Taming graph kernels with random features

Krzysztof Choromanski [1 2]

Abstract

We introduce in this paper the mechanism of *graph random features* (GRFs). GRFs can be used to construct **unbiased** randomized estimators of several important kernels defined on graphs' nodes, in particular the *regularized Laplacian kernel*. As regular RFs for non-graph kernels, they provide means to scale up kernel methods defined on graphs to larger networks. Importantly, they give substantial computational gains also for smaller graphs, while applied in downstream applications. Consequently, GRFs address the notoriously difficult problem of cubic (in the number of the nodes of the graph) time complexity of graph kernels algorithms. We provide a detailed theoretical analysis of GRFs and an extensive empirical evaluation: from speed tests, through Frobenius relative error analysis to kmeans graph-clustering with graph kernels. We show that the computation of GRFs admits an embarrassingly simple distributed algorithm that can be applied if the graph under consideration needs to be split across several machines. We also introduce a (still unbiased) *quasi Monte Carlo* variant of GRFs, q-$GRFs$, relying on the so-called *reinforced random walks* that might be used to optimize the variance of GRFs. As a byproduct, we obtain a novel approach to solve certain classes of linear equations with positive and symmetric matrices.

1. Introduction

Consider a positive definite kernel (similarity function) $\mathrm{K} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$. Kernel methods (Campbell, 2002; Kontorovich et al., 2008; Smola & Schölkopf, 2002; Canu & Smola, 2006; Ghojogh et al., 2021; Cumby & Roth, 2003) provide powerful mechanisms for modeling non-linear relationships between data points. However, as relying on the so-called *kernel matrices* $\mathbf{K} = [\mathrm{K}(\mathbf{x}_i, \mathbf{x}_j)]_{i,j}$ for datasets $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ under consideration, they have time complexity at last quadratic in the size of a dataset.

To address this limitation, a mechanism of *random features* (RFs) (Liu et al., 2022; Rahimi & Recht, 2007; 2008) to linearize kernel functions was proposed. Random features rely on randomized functions: $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^m$ that map datapoints from the original to the new space, where the linear (dot-product) kernels corresponds to the original kernel. i.e:

$$\mathrm{K}(\mathbf{x}, \mathbf{y}) = \mathbb{E}[\phi(\mathbf{x})^\top \phi(\mathbf{y})] \tag{1}$$

The theory of random features for kernels was born with the seminal paper (Johnson, 1984a), proposing a simple randomized linear transformation ϕ (with Gaussian random matrices) to approximate dot-product kernels via dimensionality reduction ($m \ll d$). That mapping ϕ is often referred to as the *Johnson-Lindenstrauss Transform* (JLT).

Interestingly, it turned out that for several nonlinear kernels, JLT can be modified by adding a nonlinear functions acting element-wise on the linearly transformed dimensions of the input vectors (and potentially by replacing Gaussian sampling mechanism and altering the constant multiplicative renormalization term), to obtain Eq. 1. For the radial basis function kernels (RBFs) (Kégl et al., 1998) and softmax kernels, this nonlinear functions are trigonometric transformations (Rahimi & Recht, 2007). For the Gaussian (an instantiation of RBFs) and softmax kernels one can also apply the so-called *positive random features* leveraging exponential nonlinearties (Choromanski et al., 2020). For the angular kernel, the sign function is used (Goemans & Williamson, 2004; Choromanski et al., 2017). Thus linearization schemes for different kernels are obtained by applying different element-wise transformations, but the overall structure of ϕ is the same across different kernels.

RFs provide **unbiased** low-rank decomposition of the kernel matrix $\mathbf{K}$, effectively enabling the (approximate) computations with $\mathbf{K}$ to be conducted in sub-quadratic time, by-passing explicit materialization of $\mathbf{K}$. Even more importantly, they help to create nonlinear variants of the simpler linear methods (e.g kernel-SVM (Boser et al., 1992) as opposed to regular SVM (Cortes & Vapnik, 1995) or kernel regression (Chávez et al., 2020) as opposed to linear regression (Goldin, 2010)) via the ϕ-transformation.

So far we have considered points embedded in the regular

[*]Equal contribution [1]Google DeepMind [2]Columbia University. Correspondence to: Krzysztof Choromanski <kchoro@google.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Euclidean $\mathbb{R}^d$-space, but what if the space itself is non-Euclidean ? Upon discretization, such a space can be approximated by undirected weighted graphs (often equipped with shortest-path-distance metric) and in that picture the points correspond to graph nodes. It is thus natural to consider in that context *graph kernels* $\mathrm{K} : \mathrm{V} \times \mathrm{V} \rightarrow \mathbb{R}$ defined on the set of nodes V of the graph. And indeed, several such kernels were proposed: *diffusion*, *regularized Laplacian*, *p-step random walk*, *inverse cosine* and more (Smola & Kondor, 2003; Kondor & Lafferty, 2002; Chung & Yau, 1999). The term *graph kernels* is also often used for kernels taking as input graphs, thus computing similarities between two graphs rather than two nodes in a given graph (Vishwanathan et al., 2010). Those are not a subject of this paper.

Unfortunately, for most graph kernels, the computation of the kernel matrix $\mathbf{K}$ is at least **cubic** in the number of nodes of the graph. This is a major obstacle on the quest to make those methods more practical and mainstream kernel techniques. A tempting idea is to try to apply random features also for graph kernels, yet so far no analogue of the mechanism presented above for regular kernels was proposed (see: Sec. 2 for the additional discussion on some efforts to apply RF-based techniques in the theory of graph kernels in the broad sense). This is in striking contrast to kernels operating in $\mathbb{R}^d$, where not only are RFs widely adopted, but (as already discussed) the core underlying mechanism is the same for a large plethora of different kernels, irrespectively of their taxonomy, e.g. shift-invariant (Gaussian, Laplace, Cauchy) or not (softmax, dot-product). One of the main challenges is that no longer is the value of the graph kernel solely the property of its input-nodes, but also the medium (the graph itself), where those nodes reside. In fact, it does not even make sense to talk about the properties of graphs' nodes by abstracting from the whole graph (in this paper we do not assume any internal graph-node structure).

This work is one of the first steps to develop rigorous theory of random features for graph kernels. We introduce here the mechanism of *graph random features* (GRFs). GRFs can be used to construct **unbiased** randomized estimators of several important kernels defined on graphs' nodes, in particular the *regularized Laplacian kernel*. As regular RFs for non-graph kernels, they provide means to scale up kernel methods defined on graphs to larger networks. Importantly, they give substantial computational gains also for smaller graphs, while applied in downstream applications. Consequently, GRFs address the notoriously difficult problem of cubic (in the number of the nodes of the graph) time complexity of graph kernels algorithms. We provide a detailed theoretical analysis of GRFs and an extensive empirical evaluation: from speed tests, through Frobenius relative error analysis to k-means graph-clustering with graph kernels. We show that the computation of GRFs admits an embarrassingly simple distributed algorithm that can be applied for particularly massive graphs. We also introduce a (still unbiased) *quasi Monte Carlo* variant of GRFs, q-$GRFs$, relying on the so-called *reinforced random walks* (Figueiredo & Garetto, 2017) that might be used to optimize the variance of GRFs. As a byproduct, we also obtain a novel approach for solving linear equations with positive and symmetric matrices.

The q-GRF variants of our algorithm (that, as we show, still provide unbiased estimation) aim to translate the geometric techniques of structured random projections, that were recently successfully applied in the theory of RFs (see: Sec. 2 for detailed discussion) to non-Euclidean domains defined by graphs. To the best of our knowledge, it was never done before. We propose this idea in the paper, yet leave to future work comprehensive theoretical and empirical analysis.

2. Related work

Probabilistic techniques are used in different contexts in the theory of graph kernels. Sampling algorithms can be applied to make the computations of kernels taking graphs as inputs (rather than graphs' nodes) feasible (Yanardag & Vishwanathan, 2015; Shervashidze et al., 2009; Leskovec & Faloutsos, 2006; Orbanz, 2017; Bressan, 2020; Wang et al., 2011). Examples include in particular *graphlet* kernels, where counters of various small-size sub-graphs are replaced by their counterparts obtained via sub-graph sub-sampling (often via random walks) (Ribeiro et al., 2022; Wu et al., 2019; Chen et al., 2016). They provide more tractable, yet biased estimation of the original objective.

Other approaches define new classes of graph kernels starting from the RF-inspired representations. Examples include structure-aware random Fourier kernels from (Fang et al., 2021) that apply the so-called *L-hop sub-graphs* processed by graph neural networks (GNNs) to obtain their latent embeddings and define a kernel by taking their dot products. Even though here we proceed in a reverse order - starting from well-known graph kernels defined by deterministic closed-form expressions and producing their unbiased RF-based representations, there is an interesting link between that line of research and our work. GRFs can be cast as representations of similar core structure, but much more specific, GNN-free and consequently, providing simplicity, low computational time and strong theoretical guarantees.

Quasi Monte Carlo (q-MC) methods, where independent samples are replaced by the correlated ensembles for more accurate estimation are powerful MC techniques. In the theory of RFs for regular kernels, new q-MC methods applying block-orthogonal ensembles of the random Gaussian vectors (the so-called *orthogonal random features* or ORFs) were shown to (sometimes only asymptotically) reduce the approximation variance of various kernels (Yu et al., 2016; Lin et al., 2020; Choromanski et al., 2018b; 2017; 2020; Likhosherstov et al., 2022; Choromanski et al., 2018a).

3. Graph random features (GRFs)

3.1. Hidden Gram-structure of squared inverse matrices

We start our analysis, that will eventually lead us to the definition of GRFs, by forgetting (only temporarily) about graph theory and focusing on positive symmetric matrices.

3.1.1. Preliminaries

Consider a positive symmetric matrix $\mathbf{U} \in \mathbb{R}^{N \times N}$ of spectral radius $\rho(\mathbf{U}) \stackrel{\text{def}}{=} \max_{\lambda \in \Lambda(\mathbf{U})} |\lambda|$ satisfying: $\rho(\mathbf{U}) < 1$, where $\Lambda(\mathbf{U})$ stands for the set of eigenvalues of $\mathbf{U}$ ($\Lambda(\mathbf{U}) \subseteq \mathbb{R}$ since $\mathbf{U}$ is symmetric). Note that under this condition, the following series converges:

$$\mathbf{I}_N + \mathbf{U} + \mathbf{U}^2 + ... \tag{2}$$

and is equal to $(\mathbf{I}_N - \mathbf{U})^{-1}$. Furthermore, we have:

Lemma 3.1. *If $\mathbf{U} \in \mathbb{R}^{N \times N}$ is a a positive symmetric matrix with $\rho(\mathbf{U}) < 1$ then the following holds:*

$$\mathbf{I}_N + 2\mathbf{U} + 3\mathbf{U}^2 + ... = (\mathbf{I_N} - \mathbf{U})^{-2} \tag{3}$$

Proof. We have: $\mathbf{I}_N + 2\mathbf{U} + 3\mathbf{U}^2 + ... = (\mathbf{I}_N + \mathbf{U} + \mathbf{U}^2 + ...) + (\mathbf{U} + 2\mathbf{U}^2 + 3\mathbf{U}^3 + ...)$. Thus we have:

$$\begin{aligned}\mathbf{I}_N + 2\mathbf{U} + 3\mathbf{U}^2 + ... = (\mathbf{I}_N + \mathbf{U} + \mathbf{U}^2 + ...) + \\ \mathbf{U}(\mathbf{I}_N + 2\mathbf{U} + 3\mathbf{U}^2 + ...)\end{aligned} \tag{4}$$

Therefore, from what we have said above, we obtain:

$$(\mathbf{I}_N - \mathbf{U})(\mathbf{I}_N + 2\mathbf{U} + 3\mathbf{U}^2 + ...) = (\mathbf{I}_N - \mathbf{U})^{-1} \tag{5}$$

and, consequently: $\mathbf{I}_N + 2\mathbf{U} + 3\mathbf{U}^2 + ... = (\mathbf{I}_N - \mathbf{U})^{-2}$. □

We will derive an algorithm for unbiasedly estimating $(\mathbf{I}_N - \mathbf{U})^{-2}$ with the Gram-matrix $\mathbf{M} = [\phi(i)^\top \phi(j)]_{i,j=1,...,N}$ for a randomized mapping: $\phi : \mathbb{N} \to \mathbb{R}^N$. We will refer to $\phi(k)$ for a given $k \in \{1, ..., N\}$ as a *signature vector* for k (see also Fig. 1 for the illustration of signature vectors).

3.1.2. Computing signature vectors

We are ready to bring back the graphs. We interpret $\mathbf{U} = [u_{i,j}]_{i,j=1,...,N}$ as a weighted adjacency matrix of the weighted graph $\mathrm{G}_\mathbf{U}$ with vertex set $\mathrm{V} = \mathrm{V}(\mathrm{G}_\mathbf{U})$ of N vertices, where the weight between node i and j is given as $u_{i,j}$. We then calculate signature vectors via random walks on $\mathrm{G}_\mathbf{U}$. The algorithm for computing $\phi(i)$ for a specific node i is given in Algorithm 1 box. Calculations of $\phi(i)$ for different i are done independently (and therefore can be easily parallelized). The algorithm works as follows.

The signature vector is zeroed out in initialization. In the initialization, we also zero-out data structure *history* $\mathcal{H}$ which maintains some particular function of the list of visited edges (more details later). A number m of random walks, all initiated at point i is conducted. In principle, they can also be parallelized, but here, for the clarity of the analysis, we assume that they are executed in a given order.

Algorithm 1 Computing a signature vector for a given i.

Input : graph $\mathrm{G}_\mathbf{U}$, termination probability p_{term}, sampling strategy sample, # of random walks m, node i.
Result: signature vector $\phi(i)$.
Main:
1. Initialize: $\phi(i)[j] = 0$ for $j = 1, 2, ..., N$.
2. Initialize: $\mathcal{H} \leftarrow \emptyset$.
3. **for** $t = 1, ..., m$ **do**
   - **(a)** initialize: load $= 1$
   - **(b)** initialize: current_vertex $\leftarrow i$
   - **(c)** update: $\phi(i)[i] \leftarrow \phi(i)[i] + 1$
   - **(d) while** (not terminated) **do**
     1. assign: $v =$ current_vertex
     2. $w, p = \text{sample}(v, \mathrm{G}_\mathbf{U}, \mathcal{H})$
     3. update: current_vertex $\leftarrow w$
     4. update: load $\leftarrow$ load $\cdot\, u_{v,w} \,/\, p(1 - p_{\text{term}})$
     5. update: $\phi(i)[w] \leftarrow \phi(i)[w] +$ load
     6. update: $\mathcal{H} \leftarrow \mathcal{H}.\text{add}((v, w))$
4. renormalize: $\phi(i) \leftarrow \phi(i) \,/\, m$

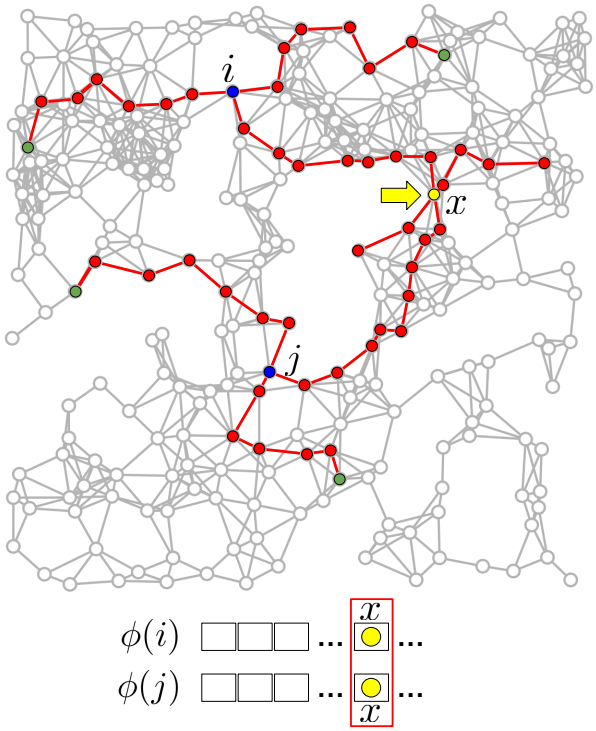

Figure 1. The pictorial representation of signature vectors. The contibutions to the dot-product between two signature vectors come from the vertices that were visited by both vectors.

Each random walk carries a load that is being updated every time the termination state has not been reached. Before each

transition, the termination is reached independently with a given probability p_{term}. The update-rule of the load is given in point 4 in the Algorithm 1 box. Strategy sample takes as input: (1) the current vertex v, (2) entire graph, (3) the function of the history of visits and outputs: (1) a neighbor w of v according to a particular randomized sampling strategy as well as: (2) its corresponding probability p. Every time a new vertex is reached, the value of its corresponding signature vector dimension is increased by the most updated value of the load, as given in point 5 in the Algorithm 1 box. When all the walks terminate, the signature vector is renormalized by the multiplicative term $\frac{1}{m}$ and returned.

Sampler & the history of visits: The simplest sample strategy is to choose a neighbor w of a current vertex v uniformly at random from the set of all neighbors N_v of v in $\mathrm{G}_{\mathbf{U}}$. In this setting: $p = \frac{1}{\mathrm{deg}(v)}$, where $\mathrm{deg}(v)$ stands for the degree of v defined as the size $|\mathrm{N}_v|$ of its neighborhood. In that setting, $\mathcal{H}$ is not needed and thus we have: $\mathcal{H} = \emptyset$. We show in Sec. 5 that this strategy works very well in practice. Another option is to take: $p = p(v, w) = \frac{u_{v,w}}{\sum_{z \in \mathrm{N}_v} u_{v,z}}$, i.e. make probabilities proportional to weights. In that setting, history is also not needed. In Sec. 4, we analyze in more detail the q-GRF setting, where nontrivial $\mathcal{H}$ is useful. This history can be used to diversify the set of random walks, by de-prioritizing edges that were already selected in previous random walks. Thus it might have the same positive effect on the estimation accuracy as ORFs from (Yu et al., 2016) that diversify the set of random projection vectors. However detailed analysis of such strategies is beyond the scope of this paper. Most importantly, as we show next, regardless of the particular instantiations of sample, signature vectors computed in Algorithm 1 indeed provide an unbiased estimation of the matrix $(\mathbf{I}_N - \mathbf{U})^{-2}$ (proof in Sec. 4):

Theorem 3.2. *Denote by $\mathbf{B} \in \mathbb{R}^{N \times N}$ a matrix with rows given by $\phi(i)^\top$ for $i = 1, ..., N$, computed according to Algorithm 1 and by $\mathbf{B}'$ its independent copy (obtained in practice by constructing another indeepentent set of random walks). Assume that method* sample *is not degenerate, i.e. it assigns a positive probability $p(v, w)$ for every neighbor w of v. Then the following is true:*

$$(\mathbf{I}_N - \mathbf{U})^{-2} = \mathbb{E}[\mathbf{B}(\mathbf{B}')^\top]. \tag{6}$$

Time complexity & signature vector maps: To compute $\phi(i)$ for every i, we need to run in every node m random walks, each of the expected length $l = \frac{1}{p_{\mathrm{term}}}$, thus the expected time complexity of computing $\mathbf{B}, \mathbf{B}'$ is $O(\frac{N m t_{\mathrm{sam}}}{p_{\mathrm{term}}} + T_{\mathrm{rep}})$, where: t_{sam} stands for time complexity of the mechanism sample and T_{rep} for the space complexity of storing the representation of $\mathrm{G}_{\mathbf{U}}$. For instance, if $\mathbf{U}$ is sparse, graph adjacency list representation is a right choice. In principle, even more compact representations are possible if graph-edges do not need to be stored explicitly. For the most straightforward uniform sampling strategy, we have: $t_{\mathrm{sam}} = O(1)$ and for many more sophisticated ones t_{sam} is proportional to the average degree of a vertex. We assume that in all considered representations there is an indexing mechanism for the neighbors of any given vertex and a vertex of a particular index can be retrieved in $O(1)$ time. Note that for $p_{\mathrm{term}} \gg \frac{m t_{\mathrm{sam}}}{N}$, $\mathbf{B}, \mathbf{B}'$ should be kept in the implicit manner (this is a pretty conservative lower bound; in practice different walks will share many vertices), since most of the entries of each of its rows are equal to zero in this case. Thus for every i, we can store a dictionary with keys corresponding to visited vertices w and values to the values of $\phi_i[w]$. We call such a dictionary a *signature vector map* (or svmap). In practice, one can choose large p_{term} without accuracy drop (see: Sec. 5).

3.1.3. Trimming RFs: anchor points and JLT

Algorithm 1 provides substantial computational gains in calculating $(\mathbf{I}_N - \mathbf{U})^{-2}$, replacing $O(N^3)$ time complexity with expected $O(\frac{Nm}{p_{\mathrm{term}}} + T_{\mathrm{rep}})$. Furthermore, signature vectors $\phi(i)$ in practice can be often stored efficiently (see: our discussion on svmaps above). Interestingly, there are also easy ways to directly control their "effective dimensionality" via one of two simple to implement tricks, provided below.

Anchor points: This technique samples randomly $K < N$ points from the set of all N vertices of $\mathrm{G}_{\mathbf{U}}$. We call this set: *anchor points*. Signature vectors are updated only in anchor points (but we still upload load for each transition) and thus effectively each of them has dimensionality K. In principle, different sampling strategies are possible and analyzing all of them in outside the scope of this work. The simplest approach is to sample uniformly at random a K-element subset of V with no repetitions. In this setting, if we replace term $\frac{1}{m}$ in point 4. of the Algorithm 1 box with $\frac{N}{Km}$ term, the estimator remains unbiased (see: Sec. 4).

JLTed signature vectors: The more algebraic approach to trimming the dimensionality of signature vectors is to apply Johnson-Lindenstrauss Transform (see: Sec. 1). We simply replace $\phi(i)$ with: $\phi_{\mathrm{JLT}}(i) = \frac{1}{\sqrt{K}} \mathbf{G} \phi(i)$ after all the computations. Here $\mathbf{G} \in \mathbb{R}^{K \times N}$ is the Gaussian matrix with entries taken independently at random from $\mathcal{N}(0, 1)$ and K is the number of RFs used in JLT. Matrix $\mathbf{G}$ is sampled independently from all the walks and is the same for all the nodes. The unbiasedness of the dot-product kernel estimation with JLTs (Johnson, 1984b) combined with Theorem 3.2, provides the unbiasedness of the overall mechanism.

3.2. From squared inverse matrices to graph kernels

3.2.1. Preliminaries

We are ready to show how the results obtained in Sec. 3.1 can be applied for graph kernels. For a given undirected weighted loopless graph G with a weighted symmetric

adjacency matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$, we consider the family of *d-regularized Laplacian kernels* defined as follows for $\mathrm{deg}_{\mathbf{W}}(i) \stackrel{\mathrm{def}}{=} \sum_{j \in \mathrm{N}_i} \mathbf{W}(i,j)$:

$$[\mathrm{K}^d_{\mathrm{lap}}(v,w)]_{i,j=1,...,N} = (\mathbf{I}_N + \sigma^2 \tilde{\mathbf{L}}_{\mathrm{G}})^{-d}, \quad (7)$$

for a symetrically normalized Laplacian $\tilde{\mathbf{L}}_{\mathrm{G}} \in \mathbb{R}^{N \times N}$:

$$\tilde{\mathbf{L}}_{\mathrm{G}}(v,w) = \begin{cases} 1 & \text{if } v = w \\ -\frac{\mathbf{W}(v,w)}{\sqrt{\mathrm{deg}_{\mathbf{W}}(v)\mathrm{deg}_{\mathbf{W}}(w)}} & \text{if } (v,w) \text{ is an edge} \end{cases} \quad (8)$$

For $d = 1$, we get the popular regularized Laplacian kernel. As we show in Sec. A.2, as long as the 1-regularized Laplacian kernel is well defined (i.e. matrix $\mathbf{I}_N + \sigma^2 \tilde{\mathbf{L}}_{\mathrm{G}}$ is invertible and the inverse has positive entries), the d-regularized Laplacian kernel is a valid positive definite kernel.

3.2.2. Signature vectors are RFs for the 2-regularized Laplacian kernel

We will start with $d = 2$ since it turns out that the mechanism of signature vectors can be straightforwardly applied there to obtain corresponding GRFs. Note that the following is true for $\mathbf{U} \in \mathbb{R}^{N \times N}$ defined as: $\mathbf{U} = [u_{i,j}]_{i,j=1,...,N}$, where $u_{i,j} = \frac{\sigma^2}{\sigma^2+1} \frac{\mathbf{W}(i,j)}{\sqrt{\mathrm{deg}_{\mathbf{W}}(i)\mathrm{deg}_{\mathbf{W}}(j)}}$:

$$\mathbf{I}_N + \sigma^2 \tilde{\mathbf{L}}_{\mathrm{G}} = (\sigma^2 + 1)(\mathbf{I}_N - \mathbf{U}) \quad (9)$$

Equation 9, together with Theorem 3.2, immediately provide the GRF-mechanism for the 2-regularized Laplacian kernel:

Corollary 3.3. *Consider a kernel matrix of the 2-regularized Laplacian kernel for a given graph* G *with* N *nodes* $\{1, 2, ..., N\}$. *Then the following holds:*

$$(\mathbf{I}_N + \sigma^2 \tilde{\mathbf{L}}_{\mathrm{G}})^{-2} = \mathbb{E}[\mathbf{C}(\mathbf{C}')^T], \quad (10)$$

where the rows of $\mathbf{C} \in \mathbb{R}^{N \times K}$ *are of the form:* $\frac{1}{\sigma^2+1}\phi(i)^\top$ *for* $i = 1, ..., N$ *and signature vectors* $\phi(i)$ *computed via Algorithm 1 for the graph* $\mathrm{G}_{\mathbf{U}}$ *and a matrix* $\mathbf{U}$ *defined above. Furthermore,* $\mathbf{C}'$ *is an independent copy of* $\mathbf{C}$.

3.2.3. GRFs & regularized Laplacian kernel

Let us now consider the case $d = 1$. From Corollary 3.3, we immediately get the following:

Corollary 3.4. *Consider a kernel matrix of the 1-regularized Laplacian kernel for a given graph* G *with* N *nodes* $\{1, 2, ..., N\}$. *Then the following holds:*

$$(\mathbf{I}_N + \sigma^2 \tilde{\mathbf{L}}_{\mathrm{G}})^{-1} = \mathbb{E}[\mathbf{C}\mathbf{D}^T], \quad (11)$$

where matrix $\mathbf{D} \in \mathbb{R}^{N \times K}$ *is defined as:* $\mathbf{D} = (\mathbf{I}_N + \sigma^2 \tilde{\mathbf{L}}_{\mathrm{G}})\mathbf{C}'$ *and* $\mathbf{C}, \mathbf{C}' \in \mathbb{R}^{N \times K}$ *are as in Corollary 3.3.*

Corollary 3.4 provides a GRF mechanism for the regularized Laplacian kernel, since it implies that we can write $\mathrm{K}^1_{\mathrm{lap}}$ as:

$$\mathrm{K}^1_{\mathrm{lap}}(i,j) = \mathbb{E}[\phi(i)^\top \psi(j)], \quad (12)$$

where $\phi(i)$ and $\psi(i)$ for $i = 1, ..., N$ are the rows of $\mathbf{C}$ and $\mathbf{D}$ respectively. Notice that this is the so-called *asymmetric RF setting* (Choromanski et al., 2022b) (where we use a different transformation for i and j). The approximate kernel matrix is not necessarily symmetric, but its expectation is always symmetric and the estimation is unbiased. However it is also easy to obtain here the GRF mechanism providing symmetric approximate kernel matrix and maintaining unbiasedness. It suffices to define:

$$\begin{aligned} \phi'(i) &= \frac{1}{\sqrt{2}}[\phi(i)^\top, \psi(i)^\top]^\top, \\ \psi'(j) &= \frac{1}{\sqrt{2}}[\psi(j)^\top, \phi(j)^\top]^\top, \end{aligned} \quad (13)$$

and the corresponding approximate kernel matrix is of the form: $\frac{1}{2}(\mathbf{C}\mathbf{D}^\top + \mathbf{D}\mathbf{C}^\top)$.

3.2.4. Going beyond $d = 2$

Let us see now what happens when we take $d > 2$. We will define GRFs recursively, having the variants for $d = 1$ and $d = 2$ already established in previous sections (they will serve as a base for our induction analysis). Take some d and assume that the following GRF mechanism defined by the unbiased low-rank decomposition of the d-regularized Laplacian kernel matrix exists for some $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{N \times K}$:

$$(\mathbf{I}_N + \sigma^2 \tilde{\mathbf{L}}_{\mathrm{G}})^{-d} = \mathbb{E}[\mathbf{X}\mathbf{Y}^\top] \quad (14)$$

Then, using Corollary 3.3, we can rewrite:

$$(\mathbf{I}_N + \sigma^2 \tilde{\mathbf{L}}_{\mathrm{G}})^{-d-2} = \mathbb{E}[\mathbf{X}\mathbf{Y}^\top \mathbf{C}(\mathbf{C}')^\top], \quad (15)$$

if we assume that matrices $\mathbf{C}, \mathbf{C}' \in \mathbb{R}^{N \times K}$ are constructed independently from $\mathbf{X}$ and $\mathbf{Y}$. We can then compute in time $O(NK^2)$ matrix $\mathbf{S} = \mathbf{Y}^\top \mathbf{C}$ and then conduct its SVD-decomposition: $\mathbf{S} = \mathbf{Z}\Sigma\mathbf{Z}^\top$ in time $O(K^3)$ for $\mathbf{Z}, \Sigma \in \mathbb{R}^{K \times K}$ and where Σ is diagonal. Then we can rewrite:

$$(\mathbf{I}_N + \sigma^2 \tilde{\mathbf{L}}_{\mathrm{G}})^{-d-2} = \mathbb{E}[(\mathbf{X}\mathbf{Z}\Sigma^{\frac{1}{2}})(\mathbf{C}'\mathbf{Z}\Sigma^{\frac{1}{2}})^\top] \quad (16)$$

for $\mathbf{X}\mathbf{Z}\Sigma^{\frac{1}{2}}, \mathbf{C}'Z\Sigma^{\frac{1}{2}} \in \mathbb{R}^{N \times K}$. The random feature vectors can be then given as the rows of $\mathbf{X}\mathbf{Z}\Sigma^{\frac{1}{2}}$ and $\mathbf{C}'Z\Sigma^{\frac{1}{2}}$.

We see that starting with $d = 1, 2$ we can then follow the above procedure to produce GRF-based low-rank decomposition of the kernel matrix for the d-regularized Laplacian kernel for any $d \in \mathbb{N}_+$.

3.2.5. GRFs for fast graph kernel methods

We will assume here that the graphs G under consideration have e edges for $e = o(N^2)$ which is a reasonable

assumption in machine learning and thus graph adjacency list representation is a default choice. Using the analysis from Sec. 3.1.2, we conclude that GRFs can be computed in time $O(\frac{Nmt_{\mathrm{sam}}}{p_{\mathrm{term}}} + Ne)$ for $d = 2$. If uniform sampling is the strategy used by sample then time complexity can be simplified to $O(\frac{Nm}{p_{\mathrm{term}}} + Ne)$. If JLT-based dimensionality is applied, an additional cost $O(N^2K)$ is incurred. This can be further reduced to $O(N^2 \log(K))$ if unbiased structured JLT variants are applied (Choromanski et al., 2017). For $d = 1$, an additional time $O(Ne)$ (for computing matrix $\mathbf{D}$, see: Corollary 3.4) is incurred. Similar analysis can be applied to larger d. It is easy to see that for $d > 2$ GRFs can be computed in time cubic in K and linear in d. Thus for $d > 2$ they are useful if $K \ll N$. We call this stage **graph pre-processing**. If the brute-force approach is applied, pre-processing involves explicit computation of the graph kernel matrix and for the graph kernels considered in this paper (as well as many others) takes time $O(N^3)$. In some applications, pre-processing is a one-time procedure (per training), but in many others needs to be applied several times in training (e.g. in graph Transformers, where graph kernels can be used to topologically modulate regular attention mechanism) (Choromanski et al., 2022a).

When the graph pre-processing is completed, graph kernels usually interact with the downstream algorithm via their matrix-vector multiplication interface, e.g. the algorithm computes $\mathbf{Kx}$ for a given graph kernel matrix $\mathbf{K} = [\mathrm{K}(i,j)]_{i,j=1,\ldots,N} \in \mathbf{R}^{N \times N}$ and a series of vectors $\mathbf{x} \in \mathbb{R}^N$ (see: k-means clustering with graph kernels: (Dhillon et al., 2004)). We will refer to this phase as **inference**. Brute-force inference can be trivially computed in time $O(N^2)$ per matrix-vector multiplication. Using GRFs strategy, an **unbiased** estimation of the matrix-vector product can be computed via a series of matrix-vector multiplications obtained from (a chain of) decompositions constructed to define GRFs. For $d \neq 2$ those decompositions involve at least three matrices and thus the GRFs do not even need to be explicitly constructed.

For instance, for $d = 2$, inference can be conducted in time $O(NK)$ and this is a pretty conservative bound. Even if one applies $K = \Omega(N)$, most of the entries of random feature vectors can be still equal to zero (if p_{term} is not too small) and in such a setting svmaps from Sec. 3.1.2 can be applied to provide sub-quadratic in N time complexity. For $d = 1$, an additional cost $O(Ne)$ of multiplications with matrices $\mathbf{I}_N + \sigma^2 \tilde{\mathbf{L}}_{\mathrm{G}}$ needs to be incurred. For $d > 2$, inference can be directly conducted in time $O(NK)$.

3.2.6. Systems of linear equations & GRFs

It is easy to see that there is nothing particularly special about the Laplacian matrix $\mathbf{I}_N + \sigma^2 \tilde{\mathbf{L}}_{\mathrm{G}}$ from Corollary 3.4 and that for any invertible matrix of the form: $\lambda(\mathbf{I}_N - \mathbf{U})$ and symmetric $\mathbf{U} \in \mathbb{R}^{N \times N}$ of $o(N^2)$ nonzero entries, one can find a decomposition:

$$\frac{1}{\lambda}(\mathbf{I}_N - \mathbf{U})^{-1} = \mathbb{E}[\mathbf{C}\mathbf{D}^\top] \tag{17}$$

in sub-cubic time, using our methods. That however provides immediately a sub-cubic randomized algorithm for unbiasedly solving linear systems of the form: $(\mathbf{I}_N - \mathbf{U})\mathbf{x} = \mathbf{b}$ for any given $\mathbf{b} \in \mathbb{R}^N$ as: $\mathbf{x} = \lambda(\mathbf{C}(\mathbf{D}^\top \mathbf{b}))$, where brackets indicate the order of computations.

3.2.7. Final remarks

The construction of GRFs presented in this section can be thought of as a low-rank decomposition of the kernel matrix $\mathbf{K}$. Then one can argue that instead of using presented algorithm, a standard low-rank decomposition method can be applied (such as spectral analysis). This however defeats the main goal of this paper - sub-cubic time complexity (the matrix to be decomposed would need to be explicitly constructed) and unbiased estimation (such an approximation would be biased). The latter remains a problem for the alternative approaches even if more refined low-rank decomposition techniques not enforcing materialization of the kernel matrix $\mathbf{K}$ are applied (such methods would however rely on efficient algorithms for computing $\mathbf{Kx}$ which are non-trivial: for $d = 1$ would need to solve systems of linear equations with symmetric matrices as sub-routines).

4. Theoretical results

4.1. GRFs provide unbiased graph kernel estimation

We start by proving Theorem3.2. Concentration results for GRFs in the base setting, where the sampler chooses a neighbor uniformly at random, are given in Sec. A.3. We leave as an exciting open question the analysis of the concentration results beyond second moment methods.

Proof. For any two nodes $a, b \in \mathrm{V}(\mathrm{G}_{\mathbf{U}})$, denote by $\Omega(a,b)$ the set of walks from a to b in $\mathrm{G}_{\mathbf{U}}$. Furthermore, symbol $\subseteq_{\mathrm{pre}}$ indicates than one walk is a prefix of the other. Then:

$$\phi(i)^\top \phi(j) = \frac{1}{m^2} \sum_{x \in \mathrm{V}(\mathrm{G}_{\mathbf{U}})} \sum_{k=1}^{m} \sum_{l=1}^{m} \sum_{\omega_1 \in \Omega(i,x)} \sum_{\omega_2 \in \Omega(j,x)} w(\omega_1) \cdot w(\omega_2) \prod_{s=1}^{l_1} \frac{(p^i_{k,s})^{-1}}{1 - p_{\mathrm{term}}} \prod_{t=1}^{l_2} \frac{(p^j_{l,t})^{-1}}{1 - p_{\mathrm{term}}} \mathbb{1}[\omega_1 \subseteq_{\mathrm{pre}} \bar{\Omega}(k,i)] \mathbb{1}[\omega_2 \subseteq_{\mathrm{pre}} \bar{\Omega}(l,j)], \tag{18}$$

where $\bar{\Omega}(k,i)$ and $\bar{\Omega}(l,j)$ stand for the kth and l^{th} sampled random walk from i and j respectively, function w outputs the product of the edge-weights of its given input walk and $p^i_{k,s}$ and $p^j_{l,t}$ are the probabilities returned by sample for the sth/t^{th} vertex of the kth/l^{th} sampled random walk starting

at i/j. Furthermore, l_1 and l_2 refer to the number of edges of ω_1 and ω_2 respectively. Note that those probabilities are in general random variables (if they are chosen as nontrivial functions of the history of sampled walks).

Therefore, for $\mathrm{len}(\omega)$ denoting the number of edges of ω:

$$\mathbb{E}[\phi(i)^\top \phi(j)] = \sum_{x \in \mathrm{V}(\mathrm{G}_\mathbf{U})} \sum_{\omega_1 \in \Omega(i,x)} \sum_{\omega_2 \in \Omega(j,x)} w(\omega_1) w(\omega_2) = \sum_{\omega \in \Omega(i,j)} (\mathrm{len}(\omega)+1) w(\omega) \tag{19}$$

The proof is completed by an observation that:

$$(\mathbf{I}_N + 2\mathbf{U} + 3\mathbf{U}^2 + ...)(i,j) = \sum_{\omega \in \Omega(i,j)} (\mathrm{len}(\omega)+1) w(\omega) \tag{20}$$

This in turn follows from the fact that for any $k \geq 0$: $\mathbf{U}^k(i,j)$ can be interpreted as the sum of $w(\omega)$ over all walks ω from i to j of k edges. □

The extension to the anchor point setting is obtained by following the same proof, but with x iterating only over sampled anchor points and results in adding a multiplicative constant, as explained in Sec. 3.1.3.

4.2. q-GRFs and correlated random walks

As already mentioned, the idea of q-GRFs is to correlate different random walks initiated in a given node i, to "more efficiently" explore the entire graph (in particular, by avoiding reconstructing similar walks) and can be thought of as an analogue of the ORF or low discrepancy sequences techniques applied for regular RFs (Yang et al., 2014). A natural candidate for the q-GRF variant is the one, where method sample implements the so-called *reinforced random walk* strategy (Kozma, 2012). The probability of choosing a neighbor w of v (see: Sec. 3.1.2) can be defined as:

$$p(v,w) = \frac{f(N(v,w))}{\sum_{z \in \mathrm{N}_v} f(N(v,z))}, \tag{21}$$

where $\mathrm{N}(x,y)$ stands for the number of times that an edge $\{x,y\}$ has been already used (across all the walks, or previous walks) and $f : \mathbb{N} \rightarrow \mathbb{R}$ is a fixed function. Note that, since we want to deprioritize edges that were already frequently visited, f should be a decreasing function.

Implementing such a strategy can be done straightforwardly with an additional multiplicative factor d_{ave} in time-complexity, where d_{ave} stands for the average degree of a graph node. We leave detailed theoretical and empirical analysis of this class of methods to future work.

5. Experiments

We empirically verify the quality of GRFs via various experiments, including downstream applications of graph kernels. The real-world graphs were accessed from the repositories described in (Ivashkin, 2023).

5.1. Speed tests

Here we compared the total number of FLOPS used by the GRF-variant of of the regularized Laplacian kernel in pre-processing followed by a single inference call involving computing the action of the kernel matrix on a given vector (this is for instance the way in which kernels are applied in the kernelized k-means algorithm, see: Sec. 5.3) with the correesponding time for various linear system solvers as well as a brute-force (BF) algorithm. Note that for d-regularized Laplacian kernels with $d = 1$, inference straightforwardly reduces to solving linear systems of equations. We used graphs of different size. The results are presented in Table 1. GRFs provide substantial reduction of FLOPS, even for smaller graphs. We took $p_{\mathrm{term}} = 0.1$ since it worked well in several other tests (see: Sec. 5.2, Sec. 5.3).

GRF	BF	Gauss-Seidel	Conjugate Gradient	Jacobi
960K	512.64M	6400K	512M	6400K
1.4M	1001M	10M	1000M	10M
10.2M	27B	90M	27B	90M

Table 1. The comparison of the number of FLOPS for the setting from Sec. 5.1 for different linear systems solvers and GRFs. Different rows correspond to: $N = 800$, $N = 1000$ and $N = 3000$.

5.2. Relative Frobenius norm error

We took several undirected graphs of different sizes and computed for them groundtruth kernel matrices $\mathbf{K}^d = [\mathrm{K}^d_{\mathrm{lap}}(i,j)]_{i,j=1,...,N}$ using d-regularized Laplacian kernels with $d = 1, 2$. We then computed their counterparts $\widehat{\mathbf{K}}^d$ obtained via GRFs and the corresponding relative Frobenius norm error defined as:

$$\epsilon = \frac{\|\mathbf{K}^d - \widehat{\mathbf{K}}^d\|_{\mathrm{F}}}{\|\mathbf{K}^d\|_{\mathrm{F}}} \tag{22}$$

We considered different termination probabilities: $p_{\mathrm{term}} \in \{0.1, 0.06, 0.01\}$ leading to average random walk lengths: $10, \frac{50}{3}, 100$ respectively as well as different number m of random walks for the construction of the GRF-vector: $m = 1, 2, 10, 20, 40, 80$. We fixed: $\sigma^2 = 0.2$. The reported empirical relative Frobenium norm errors were obtained by averaging over $s = 10$ independent experiments. We also reported their corresponding standard deviations.

The results are presented in Fig. 2. We see that different p_{term} result in very similar error-curves and thus in practice it suffices to take $p_{\mathrm{term}} = 0.1$ (average walk length $l = 10$), regardless of the density of the graph, even for graphs with $N > 1000$ nodes. Interestingly, error-curves are also very similar across all the graphs. Furthermore, small number of random walks $m = 80$ suffices to obtain $\epsilon < 2\%$. The standard deviations are reported on the plots, but since they are very small, we also include them in Sec. A.1.

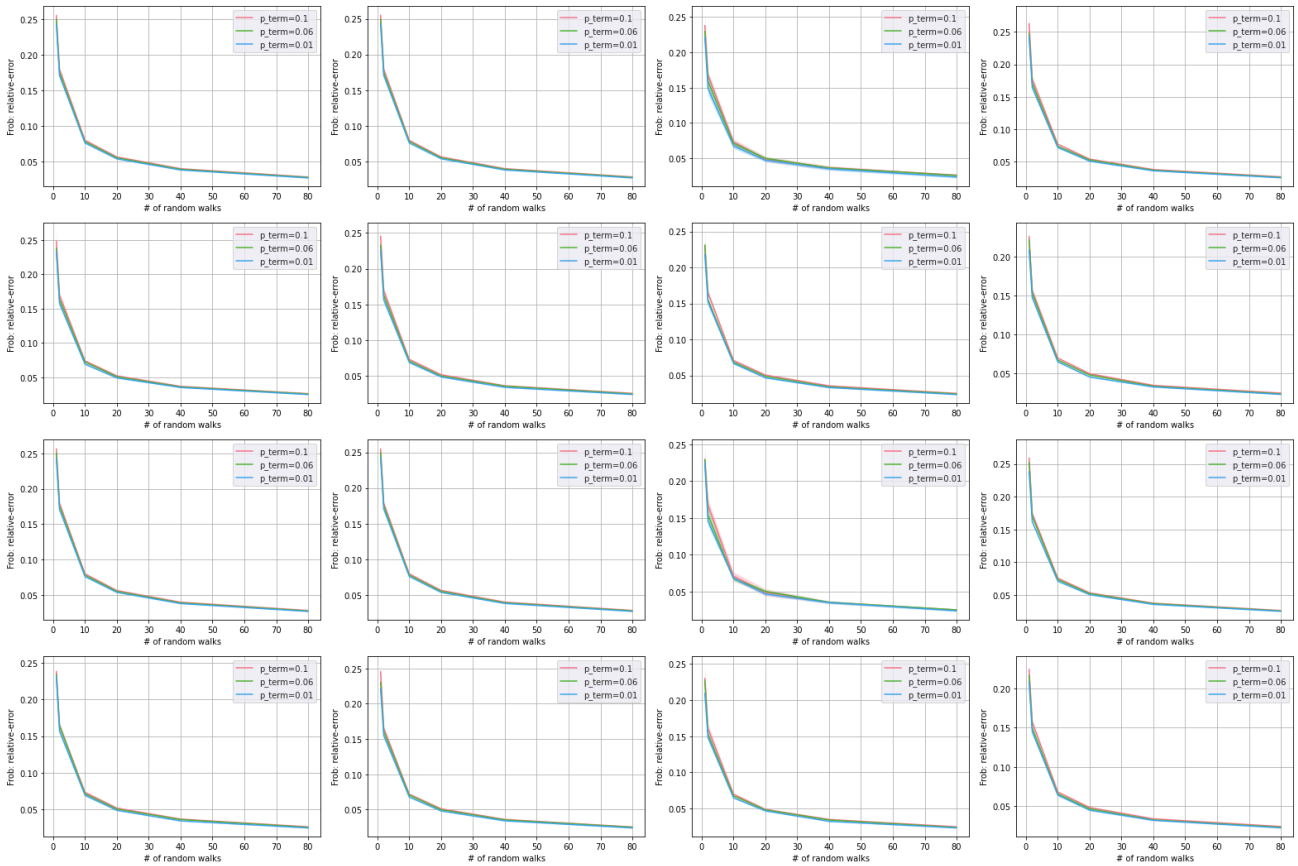

Figure 2. Relative Frobenius norm error for the setting described in Sec. 5.2. First two rows correspond to $d = 1$ and last two to $d = 2$. We considered the following graphs (from upper-left to lower-right): Erdos-Renyi graph with edge prob. 0.4 (ER-0.4, $N = 1000$), ER-0.1 ($N = 1000$), dolphins ($N = 62$), eurosis ($N = 1272$), Networking ($N = 1249$), Databases ($N = 1046$), Encryption-and-Compression ($N = 864$), Hardware-and-Architecture ($N = 763$).

5.3. K-means algorithm with graph kernels

We applied GRFs in kernelized k-means algorithm (Dhillon et al., 2004) to cluster graph nodes using graph kernels. As in the previous section, we applied d-regularized Laplacian graph kernels with $d = 1, 2$. Furthermore, we have measured the impact of the additional dimensionality reduction techniques (anchor points and JLT-based reduction) on the clustering quality. For each variant, we reported the so-called *clustering error* defined as $\epsilon = \frac{P_{\text{error}}}{\binom{N}{2}}$, where P_{error} stands for the number of pairs of nodes $\{i, j\}$ of the graph that were classified differently by the groundtruth algorithm and its GRF-variant (e.g. they belonged to the same cluster in the groundtruth variant and two different clusters in the GRF one or vice versa). In all the experiments we used $\sigma^2 = 0.2$, $p_{\text{term}}^{-1} m \leq 400$ and $K \leq 0.6N$. We chose the no of clusters $\text{nb_clusters} = 3$. The results are in Table 2. GRFs, even with more accurate of the: anchor points and JLTs methods provide accurate approximation, for some graphs (e.g. citeseer with $N = 2120$) with error $< 1\%$.

Kernel, method	citeseer	Databases	polbooks	karate
d=1, reg	0.020	0.170	0.28	0.11
d=1, min(jlt, anc)	0.010	0.097	0.323	0.314
d=2, reg	0.008	0.140	0.12	0.032
d=2, min(jlt, anc)	0.010	0.098	0.264	0.198

Table 2. Clustering errors for the experiments from Sec. 5.3 (for the d-regularized Laplacian kernel). Different tested methods: (1) regular GRFs (reg, no compression), (2) GRFs with the more accurate among: JLT (jlt), and anchor points (anc) techniques.

6. Conclusion

We have proposed in this paper a new paradigm of graph random features (GRFs) for the unbiased and computationally efficient estimation of various kernels defined on the nodes of the graph. GRFs rely on the families of random walks initiated in different nodes, depositing loads in the visited vertices of the graph. The computation of GRFs can be also easily paralellized. We have provided detailed theoretical analysis of our method and exhaustive experiments, involving in particular kmeans clustering on graphs.

References

Boser, B. E., Guyon, I., and Vapnik, V. A training algorithm for optimal margin classifiers. In Haussler, D. (ed.), *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992*, pp. 144–152. ACM, 1992. doi: 10.1145/130385.130401. URL https://doi.org/10.1145/130385.130401.

Bressan, M. Faster algorithms for sampling connected induced subgraphs. *CoRR*, abs/2007.12102, 2020. URL https://arxiv.org/abs/2007.12102.

Campbell, C. Kernel methods: a survey of current techniques. *Neurocomputing*, 48(1-4):63–84, 2002. doi: 10.1016/S0925-2312(01)00643-9. URL https://doi.org/10.1016/S0925-2312(01)00643-9.

Canu, S. and Smola, A. J. Kernel methods and the exponential family. *Neurocomputing*, 69(7-9):714–720, 2006. doi: 10.1016/j.neucom.2005.12.009. URL https://doi.org/10.1016/j.neucom.2005.12.009.

Chávez, G., Liu, Y., Ghysels, P., Li, X. S., and Rebrova, E. Scalable and memory-efficient kernel ridge regression. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), New Orleans, LA, USA, May 18-22, 2020*, pp. 956–965. IEEE, 2020. doi: 10.1109/IPDPS47924.2020.00102. URL https://doi.org/10.1109/IPDPS47924.2020.00102.

Chen, X., Li, Y., Wang, P., and Lui, J. C. S. A general framework for estimating graphlet statistics via random walk. *Proc. VLDB Endow.*, 10(3):253–264, 2016. doi: 10.14778/3021924.3021940. URL http://www.vldb.org/pvldb/vol10/p253-chen.pdf.

Choromanski, K., Downey, C., and Boots, B. Initialization matters: Orthogonal predictive state recurrent neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018a. URL https://openreview.net/forum?id=HJJ23bW0b.

Choromanski, K., Rowland, M., Sarlós, T., Sindhwani, V., Turner, R. E., and Weller, A. The geometry of random features. In Storkey, A. J. and Pérez-Cruz, F. (eds.), *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1–9. PMLR, 2018b. URL http://proceedings.mlr.press/v84/choromanski18a.html.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Choromanski, K., Lin, H., Chen, H., Zhang, T., Sehanobish, A., Likhosherstov, V., Parker-Holder, J., Sarlós, T., Weller, A., and Weingarten, T. From block-toeplitz matrices to differential equations on graphs: towards a general theory for scalable masked transformers. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S. (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 3962–3983. PMLR, 2022a. URL https://proceedings.mlr.press/v162/choromanski22a.html.

Choromanski, K. M., Rowland, M., and Weller, A. The unreasonable effectiveness of structured random orthogonal embeddings. *Advances in neural information processing systems*, 30, 2017.

Choromanski, K. M., Lin, H., Chen, H., Sehanobish, A., Ma, Y., Jain, D., Varley, J., Zeng, A., Ryoo, M. S., Likhosherstov, V., Kalashnikov, D., Sindhwani, V., and Weller, A. Hybrid random features. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022b. URL https://openreview.net/forum?id=EMigfE6ZeS.

Chung, F. R. K. and Yau, S. Coverings, heat kernels and spanning trees. *Electron. J. Comb.*, 6, 1999. doi: 10.37236/1444. URL https://doi.org/10.37236/1444.

Cortes, C. and Vapnik, V. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995. doi: 10.1007/BF00994018. URL https://doi.org/10.1007/BF00994018.

Cumby, C. M. and Roth, D. On kernel methods for relational learning. In Fawcett, T. and Mishra, N. (eds.), *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pp. 107–114. AAAI Press, 2003. URL http://www.aaai.org/Library/ICML/2003/icml03-017.php.

Dhillon, I. S., Guan, Y., and Kulis, B. Kernel k-means: spectral clustering and normalized cuts. In Kim, W., Kohavi, R., Gehrke, J., and DuMouchel, W. (eds.), *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pp. 551–556. ACM, 2004. doi: 10.1145/1014052.1014118. URL https://doi.org/10.1145/1014052.1014118.

Fang, J., Zhang, Q., Meng, Z., and Liang, S. Structure-aware random fourier kernel for graphs. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 17681–17694, 2021.

Figueiredo, D. R. and Garetto, M. On the emergence of shortest paths by reinforced random walks. *IEEE Trans. Netw. Sci. Eng.*, 4(1):55–69, 2017. doi: 10.1109/TNSE.2016.2618063. URL https://doi.org/10.1109/TNSE.2016.2618063.

Ghojogh, B., Ghodsi, A., Karray, F., and Crowley, M. Reproducing kernel hilbert space, mercer's theorem, eigenfunctions, nyström method, and use of kernels in machine learning: Tutorial and survey. *CoRR*, abs/2106.08443, 2021. URL https://arxiv.org/abs/2106.08443.

Goemans, M. X. and Williamson, D. P. Approximation algorithms for $\mathrm{m_{ax}}$-3-$\mathrm{c_{ut}}$ and other problems via complex semidefinite programming. *J. Comput. Syst. Sci.*, 68(2):442–470, 2004. doi: 10.1016/j.jcss.2003.07.012. URL https://doi.org/10.1016/j.jcss.2003.07.012.

Goldin, R. F. Review: Statistical models: Theory and practice (revised edition). cambridge university press, new york, new york, 2009, xiv + 442 pp., ISBN 978-0-521-74385-3, $40. by david a. freedman. *Am. Math. Mon.*, 117(9):844–847, 2010. doi: 10.4169/000298910X521733. URL https://doi.org/10.4169/000298910X521733.

Ivashkin, V. Community graphs repository, 2023. URL https://github.com/vlivashkin/community-graphs.

Johnson, W. B. Extensions of lipschitz mappings into hilbert space. *Contemporary mathematics*, 26:189–206, 1984a.

Johnson, W. B. Extensions of lipschitz mappings into a hilbert space. *Contemp. Math.*, 26:189–206, 1984b.

Kégl, B., Krzyzak, A., and Niemann, H. Radial basis function networks in nonparametric classification and function learning. In Jain, A. K., Venkatesh, S., and Lovell, B. C. (eds.), *Fourteenth International Conference on Pattern Recognition, ICPR 1998, Brisbane, Australia, 16-20 August, 1998*, pp. 565–570. IEEE Computer Society, 1998. doi: 10.1109/ICPR.1998.711206. URL https://doi.org/10.1109/ICPR.1998.711206.

Kondor, R. and Lafferty, J. D. Diffusion kernels on graphs and other discrete input spaces. In Sammut, C. and Hoffmann, A. G. (eds.), *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*, pp. 315–322. Morgan Kaufmann, 2002.

Kontorovich, L., Cortes, C., and Mohri, M. Kernel methods for learning languages. *Theor. Comput. Sci.*, 405(3):223–236, 2008. doi: 10.1016/j.tcs.2008.06.037. URL https://doi.org/10.1016/j.tcs.2008.06.037.

Kozma, G. Reinforced random walk, 2012. URL https://arxiv.org/abs/1208.0364.

Leskovec, J. and Faloutsos, C. Sampling from large graphs. In Eliassi-Rad, T., Ungar, L. H., Craven, M., and Gunopulos, D. (eds.), *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pp. 631–636. ACM, 2006. doi: 10.1145/1150402.1150479. URL https://doi.org/10.1145/1150402.1150479.

Likhosherstov, V., Choromanski, K., Dubey, A., Liu, F., Sarlos, T., and Weller, A. Chefs' random tables: Non-trigonometric random features. In *NeurIPS*, 2022.

Lin, H., Chen, H., Choromanski, K. M., Zhang, T., and Laroche, C. Demystifying orthogonal monte carlo and beyond. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Liu, F., Huang, X., Chen, Y., and Suykens, J. A. K. Random features for kernel approximation: A survey on algorithms, theory, and beyond. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(10):7128–7148, 2022. doi: 10.1109/TPAMI.2021.3097011. URL https://doi.org/10.1109/TPAMI.2021.3097011.

Orbanz, P. Subsampling large graphs and invariance in networks. *arXiv: Statistics Theory*, 2017.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Rahimi, A. and Recht, B. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pp. 1313–1320. Curran Associates, Inc., 2008.

Ribeiro, P., Paredes, P., Silva, M. E. P., Aparício, D., and Silva, F. M. A. A survey on subgraph counting: Concepts, algorithms, and applications to network motifs and graphlets. *ACM Comput. Surv.*, 54(2):28:1–28:36, 2022. doi: 10.1145/3433652. URL https://doi.org/10.1145/3433652.

Shervashidze, N., Vishwanathan, S. V. N., Petri, T., Mehlhorn, K., and Borgwardt, K. M. Efficient graphlet kernels for large graph comparison. In Dyk, D. A. V. and Welling, M. (eds.), *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, volume 5 of *JMLR Proceedings*, pp. 488–495. JMLR.org, 2009. URL http://proceedings.mlr.press/v5/shervashidze09a.html.

Smola, A. J. and Kondor, R. Kernels and regularization on graphs. In Schölkopf, B. and Warmuth, M. K. (eds.), *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, volume 2777 of *Lecture Notes in Computer Science*, pp. 144–158. Springer, 2003. doi: 10.1007/978-3-540-45167-9_12. URL https://doi.org/10.1007/978-3-540-45167-9_12.

Smola, A. J. and Schölkopf, B. Bayesian kernel methods. In Mendelson, S. and Smola, A. J. (eds.), *Advanced Lectures on Machine Learning, Machine Learning Summer School 2002, Canberra, Australia, February 11-22, 2002, Revised Lectures*, volume 2600 of *Lecture Notes in Computer Science*, pp. 65–117. Springer, 2002. doi: 10.1007/3-540-36434-X_3. URL https://doi.org/10.1007/3-540-36434-X_3.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242, 2010. doi: 10.5555/1756006.1859891. URL https://dl.acm.org/doi/10.5555/1756006.1859891.

Wang, T., Chen, Y., Zhang, Z., Xu, T., Jin, L., Hui, P., Deng, B., and Li, X. Understanding graph sampling algorithms for social network analysis. In *31st IEEE International Conference on Distributed Computing Systems Workshops (ICDCS 2011 Workshops), 20-24 June 2011, Minneapolis, Minnesota, USA*, pp. 123–128. IEEE Computer Society, 2011. doi: 10.1109/ICDCSW.2011.34. URL https://doi.org/10.1109/ICDCSW.2011.34.

Wu, L., Yen, I. E., Zhang, Z., Xu, K., Zhao, L., Peng, X., Xia, Y., and Aggarwal, C. C. Scalable global alignment graph kernel using random features: From node embedding to graph embedding. In Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., and Karypis, G. (eds.), *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pp. 1418–1428. ACM, 2019. doi: 10.1145/3292500.3330918. URL https://doi.org/10.1145/3292500.3330918.

Yanardag, P. and Vishwanathan, S. V. N. Deep graph kernels. In Cao, L., Zhang, C., Joachims, T., Webb, G. I., Margineantu, D. D., and Williams, G. (eds.), *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pp. 1365–1374. ACM, 2015. doi: 10.1145/2783258.2783417. URL https://doi.org/10.1145/2783258.2783417.

Yang, J., Sindhwani, V., Avron, H., and Mahoney, M. W. Quasi-monte carlo feature maps for shift-invariant kernels. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 485–493. JMLR.org, 2014. URL http://proceedings.mlr.press/v32/yangb14.html.

Yu, F. X. X., Suresh, A. T., Choromanski, K. M., Holtmann-Rice, D. N., and Kumar, S. Orthogonal random features. *Advances in neural information processing systems*, 29, 2016.

Kernel, Graph, p_{term}	m=2	m=10	m=20	m=40	m=80
d=1, ER-0.4-1000, 0.1	0.0007259063790268863	0.0001903398359981377	0.00011763026820191872	8.79471573831236e-05	4.29321194947306e-05
d=1, ER-0.4-1000, 0.06	0.0005435175726472963	0.00013151409601958063	9.575658877244471e-05	4.3906087463097626e-05	4.0755077316485244e-05
d=1, ER-0.4-1000, 0.01	0.00022105611802712477	9.098043091730836e-05	7.626805728093205e-05	3.87410902555975e-05	3.7022093881032226e-05
d=1, ER-0.1-1000, 0.1	0.00616767931977833	0.004336727765953796	0.003934111721006881	0.0027504647840658337	0.0013502125948369828
d=1, ER-0.1-1000, 0.06	0.011169983712293842	0.004103948449818662	0.0021230519714178 01	0.0015286651824786544	0.00180920093098675 4
d=1, ER-0.1-1000, 0.01	0.008926804414936252	0.003593843664943383	0.002372436626864906	0.002528684013474355	0.0013236120502376739
d=1, dolphins, 0.1	0.00616767931977833	0.004336727765953796	0.003934111721006881	0.0027504647840658337	0.0013502125948369828
d=1, dolphins, 0.06	0.011169983712293842	0.004103948449818662	0.0021230519714178 01	0.0015286651824786544	0.00180920093098675 4
d=1, dolphins, 0.01	0.008926804414936252	0.003593843664943383	0.002372436626864906	0.002528684013474355	0.0013236120502376739
d=1, eurosis, 0.1	0.0060276707312881305	0.00092538589272756	0.0006342279681653932	0.0004384555835094276	0.00018536188888052218
d=1, eurosis, 0.06	0.004751929441074538	0.001020005565545354	0.000889042372679839	0.0005086950138860196	0.0002556955468331719
d=1, eurosis, 0.01	0.0037293050211431988	0.0007734429750918471	0.0006679713650390155	0.0004663147033666988	0.00029012600990115697
d=1, Networking, 0.1	0.00354911912464601	0.0008569845357120783	0.0007800162218792217	0.000486001643398867	0.0003402086731748513
d=1, Networking, 0.06	0.0043304513219250 86	0.0011314015990526772	0.0009955841652321848	0.0005356434310080785	0.0002993746309815702
d=1, Networking, 0.01	0.0035153340017524855	0.0015484554089057877	0.0009127004247725001	0.0002422748224069301 9	0.0002548717948222871
d=1, Databases, 0.1	0.004920247514564321	0.0010238037860984882	0.0006558270450890158	0.0004630486276316611 4	0.0003245028197695966
d=1, Databases, 0.06	0.004526013840391158	0.000963926605473951	0.0007692921395363883	0.0005198262247493105	0.0004310588594538151
d=1, Databases, 0.01	0.004943075493204197	0.0008015939178439096	0.00078027641060082 7	0.0003613319043567048	0.0003127005621608197
d=1, Enc&Comp, 0.1	0.0048278192814453745	0.0010825306683229289	0.0008869476479790293	0.0007716890723512931	0.0003704820179027577
d=1, Enc&Comp, 0.06	0.0020893222009769137	0.0012842314605827185	0.0004723607601931406	0.0006878263190053518	0.0003750554030785581
d=1, Enc&Comp, 0.01	0.00504474860734935	0.0013024704440231251	0.0009294029364896054	0.000660959802814907	0.00020474232102263236
d=1, Hard&Arch, 0.1	0.004611366876905434	0.0009512468516057303	0.0012081728737839681	0.0007626404640344821	0.0004050853204590919 5
d=1, Hard&Arch, 0.06	0.005063954232425586	0.001065299727136788	0.0012631414043132634	0.0006979985117773265	0.00042153280985097 3
d=1, Hard&Arch, 0.01	0.00274347868585054	0.0007186205865051595	0.0007365735158683657	0.00045503009550719795	0.0004096699343518231 6
d=2, ER-0.4-1000, 0.1	0.0005611717968776918	0.00027274809547358435	0.00011320568751584864	6.757544191642404e-05	6.039762637275414e-05
d=2, ER-0.4-1000, 0.06	0.0004256206092542753	0.0001255914552193544	8.877177278468484e-05	4.854379424147713e-05	4.386961653719971e-05
d=2, ER-0.4-1000, 0.01	0.0003048387437939227	8.612938808993734e-05	8.861321498224955e-05	3.118698564760755e-05	2.6440001535795706e-05
d=2, ER-0.1-1000, 0.1	0.0005236255151350093	0.00022767131458694158	0.00013017955748542738	9.70665903350761e-05	4.911946309935634e-05
d=2, ER-0.1-1000, 0.06	0.0005865169497721389	0.00025410090630458796	0.00019059306793781872	0.00011444739288479127	7.032284297507503e-05
d=2, ER-0.1-1000, 0.01	0.00046667795838897455	0.00027540444251076277	0.00010490895428890273	0.00013378373593132866	7.724500570322393e-05
d=2, dolphins, 0.1	0.008072537726967633	0.00571208308059399	0.005119100184170540 5	0.0017181464186888107	0.001090082709032633
d=2, dolphins, 0.06	0.009755110708905598	0.00389170462370401	0.0025002668985388667	0.0016741448929614132	0.001371113891426613
d=2, dolphins, 0.01	0.006479716073801618	0.0024690593909838824	0.0016779353904653569	0.0018280376993101016	0.0008140444788820552
d=2, eurosis, 0.1	0.002985106085538542	0.000876946183305746	0.0007680934798181345	0.0005249300647113408	0.0002331259532404271
d=2, eurosis, 0.06	0.0041183672021622205	0.0008723728098050511	0.0007700295259631496	0.0005934744828608486	0.00017965740908675555
d=2, eurosis, 0.01	0.003124420288948241	0.0012421356753991252	0.000522089061052929	0.0003670637621586290 6	0.0001912315167467761 6
d=2, Networking, 0.1	0.003894812593269389	0.0012115481023431956	0.000687373457322389	0.0003709238725270653	0.000291758017445287 1
d=2, Networking, 0.06	0.004866701669257291	0.0010178452297390271 4	0.0007199911928308271	0.000574302270555022	0.0004029005603562638 7
d=2, Networking, 0.01	0.003824906545448091	0.0014371944936820223	0.0003065956016769412	0.0003731510983670436	0.0003803384907960821
d=2, Databases, 0.1	0.004920247514564321	0.0010238037860984882	0.0006558270450890158	0.0004630486276316611 4	0.0003245028197695966
d=2, Databases, 0.06	0.004526013840391158	0.000963926605473951	0.0007692921395363883	0.0005198262247493105	0.0004310588594538151
d=2, Databases, 0.01	0.004943075493204197	0.0008015939178439096	0.00078027641060082 7	0.0003613319043567048	0.0003127005621608197
d=2, Enc&Comp, 0.1	0.006724706042286198	0.0014845981818248232	0.0010458737350497948	0.0008081389763835953	0.00035644825231180745
d=2, Enc&Comp, 0.06	0.004100807796697153	0.0011827406795993489	0.0008530016288033847	0.0007814930040936276	0.00024164229327689848
d=2, Enc&Comp, 0.01	0.002887642103947921	0.0009055525946103308	0.0006739056569936388	0.0005584100220945653	0.00042133127128199215
d=2, Hard&Arch, 0.1	0.003095039050038908	0.00126431072738253	0.000531917678481419	0.00047539738617717955	0.0005823370820758574
d=2, Hard&Arch, 0.06	0.0028895869476586425	0.0017287614112698189	0.0012669302897452082	0.0005934700975376661	0.0003548026338104648
d=2, Hard&Arch, 0.01	0.0029574141141546616	0.0014192560156932896	0.0008739945261346017	0.0006409254479486006	0.0005212426748758263

Table 3. Standard deviations for the experiments from Sec. 5.2 (for the d-regularized Laplacian kernel) for $m > 1$.

A. Appendix

A.1. Additional experimental details

In this section, we report standard deviations for the experiments conducted in Sec. 5.2. The results are presented in Table 3.

A.2. D-regularized Laplacian kernels

We show here that as long as the regularized Laplacian kernel is well-defined, the d-regularized variants for $d > 1$ are valid positive definite kernels. The following is true:

Theorem A.1. *If a matrix $\mathbf{I}_N + \sigma^2 \tilde{\mathbf{L}}_{\mathrm{G}}$ is invertible and the inverse has positive entries, then the d-regularized Laplacian kernel is a valid positive definite kernel.*

Proof. Denote $\mathbf{A} = \sigma^2 \tilde{\mathbf{L}}_{\mathrm{G}}$. Note first that, by Perron-Frobenius Theorem, we have: $\rho(\mathbf{A}) = \lambda_{\max}(\mathbf{A})$, where $\lambda_{\max}(\mathbf{A})$ stands for the largest eigenvalue of $\mathbf{A}$ (note that all eigenvalues of $\mathbf{A}$ are real since $\mathbf{A}$ is symmetric). Then for n sufficiently large we have: $\|\mathbf{A}\|^n \leq (1-\xi)^k$ for some $\xi \in (0,1)$. Thus the Neumann series: $\sum_{n=0}^{\infty} \mathbf{A}^n$ converges to: $(\mathbf{I}_N - \mathbf{A})^{-1}$. Thus we can rewrite: $(\mathbf{I}_N - \mathbf{A})^{-d} = (\sum_{n=0}^{\infty} \mathbf{A}^n)^d$. Therefore we have: $(\mathbf{I}_N - \mathbf{A})^{-d} = \sum_n c_{d,n} \mathbf{A}^n$ for some sequence of

coefficients: $(c_{d,0}, c_{d,1}, ...)$. We conclude that $(\mathbf{I}_N - \mathbf{A})^{-d}$ is symmetric for any $d = 1, 2, ...$ (since $\mathbf{A}$ is). We can rewrite: $(\mathbf{I}_N - \mathbf{A})^{-(d+1)} = \mathbf{XY}$, where: $\mathbf{X} = (\mathbf{I}_N - \mathbf{A})^{-d}$ and $\mathbf{Y} = (\mathbf{I}_N - \mathbf{A})^{-1}$. We will proceed by induction on d. We can then conclude that matrix $(\mathbf{I}_N - \mathbf{A})^{-(d+1)} = \mathbf{XY}$ is a product of two positive definite matrices. Furthermore, $(\mathbf{I}_N - \mathbf{A})^{-(d+1)}$ is symmetric, as we have already observed. But then it is a positive definite matrix. □

A.3. Concentration results for GRFs with the base sampler

We inherit the notation from the main body of the paper, in particular from the proof of Theorem 3.2. We also denote: $A(\omega) = \prod_{s=1}^{l_1} \frac{\deg(\mathrm{v_s})}{1-p_{\mathrm{term}}}$, where: $\omega = (v_1, v_2, ..., v_{l_{1+1}})$. Furthermore, for two walks: ω_1, ω_2,we use the following notation: $\omega_1 \subset_{\mathrm{pre}} \omega_2$ if ω_1 is a **strict** prefix of ω_2 and $\omega_1 \subseteq_{\mathrm{pre}} \omega_2$ if ω_1 is a prefix (not necessarily strict) of ω_2. We prove the following concentration result.

Theorem A.2. *Consider an unbiased estimation of the matrix $\mathbf{K} = (\mathbf{I}_N - \mathbf{U})^{-2}$ via $\mathbf{M} = \mathbf{B}(\mathbf{B}')^\top$ for matrices $\mathbf{B}, \mathbf{B}' \in \mathbb{R}^{N \times N}$, as in Theorem 3.2. Then the following is true for any $i, j \in \{1, ..., |\mathrm{V}(\mathrm{G}_{\mathbf{U}})|\}$, $i \neq j$:*

$$\mathrm{Var}(\mathbf{M}(i,j)) = \frac{1}{m^2}(\Lambda - \mathbf{K}^2(i,j)) \tag{23}$$

for Λ defined as follows:

$$\Lambda = \sum_{x_1 \in \mathrm{V}} \sum_{x_2 \in \mathrm{V}} \sum_{\omega_1 \in \Omega(i,x_1)} \sum_{\omega_2 \in \Omega(j,x_1)} \sum_{\omega_3 \in \Omega(i,x_2)} \sum_{\omega_4 \in \Omega(j,x_2)} w(\omega_1)w(\omega_2)w(\omega_3)w(\omega_4)\Gamma(\omega_1, \omega_2, \omega_3, \omega_4) \tag{24}$$

and where:

$$\Gamma(\omega_1, \omega_2, \omega_3, \omega_4) = \begin{cases} \frac{1}{A(\omega_2)A(\omega_4)} & \textit{if } \omega_1 \subset_{\mathrm{pre}} \omega_2 \textit{ and } \omega_3 \subset_{\mathrm{pre}} \omega_4 \\ \frac{1}{A(\omega_2)A(\omega_3)} & \textit{if } \omega_1 \subset_{\mathrm{pre}} \omega_2 \textit{ and } \omega_4 \subseteq_{\mathrm{pre}} \omega_3 \\ \frac{1}{A(\omega_1)A(\omega_4)} & \textit{if } \omega_2 \subseteq_{\mathrm{pre}} \omega_1 \textit{ and } \omega_3 \subset_{\mathrm{pre}} \omega_4 \\ \frac{1}{A(\omega_1)A(\omega_3)} & \textit{if } \omega_2 \subseteq_{\mathrm{pre}} \omega_1 \textit{ and } \omega_4 \subseteq_{\mathrm{pre}} \omega_3 \end{cases} \tag{25}$$

Proof. We have the following:

$$\mathrm{Var}(\mathbf{M}(i,j)) = \frac{1}{m^2} \sum_{k=1}^{m} \sum_{l=1}^{m} X_{k,l}, \tag{26}$$

where:

$$X_{k,l} = \sum_{x \in \mathrm{V}} \sum_{\omega_1 \in \Omega(i,x)} \sum_{\omega_2 \in \Omega(j,x)} w(\omega_1)w(\omega_2)\mathbb{1}[\omega_1 \subseteq_{\mathrm{pre}} \tilde{\Omega}(k,i)]\mathbb{1}[\omega_2 \subseteq_{\mathrm{pre}} \tilde{\Omega}(l,j)] \tag{27}$$

Note that since different random walks are chosen independently, all random variables $X_{k,l}$ are independent and therefore:

$$\mathrm{Var}(\mathbf{M}(i,j)) = \frac{1}{m^2}(\mathbb{E}[X_{1,1}^2] - (\mathbb{E}[X_{1,1}])^2) \tag{28}$$

Thus, from the unbiasedness of the estimator, we obtain:

$$\mathrm{Var}(\mathbf{M}(i,j)) = \frac{1}{m^2}(\mathbb{E}[X_{1,1}^2] - \mathbf{K}^2(i,j)) \tag{29}$$

It suffices to prove that: $\mathbb{E}[X_{1,1}^2] = \Lambda$. Note that we have:

$$\begin{aligned} \mathbb{E}[X_{1,1}^2] = \mathbb{E}[\sum_{x_1 \in \mathrm{V}} \sum_{x_2 \in \mathrm{V}} \sum_{\omega_1 \in \Omega(i,x_1)} \sum_{\omega_2 \in \Omega(j,x_1)} \sum_{\omega_3 \in \Omega(i,x_2)} \sum_{\omega_4 \in \Omega(j,x_2)} & w(\omega_1)w(\omega_2)w(\omega_3)w(\omega_4) \\ & \mathbb{1}[\omega_1 \subseteq_{\mathrm{pre}} \tilde{\Omega}(1,i)]\mathbb{1}[\omega_2 \subseteq_{\mathrm{pre}} \tilde{\Omega}(1,j)] \\ & \mathbb{1}[\omega_3 \subseteq_{\mathrm{pre}} \tilde{\Omega}(1,i)]\mathbb{1}[\omega_4 \subseteq_{\mathrm{pre}} \tilde{\Omega}(1,j)]] \end{aligned} \tag{30}$$

Thus we have:

$$\begin{aligned} \mathbb{E}[X_{1,1}^2] = \sum_{x_1 \in \mathrm{V}} \sum_{x_2 \in \mathrm{V}} \sum_{\omega_1 \in \Omega(i,x_1)} \sum_{\omega_2 \in \Omega(j,x_1)} \sum_{\omega_3 \in \Omega(i,x_2)} \sum_{\omega_4 \in \Omega(j,x_2)} & w(\omega_1)w(\omega_2)w(\omega_3)w(\omega_4) \\ & \mathbb{P}[\mathbb{1}[\omega_1 \subseteq_{\mathrm{pre}} \tilde{\Omega}(1,i)]\mathbb{1}[\omega_2 \subseteq_{\mathrm{pre}} \tilde{\Omega}(1,j)] \\ & \mathbb{1}[\omega_3 \subseteq_{\mathrm{pre}} \tilde{\Omega}(1,i)]\mathbb{1}[\omega_4 \subseteq_{\mathrm{pre}} \tilde{\Omega}(1,j)]] \end{aligned} \tag{31}$$

We conclude the proof, observing that:

$$\Gamma(\omega_1, \omega_2, \omega_3, \omega_4) = \mathbb{P}\left[\mathbb{1}[\omega_1 \subseteq_{\text{pre}} \tilde{\Omega}(1,i)]\mathbb{1}[\omega_2 \subseteq_{\text{pre}} \tilde{\Omega}(1,j)]\mathbb{1}[\omega_3 \subseteq_{\text{pre}} \tilde{\Omega}(1,i)]\mathbb{1}[\omega_4 \subseteq_{\text{pre}} \tilde{\Omega}(1,j)]\right] \tag{32}$$

□

Completely analogous analysis can be conducted for $i = j$, but the formula is more convoluted since certain pairs of walks sampled from i and j are clearly no longer independent as being exactly the same (namely: the kth sampled walk from i and the kth sampled walk from j for $k = 1, 2, ..., m$).

A.4. Additional derivations leading to Equation 18

For Reader's convenience, we will here explain in more detail the derivations leading to Eq. 18. For the simplicity, we will assume that $m = 1$ since for larger m the formula is obtained simply by averaging over all pairs of random walks from node i and j.

Note first that directly from the definition of GRFs and Algorithm 1, we get:

$$\phi(i)^\top \phi(j) = \sum_{x \in \mathrm{V}(G_{\mathbf{U}})} \left(\sum_{r \in \mathcal{K}_x(\bar{\Omega}(1,i))} \mathrm{load}^{\bar{\Omega}(1,i)}(x, r) \right) \left(\sum_{v \in \mathcal{K}_x(\bar{\Omega}(1,j))} \mathrm{load}^{\bar{\Omega}(1,j)}(x, v) \right), \tag{33}$$

where $\mathcal{K}_x(\omega)$ for a given walk ω returns the set of these time indices when ω hits x (first vertex of the walk gets time index zero, next one, time index one, etc.) and furthermore $\mathrm{load}^{\omega}(x, c)$ returns the increment of the load in x added to the current load in time c (see: Algorithm 1, while-loop, point 5). Now note that each r can be identified with its corresponding prefix-subwalk ω_1 and each v can be identified with its corresponding prefix-subwalk ω_2 Finally, observe that the increment of the load that ω_1 contributes to is precisely: $w(\omega_1) \prod_{s=1}^{l_1} \frac{(p^i_{1,s})^{-1}}{1-p_{\text{term}}}$ and the increment of the load that ω_2 contributes to is precisely: $w(\omega_2) \prod_{t=1}^{l_2} \frac{(p^j_{1,t})^{-1}}{1-p_{\text{term}}}$ (see: Algorithm 1, while-loop, point 4). That gives us Eq. 18.