–295, 2004.

[38] S. Hachul and M. Jünger. Large-Graph Layout Algorithms at Work: An Experimental Study. *Journal of Graph Algorithms and Applications*, 11(2):345–369, 2007.

[39] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proc. Python in Science Conference*, pages 11–15, 2008.

[40] D. Harel and Y. Koren. A Fast Multi-Scale Method for Drawing Large Graphs. *Journal of Graph Algorithms and Applications*, 6(3):179–202, 2002.

[41] D. Harel and Y. Koren. Graph Drawing by High-Dimensional Embedding. *Journal of Graph Algorithms and Applications*, 8(2):195–214, 2004.

[42] D. Haussler. Convolution Kernels on Discrete Structures. Technical Report UCSC-CRL-99-10, University of California, Santa Cruz, 1999.

[43] I. Herman, G. Melançon, and M. S. Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[44] T. Horváth, T. Gärtner, and S. Wrobel. Cyclic Pattern Kernels for Predictive Graph Mining. In *Proc. ACM SIGKDD Conference on Knowledge Discovery And Data Mining*, pages 158–167, 2004.

[45] Y. Hu. Efficient and High Quality Force-Directed Graph Drawing. *Mathematica Journal*, 10(1):37–71, 2005.

[46] W. Huang, S.-H. Hong, and P. Eades. Effects of Sociogram Drawing Conventions and Edge Crossings in Social Network Visualization. *Journal of Graph Algorithms and Applications*, 11(2):397–429, 2007.

[47] G. F. Jenks. The Data Model Concept in Statistical Mapping. *International Yearbook of Cartography*, 7:186–190, 1967.

[48] F. Kaden. Graphmetriken und Distanzgraphen. *ZKI–Informationen, Akad. Wiss. DDR*, 2(82):1–63, 1982.

[49] T. Kamada and S. Kawai. An Algorithm for Drawing General Undirected Graphs. *Information Processing Letters*, 31(1):7–15, 1989.

[50] H. Kashima, K. Tsuda, and A. Inokuchi. Kernels for Graphs. In K. Tsuda, B. Scholkopf, and J.-P. Vert, editors, *Kernels and Bioinformatics*, pages 155–170. MIT Press, 2004.

[51] J. Katsnelson and S. Kotz. On the Upper Limits of Some Measures of Variability. *Archiv für Meteorologie, Geophysik und Bioklimatologie, Serie B*, 8(1):103–107, 1957.

[52] S. Kieffer, T. Dwyer, K. Marriott, and M. Wybrow. HOLA: Human-like Orthogonal Network Layout. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):349–358, 2016.

[53] Y. Koren. Drawing Graphs by Eigenvectors: Theory and Practice. *Com-*

puters and Mathematics with Applications, 49(11–12):1867–1888, 2005.

[54] Y. Koren, L. Carmel, and D. Harel. ACE: A Fast Multiscale Eigenvectors Computation for Drawing Huge Graphs. In *Proc. IEEE Symposium on Information Visualization*, pages 137–144, 2002.

[55] T. Masui. Evolutionary Learning of Graph Layout Constraints from Examples. In *Proc. ACM Symposium on User Interface Software and Technology*, pages 103–108, 1994.

[56] B. J. McGrath, C. and D. Krackhardt. Seeing Groups in Graph Layout. *Connections*, 19(2):22–29, 1996.

[57] B. J. McGrath, C. and D. Krackhardt. The Effect of Spatial Arrangement on Judgments and Errors in Interpreting Graphs. *Social Networks*, 19(3):223–242, 1997.

[58] B. Meyer. Self-Organizing Graphs – A Neural Network Perspective of Graph Layout. In *Proc. Graph Drawing*, pages 246–262, 1998.

[59] T. Milenković, V. Memišević, A. K. Ganesan, and N. Pržulj. Systems-Level Cancer Gene Identification from Protein Interaction Network Topology Applied to Melanogenesis-related Functional Genomics Data. *Journal of the Royal Society Interface*, 7(44):423–437, 2010.

[60] C. W. Muelder and K.-L. Ma. A Treemap Based Method for Rapid Layout of Large Graphs. In *Proc. IEEE Pacific Visualization Symposium*, pages 231–238, 2008.

[61] C. W. Muelder and K.-L. Ma. Rapid Graph Layout Using Space Filling Curves. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1301–1308, 2008.

[62] Q. H. Nguyen, S. H. Hong, P. Eades, and A. Meidiana. Proxy Graph: Visual Quality Metrics of Big Graph Sampling. *IEEE Transactions on Visualization and Computer Graphics*, 23(6):1600–1611, 2017.

[63] O. Niggemann and B. Stein. A Meta Heuristic for Graph Drawing: Learning the Optimal Graph-Drawing Method for Clustered Graphs. In *Proc. Working Conference on Advanced Visual Interfaces*, pages 286–289, 2000.

[64] Nino Shervashidze. *Scalable Graph Kernels*. PhD thesis, Universität Tübingen, 2012.

[65] T. P. Peixoto. The graph-tool Python Library. `https://graph-tool.skewed.de`, 2014.

[66] N. Przulj. Biological Network Comparison Using Graphlet Degree Distribution. *Bioinformatics*, 23(2):e177–e188, 2007.

[67] N. Przulj and J. I. Corneil, D. G. Modeling Interactome: Scale-free or Geometric? *Bioinformatics*, 20(18):3508–3515, 2004.

[68] H. C. Purchase. Metrics for Graph Drawing Aesthetics. *Journal of Visual Languages and Computing*, 13(5):501–516, 2002.

[69] H. C. Purchase, J.-A. Allder, and D. Carrington. Graph Layout Aesthetics in UML Diagrams: User Preferences. *Journal of Graph Algorithms and Applications*, 6(3):255–279, 2002.

[70] H. C. Purchase, C. Pilcher, and B. Plimmer. Graph Drawing Aesthetics–Created by Users, Not Algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 18(1):81–92, 2012.

[71] M. Rahman, M. A. Bhuiyan, M. Rahman, and M. A. Hasan. GUISE: A Uniform Sampler for Constructing Frequency Histogram of Graphlets. *Knowledge and Information Systems*, 38(3):511–536, 2014.

[72] J. Ramon and T. Gärtner. Expressivity versus Efficiency of Graph Kernels. In *Proc. International Workshop on Mining Graphs, Trees and Sequences*, 2003.

[73] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proc. LREC Workshop on New Challenges for NLP Frameworks*, pages 45–50, 2010.

[74] N. Shervashidze and K. M. Borgwardt. Fast Subtree Kernels on Graphs. In *Proc. Conference on Neural Information Processing Systems*, pages 1660–1668, 2009.

[75] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

[76] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient Graphlet Kernels for Large Graph Comparison. In *Proc. International Conference on Artificial Intelligence and Statistics*, pages 488–495, 2009.

[77] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[78] A. J. Smola and B. Schölkopf. A Tutorial on Support Vector Regression. *Statistics and Computing*, 14(3):199–222, 2004.

[79] F. Sobik. Graphmetriken und Klassifikation Strukturierter Objekte. *ZKI–Informationen, Akad. Wiss. DDR*, 2(82):63–122, 1982.

[80] M. Spönemann, B. Duderstadt, and R. von Hanxleden. Evolutionary Meta Layout of Graphs. In *Proc. Diagrams*, pages 16–30, 2014.

[81] R. Tamassia, editor. *Handbook of Graph Drawing and Visualization*. CRC Press, 2013.

[82] J. Ugander, L. Backstrom, and J. Kleinberg. Subgraph Frequencies: Mapping the Empirical and Extremal Geography of Large Graph Collections. In *Proc. International Conference on World Wide Web*, pages 1307–1318, 2013.

[83] L. van der Maaten and G. Hinton. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[84] F. van Ham and B. Rogowitz. Perceptual Organization in User-Generated Graph Layouts. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1333–1339, 2008.

[85] S. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph Kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[86] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. van Wijk, J.-D. Fekete, and D. Fellner. Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011.

[87] C. Walshaw. A Multilevel Algorithm for Force-Directed Graph-Drawing. *Journal of Graph Algorithms and Applications*, 7(3):253–285, 2003.

[88] P. Wang, J. C. S. Lui, B. Ribeiro, D. Towsley, J. Zhao, and X. Guan. Efficiently Estimating Motif Statistics of Large Networks. *ACM Transactions on Knowledge Discovery from Data*, 9(2):8:1–8:27, 2014.

[89] R.-L. Wang and O. K. Artificial Neural Network for Minimum Crossing Number Problem. In *Proc. International Conference on Machine Learning and Cybernetics*, volume 7, pages 4201–4204, 2005.

[90] Y. Wu, N. Cao, D. Archambault, Q. Shen, H. Qu, and W. Cui. Evaluation of Graph Sampling: A Visualization Perspective. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):401–410, 2017.

[91] P. Yanardag and S. Vishwanathan. Deep Graph Kernels. In *Proc. ACM SIGKDD Conference on Knowledge Discovery And Data Mining*, pages 1365–1374, 2015.

[92] B. Zelinka. On a Certain Distance between Isomorphism Classes of Graphs. *Časopis pro pěstování matematiky*, 100(4):371–373, 1975.