## A Random Walk Descriptor Generator

In this appendix, we give the definition of the random walk (RW) descriptor generator, and show that its iterative updating process will eventually converge to a stationary distribution, which makes the generated descriptor irrelevant to the initial labeling of a node and its neighborhood.

We begin with the definition of the RW descriptor generator. We will work on undirected connected graphs. Let the alphabet of the categorical attributes $\Sigma = \{l_1, \ldots, l_L\}$, $\mathcal{L}_c$ be the labeling function, then the RW descriptor of a node $v$, $f_{rw}(v) = (A_1(v), \ldots, A_L(v))$, is a vector of length $L$, with the $i$th component $A_i(v)$ associated with the label $l_i$. Let $\lambda \in [0, 1]$ be a scalar, $d(u)$ be the degree of node $u$. $f_{rw}^{(r)}(v)$ are recursively defined in the following way:

(1) Initialization:

$$A_i^{(0)}(v) = \begin{cases} 1 & \text{if } \mathcal{L}_c(v) = l_i, \\ 0 & \text{otherwise} \end{cases}$$

(2) Updating:

$$A_i^{(r+1)}(v) = A_i^{(r)}(v) \times (1-\lambda) - \sum_{u \in \mathcal{N}(v)} A_i^{(r)}(v) \times \frac{\lambda}{d(u)}$$

$$for\ i = 1, \ldots, L, \forall r \in \mathbb{N}$$

For some label $l_i$, initially, $A_i^{(0)}(v)$ is set to be 1 if $l_i$ is $v$'s categorical label, and otherwise 0. In each iteration, each node evenly distributes a $\lambda$ fraction of its value to all of its neighbors and obtains some value from its neighbors in the same way. Therefore, the total value $\sum_{v \in G} A_i(v)$ will not change during the iterative process. Let $C$ be the adjacency matrix of graph $G$, $D = diag(d(v_1), \ldots, d(v_n))$ the diagonal matrix of the degrees of the nodes in $G$, and $M = \lambda I + (1-\lambda)D^{-1}C$, where $I$ is the identity matrix. The following lemma directly follows from the definition.

LEMMA A.1. *Let $A^{(r)}$ be a $L$-by-$n$ matrix constructed by stacking $f_{rw}^{(r)}(v_i), 1 \leq i \leq n$ as columns, then $A^{(r+1)} = A^{(r)}M$.*

It is easy to see that each column of $M$ sums to 1, which implies that $M$ is the transition matrix of a random walk on the graph. Let's first focus on a single label $l_i$ and a graph $G$ on which only one node $v$ is labeled $l_i$. Then the above iterative process corresponds to a random walk on the graph starting at $v$ with transition matrix $M$. Let $S = (\frac{d(v_1)}{2m}, \ldots, \frac{d(v_n)}{2m})$, where $n$ and $m$ are the number of nodes and edges in the graph. We show that $S$ is a stationary distribution of the random walk in the following theorem.

THEOREM A.1. *$S$ is a stationary distribution of the random walk defined by the transition matrix $M$, namely $S = SM$.*

*Proof.*

$$\begin{aligned} SM &= S(\lambda I + (1-\lambda)D^{-1}C) \\ &= \lambda S + (1-\lambda)(SD^{-1})C \\ &= \lambda S + (1-\lambda)\frac{1}{2m}(1, 1, \ldots, 1)C \\ &= \lambda S + (1-\lambda)S \\ &= S \end{aligned}$$

If this random walk is irreducible, which is the case when $\lambda < 1$, the stationary distribution $S$ is also unique [3]. Therefore, regardless of the starting node, the iterative updating process of the RW descriptor generator on $G$ will always converge to $S$. If there are $n_i > 1$ nodes labeled $l_i$ in $G$, the iterative updating process will correspond to $n_i$ independent random walks starting at each of those nodes, and it will simply converge to $n_i S$. This clearly shows the weakness of the RW descriptor generator: two nodes in the same graph with same degree will eventually end up with the same descriptor, regardless of whether they have the same label, and regardless of the probably different labeling of their neighborhood. Therefore, the RW descriptor generator doesn't have the descriptor property.

Finally, when $\lambda = 1$, the above iterative updating process may not converge. A simple example is a graph with only two nodes $v_1$ and $v_2$, and $\mathcal{L}_c(v_1) = l_1, \mathcal{L}_c(v_2) = l_2$, then $f_{rw}(v_1)$ will oscillate between $(0, 1)$ and $(1, 0)$, which is obviously not a good descriptor for $v_1$.

## B VG Kernel Example

We illustrate in Figure 1 how the VG pyramid matching works with $t = 3$ and $b = 2$. The graph in the top shows the hierarchical clustering structure, while the rest illustrates the construction and matching of the multi-resolution histograms for two graphs. (1) For the hierarchical clustering, the level 0 cluster contains all descriptors (points) in $\mathcal{F}(\mathcal{G})$, and the solid line indicates the boundary of the two level 1 clusters, while the dashed lines indicate the boundaries of level 2 clusters. (2) The multi-resolution histograms are constructed as in the bottom. Take $G_1$ for example, the level 0 histogram $H_0(G_1) = 11$ as there is only one bin and there are 11 descriptors of $G_1$ in the level 0 cluster. The level 1 histogram $H_1(G_1) = [5, 6]$ as there are 5 descriptors in the first level 1 cluster and 6 in the second. The level 2 histogram is set similarly. (3) Matching. Take the first bin of the level 1 histogram and its two child

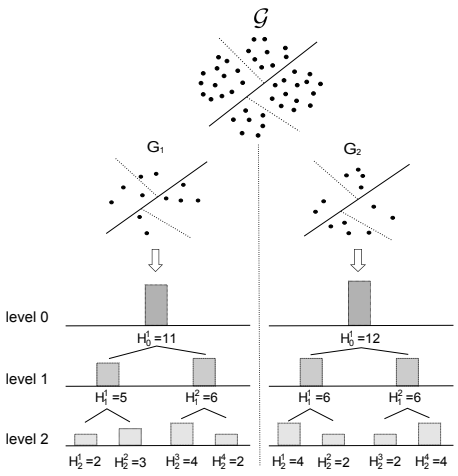


Figure 1: Example for the VG pyramid matching kernel.

bins as example. $\mathcal{J}_2^1(G_1, G_2) = \mathcal{I}_2^1(G_1, G_2) = min(H_2^1(G_1), H_2^1(G_2)) = 2$, $\mathcal{J}_2^2(G_1, G_2) = \mathcal{I}_2^2(G_1, G_2) = min(H_2^2(G_1), H_2^2(G_2)) = 2$, and $\mathcal{I}_1^1(G_1, G_2) = min(H_1^1(G_1), H_1^1(G_2)) = 5$, therefore $\mathcal{J}_1^1(G_1, G_2) = \mathcal{I}_1^1(G_1, G_2) - \mathcal{J}_2^1(G_1, G_2) - \mathcal{J}_2^2(G_1, G_2) = 1$. In other words, there are 2 descriptors matched in each of the first two bins in the level 2 histogram, and one more match is found in the first bin of the level 1 histogram.

## C Setup for Experiments on Real-world Datasets

Here we give the detailed experiment setup. SP, WLSP, CSM and GH require additional kernels to evaluate attributes. Categorical attributes are evaluated by the Dirac kernel for all the graph kernels. Partial charge and SSE length are evaluated by the Brownian bridge kernel (see [1]). The parameter $c$ of the Brownian bridge kernel for partial charge is selected from $\{0.5, 1, 1.5, 2\}$, and that for the SSE length is selected from $\{1, 3, 5, 7\}$. For SP and WLSP, a triangular kernel whose $c$ is chosen from $\{1, 3, 5, 7, 9, 11\}$ is employed to compare 3D distance between atoms. For CSM, the 3D distance is also evaluated by a triangular kernel, but $c$ is fixed to 3, as suggested by the author. For GH, node numerical attributes are all evaluated by a Gaussian kernel with $\lambda$ selected from $\{10^{-4}, 10^{-3}, \ldots, 10^{4}\}$ (see [2]). For PK, we use the label diffusion process as propagation scheme. The LSH metrics are $m_l = tv$ and $m_a = l_1$. The LSH bin-width for categorical attributes $w_l$ is selected from $\{10^{-5}, 10^{-4}, 10^{-3}\}$, and that for numerical attributes, $w_a$, is fixed to 1. Both categorical and numerical attributes are propagated (see [4]).

According to the suggestion from the original papers, we choose $h$ from $\{0, ..., 10\}$ for WL, from $\{0, ..., 3\}$ for WLSP. The maximum subgraph size $k$ for the CSM kernel is selected from $\{1, \ldots, 7\}$. For PK and DM, $h$ is selected from $\{0, ..., 15\}$. The decay factor $\eta$ is selected from $\{1/2, 1/3\}$. For the VG kernel, $t$ is selected from $\{4, 5\}$, and $b$ is selected from $\{10, 20\}$.

In order to adapt DM to handle edge numerical attributes, we first generate node descriptors, and compute the descriptor of an edge as the average of the descriptors of its endpoints. Edge numerical attributes are then appended. In this way, a graph is converted into two sets of descriptors, node descriptors $\mathcal{F}_v$ and edge descriptors $\mathcal{F}_e$, and the DM kernel is then defined as

$$k_{dm}(G_1, G_2) = k(\mathcal{F}_v(G_1), \mathcal{F}_v(G_2)) \times k(\mathcal{F}_e(G_1), \mathcal{F}_e(G_2)),$$

which is still a valid kernel since it is a product of valid kernels.

Finally, all kernel matrices are normalized before classification. The parameter $C$ for SVM is selected from $\{10^{-7}, 10^{-5}, \ldots, 10^{7}\}/N$, where $N$ is the number of graphs.

## References


[1] K. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl 1):i47–i56, 2005.

[2] A. Feragen, N. Kasenburg, J. Petersen, M. de Bruijne, and K. Borgwardt. Scalable kernels for graphs with continuous attributes. In *NIPS*, 2013.

[3] L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, 2(1):1–46, 1993.

[4] M. Neumann, R. Garnett, C. Bauckhage, and K. Kersting. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning*, pages 1–37, 2015.