From Node Embedding to Graph Embedding: Scalable Global Graph Kernel via Random Features

Lingfei Wu[1], **Ian En-Hsu Yen**[2], **Kun Xu**[3], **Liang Zhao**[4], **Yinglong Xia**[5], **Michael Witbrock**[1]
[1] IBM Research, [2] Carnegie Mellon University, [3] Tencent AI, [4] George Mason University, [5] Huawei
{wuli,witbrock}@us.ibm.com, {syxu828}@gmail.com,
{lzhao9}@gmu.edu, {yinglong.xia.2010}@ieee.org

1 Introduction

Graph kernels are one of the most important methods for graph data analysis and have been successfully applied in diverse applications. We can generally categorize existing graph kernels into two groups: kernels based on local sub-structures, and kernels based on global properties. The first line of research compares sub-structures of graphs such as random walks [1], shortest paths [2], and graphlets [3]. 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. However, the aforementioned approaches only consider local patterns rather than global properties, which may substantially limit effectiveness in some applications. Equally importantly, most of these graph kernels scale poorly to large graphs due to their at-least-quadratic complexity in the number of graphs and cubic complexity in the size of each graph.

Another family of research is the use of geometric embeddings of graph nodes to capture global properties, which has shown great promise, achieving state-of-the-art performance in graph classification [4–8]. Unfortunately, these global kernel methods do not yield a *valid positive-definite (p.d.)* kernel and thus delivers a serious blow to hopes of using kernel support machine. Two recent graph kernels, the multiscale laplacian kernel [8] and optimal assignment kernel [7] were developed to overcome these limitations by building a p.d. kernel between node distributions or through histogram intersection. However, 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.

In this paper, we propose a new family of graph kernels that take into account the global properties of graphs, based on recent advances in the distance kernel learning framework [9]. The proposed kernels are truly p.d. kernels constructed from an explicit feature map given by a transportation distance [10] between a set of geometric node embeddings of raw graphs and those of a distribution over random graphs. In particular, we make full use of the well-known *Earth Mover's Distance (EMD)*, computing the minimum cost to transport a set of node embeddings of raw graphs onto the ones from 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. Our experiments on 9 benchmark graph kernel and social network datasets demonstrate that RGE matches or outperforms state-of-the-art graph kernel and deep graph neural networks. In addition, RGE has shown to achieve (quasi-)linear scalability when increasing the number of the graphs and graph size.

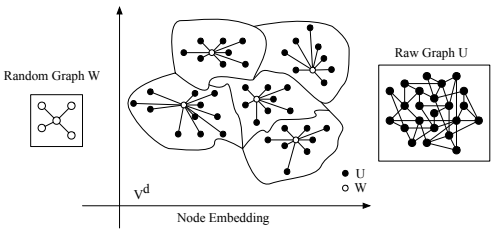


Figure 1: An example of how the EMD is used to measure the distance between a random graph and a raw graph. Each random graph implicitly partitions the larger raw graph through node alignments in a low dimensional node embedding space.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

2 Earth Mover's Distance Based Global Graph Kernel

The following notation will be used throughout the paper. Let a graph 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 $\ell : V \rightarrow \Sigma$ is a function that assigns labels to nodes from an alphabet in the graph. For simplicity, we assume that each graph has n nodes, m edges, and l 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 labels corresponding to each graph in $\mathcal{G}$ where $\mathcal{Y} = \{Y_i\}_{i=1}^N$.

Geometric Embeddings of Graphs. 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 d is the size of latent embedding space. 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 the adjacency matrix $A_{ij} = 1$ if $(v_i, v_j) \in E$ and $A_{ij} = 0$ otherwise, and D is the weighted 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 $\boldsymbol{u}_i \in \mathbb{R}^d$ corresponding to the i-th row of U.

Node Transportation via Earth Mover's Distance. We assume now that a graph G is represented by the bag-of-vectors $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n\}$. If node v_i has some number c_i of outgoing edges in the corresponding adjacency matrix A of a graph G, we denote $\boldsymbol{t}_i = (c_i / \sum_{j=1}^n c_j) \in \mathbb{R}$ 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. We cast the task as a well-known transportation problem [11], which can addressed using the Earth Mover's Distance [12] defined as follows:

$$\text{EMD}(G_x, G_y) := \min_{\mathcal{T} \in \mathbb{R}_+^{n_x \times n_y}} \langle \mathcal{D}, \mathcal{T} \rangle, s.t., \mathcal{T}\mathbf{1} = \boldsymbol{t}^{(G_x)}, \;\; \mathcal{T}^T\mathbf{1} = \boldsymbol{t}^{(G_y)}. \tag{1}$$

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. However, the EMD is expensive to compute; its computational complexity is $O(n^3 \log(n))$, and for large graphs, n is large. More importantly, building a kernel matrix using EMD does not lead to a positive p.d. kernel, which impairs its use and performance.

Global Graph Kernel using EMD. The core task is to seek a way to build a *positive-definite* graph kernel that can make full use of both computed geometric node embeddings for graphs and a distance measure matching the node embeddings. Following recent work in which a distance kernel learning framework was developed [9], we here define our global graph kernel as follows:

$$k(G_x, G_y) := \int p(G_\omega)\phi_{G_\omega}(G_x)\phi_{G_\omega}(G_y)dG_\omega, \text{where} \;\; \phi_{G_\omega}(G_x) := \exp(-\gamma \text{EMD}(G_x, G_\omega)). \tag{2}$$

Here G_ω is a random graph consisting of a number D of random nodes with their associated node embeddings $W = \{\boldsymbol{w}_i\}_{i=1}^D \in \mathcal{V}$. $p(G_\omega)$ is a distribution over the space of all random graphs of variable graph 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 (2) is to express it as

$$k(G_x, G_y) := \exp\left(-\gamma \mathit{softmin}_{p(G_\omega)}\{\text{EMD}(G_x, G_\omega) + \text{EMD}(G_\omega, G_y)\}\right) \tag{3}$$

where,

$$\mathit{softmin}_{p(G_\omega)}\{f(G_\omega)\} := -\frac{1}{\gamma} \log \int p(G_\omega) e^{-\gamma f(G_\omega)} dG_\omega \tag{4}$$

is a version of the soft minimum function parameterized by $p(G_\omega)$ and γ. Note that Equation (4) can be viewed as a smoothed version of the usual definition of soft minimum $\mathit{softmin}_i f_i := -\mathit{softmax}_i(-f_i) = -\log \sum_i e^{-f_i}$, reweighting by a probability density $p(G_\omega)$ and one more parameter γ to control the degree of smoothness. When γ is large and $f(G_\omega)$ is Lipschitz-continuous, the value of (4) is mostly determined by the minimum of $f(G_\omega)$.

Algorithm 1 Random Graph Embedding

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

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

It is worth noting that EMD is a metric and thus by the triangle inequality, we have

$$\text{EMD}(G_x, G_y) \leq \min_{G_\omega \in \Omega} \big(\text{EMD}(G_x, G_\omega) + \text{EMD}(G_\omega, G_y)\big) \tag{5}$$

and the equality holds if we allow the size of the random graph $D_{\max}$ to be no smaller than L. Therefore, the kernel (3) serves as a good approximation to the EMD between any pair of graphs G_x, G_y while being *positive-definite* by definition.

Random Graph Embedding: Approximation of Global Graph Kernel. Exact computation of the proposed kernel in (2) is often infeasible, as it does not admit a simple analytic solution. However, since we define our kernel in terms of a randomized kernel approximation [13], it naturally yields a following random approximation,

$$k(G_x, G_y) \approx \langle Z(G_x), Z(G_y)\rangle = \frac{1}{R}\sum_{i=1}^{R} \phi_{G_{\omega i}}(G_x)\phi_{G_{\omega i}}(G_y) \tag{6}$$

where $\{G_{\omega i}\}_{i=1}^R$ are i.i.d. random graphs drawn from $p(G_\omega)$ and $Z(G_x) := (\frac{1}{\sqrt{R}}\phi_{G_{\omega i}}(G_x))_{i=1}^R$ gives a vector representation of graph G_x. We call this random approximation *Random Graph Embedding (RGE)*, which we will show in the next section this random approximation (6) converges to the exact kernel (2) uniformly over all pairs of graphs (G_x, G_y).

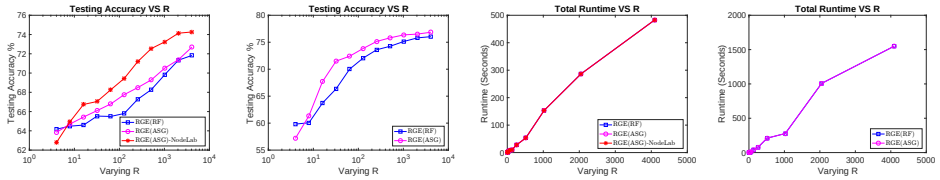
Algorithm 1 summarizes the procedure to generate feature vectors for data graphs. There are several important comments to make here. By efficiently approximating the proposed global graph kernel using RGE, we enjoy the double benefits of improved accuracy and reduced computation. A conventional evaluation of EMD has complexity $O(n^3 log(n))$ assuming that all graphs have similar size n. In contrast, our RGE approximation only requires computation with the quasi-linear complexity $O(nlog(n))$ if D is treated as a constant. This is because one evaluation of EMD only requires $O(D^2 nlog(n))$ [14] thanks to the small size of random graphs. Recall that with a state-of-the-art eigensolver [15, 16], we can effectively compute the d largest eigenvectors with linear complexity $O(dmz)$. Therefore, the total computational complexity is $O(NRnlog(n) + dmz)$, (quasi-)linear complexity in terms of the number of graphs N and the number of graph nodes n or graph edges m; we will empirically assess the effective complexity in the subsequent experimental sections.

3 Experiments

We performed experiments to demonstrate the effectiveness and efficiency of the proposed method, and compared against a total of 12 graph kernels and deep graph neural networks on 9 benchmark datasets[1] widely used for testing the performance of graph kernels.

Baselines. Due to the large literature, we only compare to 3 classical kernels [1, 3, 2]and 4 representative global kernels [6, 5, 7] related to our approach. Furthermore, we also compare RGE against 4 recently developed deep learning approaches with node labels, including Deep Graph Convolutional Neural Networks (DGCNN), [17]; PATCHY-SAN (PSCN) [18], Diffusion

[1]http://members.cbio.mines-paristech.fr/ nshervashidze/code/



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

Figure 2: Test accuracies and runtime of RGE w/ and w/o node labels when varying R.

CNN (DCNN) [19], and Deep Graphlet Kernel (DGK) [20]. The first three models are built on Convolutional Neural Networks on graphs while the last one is based on Word2Vec.

Setup. 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 [3:3:30], respectively. All parameters of the SVM and hyperparameters of our method were optimized only on the training dataset. For all baselines we have taken the best reported number, except for EMD, where we reran the experiments for fair comparison in terms of both accuracy and runtime.

Table 1: Comparison of classification accuracy against graph kernel methods with node labels.

Datasets	PTC	ENZYMES	PROTEINS	NCI1	NCI019
Methods	Accu	Accu	Accu	Accu	Accu
RGE	**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 [6]	57.67 ± 2.11 (42s)	42.85 ± 0.72 (196s)	**76.03 ± 0.28** (1936s)	75.89 ± 0.16 (7942s)	73.63 ± 0.33 (8073s)
PM [6]	60.22 ± 0.86	40.33 ± 0.34	74.39 ± 0.45	72.91 ± 0.53	71.97 ± 0.15
OA-E_λ(A) [5]	58.76 ± 0.92	43.56 ± 0.66	—	69.83 ± 0.30	68.96 ± 0.35
V-OA [7]	56.4 ± 1.8	35.1 ± 1.1	73.8 ± 0.5	65.6 ± 0.4	65.1 ± 0.4
RW [1]	57.06 ± 0.86	19.33 ± 0.62	71.67 ± 0.78	63.34 ± 0.27	63.51 ± 0.18
GL [3]	59.41 ± 0.94	32.70 ± 1.20	71.63 ± 0.33	66.00 ± 0.07	66.59 ± 0.08
SP [2]	60.00 ± 0.72	41.68 ± 1.79	75.61 ± 0.45	73.47 ± 0.11	73.07 ± 0.11

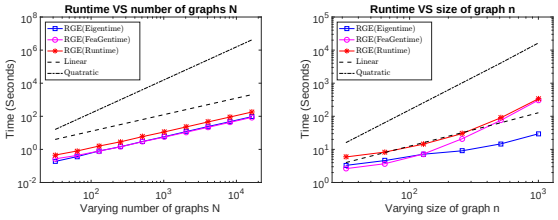
Table 2: Comparison of classification accuracy against recent deep learning models on graphs.

Datasets	PROTEINS	NCI1	IMDB-B	IMDB-M	COLLAB
Methods	Accu	Accu	Accu	Accu	Accu
RGE	**75.98 ± 0.71**	**76.46 ± 0.45**	**71.48 ± 1.01**	47.26 ± 0.89	**76.85 ± 0.23**
DGCNN	75.54 ± 0.94	74.44 ± 0.47	70.03 ± 0.86	**47.83 ± 0.85**	73.76 ± 0.49
PSCN	75.00 ± 2.51	76.34 ± 1.68	71.00 ± 2.29	45.23 ± 2.84	72.60 ± 2.15
DCNN	61.29 ± 1.60	56.61 ± 1.04	49.06 ± 1.37	33.49 ± 1.42	52.11 ± 0.53
DGK	27.08 ± 0.79	62.48 ±0.25	66.96 ± 0.56	44.55 ± 0.52	73.09 ± 0.25

Comparison with All Baselines. Tables 1, and 2 show that RGE consistently outperforms or matches other state-of-the-art graph kernels and deep learning approaches in terms of classification accuracy and computational time (only compared to EMD).

Impacts of R on Accuracy and Runtime of RGE. 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.

Scalability of RGE. Fig.3 shows the linear scalability of RGE in the number of graphs, confirming our complexity analysis in Section 2.



(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.

4 Conclusion and Future Work

In this work, we have presented a new family of p.d. graph kernels that take into account global properties of graphs based on an RGE approximation. The benefits of RGE are demonstrated by its much higher graph classification accuracy compared with all existing graph kernels and its (quasi-)linear scalability in terms of the number of graphs and graph size.

References

[1] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Learning Theory and Kernel Machines*. Springer, 2003, pp. 129–143.

[2] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," in *Data Mining, Fifth IEEE International Conference on*. IEEE, 2005, pp. 8–pp.

[3] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Artificial Intelligence and Statistics*, 2009, pp. 488–495.

[4] F. Johansson, V. Jethava, D. Dubhashi, and C. Bhattacharyya, "Global graph kernels using geometric embeddings," in *Proceedings of the 31st International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, 2014.

[5] F. D. Johansson and D. Dubhashi, "Learning with similarity functions on graphs using matchings of geometric embeddings," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 467–476.

[6] G. Nikolentzos, P. Meladianos, and M. Vazirgiannis, "Matching node embeddings for graph similarity." in *AAAI*, 2017, pp. 2429–2435.

[7] N. M. Kriege, P.-L. Giscard, and R. Wilson, "On valid optimal assignment kernels and applications to graph classification," in *Advances in Neural Information Processing Systems*, 2016, pp. 1623–1631.

[8] R. Kondor and H. Pan, "The multiscale laplacian graph kernel," in *Advances in Neural Information Processing Systems*, 2016, pp. 2990–2998.

[9] L. Wu, I. E.-H. Yen, F. Xu, P. Ravikuma, and M. Witbrock, "D2ke: From distance to kernel and embedding," *arXiv preprint arXiv:1802.04956*, 2018.

[10] J. Solomon, R. Rustamov, L. Guibas, and A. Butscher, "Continuous-flow graph transportation distances," *arXiv preprint arXiv:1603.06927*, 2016.

[11] F. L. Hitchcock, "The distribution of a product from several sources to numerous localities," *Studies in Applied Mathematics*, vol. 20, no. 1-4, pp. 224–230, 1941.

[12] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

[13] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, 2008, pp. 1177–1184.

[14] F. Bourgeois and J.-C. Lassalle, "An extension of the munkres algorithm for the assignment problem to rectangular matrices," *Communications of the ACM*, vol. 14, no. 12, pp. 802–804, 1971.

[15] A. Stathopoulos and J. R. McCombs, "Primme: preconditioned iterative multimethod eigensolver—methods and software description," *ACM Transactions on Mathematical Software (TOMS)*, vol. 37, no. 2, p. 21, 2010.

[16] L. Wu, E. Romero, and A. Stathopoulos, "Primme_svds: A high-performance preconditioned svd solver for accurate large-scale computations," *SIAM Journal on Scientific Computing*, vol. 39, no. 5, pp. S248–S271, 2017.

[17] B. Wu, Y. Liu, B. Lang, and L. Huang, "Dgcnn: Disordered graph convolutional neural network based on the gaussian mixture model," *arXiv preprint arXiv:1712.03563*, 2017.

[18] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, 2016, pp. 2014–2023.

[19] J. Atwood, S. Pal, D. Towsley, and A. Swami, "Sparse diffusion-convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2016.

[20] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1365–1374.