Scalable Global Alignment Graph Kernel Using Random Features: From Node Embedding to Graph Embedding

Lingfei Wu[★], Ian En-Hsu Yen[*], Zhen Zhang[†], Kun Xu[★], Liang Zhao[+], Xi Peng[°], Yinglong Xia[•], Charu Aggarwal[★ *]

IBM Research[★], CMU[*], WUSTL[†], GMU[+], UDEL[°], Huawei[•]

ABSTRACT

Graph kernels are widely used for measuring the similarity between graphs. Many existing graph kernels, which focus on local patterns within graphs rather than their global properties, suffer from significant structure information loss when representing graphs. Some recent global graph kernels, which utilizes the alignment of geometric node embeddings of graphs, yield state-of-the-art performance. However, these graph kernels are not necessarily positive-definite. More importantly, computing the graph kernel matrix will have at least quadratic time complexity in terms of the number and the size of the graphs. In this paper, we propose a new family of global alignment graph kernels, which take into account the global properties of graphs by using geometric node embeddings and an associated node transportation based on earth mover's distance. Compared to existing global kernels, the proposed kernel is positive-definite. Our graph kernel is obtained by defining a distribution over *random graphs*, which can naturally yield random feature approximations. The random feature approximations lead to our graph embeddings, which is named as "random graph embeddings" (RGE). In particular, RGE is shown to achieve *(quasi-)linear scalability* with respect to the number and the size of the graphs. The experimental results on nine benchmark datasets demonstrate that RGE outperforms or matches twelve state-of-the-art graph classification algorithms.

CCS CONCEPTS

• **Computing methodologies** → **Kernel methods**.

KEYWORDS

Graph Kernel, Graph Representation Learning, Graph Embedding, Global Alignment, Random Features

ACM Reference Format:
Lingfei Wu[★], Ian En-Hsu Yen[*], Zhen Zhang[†], Kun Xu[★], Liang Zhao[+], Xi Peng[°], Yinglong Xia[•], Charu Aggarwal[★] . 2019. Scalable Global Alignment Graph Kernel Using Random Features: From Node Embedding to Graph Embedding. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330918

[*]Corresponding author: Lingfei Wu. Email: wuli@us.ibm.com

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330918

1 INTRODUCTION

Graph kernels are one of the most important methods for graph data analysis and have been successfully applied in diverse fields such as disease and brain analysis [6, 21], chemical analysis [25], image action recognition and scene modeling [8, 37], and malware analysis[36]. Since there are no explicit features in graphs, a kernel function corresponding to a high-dimensional feature space provides a flexible way to represent each graph and to compute similarities between them. Hence, much effort has been devoted to designing feature spaces or kernel functions for capturing similarities between structural properties of graphs.

The first line of research focuses on local patterns within graphs [9, 28]. Specifically, these kernels recursively decompose the graphs into small sub-structures, and then define a feature map over these sub-structures for the resulting graph kernel. Conceptually, these notable graph kernels can be viewed as instances of a general kernel-learning framework called R-convolution for discrete objects [10, 29]. However, the aforementioned approaches consider only local patterns rather than global properties, which may substantially limit effectiveness in some applications, depending on the underlying structure of graphs. Equally importantly, most of these graph kernels scale poorly to large graphs due to their at least quadratic time complexity in terms of the number of graphs and cubic time complexity in terms of the size of graphs.

Another family of methods use geometric embeddings of graph nodes to capture global properties, which has shown great promise, achieving state-of-the-art performance in graph classification [14, 15, 23]. However, these global graph kernels based on matching node embeddings between graphs may suffer from the loss of positive definiteness. Furthermore, the majority of these approaches have at least quadratic complexity in terms of either the number of graph samples or the size of the graph.

To address these limitations of existing graph kernels, we propose a new family of global graph kernels that take into account the global properties of graphs, based on recent advances in the distance kernel learning framework [42]. The proposed kernels are truly *positive-definite (p.d.)* kernels constructed from a random feature map given by a transportation distance between a set of geometric node embeddings of raw graphs and those of random graphs sampled from a distribution. In particular, we make full use of the well-known *Earth Mover's Distance (EMD)*, computing the minimum cost of transporting a set of node embeddings of raw

graphs to the ones of random graphs. To yield an efficient computation of the kernel, we derive a *Random Features (RF)* approximation using a limited number of random graphs drawn from either data-independent or data-dependent distributions. The methods used to generate high-quality random graphs have a significant impact on graph learning. We propose two different sampling strategies depending on whether we use node label information or not. Furthermore, we note that each building block in this paper - geometric node embeddings and EMD - can be replaced by other node embeddings methods [15, 46] and transportation distances [32]. Our code is available at https://github.com/IBM/RandomGraphEmbeddings.

We highlight the main contributions as follows:

- We propose a class of p.d. global alignment graph kernels based on their global properties derived from geometric node embeddings and the corresponding node transportation.
- We present *Random Graph Embeddings (RGE)*, a by-product of the RF approximation, which yields an expressive graph embedding. Based on this graph embedding, we significantly reduce computational complexity at least *from quadratic to (quasi-)linear* in both the number and the size of the graphs.
- We theoretically show the uniform convergence of RGE. We prove that given $\Omega(1/\epsilon^2)$ random graphs, the inner product of RGE can uniformly approximates the corresponding exact graph kernel within ϵ−precision, with high probability.
- Our experimental results on nine benchmark datasets demonstrate that RGE outperforms or matches twelve state-of-the-art graph classification algorithms including graph kernels and deep graph neural networks. In addition, we numerically show that RGE can achieve (quasi-)linear scalability with respect to both the number and the size of graphs.

2 RELATED WORK

In this section, we first make a brief survey of the existing graph kernels and then detail the difference between conventional random features method for vector inputs [24] and our random features method for structured inputs.

2.1 Graph Kernels

Generally speaking, we can categorize the existing graph kernels into two groups: kernels based on local sub-structures, and kernels based on global properties.

The first group of graph kernels compare sub-structures of graphs, following a general kernel-learning framework, i.e., R-convolution for discrete objects [10]. The major difference among these graph kernels is rooted in how they define and explore sub-structures to define a graph kernel, including random walks [9], shortest paths [4], cycles [12], subtree patterns [28], and graphlets [30]. A thread of research attempts to utilize node label information using the Weisfeiler-Leman (WL) test of isomorphism [29] and takes structural similarity between sub-structures into account [44, 45] to further improve the performance of kernels.

Recently, a new class of graph kernels, which focus on the use of geometric node embeddings of graph to capture global properties, are proposed. These kernels have achieved state-of-the-art performance in the graph classification task [14, 15, 23]. The first global kernel was based on the Lovász number [20] and its associated orthonormal representation [14]. However, these kernels can only be applied on unlabelled graphs. Later approaches directly learn graph embeddings by using landmarks [15] or compute a similarity matrix [23] by exploiting different matching schemes between geometric embeddings of nodes of a pair of graphs. Unfortunately, the resulting kernel matrix does not yield a *valid p.d.* kernel and thus delivers a serious blow to hopes of using kernel support machine. Two recent graph kernels, the multiscale laplacian kernel [16] and optimal assignment kernel [17] were developed to overcome these limitations by building a p.d. kernel between node distributions or histogram intersection.

However, most of existing kernels only focus on learning kernel matrix for graphs instead of graph-level representation, which can only be used for graph classification rather than other graph related tasks (e.g., graph matching). More importantly, how to align the nodes in two graphs plays a central role in learning a similarity score. In this paper, we rely on an optimal transportation distance (e.g., Earch Mover's Distance) to learn the alignment between corresponding nodes that have similar structural roles in graphs, and directly generate a graph-level representation (embedding) for each graph instead of explicitly computing a kernel matrix.

2.2 Random Features for Kernel Machines

Over the last decade, the most popular approaches for scaling up kernel method is arguably random features approximation and its fruitful variants [3, 24, 31, 39]. Given a predefined kernel function, the inner product of RF directly approximates the exact kernel via sampling from a distribution, which leads to a fast linear method for computing kernel based on the learned low-dimensional feature representation. However, these RF approximation methods can only be applied to the shift-invariant kernels (e.g., the Gaussian or Laplacian kernels) with vector-form input data. Since a graph is a complex object, the developed graph kernels are neither shift-invariant kernels nor with vector-form inputs. Due to these challenges, to the best of our knowledge, there are no existing studies on how to develop the RF approximation for graph kernels.

A recent work, called D2KE (distances to kernels and embeddings) [42], proposes the general methodology of the derivation of a positive-definite kernel through a RF map from any given distance function, which enjoys better theoretical guarantees than other distance-based methods. In [43], D2KE was extended to design a specialized time-series embedding and showed the strong empirical performance for time-series classification and clustering. We believe there is no work on applying D2KE to the graph kernel domain [1]. Our work is the first one to build effective and scalable global graph kernels using Random Features.

3 GEOMETRIC EMBEDDINGS OF GRAPHS AND EARTH MOVER'S DISTANCE

In this section, we will introduce two important building blocks of our method, the geometric node embeddings that are used to represent a graph as a bag-of-vectors, and the well-known transportation distance EMD.

[1]Upon acceptance of this paper, a parallel work [1] also adopted D2KE to develop an unsupervised neural network model for learning graph-level embedding.

3.1 Geometric Embeddings of Graphs

The following notation will be used throughout the paper. Let a graph consisting of n nodes, m edges, and l discrete node labels be represented as a triplet $G = (V, E, \ell)$, where $V = \{v_i\}_{i=1}^n$ is the set of vertices, $E \subseteq (V \times V)$ is the set of undirected edges, and $\ell : V \to \Sigma$ is a function that assigns the label information to nodes from an alphabet set Σ. In this paper, we will consider both unlabeled graphs and graphs with discrete node labels. Let $\mathcal{G}$ be a set of N graphs where $\mathcal{G} = \{G_i\}_{i=1}^N$ and let $\mathcal{Y}$ be a set of graph labels [2] corresponding to each graph in $\mathcal{G}$ where $\mathcal{Y} = \{Y_i\}_{i=1}^N$. Let the geometric embeddings of a graph G be a set of vectors $U = \{\boldsymbol{u}_i\}_{i=1}^n \in \mathbb{R}^{n \times d}$ for all nodes, where the vector $\boldsymbol{u}_i$ in U is the representation of the node v_i, and d is the size of latent node embedding space.

Typically, with different underlying learning tasks, a graph G can be characterized by different forms of matrices. Without loss of generality, we use the normalized Laplacian matrix $L = D^{-1/2}(D - A)D^{-1/2} = I - D^{-1/2}AD^{-1/2}$, where A is the adjacency matrix with $A_{ij} = 1$ if $(v_i, v_j) \in E$ and $A_{ij} = 0$ otherwise, and D is the degree matrix. We then compute the d smallest eigenvectors of L to obtain U as its geometric embeddings through the partial eigendecomposition of $L = U\Lambda U^T$. Then each node v_i will be assigned an embedding vector $\boldsymbol{u}_i \in \mathbb{R}^d$ where $\boldsymbol{u}_i$ is the i-th row of the absolute U. Note that since the signs of the eigenvectors are arbitrary, we use the absolute values. Let $\boldsymbol{u}_{ij}$ be the jth item of the vector $\boldsymbol{u}_i$, then it satisfies $|\boldsymbol{u}_{ij}| \leq 1$. Therefore, the node embedding vectors can be viewed as points in a d-dimensional unit hypercube. This fact plays an important role in our following sampling strategy.

Note that although the standard dense eigensolvers require at least cubic time complexity in the number of graph nodes, with a state-of-the-art iterative eigensolver [33, 40], we can efficiently solve eigendecomposition with complexity that is linear in the number of graph edges. It is also worth noting that the resulting geometric nodes embeddings well capture global properties of the graph since the eigenvectors associated with low eigenvalues of L encode the information about the overall structure of G based on the spectral graph theory [35].

In the traditional model of Natural Language Processing, a bag-of-words had been the most common way to represent a document. With modern deep learning approaches, each element such as a word in the document or a character in the string is embedded into a low-dimensional vector and is fed within a bag-of-vectors into recurrent neural networks that perform document and string classification. Similarly, we also represent each graph as bag-of-vectors using a set of geometric node embeddings. However, although there is canonical ordering for the nodes of a graph, it is not reliable in most case. Therefore, it is important to find an optimal matching between two sets of node embeddings when comparing two graphs.

3.2 Node Transportation via EMD

Now we assume that a graph G is represented by the bag-of-vectors $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n\}$. To use the bag-of-words model, we also need to compute weights associated with each node vector. To be precise, if node v_i has c_i outgoing edges, we use $\boldsymbol{t}_i = (c_i / \sum_{j=1}^n c_j) \in \mathbb{R}$ as a normalized bag-of-words (nBOW) weight for each node. Our goal is to measure the similarity between a pair of graphs (G_i, G_j) using a proper distance measure. Instead of treating it as an assignment problem solved by maximum weight matching as in [15], we cast the task as a well-known transportation problem [11], which can be addressed by using the Earth Mover's Distance [26].

Using EMD, one can easily measure the dissimilarity between a pair of graphs (G_x, G_y) through node transportation, which essentially takes into account alignments between nodes. Let $n = max(n_x, n_y)$ denote the maximum number of nodes in a pair of graphs (G_x, G_y). Since $\boldsymbol{t}^{(G_x)}$ is the nBOW weight vector for the graph G_x, it is easy to obtain that $(\boldsymbol{t}^{(G_x)})^T \mathbf{1} = 1$. Similarly, we have $(\boldsymbol{t}^{(G_y)})^T \mathbf{1} = 1$. Then the EMD is defined as

$$\begin{aligned} \text{EMD}(G_x, G_y) &:= \min_{\mathcal{T} \in \mathbb{R}_+^{n_x \times n_y}} \langle \mathcal{D}, \mathcal{T} \rangle, \\ \text{subject to} &: \mathcal{T}\mathbf{1} = \boldsymbol{t}^{(G_x)}, \quad \mathcal{T}^T \mathbf{1} = \boldsymbol{t}^{(G_y)}. \end{aligned} \tag{1}$$

where $\mathcal{T}$ is the transportation flow matrix with $\mathcal{T}_{ij}$ denoting how much of node v_i in G_x travels to node v_j in G_y, and $\mathcal{D}$ is the transportation cost matrix where each item $\mathcal{D}_{ij} = d(\boldsymbol{u}_i, \boldsymbol{u}_j)$ denotes the distance between two nodes measured in their embedding space. Typically, the Euclidean distance $d(\boldsymbol{u}_i, \boldsymbol{u}_j) = \|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2$ is adopted. We note that with the distance $d(\boldsymbol{u}_i, \boldsymbol{u}_j)$ is a *metric* in the embedding space, the EMD (1) also define a *metric* between two graphs [26]. An attractive attribute of the EMD is that it provides an accurate measurement of the distance between graphs with different nodes that are contextually similar but in different positions in the graph. The EMD distance has been observed to perform well on text categorization [18] and graph classification [23]. A straightforward way that defines a kernel matrix based on EMD that measures the similarity between graphs has been shown in [23] as follows:

$$K = -\frac{1}{2} JDJ \tag{2}$$

where J is the centering matrix $J = I - \frac{1}{N}\mathbf{1}\mathbf{1}^T$ and D is the EMD distance matrix from all the pairs of graphs. However, there are three problems. The first one is that the Kernel matrix in (2) is not necessarily positive-definite. The second problem is that the EMD is expensive to compute, since its time complexity is $O(n^3 \log(n))$. In addition, computing the EMD for each pair of graphs requires the quadratic time complexity $O(N^2)$ in the number of graphs, which is highly undesirable for large-scale graph data. In this paper, we propose a scalable global alignment graph kernel using the random features to simultaneously address all these issues.

4 SCALABLE GLOBAL ALIGNMENT GRAPH KERNEL USING RANDOM FEATURES

In this section, we first show how to construct a class of the p.d. global alignment graph kernels from an optimal transportation distance (e.g., EMD) and then present a simple yet scalable way to compute expressive graph embeddings through the RF approximation. We also show that the inner product of the resulting graph embeddings uniformly converge to the exact kernel.

[2] Note that there are two types of labels involved in our paper, i.e., the node labels and the graph labels. The node labels characterize the property of nodes. The graph labels are the classes that graph belongs to.

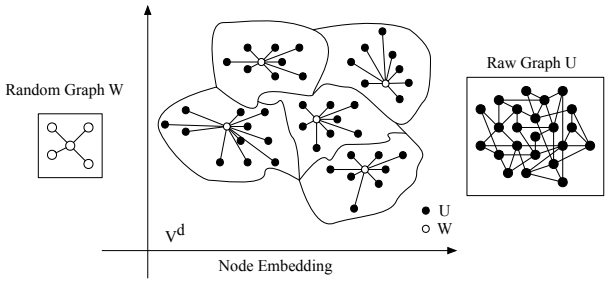


Figure 1: An illustration of how the EMD is used to measure the distance between a random graph and a raw graph. Each small random graph implicitly partitions the larger raw graph through node alignments in a low dimensional node embedding space.

4.1 Global Alignment Graph Kernel Using EMD and RF

The core task is to build a *positive-definite* graph kernel that can make full use of both computed geometric node embeddings for graphs and a distance measure considering the alignment of the node embeddings. We here define our global graph kernel as follows:

$$k(G_x, G_y) := \int p(G_\omega)\phi_{G_\omega}(G_x)\phi_{G_\omega}(G_y)dG_\omega, \quad \text{where } \phi_{G_\omega}(G_x) := \exp(-\gamma \text{EMD}(G_x, G_\omega)). \tag{3}$$

Here G_ω is a random graph consisting of D random nodes with their associated node embeddings $W = \{\boldsymbol{w}_i\}_{i=1}^D$, where each random node embedding $\boldsymbol{w}_i$ is sampled from a d-dimensional vector space $\mathcal{V} \in \mathbb{R}^d$. Thus, $p(G_\omega)$ is a distribution over the space of all random graphs of variable sizes $\Omega := \bigcup_{D=1}^{D_{max}} \mathcal{V}^D$. Then we can derive an infinite-dimensional feature map $\phi_{G_\omega}(G_x)$ from the EMD between G_x and all possible random graphs $G_\omega \in \Omega$. One explanation of how our proposed kernel works is that a small random graph can implicitly partition a larger raw graph through node transportation (or node alignments) in the corresponding node embedding space using EMD, as illustrated in Fig. 1.

A more formal and revealing way to interpret our kernel defined in (3) is to express it as

$$k(G_x, G_y) := \exp\left(-\gamma\, softmin_{p(G_\omega)}\{\text{EMD}(G_x, G_\omega) + \text{EMD}(G_\omega, G_y)\}\right) \tag{4}$$

where,

$$softmin_{p(G_\omega)}\{f(G_\omega)\} := -\frac{1}{\gamma}\log \int p(G_\omega)e^{-\gamma f(G_\omega)}dG_\omega \tag{5}$$

can be treated as the soft minimum function defined by two parameters $p(G_\omega)$ and γ. Since the usual soft minimum is defined as $softmin_i f_i := -softmax_i(-f_i) = -\log \sum_i e^{-f_i}$, then Equation (5) can be regarded as its smoothed version, which uses parameter γ to control the degree of smoothness and is reweighted by a probability density $p(G_\omega)$. Interestingly, the value of (5) is mostly determined by the minimum of $f(G_\omega)$, when $f(G_\omega)$ is Lipschitz-continuous and γ is large. Since EMD is a metric as discussed above, we have

$$\text{EMD}(G_x, G_y) \le \min_{G_\omega \in \Omega} \big(\text{EMD}(G_x, G_\omega) + \text{EMD}(G_\omega, G_y)\big)$$

by the triangle inequality. The equality holds if the maximum size of the random graph, $D_{\max}$, is equal or greater than the original graph size n. Therefore, the kernel value in (4) serves as a good approximation to the EMD between any pair of graphs G_x and G_y. By the kernel definition, it must be *positive-definite*.

4.2 Random Graph Embedding: Random Features of Global Alignment Graph Kernel

In this section, we will introduce how to efficiently compute the proposed global alignment graph kernels and derive the random graph embedding that can be used for representing graph-level embedding from the geometric node embeddings.

4.2.1 *Efficient Computation of RGE.*

Exact computation of the proposed kernel in (3) is often infeasible, as it does not admit a simple analytic solution. A natural way to compute such kernel is to resort to a kernel approximation that is easy to compute while uniformly converges to the exact kernel. As one of the most effective kernel approximation techniques, random features method has been demonstrated great successes in approximating Gaussian Kernel [19, 24] and Laplacian Kernel [41] in various applications. However, as we discussed before in Sec. 2.2, conventional RF methods cannot be directly applied to our graph kernels since they are not shift-invariant and cannot deal with the inputs that are not vector-form. Moreover, for traditional RF methods, we have to know the kernel function prior before hand, which is also not available in our case. However, fortunately, since we can define our kernel in terms of a randomized feature map, it naturally yields the following random approximation that does not require aforementioned assumptions,

$$\tilde{k}(G_x, G_y) = \langle Z(G_x), Z(G_y)\rangle = \frac{1}{R}\sum_{i=1}^{R}\phi_{G_{\omega i}}(G_x)\phi_{G_{\omega i}}(G_y) \to k(G_x, G_y), \text{ as } R \to \infty. \tag{6}$$

where $Z(G_x)$ is a R-dimensional vector with the $i-$th term $Z(G_x)_i = \frac{1}{\sqrt{R}}\phi_{G_{\omega i}}(G_x)$, and $\{G_{\omega i}\}_{i=1}^R$ are i.i.d. samples drawn from $p(G_\omega)$. Note that the vector $Z(G_x)$ just can be considered as the representation (embedding) of graph G_x. We call this random approximation "*random graph embedding (RGE)*", a generalized concept of "random features" for our graph inputs. We will also show that this random approximation RGE admits the uniform convergence to the original kernel (3) over all pairs of graphs (G_x, G_y).

Algorithm 1 summarizes the procedure to generate feature vectors for data graphs. There are several comments to make here. First of all, the distribution $p(G_\omega)$ is the key to generating high-quality node embeddings for random graphs. We propose two different ways to generate random graphs, which we will illustrate in detail later. Second, the size D of the random graphs is typically quite small. An intuitive explanation why a small random graph captures important global information of raw graphs has been discussed in the previous section. However, since there is no prior information to determine how many random nodes is needed to segment the data graph for learning discriminatory features, we sample the size of the random graphs from a uniform distribution $[1, D_{max}]$ to obtain an unbiased estimate of D. Finally, both node embedding

and distance measures can be further improved by exploiting the latest advancements in these techniques.

Algorithm 1 Random Graph Embedding

Input: Data graphs $\{G_i\}_{i=1}^N$, node embedding size d, maximum size of random graphs D_{max}, graph embedding size R.
Output: Feature matrix $Z_{N\times R}$ for data graphs

1: Compute nBOW weights vectors $\{\boldsymbol{t}^{(G_i)}\}_{i=1}^N$ of the normalized Laplacian L of all graphs
2: Obtain node embedding vectors $\{\boldsymbol{u}_i\}_{i=1}^n$ by computing d smallest eigenvectors of L
3: **for** $j = 1, \ldots, R$ **do**
4: Draw D_j uniformly from $[1, D_{max}]$.
5: Generate a random graph $G_{\omega j}$ with D_j number of nodes embeddings W from Algorithm 2.
6: Compute a feature vector $Z_j = \phi_{G_{\omega_j}}(\{G_i\}_{i=1}^N))$ using EMD or other optimal transportation distance in Equation (3).
7: **end for**
8: Return feature matrix $Z(\{G_i\}_{i=1}^N) = \frac{1}{\sqrt{R}}\{Z_i\}_{i=1}^R$

By efficiently approximating the proposed global alignment graph kernel using RGE, we obtain the benefits of both improved accuracy and reduced computational complexity. Recall that the computation of EMD has time complexity $O(n^3 log(n))$ and thus the existing graph kernels require at least $O(N^2 n^3 log(n))$ computational complexity and $O(N^2)$ memory consumption, where N and n are the number of graphs and the average size of graphs, respectively. Because of the small size of random graphs, the computation of EMD in our RGE approximation only requires $O(D^2 nlog(n))$ [5]. It means that our RGE approximation only requires computation with the quasi-linear complexity $O(nlog(n))$ if we treat D as a constant (or a small number). Note that with a state-of-the-art eigensolver [33, 40], we can effectively compute the d largest eigenvectors with linear complexity $O(dmz)$, where m is the number of graph edges and z is the number, typically quite small, of iterations of iterative eigensolver. Therefore, the total computational complexity and memory consumption of RGE are $O(NRnlog(n) + dmz)$ and $O(NR)$ respectively. Compared to other graph kernels, our method reduces computational complexity from quadratic to linear in terms of the number of graphs, and from (quasi-)cubic to (quasi-)linear in terms of the graph size. We will empirically assess the computational runtime in the subsequent experimental section.

4.2.2 Data-independent and Data-dependent Distributions. Algorithm 2 details the two sampling strategies (data-independent and data-dependent distributions) for generating a set of node embeddings of a random graph. The first scheme is to produce random graphs from a data-independent distribution. Traditionally, conventional RF approximation has to obtain random samples from a distribution corresponding to the user predefined kernel (e.g., Gaussian or Laplacian kernels). However, since we reverse the order by firstly defining the distribution and then defining a kernel similar to [42], we are free to select any distribution that can capture the characteristics of the graph data well. Given that all node embeddings are distributed in a d-dimensional unit hypercube space, we first compute the largest and smallest elements in all node embeddings and then use a uniform distribution in the range of these two values to generate a set of d-dimensional vectors for random node embeddings in a random graph. Since node embeddings are roughly dispersed uniformly in the d-dimensional unit hypercube space, we found this scheme works well in most of cases. Like the traditional RF, this sampling scheme is data-independent. So we call it RGE(RF).

Another scheme is conceptually similar to recently proposed work on deriving data-dependent traditional random features [13] for vector-inputs, which have been shown to have a lower generalization error than data-independent random features [27]. However, unlike these conventional RF methods [13, 27] and the conventional landmarks method that selects a representative set of whole graphs [15], we propose a new way to sample parts of graphs (only from training data) as random graphs, which we refer to as the *Anchor Sub-Graphs (ASG)* scheme RGE(ASG). There are several potential advantages compared to lankmarks and RF methods. First of all, ASG opens the door to defining an indefinite feature space since there are conceptually unlimited numbers of sub-graphs, compared to the limited size (up to the number of graphs) of landmarks. Second, ASG produces a random graph by permuting graph nodes of the original graph and by resembling randomly their corresponding node embeddings in the node embedding space, which may help to identify more hidden global structural information instead of only considering the raw graph topology. Thanks to EMD, hidden global structure can be captured well through node alignments. Finally, unlike RGE(RF), the ASG scheme allows exploiting node-label information in raw graphs since this information is also accessible through the sampled nodes in sub-graphs.

Incorporating the node label information into RGE(ASG) is fairly straightforward; it is desirable to assign nodes with same labels a smaller distance than these with different labels. Therefore, we can simply set the distance $d(\boldsymbol{u}_i, \boldsymbol{u}_j) = max(\|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2, \sqrt{d})$ if nodes v_i and v_j have different node labels since $\sqrt{d}$ is the largest distance in a d-dimensional unit hypercube space.

4.3 Convergence of Random Graph Embedding

In this section, we establish a bound on the number of random graphs required to guarantee an ϵ approximation between the exact kernel (3) and its random feature approximation (6) denoted by $\tilde{k}(G_x, G_y)$. We first establish a covering number for the space $\mathcal{X}$ under the EMD metric.

Lemma 1. There is an ϵ-covering $\mathcal{E}$ of $\mathcal{X}$ under the metric defined by EMD with Euclidean ground distance such that

$$\forall G \in \mathcal{X}, \exists G_i \in \mathcal{E},\ \text{EMD}(G, G_i) \leq \epsilon.$$

with $|\mathcal{E}| \leq (1 + \frac{2}{\epsilon})^{Md}$, where M is an upper bound on the number of nodes for any graph $G \in \mathcal{X}$.

Proposition 1. Let $\Delta_R(G_x, G_y) = k(G_x, G_y) - \tilde{k}(G_x, G_y)$. We have that if $|\Delta_R(G_i, G_j)| \leq t$, $\forall G_i, G_j \in \mathcal{E}$, where $\mathcal{E}$ is an $\frac{t}{4\gamma}-$covering of $\mathcal{X}$, and γ is the parameter of ϕ_{G_ω}, then $|\Delta_R(G_x, G_y)| \leq 2t$, $\forall G_x, G_y \in \mathcal{X}$.

Thus, given $\Omega(\frac{1}{\epsilon^2})$ random graphs, the inner product of RGE can uniformly approximates the corresponding exact graph kernel

Algorithm 2 Random Graph Generation

Input: Node embeddings $U = \{\boldsymbol{u}_i\}_{i=1}^{n}$, node embedding size d, size of random graph D_j.

Output: Random node embeddings $W = \{\boldsymbol{w}_i\}_{i=1}^{D_j}$

1: **if** Choose RGE(RF) **then**
2: Compute maximum value u_{max} and minimum value u_{min} in U.
3: Generate a number D_j of random node embedding vectors $\{\boldsymbol{w}_i\}_{i=1}^{D_j}$ in a random graph drawn from $\big(u_{min} + (u_{max} - u_{min}) \times rand(d, D_j)\big)$.
4: **else if** Choose RGE(ASG) **then**
5: Uniformly draw graph index $k = rand(1, N)$ and select the k-th raw graph
6: Uniformly draw a number D_j of node indices $\{n_1, n_2, \ldots, n_{D_j}\}$ in the k-th raw graph
7: Generate a number D_j of random node embedding vectors $\{\boldsymbol{w}_i\}_{i=1}^{D_j} = \{\boldsymbol{u}_{n_1}, \boldsymbol{u}_{n_2}, \ldots, \boldsymbol{u}_{n_{D_j}}\}$ as well as its associated node labels for a random graph
8: **end if**
9: Return nodes embeddings $W = \{\boldsymbol{w}_i\}_{i=1}^{D_j}$ for a random graph

within $\epsilon-$precision, with high probability, as shown in the following Theorem.

Theorem 1. The uniform convergence rate is

$$P\left\{\sup_{G_x, G_y \in \mathcal{X}} |\Delta_R(G_x, G_y)| \leq \epsilon\right\} \geq 1 - 2(1 + \frac{16\gamma}{\epsilon})^{2dM} \exp(-R\epsilon^2/8).$$

Therefore, to guarantee $|\Delta_R(G_x, G_y)| \leq \epsilon$ with probability at least $1 - \delta$, it suffices to have

$$R = \Omega\left(\frac{Md}{\epsilon^2} \log(1 + \frac{16\gamma}{\epsilon}) + \frac{1}{\epsilon^2}\big[\log(\frac{1}{\delta}) + \text{const}\big]\right).$$

Proof. Based on Proposition 1, we have

$$\begin{aligned} &P\left\{\sup_{G_x, G_y \in \mathcal{X}} |\Delta_R(G_x, G_y)| \leq 2t\right\} \\ \geq &P\left\{\sup_{G_i, G_j \in \mathcal{E}} |\Delta_R(G_x, G_y)| \leq t\right\}. \end{aligned} \tag{7}$$

For any $G_i, G_j \in \mathcal{E}$, since $E[\Delta_R(G_i, G_j)] = 0$ and $|\Delta_R(G_i, G_j)| \leq 1$, from the Hoeffding inequality, we have

$$P\left\{|\Delta_R(G_i, G_j)| \geq t\right\} \leq 2\exp(-Rt^2/2). \tag{8}$$

Therefore,

$$\begin{aligned} &P\left\{\sup_{G_i, G_j \in \mathcal{E}} |\Delta_R(G_i, G_j)| \geq t\right\} \\ \leq &\sum_{G_i, G_j \in \mathcal{E}} P\{|\Delta_R(G_i, G_j)| \geq t\} \\ \leq &2|\mathcal{E}|^2 \exp(-Rt^2/2) \leq 2(1 + \frac{8\gamma}{t})^{2dM} \exp(-Rt^2/2). \end{aligned} \tag{9}$$

Combining (7) and (9), and setting $t = \frac{\epsilon}{2}$, we obtain the desired result. □

The above theorem states that, to find an ϵ approximation to the exact kernel, it suffices to have number of random features $R = \Omega(\frac{1}{\epsilon^2})$. We refer interested readers to the details of the proof of Theorem (1) in Appendix A.

5 EXPERIMENTS

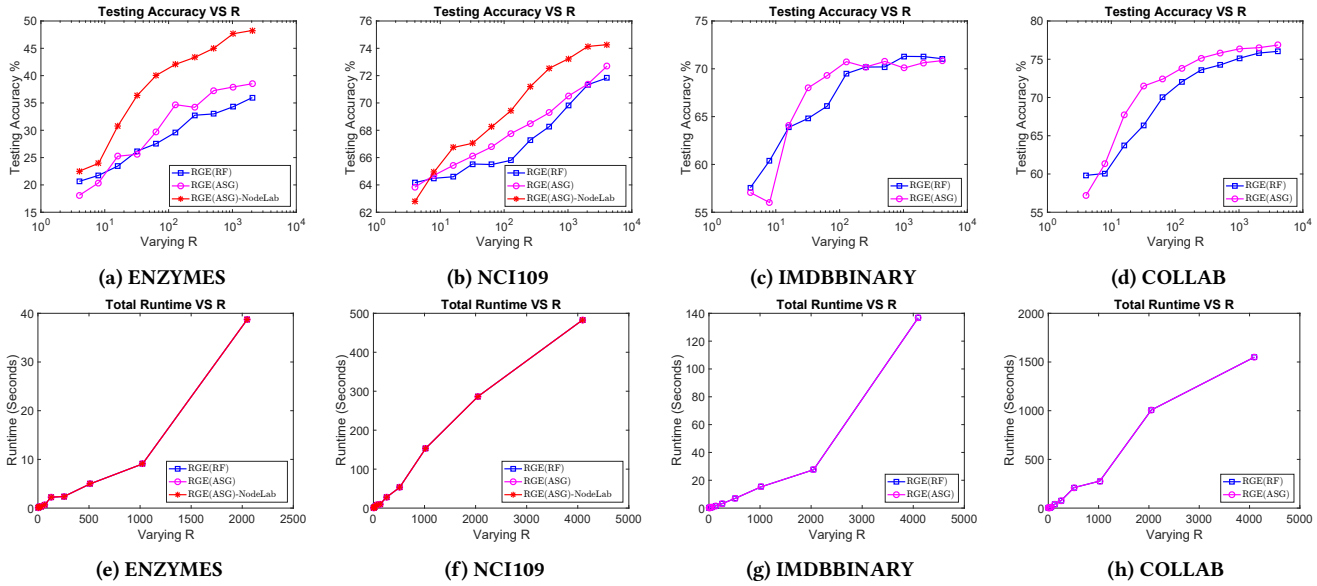
We performed experiments to demonstrate the effectiveness and efficiency of the proposed method, and compared against a total of twelve graph kernels and deep graph neural networks on nine benchmark datasets[3] widely used for testing the performance of graph kernels. We implemented our method in Matlab and utilized the C-MEX function[4] for the computationally expensive component of EMD. All computations were carried out on a DELL system with Intel Xeon processors 272 at 2.93GHz for a total of 16 cores and 250 GB of memory, running the SUSE Linux operating system.

Datasets. We applied our method to widely-used graph classification benchmarks from multiple domains [29, 34, 44]; MUTAG, PTC-MR, ENZYMES, PROTEINS, NCI1, and NCI109 are graphs derived from small molecules and macromolecules, and IMDB-B, IMDB-M, and COLLAB are derived from social networks. All datasets have binary labels except ENZYMES, IMDB-M, and COLLAB which have 6, 3 and 3 classes, respectively. All bioinformatics graph datasets have node labels while all other social network graphs have no node labels. Detailed descriptions of these 9 datasets, including statistical properties, are provided in the Appendix.

Baselines. Due to the large literature, we compare our method RGE against five representative global kernels related to our approach and three classical graph kernels, including EMD-based Indefinite Kernel (EMD) [23], Pyramid Match Kernel (PM) [23], Lovász θ Kernel (Lo-θ) [14], Optimal Assignment Matching (OA-E_λ(A)) [15], Vertex Optimal Assignment Kernel (V-OA) [17], Random Walk Kernel (RW) [9], Graphlet Kernel (GL) [30], and Shortest Path Kernel (SP) [4]. Furthermore, we also compare RGE with several variants of Weisfeler-Leman Graph Kernel (WL-ST [29], WL-SP [29], and WL-OA-E_λ(A) [15]). Finally, we compare RGE against four recently developed deep learning models with node labels, including Deep Graph Convolutional Neural Networks (DGCNN), [38]; PATCHY-SAN (PSCN) [22], Diffusion CNN (DCNN) [2], and Deep Graphlet Kernel (DGK) [44]. The first three models are built on convolutional neural networks on graphs while the last one is based on Word2Vec model. Since WL test is a generic technique to utilize discrete node labels for improving many stand-alone graph kernels, in this study, we first focus on testing the capability of each graph kernel without node labels and then assess the performance of each graph kernel with plain node labels and with WL techniques [5].

Setup. Since RGE is a graph embedding, we directly employ a linear SVM implemented in LIBLINEAR [7] since it can faithfully separate the effectiveness of our feature representation from the power of the nonlinear learning solvers. Following the convention of the graph kernel literature, we perform 10-fold cross-validation, using 9 folds for training and 1 for testing, and repeat the whole experiments ten times (thus 100 runs per dataset) and report the average prediction accuracies and standard deviations. The ranges of hyperparameters γ and D_max are [1e-3 1e-2 1e-1 1 10] and

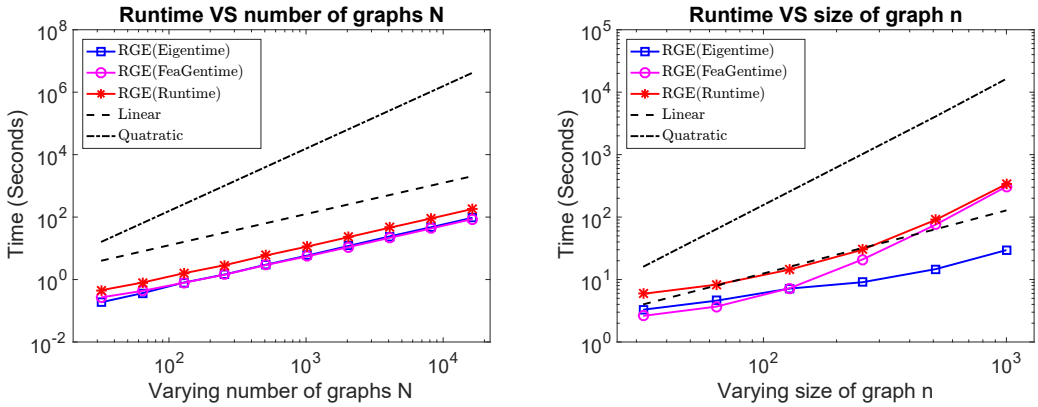
[3] http://members.cbio.mines-paristech.fr/ nshervashidze/code/
[4] http://ai.stanford.edu/~rubner/emd/default.htm
[5] Our approach to combine RGE(ASG) with WL techniques is to first use WL to generate new node labels and then apply RGE(ASG) with these node labels.



(a) ENZYMES (b) NCI109 (c) IMDBBINARY (d) COLLAB

(e) ENZYMES (f) NCI109 (g) IMDBBINARY (h) COLLAB

Figure 2: Test accuracies and runtime of three variants of RGE with and without node labels when varying R.



(a) Number of graphs N (b) Size of graph n

Figure 3: Runtime for computing node embeddings and RGE graph embeddings, and overall runtime when varying number of graphs N and size of graph n. (Default values: number of graphs $N = 1000$, graph size $n = 100$, edge size $m = 200$). Linear and quadratic complexity are also plotted for easy comparison.

[3:3:30], respectively. All parameters of the SVM and hyperparameters of our method were optimized only on the training dataset. To eliminate the random effects, we repeat the whole experiments ten times and report the average prediction accuracies and standard deviations. For all baselines we have taken the best reported number from their papers. Since EMD is the closet method to ours, we execute both methods under the same setting for fair comparison and report both accuracy and computational time.

Impacts of R on Accuracy and Runtime of RGE. We conducted experiments investigating the convergence behavior and the scalability of three variants of RGE with or without using node labels when increasing the number R of random graphs. The hyperparameter D is obtained from the previous cross-validations on the training set. We report both testing accuracy and runtime when increasing graph embedding size R. As shown in Fig. 2, all variants of RGE converge very rapidly when increasing R from a small number (R = 4) to relatively large number ($R = 2^k > n$). This confirms our analysis in Theorem 1 that the RGE approximation can guarantee rapid convergence to the exact kernel. The second observation is that RGE exhibits quasi-linear scalability with respect to R, as predicted by our computational analysis. This is particularly important for large scale graph data since most graph kernels have quadratic complexity in the number of graphs and/or in the size of graphs.

Scalability of RGE varying N graphs and n nodes. We further assess the scalability of RGE when varying number of graphs N and size of graph n for randomly generated graphs. We change the number of graphs in the range of $N = [8\ 16384]$ and the size of

Table 1: Comparison of classification accuracy against graph kernel methods without node labels.

Datasets	MUTAG	PTC-MR	ENZYMES	NCI1	NCI019
RGE(RF)	**86.33** ± 1.39(1s)	59.82 ± 1.42(1s)	35.98 ± 0.89(38s)	**74.70** ± 0.56(727s)	72.50 ± 0.32(865s)
RGE(ASG)	85.56 ± 0.91(2s)	**59.97** ± 1.65 (1s)	**38.52** ± 0.91(18s)	74.30 ± 0.45(579s)	**72.70** ± 0.42(572s)
EMD	84.66 ± 2.69 (7s)	57.65 ± 0.59 (46s)	35.45 ± 0.93 (216s)	72.65 ± 0.34 (8359s)	70.84 ± 0.18 (8281s)
PM	83.83 ± 2.86	59.41 ± 0.68	28.17 ± 0.37	69.73 ± 0.11	68.37 ± 0.14
Lo-θ	82.58 ± 0.79	55.21 ± 0.72	26.5 ± 0.54	62.28 ± 0.34	62.52 ± 0.29
OA-E_λ(A)	79.89 ± 0.98	56.77 ± 0.85	36.12 ± 0.81	67.99 ± 0.28	67.14 ± 0.26
RW	77.78 ± 0.98	56.18 ± 1.12	20.17 ± 0.83	56.89 ± 0.34	56.13 ± 0.31
GL	66.11 ± 1.31	57.05 ± 0.83	18.16 ± 0.47	47.37 ± 0.15	48.39 ± 0.18
SP	82.22 ± 1.14	56.18 ± 0.56	28.17 ± 0.64	62.02 ± 0.17	61.41 ± 0.32

Table 2: Comparison of classification accuracy against graph kernel methods with node labels or WL technique.

Datasets	PTC-MR	ENZYMES	PROTEINS	NCI1	NCI019
RGE(ASG)	**61.5** ± 2.34(1s)	**48.27** ± 0.99(28s)	75.98 ± 0.71(20s)	**76.46** ± 0.45(379s)	**74.42** ± 0.30(526s)
EMD	57.67 ± 2.11 (42s)	42.85 ± 0.72 (296s)	**76.03** ± 0.28 (1936s)	75.89 ± 0.16 (7942s)	73.63 ± 0.33 (8073s)
PM	60.38 ± 0.86	40.33 ± 0.34	74.39 ± 0.45	72.91 ± 0.53	71.97 ± 0.15
OA-E_λ(A)	58.76 ± 0.92	43.56 ± 0.66	—	69.83 ± 0.30	68.96 ± 0.35
V-OA	56.4 ± 1.8	35.1 ± 1.1	73.8 ± 0.5	65.6 ± 0.4	65.1 ± 0.4
RW	57.06 ± 0.86	19.33 ± 0.62	71.67 ± 0.78	63.34 ± 0.27	63.51 ± 0.18
GL	59.41 ± 0.94	32.70 ± 1.20	71.63 ± 0.33	66.00 ± 0.07	66.59 ± 0.08
SP	60.00 ± 0.72	41.68 ± 1.79	73.32 ± 0.45	73.47 ± 0.11	73.07 ± 0.11
WL-RGE(ASG)	**62.20** ± 1.67(1s)	57.97 ± 1.16(38s)	**76.63** ± 0.82(30s)	**85.85** ± 0.42(401s)	**85.32** ± 0.29(798s)
WL-ST	57.64 ± 0.68	52.22 ± 0.71	72.92 ± 0.67	82.19 ± 0.18	82.46 ± 0.24
WL-SP	56.76 ± 0.78	**59.05** ± 1.05	74.49 ± 0.74	84.55 ± 0.36	83.53 ± 0.30
WL-OA-E_λ(A)	59.72 ± 1.10	53.76 ± 0.82	—	84.75 ± 0.21	84.23 ± 0.19

Table 3: Comparison of classification accuracy against recent deep learning models on graphs.

Datasets	PTC-MR	PROTEINS	NCI1	IMDB-B	IMDB-M	COLLAB
(WL-)RGE(ASG)	62.20 ± 1.67	**76.63** ± 0.82	**85.85** ± 0.42	**71.48** ± 1.01	47.26 ± 0.89	**76.85** ± 0.34
DGCNN	58.59 ± 2.47	75.54 ± 0.94	74.44 ± 0.47	70.03 ± 0.86	**47.83** ± 0.85	73.76 ± 0.49
PSCN	**62.30** ± 5.70	75.00 ± 2.51	76.34 ± 1.68	71.00 ± 2.29	45.23 ± 2.84	72.60 ± 2.15
DCNN	56.6 ± 1.20	61.29 ± 1.60	56.61 ± 1.04	49.06 ± 1.37	33.49 ± 1.42	52.11 ± 0.53
DGK	57.32 ± 1.13	71.68 ± 0.50	62.48 ±0.25	66.96 ± 0.56	44.55 ± 0.52	73.09 ± 0.25

graph in the range of $n = [8\ 1024]$, respectively. When generating random adjacency matrices, we set the number of edges always be twice the number of nodes in a graph. We report the runtime for computing node embeddings using a state-of-the-art eigensolver [40], generating RGE graph embeddings, and the overall computation of graph classification, accordingly. Fig. 3(a) shows the linear scalability of RGE when increasing the number of graphs, confirming our complexity analysis in the previous Section. In addition, as shown in Fig. 3(b), RGE still exhibits linear scalablity in computing eigenvectors but slightly quasi-linear scalablity in RGE generation time and overall time, when increasing the size of graph. This is because that even though RGE reduces conventional EMD's complexity from super-cubic $O(n^3 log(n)$ to $O(D^2 nlog(n)$ (where D is a small constant), the log factor starts to show its impact on computing EMD between raw graphs and small random graphs when n becomes large (e.g. close to 1000). Interestingly, with a state-of-the-art eigensolver, the complexity of computing a few eigenvectors is linearly proportional to the graph size n [40]. This is highly desired property of our RGE embeddings, which open the door to large-scale applications of graph kernels for various applications such as social networks analysis and computational biology.

Comparison with All Baselines. Tables 1, 2, and 3 show that RGE consistently outperforms or matches other state-of-the-art graph kernels and deep learning approaches in terms of classification accuracy. There are several further observations worth making here. First, EMD, the closest method to RGE, shows good performance compared to most of other methods but often has significantly worse performance than RGE, highlighting the utility the novel graph kernel design using a feature map of random graphs and the effectiveness of a truly p.d. kernel. Importantly, RGE is also orders of magnitude faster than EMD in all cases, especially for data with a large graph size (like PROTEINS) or large number of graphs (like NCI1 and NCI109).

Second, the performance of RGE renders clear the importance of considering global properties graphs, and of having a distance measure able to align contextually-similar but positionally-different nodes, for learning expressive representations of graphs. In addition, as shown in Table 2, we observe that all methods (including

RGE) gain performance benefits when considering the node label information or utilizing WL iterations based on node labels. With node label information, the gaps between RGE and other methods diminish but still showing very clear advantages of RGE.

Finally, as shown in Table 3, for biological datasets we used the WL-RGE(ASG) to obtain the best performance with WL iteration. For social network datasets, we used the RGE(ASG) without node label since there are no node labels on these datasets. Compared to supervised deep learning based approaches, our unsupervised RGE method yet still shows clear advantages, highlighting the importance of aligning the structural roles of each node when comparing two graphs. In contrast, most of deep learning based methods focus on node-level representations instead of graph-level representation (typically using mean-pooling), which cannot take into account these important structural roles of each node in graphs.

6 CONCLUSION AND FUTURE WORK

In this work, we have presented a new family of p.d. and scalable global graph kernels that take into account global properties of graphs. The benefits of RGE are demonstrated by its much higher graph classification accuracy compared with other graph kernels and its (quasi-)linear scalability in terms of the number of graphs and graph size. Several interesting directions for future work are indicated: i) the graph embeddings generated by our technique can be applied and generalized to other learning problems such as graph (subgraph) matching or searching; ii) extensions of the RGE kernel for graphs with continuous node attributes and edge attributes should be explored.

REFERENCES

[1] Rami Al-Rfou, Dustin Zelle, and Bryan Perozzi. 2019. DDGK: Learning Graph Representations for Deep Divergence Graph Kernels. *arXiv:1904.09671* (2019).

[2] James Atwood, Siddharth Pal, Don Towsley, and Ananthram Swami. 2016. Sparse Diffusion-Convolutional Neural Networks. In *NIPS*.

[3] Francis Bach. 2017. On the equivalence between kernel quadrature rules and random feature expansions. *Journal of Machine Learning Research* 18, 21 (2017), 1–38.

[4] Karsten M Borgwardt and Hans-Peter Kriegel. 2005. Shortest-path kernels on graphs. In *Data Mining, Fifth IEEE International Conference on*. IEEE, 8–pp.

[5] François Bourgeois and Jean-Claude Lassalle. 1971. An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Commun. ACM* 14, 12 (1971), 802–804.

[6] Pin-Yu Chen and Lingfei Wu. 2017. Revisiting spectral graph clustering with generative community models. In *ICDM*. 51–60.

[7] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of machine learning research* 9, Aug (2008), 1871–1874.

[8] Matthew Fisher, Manolis Savva, and Pat Hanrahan. 2011. Characterizing structural relationships in scenes using graph kernels. *ACM Transactions on Graphics (TOG)* 30, 4 (2011), 34.

[9] Thomas Gärtner, Peter Flach, and Stefan Wrobel. 2003. On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*. Springer, 129–143.

[10] David Haussler. 1999. *Convolution kernels on discrete structures*. Technical Report. Department of Computer Science, University of California at Santa Cruz.

[11] Frank L Hitchcock. 1941. The distribution of a product from several sources to numerous localities. *Studies in Applied Mathematics* 20, 1-4 (1941), 224–230.

[12] Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. 2004. Cyclic pattern kernels for predictive graph mining. In *KDD*. ACM, 158–167.

[13] Catalin Ionescu, Alin Popa, and Cristian Sminchisescu. 2017. Large-scale data-dependent kernel approximation. In *Artificial Intelligence and Statistics*. 19–27.

[14] Fredrik Johansson, Vinay Jethava, Devdatt Dubhashi, and Chiranjib Bhattacharyya. 2014. Global graph kernels using geometric embeddings. In *ICML*.

[15] Fredrik D Johansson and Devdatt Dubhashi. 2015. Learning with similarity functions on graphs using matchings of geometric embeddings. In *KDD*. ACM, 467–476.

[16] Risi Kondor and Horace Pan. 2016. The multiscale laplacian graph kernel. In *NIPS*. 2990–2998.

[17] Nils M Kriege, Pierre-Louis Giscard, and Richard Wilson. 2016. On valid optimal assignment kernels and applications to graph classification. In *NIPS*. 1623–1631.

[18] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *ICML*. 957–966.

[19] Quoc Le, Tamás Sarlós, and Alex Smola. 2013. Fastfood-approximating kernel expansions in loglinear time. In *ICML*, Vol. 85.

[20] László Lovász. 1979. On the Shannon capacity of a graph. *IEEE Transactions on Information theory* 25, 1 (1979), 1–7.

[21] Fatemeh Mokhtari and Gholam-Ali Hossein-Zadeh. 2013. Decoding brain states using backward edge elimination and graph kernels in fMRI connectivity networks. *Journal of neuroscience methods* 212, 2 (2013), 259–268.

[22] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *ICML*. 2014–2023.

[23] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. 2017. Matching Node Embeddings for Graph Similarity.. In *AAAI*. 2429–2435.

[24] Ali Rahimi and Benjamin Recht. 2008. Random features for large-scale kernel machines. In *NIPS*. 1177–1184.

[25] Liva Ralaivola, Sanjay J Swamidass, Hiroto Saigo, and Pierre Baldi. 2005. Graph kernels for chemical informatics. *Neural networks* 18, 8 (2005), 1093–1110.

[26] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision* 40, 2 (2000), 99–121.

[27] Alessandro Rudi and Lorenzo Rosasco. 2017. Generalization properties of learning with random features. In *NIPS*. 3218–3228.

[28] Nino Shervashidze and Karsten M Borgwardt. 2009. Fast subtree kernels on graphs. In *NIPS*. 1660–1668.

[29] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12, Sep (2011), 2539–2561.

[30] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. 2009. Efficient graphlet kernels for large graph comparison. In *AIStats*. 488–495.

[31] Aman Sinha and John C Duchi. 2016. Learning kernels with random features. In *NIPS*. 1298–1306.

[32] Justin Solomon, Raif Rustamov, Leonidas Guibas, and Adrian Butscher. 2016. Continuous-flow graph transportation distances. *arXiv:1603.06927* (2016).

[33] Andreas Stathopoulos and James R McCombs. 2010. PRIMME: preconditioned iterative multimethod eigensolverâĂŤmethods and software description. *ACM Transactions on Mathematical Software (TOMS)* 37, 2 (2010), 21.

[34] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. *Journal of Machine Learning Research* 11 (2010), 1201–1242.

[35] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.

[36] Cynthia Wagner, Gerard Wagener, Radu State, and Thomas Engel. 2009. Malware analysis with graph kernels and support vector machines. In *Malicious and Unwanted Software, 2009 4th International Conference on*. IEEE, 63–68.

[37] Ling Wang and Hichem Sahbi. 2013. Directed acyclic graph kernels for action recognition. In *ICCV*. IEEE, 3168–3175.

[38] Bo Wu, Yang Liu, Bo Lang, and Lei Huang. 2017. DGCNN: Disordered Graph Convolutional Neural Network Based on the Gaussian Mixture Model. *arXiv:1712.03563* (2017).

[39] Lingfei Wu, Pin-Yu Chen, Ian En-Hsu Yen, Fangli Xu, Yinglong Xia, and Charu Aggarwal. 2018. Scalable spectral clustering using random binning features. In *KDD*. ACM, 2506–2515.

[40] Lingfei Wu, Eloy Romero, and Andreas Stathopoulos. 2017. PRIMME_SVDS: A high-performance preconditioned SVD solver for accurate large-scale computations. *SIAM Journal on Scientific Computing* 39, 5 (2017), S248–S271.

[41] Lingfei Wu, Ian EH Yen, Jie Chen, and Rui Yan. 2016. Revisiting random binning features: Fast convergence and strong parallelizability. In *KDD*. ACM, 1265–1274.

[42] Lingfei Wu, Ian En-Hsu Yen, Fangli Xu, Pradeep Ravikuma, and Michael Witbrock. 2018. D2KE: From Distance to Kernel and Embedding. *arXiv preprint arXiv:1802.04956* (2018).

[43] Lingfei Wu, Ian En-Hsu Yen, Jinfeng Yi, Fangli Xu, Qi Lei, and Michael Witbrock. 2018. Random Warping Series: A Random Features Method for Time-Series Embedding. In *International Conference on Artificial Intelligence and Statistics*. 793–802.

[44] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *KDD*. ACM, 1365–1374.

[45] Pinar Yanardag and SVN Vishwanathan. 2015. A structural smoothing framework for robust graph comparison. In *NIPS*. 2134–2142.

[46] Zhen Zhang, Mianzhi Wang, Yijian Xiang, Yan Huang, and Arye Nehorai. 2018. RetGK: Graph Kernels based on Return Probabilities of Random Walks. In *NIPS*. 3968–3978.

A APPENDIX A: PROOFS OF LEMMA 1 AND THEOREM 1

A.1 Proof of Lemma 1

Proof. Since the geometric node embedding $\boldsymbol{u}_i$ uses the normalized eigenvectors of the Laplacian matrix, we have that $\|\boldsymbol{u}_i\|_2 \leq 1$, i.e., $\boldsymbol{u}_i$ belongs to a unit ball. Therefore, we can find an ϵ-covering $\mathcal{E}_v$ of size $(1+\frac{2}{\epsilon})^d$ for the unit ball. Next, we define $\mathcal{E}$ as all the possible sets of $\boldsymbol{v} \in \mathcal{E}_v$ of size no larger than M. So we have $|\mathcal{E}| = (1+\frac{2}{\epsilon})^{dM}$. For any graph $G = (\boldsymbol{v}_j)_{j=1}^n \in \mathcal{X}$, we can find $G_i \in \mathcal{E}$ with also n nodes $(\boldsymbol{u}_j)_{j=1}^n$ such that $\|\boldsymbol{u}_j - \boldsymbol{v}_j\| \leq \epsilon$. Then by the definition of EMD (1), a solution that assigns each node $\boldsymbol{v}_j$ in G to a node $\boldsymbol{u}_j$ in x_i would have overall cost less than ϵ, So $\text{EMD}(G, G_i) \leq \epsilon$. □

A.2 Proof of Proposition 1

Proof. For any $G_x, G_y \in \mathcal{X}$, we can find $G_{x_k}, G_{y_k} \in \mathcal{E}$, such that

$$\text{EMD}(G_x, G_{x_k}) \leq \frac{t}{4\gamma} \quad \text{and} \quad \text{EMD}(G_y, G_{y_k}) \leq \frac{t}{4\gamma}. \tag{10}$$

Write $\Delta_R(G_x, G_y) = \Delta_R(G_{x_k}, G_{y_k}) + \Delta_R(G_x, G_y) - \Delta_R(G_{x_k}, G_{y_k})$, then we have

$$\begin{aligned} &|\Delta_R(G_x, G_y)| \\ \leq &|\Delta_R(G_{x_k}, G_{y_k})| + |\Delta_R(G_x, G_y) - \Delta_R(G_{x_k}, G_{y_k})| \\ \leq &|\Delta_R(G_{x_k}, G_{y_k})| + |\tilde{k}_R(G_{x_k}, G_{y_k}) - \tilde{k}_R(G_x, G_y)| \\ &+ |k_R(G_{x_k}, G_{y_k}) - k_R(G_x, G_y)| \end{aligned} \tag{11}$$

Now we consider the second term.

$$\begin{aligned} &|\tilde{k}_R(G_{x_k}, G_{y_k}) - \tilde{k}_R(G_x, G_y)| \\ \leq &\frac{1}{R}\sum_{i=1}^R |\exp\big(-\gamma\text{EMD}(G_{x_k}, G_{\omega_i}) - \gamma\text{EMD}(G_{y_k}, G_{\omega_i})\big) - \\ &\exp\big(-\gamma\text{EMD}(G_x, G_{\omega_i}) - \gamma\text{EMD}(G_y, G_{\omega_i})\big)| \\ \leq &\frac{1}{R}\sum_{i=1}^R \gamma|\text{EMD}(G_{x_k}, G_{\omega_i}) + \text{EMD}(G_{y_k}, G_{\omega_i}) \\ &- \text{EMD}(G_x, G_{\omega_i}) - \text{EMD}(G_y, G_{\omega_i})| \\ \leq &\frac{1}{R}\sum_{i=1}^R \gamma|\text{EMD}(G_{x_k}, G_{\omega_i}) - \text{EMD}(G_x, G_{\omega_i})| + \\ &\frac{1}{R}\sum_{i=1}^R \gamma|\text{EMD}(G_{y_k}, G_{\omega_i}) - \text{EMD}(G_y, G_{\omega_i})| \\ \leq &\frac{1}{R}\sum_{i=1}^R \gamma\text{EMD}(G_x, G_{x_k}) + \frac{1}{R}\sum_{i=1}^R \gamma\text{EMD}(G_y, G_{y_k}) \leq \frac{t}{2}. \end{aligned} \tag{12}$$

Similarly, we can prove that the third term in (11) satisfies

$$|k_R(G_{x_k}, G_{y_k}) - k_R(G_x, G_y)| \leq \frac{t}{2}. \tag{13}$$

Combining (12), (13), and the assumption $|\Delta_R(G_{x_k}, G_{y_k})| \leq t$, we obtain the desired result. □

A.3 Proof of Theorem 1

Proof. Based on Proposition 1, we have

$$\begin{aligned} &P\left\{\sup_{G_x, G_y \in \mathcal{X}} |\Delta_R(G_x, G_y)| \leq 2t\right\} \\ \geq &P\left\{\sup_{G_i, G_j \in \mathcal{E}} |\Delta_R(G_x, G_y)| \leq t\right\}. \end{aligned} \tag{14}$$

For any $G_i, G_j \in \mathcal{E}$, since $E[\Delta_R(G_i, G_j)] = 0$ and $|\Delta_R(G_i, G_j)| \leq 1$, from the Hoeffding inequality, we have

$$P\left\{|\Delta_R(G_i, G_j)| \geq t\right\} \leq 2\exp(-Rt^2/2). \tag{15}$$

Therefore,

$$\begin{aligned} &P\left\{\sup_{G_i, G_j \in \mathcal{E}} |\Delta_R(G_i, G_j)| \geq t\right\} \\ \leq &\sum_{G_i, G_j \in \mathcal{E}} P\{|\Delta_R(G_i, G_j)| \geq t\} \\ \leq &2|\mathcal{E}|^2 \exp(-Rt^2/2) \leq 2(1+\frac{8\gamma}{t})^{2dM}\exp(-Rt^2/2). \end{aligned} \tag{16}$$

Combining (14) and (16), and setting $t = \frac{\epsilon}{2}$, we obtain the desired result. □

B APPENDIX B: ADDITIONAL EXPERIMENTAL RESULTS

General Setup. We perform experiments to demonstrate the effectiveness and efficiency of the proposed method, and compare against total 12 graph kernels and deep graph neural networks on 9 benchmark datasets (as shown in Table 4) [6] that is widely used for testing the performance of graph kernels. We implement our method in Matlab and utilize C-MEX function [7] for the computationally expensive component of EMD. To accelerate the computation, we use multithreading with total 12 threads in all experiments. All computations were carried out on a DELL dual socket system with Intel Xeon processors 272 at 2.93GHz for a total of 16 cores and 250 GB of memory, running the SUSE Linux operating system.

B.1 Additional Results and Discussions on Accuracy and Runtime of RGE Varying R

Setup. We now conduct experiments to investigate the behavior of three variants of RGE with or without using node labels by varying the number R of random graphs. The hyperparameter D is obtained from the previous cross-validations on the training set. Depending on the size of graph on each dataset, we set R in the range starting from 4 and ending with a number R just satisfying $R = 2^k > n$. We report both testing accuracy and runtime when increasing graph embedding size R.

B.2 Additional Results and Discussions on Scalability of RGE varying N graphs and n nodes

Setup. We assess the scalability of RGE when varying number of graphs N and the size of a graph n on randomly generated graphs. We change the number of graphs in the range of $N = [8\ 16384]$ and the size of graph in the range of $n = [8\ 1024]$, respectively. When generating random adjacency matrices, we set the number of edges

[6] http://members.cbio.mines-paristech.fr/ nshervashidze/code/
[7] http://ai.stanford.edu/ rubner/emd/default.htm

Table 4: Properties of the datasets.

Dataset	MUTAG	PTC	ENZYMES	PROTEINS	NCI1	NCI109	IMDB-B	IMDB-M	COLLAB
Max # Nodes	28	109	126	620	111	111	136	89	492
Min # Nodes	10	2	2	4	3	4	12	7	32
Ave # Nodes	17.9	25.6	32.6	39.05	29.9	29.7	19.77	13.0	74.49
Max # Edges	33	108	149	1049	119	119	1249	1467	40119
Min # Edges	10	1	1	5	2	3	26	12	60
Ave # Edges	19.8	26.0	62.1	72.81	32.3	32.1	96.53	65.93	2457.34
# Graph	188	344	600	1113	4110	4127	1000	1500	5000
# Graph Labels	2	2	6	2	2	2	2	3	3
# Node Labels	7	19	3	3	37	38	—	—	—

always be twice the number of nodes in a graph. We use the size of node embedding $d = 6$ just like in the previous sections. We set the hyperparameters related to RGE itself are $DMax = 10$ and $R = 128$. We report the runtime for computing node embeddings using state-of-the-art eigensolver [33, 40] and RGE graph embeddings, and the overall runtime, respectively.

B.3 Additional Results and Discussions on Comparisons Against All Baselines

Setup. Since RGE is a graph embedding, we directly employ a linear SVM implemented in LIBLIBNEAR [7] since it can faithfully examine the effectiveness of our feature representation from the power of the nonlinear learning solvers. Following the convention in the graph kernel literature, we perform 10-fold cross-validation, using 9 folds for training and 1 for testing, and repeat the whole experiments ten times (thus 100 runs per dataset) and report the average prediction accuracies and standard deviations. The ranges of hyperparameters γ and D_max are [1e-3 1e-2 1e-1 1 10] and [3:3:30], respectively. All parameters of the SVM and hyperparameters of our method were optimized only on the training dataset. The node embedding size is set to either 4, 6 or 8 but always be the same number for all variants of RGE on the same datasets. To eliminate the random effects, we repeat the whole experiments ten times and report the average prediction accuracies and standard deviations. For all baselines we take the best number reported in the papers except EMD, where we rerun the experiments for fair comparisons in terms of both accuracy and runtime. Since GRE, OA, EMD, and PM are essentially built on the same node embeddings from the adjacency matrices, we take the number of OA-E_λ(A) in [15] for a fair comparison.