# Efficient Graph Kernels by Randomization

Marion Neumann,[1] Novi Patricia,[1] Roman Garnett,[2] Kristian Kersting[1]

[1] Knowledge Discovery Department, Fraunhofer IAIS,
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
{firstname.lastname@iais.fraunhofer.de}
[2] Robotics Institute, Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, PA 15213, United States
{rgarnett@cs.cmu.edu}

**Abstract.** Learning from complex data is becoming increasingly important, and graph kernels have recently evolved into a rapidly developing branch of learning on structured data. However, previously proposed kernels rely on having discrete node label information. In this paper, we explore the power of continuous node-level features for propagation-based graph kernels. Specifically, propagation kernels exploit node label distributions from propagation schemes like label propagation, which naturally enables the construction of graph kernels for partially labeled graphs. In order to efficiently extract graph features from continuous node label distributions, and in general from continuous vector-valued node attributes, we utilize randomized techniques, which easily allow for deriving similarity measures based on propagated information. We show that propagation kernels utilizing locality-sensitive hashing reduce the runtime of existing graph kernels by several orders of magnitude. We evaluate the performance of various propagation kernels on real-world bioinformatics and image benchmark datasets.

## 1 Introduction

For attribute-valued data, sophisticated kernel approaches for classification and regression have been widely and successfully studied. Nowadays, however, the bulk of information, such as available on the world wide web, is complex and highly structured. Structured data is commonly represented by graphs, which capture relations among entities, but also naturally model the structure of whole objects. Real-world examples are proteins or molecules in bioinformatics, image scenes in computer vision, text documents in natural language processing, and object and scene models in robotics, to name but a few. Learning in such domains and in turn developing meaningful kernels to take the structure of these data into account is becoming more and more important.

In addition to the structural properties of data entities, we often have access to vast quantities of additional, possibly continuous related information, for instance meta-data for images or text documents. Incorporating such information consistently is difficult and incomplete data and missing information constitute major challenges for learning. Unfortunately, existing graph kernels [4,6,10,18,19]

rely on having discrete node labels and, besides, can only handle graphs with full node label information in a principled manner. In this paper, we propose the family of *propagation kernels* which leverage the power of continuous label distributions.

Triggered by previously introduced kernels on probabilistic models [8, 21], propagation kernels exploit distributions from propagation schemes like label propagation (LP) and enhance existing graph kernel frameworks to handle continuous, vector-valued node attributes. In particular, we define kernel inputs, i.e. graph features, that are counts of similar node label distributions on the respective graphs. This generalization additionally enables us to define graph kernels for partially labeled graphs in a natural way. Unfortunately, comparing all distributions of node labels among all graphs in the database scales as $\mathcal{O}(n^2)$, where $n$ is the total number of nodes, and aggregating all distributions on node labels in one graph to a single vector leads to significant information loss. Hence, in order to efficiently determine the similarity among node label distributions and in general to be able to deal with continuous, vector-valued node attributes, we leverage randomization techniques from the theoretical computer science community. We define locality-sensitive hash (LSH) functions to create distance-preserving signatures for each node label distribution. These functions provide a randomized algorithm that allow us to efficiently compute the distribution-based count features for our kernels in $\mathcal{O}(n)$. We are able to show that the hash values can preserve both the total variation and the Hellinger metrics. Therefore, our LSH enables us to efficiently retrieve similar distributions based on these distance measures for probability distributions.

To summarize, the main contribution of our work is the introduction of a family of fast graph kernels based on propagating node labels. Specifically, we introduce locality-sensitive hash functions to efficiently compute the count features based on the similarity of node label distributions. Furthermore, we show that applying randomized techniques reduces the running time of existing graph kernels, namely kernels based on the Weisfeiler–Lehman test of isomorphism [18], by several orders of magnitude and we propose propagation kernels utilizing locality-sensitive hashing that are even more efficient.

We proceed as follows. We start off by touching upon related work. After introducing the family of propagation kernels and giving several examples thereof, we will describe locality-sensitive hashing for handling vector-valued node label distributions. Before concluding, we present experimental results for graph classification tasks on state-of-the-art image datasets, and commonly used bioinformatics benchmark datasets.

## 2 Related Work

Propagation kernels are related to three lines of research. First of all, they are deeply connected to several graph kernels developed within the graph mining community. Graph kernels can be categorized mainly into four classes: graph kernels based on walks [4, 10, 22] and paths [2], graph kernels based on limited-

size subgraphs [7,19], graph kernels based on subtree patterns [13,16], and graph kernels based on structure propagation [18]. Whereas the efficient kernel computation such as [22] are able to compare unlabeled graphs efficiently, Shervashidze *et al.* [19] specifically consider efficient comparisons of large, labeled graphs. The Weisfeiler–Lehman (WL) subtree kernel, one instance of the family of WL-kernels introduced, essentially computes count features for each graph based on the signatures arising from iterative multi-set label determination and compression steps. In every kernel iteration, these features are then the inputs to a base kernel and the WL-kernel is the sum of those base kernels over the iterations. The challenge of comparing large, partially labeled graphs—as considered by the propagation kernels introduced in the present paper—remains to a large extent unsolved. One could mark unlabeled nodes with a unique symbol and propagate this symbol using the Weisfeiler–Lehman kernels [18]. However, this neglects any label proportion information due to the diffusion process of labels on the graph. Likewise, one could just propagate labels across the graph and then run the WL-kernel. This, however, is also likely to neglect label proportion information. Indeed, after label propagation converges, we may ignore the label proportions of a node. Before convergence, however, we shall be concerned with information encoded in intermediate label proportions at nodes. Moreover, a two-stage approach may run many unnecessary label propagation iterations.

Second, propagation kernels are deeply connected to several recent lifted message-passing approaches [1, 11, 15, 20] to probabilistic inference. They have rendered many of these large, previously intractable problems quickly solvable by exploiting the induced redundancies. Specifically, they automatically group nodes and potentials of the graphical model into supernodes and superpotentials if they have identical computation trees (i.e., the tree-structured "unrolling" of the graphical model computations rooted at the nodes). Then, they run modified message-passing approaches on this lifted (compressed) network. It is actually easy to see that the color-passing approach in [11] for computing the lifted network is as a form of the 1-dimensional Weisfeiler–Lehman algorithm as also employed by the WL-kernels [18]. However, there is a subtle difference. For lifted inference, symmetries among random variables easily break when variables become correlated by virtue of sharing asymmetrically observed evidence, that is labels of nodes are observed. Consequently, color-passing provides a lifted model that is not far from propositionalized, therefore canceling the benefits of lifted inference. For graph kernels, this is exactly what we want. The correlations help us to distinguish different graphs. This insight was the seed that grew into the idea of propagation kernels.

Finally, propagation kernels make another contact point, namely between WL-kernels and kernels that accommodate probability distributions [8, 9, 12, 14]. However, whereas the latter ones essentially build kernels based on the outcome of probabilistic inference after convergence, propagation kernels intuitively count common sub-distributions induced after each iteration of running inference in two graphs. In doing so, they are able to take structure information into account.

## 3 Propagation Kernels

In the following, we introduce the general family of *propagation kernels* and present several instances based on propagating label information. The main insight here is, that the intermediate node label distributions, e.g., the iterative distribution updates of a label propagation scheme, capture label *and* structure information of the graphs. Hence, we design kernel inputs, i.e. graph features, based on the counts of similar distributions among the respective graphs' nodes. Further, we show that propagation kernels generalize existing structure propagating WL-kernels, especially the WL-subtree kernel [18].

### 3.1 General Definition

Here we will define a similarity measure $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ among graph instances $G^{(i)} \in \mathcal{X}$, in particular, let $K$ be a positive semidefinite covariance function. Let $G_t^{(i)} = (V^{(i)}, E^{(i)}, L_t^{(i)})$ with $t = [0, .., T]$ be a sequence of graphs in $\mathcal{X}$, where $V^{(i)}$ is the set of nodes and $E^{(i)}$ is the set of edges. Further, each node in $V^{(i)}$ is endowed with one of $k$ true labels and each graph has $n_i$ nodes. $L_t^{(i)} \in \mathbb{R}^{n_i \times k}$ represents the label distributions[3] for all nodes in $V^{(i)}$ which are iteratively updated. Note that we do not assume that the node labels have to be given for all nodes. Propagation kernels can naturally be computed for partially labeled graphs as the features are only built upon the node label distributions, which can be initialized uniformly for unknown node labels. Observed node labels are represented by a trivial delta distribution.

Propagation kernels are defined by applying the following iterative procedure $T + 1$ times, beginning with an initial set of graphs $\{G_0^{(i)}\}$ with label distributions initialized as above.

**Step 1: count common node label distributions.** First, we generate feature vectors $\phi(G_t^{(i)})$ for each graph by counting common label distributions induced over the nodes among the respective graphs. Therefore, each node in each graph is placed into one of a number of "bins," each one collecting similar label distributions, and these vectors count the nodes in each bin for each graph. The exact details of this procedure are given below, in Section 3.2 and Section 4.

**Step 2: calculate current kernel contribution.** Given these vectors, for each pair of graphs $G^{(i)}$ and $G^{(j)}$, we calculate

$$k(G_t^{(i)}, G_t^{(j)}) = \langle \phi(G_t^{(i)}), \phi(G_t^{(j)}) \rangle, \tag{1}$$

where $\langle \cdot, \cdot \rangle$ is an arbitrary base kernel. This value will be an additive contribution to the final kernel value between these graphs.

**Step 3: propagate node label distributions.** Finally, we apply an iterative update scheme for the node label distributions

$$L_t^{(i)} \to L_{t+1}^{(i)}, \tag{2}$$

[3] Note that $L^{(i)}$ could also involve continuous, vector-valued node attributes, however, in this paper we focus on label distributions.

**Algorithm 1** The general propagation kernel computation.

**given** iterations $T$, initial label distributions $L_0^{(i)}$, base kernel $k(\cdot,\cdot)$
$K \leftarrow 0$
**for** $t \leftarrow 0 \dots T$ **do**
  **for all** graphs $G^{(i)}$ **do**
    $\phi(G_t^{(i)}) \leftarrow 0$
    **for** $j \leftarrow 1 \dots n_i$ **do**
      $\phi(G_t^{(i)}) \leftarrow \phi(G_t^{(i)}) + f(\ell_{t,j}^{(i)})$ ▷ count node label distributions, Eq. (4)
    **end for**
    $L_t^{(i)} \rightarrow L_{t+1}^{(i)}$ ▷ update label distribution, Eq. (2)
  **end for**
  $K \leftarrow K + k(\Phi, \Phi)$ ▷ $\Phi$ is $N \times k''$ matrix of $\phi$ vectors
**end for**

e.g. label propagation. These new label distributions replace those in the current set of graphs, and we continue with Step 1. The exact choice for this update results in different propagation kernels; examples are provided in Section 3.3.

Finally, the $T$-iteration propagation kernel between two graphs $G^{(i)}$ and $G^{(j)}$ is defined as

$$K_T(G^{(i)}, G^{(j)}) = \sum_{t=0}^{T} k(G_t^{(i)}, G_t^{(j)}). \tag{3}$$

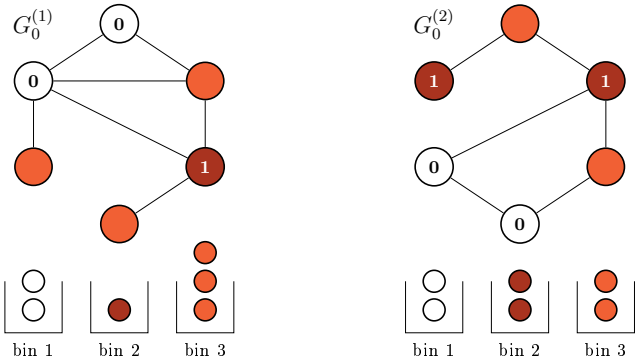
The propagation kernel computation is summarized in Algorithm 1 and an illustrative example for $t = 0$ and $t = 1$ for two graphs is shown in Figure 1.

### 3.2 Distribution-based Graph Features

The main ingredient of propagation kernels is the way distribution-based graph features are generated. Let $\ell_{t,j}^{(i)}$ be the $j$-th row of $L_t^{(i)}$ and $\mathcal{L} = \bigcup_i^N \bigcup_j^{n_i} \{\ell_{t,j}^{(i)}\}$ be the set of all uniquely occurring label distributions on the nodes of all graphs. The family of propagation kernels is characterized by generating graph features by counting node-level features on that graph. This will be captured by a function $f$ mapping from the space of distributions $\mathbb{R}^k$ into the space of standard basis vectors $E_{k'} = \{e_1, ..., e_{k'}\}$ with $k' = |\mathcal{L}| \leq n$, where $n$ is the number of nodes for all graphs $n = \sum_{i=1}^{N} n_i$. We now define

$$\phi(G_t^{(i)}) = \sum_{j=1}^{n_i} f(\ell_{t,j}^{(i)}). \tag{4}$$

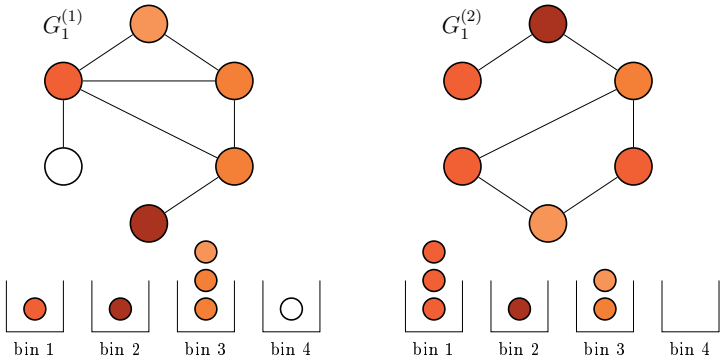
As the node label distributions $\ell_{t,j}^{(i)}$ are $k-$dimensional *continuous* vectors the cardinality of $\mathcal{L}$ might in fact be equal to the total number of nodes $n$ in the whole graph database. This, however, means that our derived features are not meaningful as, in this case, we never get the same count feature for any two




$\phi(G_0^{(1)}) = [2,\ 1,\ 3]^\top$ $\phi(G_0^{(2)}) = [2,\ 2,\ 2]^\top$

$\langle \phi(G_0^{(1)}), \phi(G_0^{(2)}) \rangle = 12$

(a) Initial label distributions and base kernel value for $t = 0$




$\phi(G_1^{(1)}) = [1,\ 1,\ 3,\ 1]^\top$ $\phi(G_1^{(2)}) = [3,\ 1,\ 2,\ 0]^\top$

$\langle \phi(G_1^{(1)}), \phi(G_1^{(2)}) \rangle = 10$

(b) Updated label distributions and base kernel value for $t = 1$

Fig. 1: **Propagation Kernel** Propagation kernel computations for two graphs $G_t^{(1)}$ and $G_t^{(2)}$ with binary node labels using one iteration of label propagation, Eq.(6), as distribution update. Node label distributions are decoded by color, white means $\ell_{0,j}^{(i)} = [1, 0]$ and dark red stands for $\ell_{0,j}^{(i)} = [0, 1]$, the initial distributions for unlabeled nodes (light red) are $\ell_{0,j}^{(i)} = [1/2, 1/2]$. Panel (a) shows the initial distributions, bins, and respective kernel computation and panel (b) depicts distributions, bins, features and linear base kernel for $t = 1$.

similarly distributed nodes and the kernel value for any two graphs as defined in Eq. (1) is always zero. To ensure the acquisition of meaningful features we

leverage *quantization* [5]. Hence, the mapping $f$ is replaced by $q\colon \mathbb{R}^k \to E_{k'}$, where $q$ is a quantization function such that $k' \ll |\mathcal{L}| \leq n$. Note, that deriving a quantization function for distributions involves considering distance metrics for distributions such as Hellinger or total variation (TV) distance. We are not defining the quantization function here but instead give an efficient solution by locality-sensitive hashing [3] in Section 4 and also show that we can construct quantization functions which are Hellinger and TV distance preserving.

### 3.3 Instances of Propagation Kernels

So far, we defined the general family of propagation kernels. Specific choices of label update schemes, cf. Eq. (2), result in different instances of the propagation kernel family. In particular, we introduce the *diffusion graph kernel*, the *label propagation kernel*, the *belief propagation kernel*, and *structure propagation kernels* as for instance the WL-subtree kernel.

**Diffusion Graph Kernel:** For the diffusion graph kernels we use the following update for the node label distributions $L_t^{(i)} \to L_{t+1}^{(i)}$. Given the adjacency matrix $A^{(i)}$ of graph $G^{(i)}$ label diffusion on each node is defined as

$$L_{t+1}^{(i)} \leftarrow T^{(i)}\, L_t^{(i)}, \tag{5}$$

where $T^{(i)}$ is the transition matrix, i.e., the row-normalized adjacency matrix $T^{(i)} = (D^{(i)})^{-1} A^{(i)}$, where $D^{(i)}$ is the diagonal degree matrix with $D_{aa}^{(i)} = \sum_b A_{ab}^{(i)}$.

**Label Propagation Kernel:** The label distribution update for the label propagation kernel differs in the fact, that before each iteration of label diffusion the labels of the originally labeled nodes are *pushed back* [25]. Let $L_0^{(i)} = \left[L_{0,[labeled]}^{(i)}, L_{0,[unlabeled]}^{(i)}\right]^\top$ be the original labels of graph $G^{(i)}$, where the distributions in $L_{0,[labeled]}^{(i)}$ represent hard labels and $L_{0,[unlabeled]}^{(i)}$ are initialized by a uniform label distribution, i.e., each entry is $1/k$. Then the label propagation is defined by

$$\begin{aligned} L_{t,[labeled]}^{(i)} &\leftarrow L_{0,[labeled]}^{(i)}, \\ L_{t+1}^{(i)} &\leftarrow T^{(i)}\, L_t^{(i)}. \end{aligned} \tag{6}$$

Note, that we can choose other step sizes for the label propagation update scheme as one. This means that we run several iterations of label propagation for each distribution update. This can be beneficial for settings with partially labeled graphs. Further, other update schemes, such as "label spreading" [24], can be used in a similar manner resulting in a *label spreading kernel*.

**Belief Propagation Kernel:** Triggered by the idea of defining the similarity of graphical models, like conditional random field (CRF) representations of images or different groundings of a Markov logic network (MLN) [17], we can also

use belief propagation [23] to get the node-level feature update in Eq. (2). To define the belief propagation kernel, we simply use marginal probabilities instead of label distributions. Due to space limitations, we do not provide any more details here and leave comprehensive derivations and experiments for future work.

**Structure Propagation Kernel — The WL-subtree Kernel:** The WL graph kernels for labeled graphs are currently state-of-the-art considering both prediction performance and runtime [18]. Therefore, we briefly introduce one instance, the WL-subtree kernel, and give its definition within our proposed framework of propagation kernels. Given two graphs $G^{(i)}$ and $G^{(j)}$, the subtree pattern kernel counts all pairs of matching substructures in subtrees rooted at all nodes of $G^{(i)}$ and $G^{(j)}$ respectively. The runtime complexity of this approach for $N$ graphs is $\mathcal{O}(n^2T4^d)$ [16], where $d$ is the maximum node degree in all graphs. The idea of using the 1-dimensional Weisfeiler–Lehman test of isomorphism is to overcome the poor ability to scale to large, labeled graphs, as it scales linearly on the number of edges of the graphs and the length of the considered graph sequence. Hence, the WL-subtree kernel on $N$ graphs with $T$ iterations can be computed in $\mathcal{O}(Tm + NTn)$ [18], where $m$ is the total number of edges in all graphs and $n$ is the total number of nodes. Given two graphs $G^{(i)}$ and $G^{(j)}$, the algorithm works as follows. First, a signature is generated for each node in each graph by concatenating its label with a sorted multiset of its neighboring nodes. Then each node is assigned a new label such that nodes with the same signature are labeled the same. This means a hard label update in Eq. (2). Indeed, the WL-subtree kernel can be defined analogously to Eq. (3). The sequence of graphs $\{G_t^{(i)}\}$ is given by hard labels reflecting the signatures for the respective subtree pattern and the features are represented by

$$\phi_{\text{WL}}(G_t^{(i)}) = \sum_{j=1}^{n_i} g(\ell_{t,j}^{(i)}), \tag{7}$$

where $\ell_{t,j}^{(i)} \in \{e_1, ..., e_{k''}\}$ with $k'' \leq n$ being the number of different subtree patterns and $g\colon i \mapsto e_i$. That means, $\ell_{t,j}^{(i)}$ represents a hard signature for every node $j$ and $g$ maps each signature to one standard basis vector. Indeed, this view opens up the possibility to handle probabilistic subtree patterns in the setting of WL-kernels for partially labeled graphs, which, in turn, is a compelling direction to enhance the power of WL-kernels.

## 4 Locality-Sensitive Hashing for Propagation Kernels

We now describe our quantization approach for implementing propagation kernels on graphs with node label distributions. We take our inspiration from locality-sensitive hashing [3], which seeks for quantization functions on metric spaces where points "close enough" to each other in that space are "probably" assigned to the same bin. In our case, we will consider each node label vector as being an element of the space of discrete probability distributions on $k$ items equipped with an appropriate probability metric.

We will begin with a formal definition. Let $\mathcal{X}$ be a metric space with metric $d\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and let $\mathcal{Y} = \{1, 2, \ldots, k'\}$. Let $\theta > 0$ be a threshold, $c > 1$ be an approximation factor, and $p_1, p_2 \in (0, 1)$ be the given success probabilities. A set of functions $\mathcal{H}$ from $\mathcal{X}$ to $\mathcal{Y}$ is called a $(\theta, c\theta, p_1, p_2)$-*locality sensitive hash* (LSH) if for any function $h \in \mathcal{H}$ chosen uniformly at random, and for any two points $x, x' \in \mathcal{X}$, we have that

- if $d(x, x') < \theta$, then $\Pr(h(x) = h(x')) > p_1$, and
- if $d(x, x') > c\theta$, then $\Pr(h(x) = h(x')) < p_2$.

It is known that we can construct LSH families for $L^p$ spaces with $p \in (0, 2]$ [3]. Let $V$ be a real-valued random variable. $V$ is called $p$-stable if for any $\{x_1, x_2, \ldots, x_d\}$, $x_i \in \mathbb{R}$ and independently sampled $v_1, v_2, \ldots, v_d$, we have $\sum x_i v_i \sim \|\mathbf{x}\|_p V$. Explicit $p$-stable distributions are known for some $p$; for example, the standard Cauchy distribution is 1-stable, and the standard normal distribution is 2-stable. Given the ability to sample from a $p$-stable distribution $V$, we may define a LSH $\mathcal{H}$ on $\mathbb{R}^d$ with the $L^p$ metric [3]. An element $h$ of $\mathcal{H}$ is specified by three parameters: a width $w \in \mathbb{R}^+$, a $d$-dimensional vector $\mathbf{v}$ whose entries are independent samples of $V$, and $b \in [0, w]$ drawn from $\mathcal{U}[0, w]$, and defined as

$$h(\mathbf{x}; w, \mathbf{v}, b) = \left\lfloor \frac{\mathbf{v}^\top \mathbf{x} + b}{w} \right\rfloor. \tag{8}$$

We may now consider $h(\cdot)$ to be a function mapping our label distributions to integer-valued bins, where similar distributions end up in the same bin. If we number the non-empty integer bins occupied by all the nodes in all graphs from 1 to $k''$, then we may define the function $f$ in Eq. (4) by $f(\cdot) = u \circ h(\cdot)$, where $u : \mathbb{N} \to E_{k''}$. To decrease the probability of collision it is common to choose more than one random vector $\mathbf{v}$. For propagation kernels, however, we only use one hyperplane, as we effectively have $T$ hyperplanes for the whole kernel computation and the probability of a hash conflict is reduced over the iterations.

The intuition behind the expression in Eq. (8) is that $p$-stability implies that two vectors that are close under the $L^p$ norm will be close after taking the dot product with $\mathbf{v}$; specifically, $(\mathbf{v}^\top \mathbf{x} - \mathbf{v}^\top \mathbf{x}')$ is distributed as $\|\mathbf{x} - \mathbf{x}'\|_p V$. In our applications, we are concerned with the space of discrete probability distributions on $k$ elements, endowed with a probability metric $d$. Here we specifically consider the *total variation* (TV) and *Hellinger* (H) distances:

$$d_{\mathrm{TV}}(p, q) = {}^1\!/_2 \sum_i |p_i - q_i|, \quad d_{\mathrm{H}}(p, q) = \left( {}^1\!/_2 \sum_i \left(\sqrt{p_i} - \sqrt{q_i}\right)^2 \right)^{1/2}.$$

The total variation distance is simply half the $L^1$ metric, and the Hellinger distance is a scaled version of the $L^2$ metric after applying the map $p \mapsto \sqrt{p}$. We may therefore create a locality-sensitive hash family for $d_{\mathrm{TV}}$ by direct application of Eq. (8), and create a locality-sensitive hash family for $d_{\mathrm{H}}$ by applying Eq. (8) after applying the square root map to our label distributions. These are the quantization schemes applied in our experiments.

## 5 Empirical Evaluation

Our intention here is to investigate the power of propagation kernels for graph classification. Specifically, we investigate the two following questions:
**(Q1)** Do propagation kernels utilizing continuous distribution-based features arising from propagating label information perform competitively as state-of-the-art graph kernels for graph classification?
**(Q2)** Does randomization, in particular locality-sensitive hashing techniques, improve efficiency over state-of-the-art graph kernels?
To this aim, we implemented propagation kernels in Matlab and considered the following experimental protocol.

### 5.1 Experimental Protocol

We compare classification accuracy and runtime for several different instances of propagation kernels: the *diffusion graph kernel* and the *label propagation kernel* with Hellinger distance and total variation distance, and a *structure propagation kernel*, namely the WL-subtree kernel. We choose the WL-subtree kernel for comparisons as it is currently the most accurate and efficient graph kernel and additionally report several results for other graph kernels from [18].

We consider two general settings, graph classification for fully and for partially labeled graphs. The latter is a reasonable situation when dealing with semantic images as fully labeled data is costly or even impossible to acquire. Note, that for partially labeled datasets we use the *label propagation kernel*, whereas for fully labeled graphs this does not make sense because of the *push back* of original labels. Here, we choose the *diffusion graph kernel*.

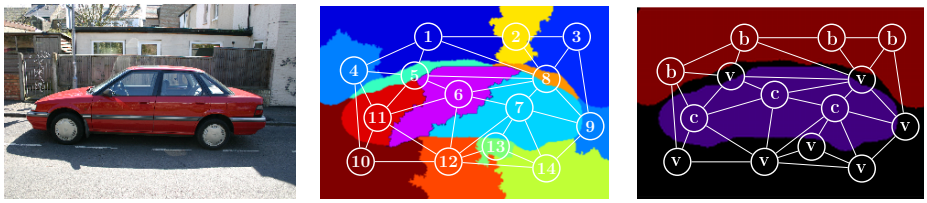
The classification performance is evaluated by running C-SVM classifications using libSVM,[4] where the cost parameter is learned by cross-validation on the training set. We compute all kernels for $T = 0, ..., 10$ and report the average of the best accuracies from 10 re-runs of a 10-fold cross-validation. For all runtime experiments all kernels are as well computed for $T = 10$ and all experiments were conducted on an Apple Mac Pro workstation with two 2.26 GHz quad-core Intel Xeon "Gainestown" processors (model E5520) and 28 GB of RAM. We used a linear base kernel for all methods and the bin width parameter for LSH was set to $w = 10^{-5}$. All results were fairly insensitive to the exact choice of $w$.

### 5.2 Datasets

We considered two real-world benchmark datasets.

**Bioinformatics Benchmark Data:** We ran experiments on the following benchmark datasets: MUTAG, ENZYMES, NCI1, NCI109, and D&D. MUTAG contains 188 sets of mutagenic aromatic and heteroaromatic nitro compounds, the label refers to their mutagenic effect on the Gram-negative bacterium *Salmonella*

[4] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`




(a) Original image (b) Graph over superpixels (c) Semantic labels

Fig. 2: **Graph Based Scene Classification** In semantic scene classification the original images (a) are represented by graphs of superpixels (b). Each superpixel node has an attached semantic label (c) (here: b = *building*, c = *car*, and v = *void*). Each graph represents the semantic scene of an image and similar scenes are classified according to a kernel capturing label and structure information.

*typhimurium*. ENZYMES has 6 EC top-level classes. It is a dataset of protein tertiary structures belonging to 600 enzymes from the BRENDA enzyme database. NCI1 and NCI109 are anti-cancer screens, in particular for cell lung cancer and ovarian cancer cell lines, respectively. D&D consists of 1178 protein structures, with the nodes in each graph represent amino acids and two nodes forming an edge if they are less than 6 Ångstroms separated. For a more comprehensive introduction and references, see [18].

**Image Benchmark Data:** The two real-world image datasets MSRC 9-class and MSRC 21-class[5] are state-of-the-art datasets in semantic image processing. Each image is represented by a conditional Markov random field graph, as illustrated in Figure 2. The nodes of each graph are derived by oversegmenting the images using the quick shift algorithm[6] with an average of 40 superpixels per graph. Hence, each node represents one superpixel and the semantic (ground-truth) node labels are derived by taking the mode ground-truth label of all pixels in the corresponding segment. Note, that the number of nodes varies from graph to graph. MSRC9 consists of 221 images, and a total of 8969 nodes. The node labels consist of nine classes *building, grass, tree, cow, sky, aeroplane, face, car, bicycle* and a label *void* to handle objects that do not fall into one of these classes. Each image can be classified into one out of eight classes. From the MSRC 21-class dataset, which is a more comprehensive and complex image dataset, we derived two datasets for our experiments. MSRC21 consists of 565 images with 24 109 labeled superpixels of 21 classes: *building, grass, tree, cow, sheep, sky, airplane, water, face, car, bicycle, flower, sign, bird, book, chair, road, cat, dog, body, boat,* and *void*. For the second dataset, MSRC21C, we extracted a subset of the most challenging scenes by removing all images having fewer than four different class labels. The resulting dataset consists of 209 graphs and 8 626 nodes

[5] `http://research.microsoft.com/en-us/projects/ObjectClassRecognition/`
[6] `http://www.vlfeat.org/overview/quickshift.html`

Table 1: Average accuracy (and standard deviation) on the image datasets for *diffusion graph kernel* $K_{\text{DIFF}}$ using Hellinger distance (+H) resp. total variation distance (+TV), and for the *WL-subtree kernel* $K_{\text{WL}}$. Bold indicates best result.

| | dataset | | |
|---|---|---|---|
| method | MSRC9 | MSRC21 | MSRC21C |
| $K_{\text{DIFF+H}}$ | 91.6 (0.5) | **83.6** (0.8) | **88.7** (0.7) |
| $K_{\text{DIFF+TV}}$ | 91.6 (0.5) | **83.6** (0.8) | **88.7** (0.7) |
| $K_{\text{WL}}$ | **92.1** (0.8) | 82.2 (1.1) | 88.5 (0.4) |

Table 2: Average accuracy (and standard deviation) on the bioinformatics benchmark datasets for *diffusion graph kernel* $K_{\text{DIFF}}$ using Hellinger distance (+H) resp. total variation distance (+TV), and for the *WL-subtree kernel* $K_{\text{WL}}$. Bold indicates best result. Results for random walk kernel and Ramon–Gärtner kernel are taken from [18] to provide a broader overview of state-of-the-art graph kernel performances; however, please note that we did not re-run the experiments and hence they have most likely been produced using different random folds.

| | dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| method | MUTAG | | ENZYMES | | NCI1 | | NCI109 | | D&D | |
| $K_{\text{DIFF+H}}$ | **87.7** | (1.3) | 47.1 | (1.2) | **84.4** | (0.2) | **84.0** | (0.3) | 79.2 | (0.4) |
| $K_{\text{DIFF+TV}}$ | 87.5 | (1.3) | 47.0 | (1.1) | 84.2 | (0.3) | 83.6 | (0.3) | 79.3 | (0.3) |
| $K_{\text{WL}}$ | 87.0 | (1.0) | **53.1** | (1.3) | 82.2 | (0.2) | 82.5 | (0.2) | **80.0** | (0.4) |
| random walk [22] | 80.7 | (0.4) | 21.7 | (0.9) | 64.3 | (0.3) | 63.5 | (0.2) | 71.7 | (0.5) |
| Ramon–Gärtner [16] | 85.7 | (0.5) | 13.4 | (0.9) | 61.9 | (0.3) | 61.7 | (0.2) | 57.2 | (0.1) |

Table 3: Average accuracy (and standard deviation) on 10 different sets of partially labeled images for *label propagation kernel* using TV distance ($K_{\text{LP+TV}}$), and for the *WL-subtree kernel* with unlabeled nodes treated as additional label $K_{\text{WL}}$ and with hard labels derived from converged LP (LP + $K_{\text{WL}}$).

| | | labels missing | | | |
|---|---|---|---|---|---|
| dataset | method | 20% | 40% | 60% | 80% |
| MSRC9 | $K_{\text{LP+TV}}$ | **90.0** (1.2) | **88.7** (1.0) | **86.6** (1.3) | **80.4** (1.8) |
| | LP + $K_{\text{WL}}$ | **90.0** (0.6) | 87.9 (1.9) | 83.2 (2.0) | 77.9 (3.1) |
| | $K_{\text{WL}}$ | 89.2 (1.5) | 88.1 (1.5) | 85.7 (1.9) | 78.5 (2.7) |
| MSRC21 | $K_{\text{LP+TV}}$ | **86.9** (0.8) | **84.7** (1.0) | **79.5** (0.9) | **69.3** (1.1) |
| | LP + $K_{\text{WL}}$ | 85.8 (0.6) | 81.5 (0.8) | 74.5 (1.0) | 64.0 (1.2) |
| | $K_{\text{WL}}$ | 85.4 (1.3) | 81.9 (1.2) | 76.0 (0.8) | 63.7 (1.3) |

in total. For both datasets based on the MSRC 21-class data each image can be classified as one out of 20 classes.

### 5.3 Experimental Results

Under our experimental protocol, propagation kernels gave the following results.

**(Q1) Predictive Performance:** The predictive performances for fully labeled graphs are summarized in Tables 1 and 2. On MSRC21, MSRC21C, MUTAG, NCI1, and NCI109, propagation kernels reached the highest accuracy. Only on ENZYMES, did $K_{\text{WL}}$ perform considerably better than propagation kernels. For the remaining datasets, the predictive performance is comparable. The Ramon–Gärtner and random-walk kernels were less competitive to the propagation kernels. To assess the predictive performance of propagation kernels on partially labeled graphs, we ran the following experiments 10 times. We randomly removed 20–80% of the labels in MSRC9, MSRC21, and MSRC21C and computed cross-validation accuracies and standard deviations. Because the WL-subtree kernel was not designed for partially labeled graphs, we compare the *label propagation kernel* to two variants: one where we treat unlabeled nodes as an additional label "*u*" ($K_{\text{WL}}$) and another where we use hard labels derived from running label propagation until convergence (LP + $K_{\text{WL}}$). The results for the first two datasets are shown in Table 3. The results for MSRC21C showed the same behavior and hence were omitted. For larger fractions of missing labels $K_{\text{LP+TV}}$ obviously outperforms the baseline methods and surprisingly running label propagation until convergence and then the WL-subtree kernel gives poorer results than $K_{\text{WL}}$. However, label propagation might be beneficial for larger amounts of missing labels. In general, the results on all experiments clearly show that question **(Q1)** can be answered affirmatively.

**(Q2) Running Time:** The runtime results are summarized in Table 4. Empirically, we observe that propagation kernels can be orders of magnitude faster then existing graph kernels. They can easily scale to graphs with thousands of nodes. On D&D, $K_{\text{DIFF+TV}}$ was computed at least twice as fast as any other method. On ENZYMES, $K_{\text{DIFF+TV}}$ takes less then a second, whereas all other methods take several seconds. Compared to $K_{\text{WL(REF)}}$, it is two orders of magnitude faster. Comparing the runtimes of $K_{\text{WL(LSH)}}$ and $K_{\text{WL(REF)}}$, we clearly see that leveraging randomization significantly outperforms the non-randomized approach. We also compared the runtime of propagation kernels using label propagation to the WL-subtree kernel on the MSRC21 dataset with partially labeled graphs. We again compare $K_{\text{LP+TV}}$ with $K_{\text{WL(LSH)}}$ and LP + $K_{\text{WL(LSH)}}$. The results are summarized in Figure 3. $K_{\text{WL(REF)}}$ is over 36 times slower than $K_{\text{LP+TV}}$. These results again confirm that propagation kernels have attractive scalability properties for large datasets. The LP + $K_{\text{WL}}$ approach wastes computation time while running LP to convergence before it can even begin calculating the kernel. The intermediate label distributions obtained during the convergence process are already extremely powerful for classification and allow one to save computation time. These results clearly answer question **(Q2)** affirmatively.

Table 4: Runtime in seconds for $T = 10$ on the bioinformatics datasets for the *diffusion graph kernel* ($K_{\text{DIFF+TV}}$) using TV distance, and for the *WL-subtree kernel* for a implementation leveraging randomization ($K_{\text{WL(LSH)}}$) and the standard implementation ($K_{\text{WL(REF)}}$) presented in [18]. Bold indicates best result. Results for WL-edge kernel ($K_{\text{WL-EDGE}}$) and graphlet count kernel are taken from [18] to provide a broader overview of state-of-the-art graph kernel performances.

| | dataset | | | | | |
|---|---|---|---|---|---|---|
| method | MUTAG | ENZYMES | NCI1 | NCI109 | D&D | total |
| $K_{\text{DIFF+TV}}$ | 0.12 s | **0.89 s** | 116 s | 133 s | **55 s** | **376 s** |
| $K_{\text{WL(LSH)}}$ | **0.03 s** | 1.6 s | 185 s | 189 s | 117 s | 493 s |
| $K_{\text{WL(REF)}}$ | 4.7 s | 29 s | 216 s | 216 s | 511 s | 977 s |
| $K_{\text{WL-EDGE}}$ [18] | 3 s | 11 s | **65 s** | **58 s** | 3 days | 3 days |
| graphlet count [19] | 3 s | 5 s | 87 s | 87 s | 23 hours | 23 hours |

To summarize, propagation kernels turned out to be competitive in terms of predictive accuracy and speed on all datasets, often by orders of magnitude. Thus, questions **(Q1)** and **(Q2)** can be answered affirmatively.

## 6 Conclusions and Future Work

Probabilistic models provide a principled way of spreading information and even treating missing information within graphs. Known labels can be used to propagate information through the graph in order to label all nodes. On the other hand, discriminative methods such as support vector machines enable us to construct flexible decision boundaries and often result in classification performance superior to that of the model based approaches. In this paper, we developed a natural way of combining both frameworks for the construction of graph kernels, called propagation kernels. Intuitively, propagation kernels count common subdistributions induced in each iteration of running inference in two graphs. For counting the continuous information — the distributional information computed for each node — efficiently, they leverage the randomized technique of locality-sensitive hashing. As our experimental results demonstrate, propagation kernels are competitive in terms of accuracy with state-of-the-art kernels on several classification benchmark datasets, even reaching the highest accuracy level on five out of eight datasets. Moreover, in terms of runtime, propagation kernels outperform other graph kernels, even the recently developed efficient WL-kernels.

Propagation kernels provide several interesting avenues for future work. While we have used classification to guide the development of propagation kernels, the results are directly applicable to regression, clustering, and ranking, among other tasks. Employing message-based probabilistic inference schemes such as (loopy) belief propagation directly paves the way to deal with more general structures

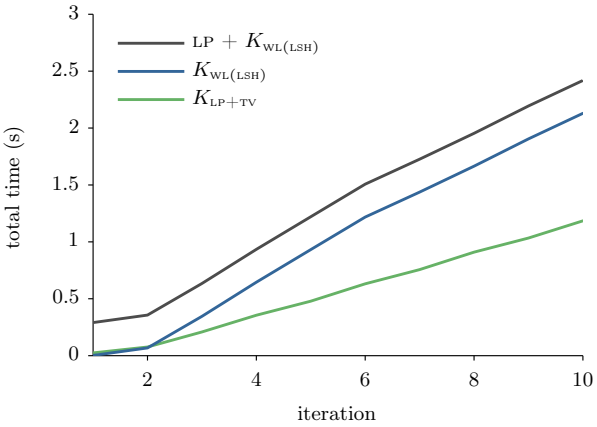



Fig. 3: **Runtime for Partially Labeled MSRC21** Average time in seconds over 10 different instances of the MSRC21 dataset with 50% labeled nodes for kernel iterations $T$ from 0 to 10. We compare the *WL-subtree kernel* with unlabeled nodes treated as additional label ($K_{\text{WL(LSH)}}$), and with hard labels derived from converged LP distributions (LP+$K_{\text{WL(LSH)}}$), and the *label propagation kernel* with TV distance ($K_{\text{LP+TV}}$). $K_{\text{WL(REF)}}$ required 36s for $T = 10$ and is not included.

then just graphs; we are currently investigating Markov logic networks [17]. By considering the computation trees—the tree-structured unrolling of a given graph rooted at the nodes—one may even realize within-network relational classification using propagation kernels.

**Acknowledgments:** This work was partly supported by the Fraunhofer ATTRACT fellowship STREAM and by the European Commission under contract number FP7-248258-First-MM.

## References

1. B. Ahmadi, K. Kersting, and S. Sanner. Multi-Evidence Lifted Message Passing, with Application to PageRank and the Kalman Filter. In *Proc. of the 22nd International Joint Conference on Artificial Intelligence (IJCAI–11)*, 2011.
2. K.M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Proceedings of International Conference on Data Mining (ICDM-2005)*, pages 74–81, 2005.
3. M. Datar and P. Indyk. Locality-sensitive hashing scheme based on $p$-stable distributions. In *Proceedings of the 20th Annual Symposium on Computational Geometry (SCG-2004)*, pages 253–262, 2004.
4. T. Gärtner, P. A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proc. of Computational Learning Theory and Kernel Machines (COLT-2003)*, pages 129–143, 2003.
5. A. Gersho and R. Gray. *Vector quantization and signal compression.* Kluwer Academic Publishers, Norwell, MA, USA, 1991.

6. S. Hido and H. Kashima. A linear-time graph kernel. In *Proc. of the 9th IEEE International Conference on Data Mining (ICDM-2009)*, pages 179–188, 2009.
7. T. Horváth, T. Gärtner, and S. Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of Knowledge Discovery in Databases (KDD-2004)*, pages 158–167, 2004.
8. T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proc. of Neural Information Processing Systems (NIPS-1998)*, pages 487–493, 1998.
9. T. Jebara, R.I. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.
10. H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proc. of the 20th International Conference on Machine Learning (ICML-2003)*, pages 321–328, 2003.
11. K. Kersting, B. Ahmadi, and S. Natarajan. Counting Belief Propagation. In *Proc. of the 25th Conference on Uncertainty in Artificial Intelligence (UAI– 09)*, 2009.
12. J.D. Lafferty and G. Lebanon. Information diffusion kernels. In *Proc. of Neural Information Processing Systems (NIPS-2002)*, pages 375–382, 2002.
13. P. Mahé and J.-P. Vert. Graph kernels based on tree patterns for molecules. *Machine Learning*, 75(1):3–35, 2009.
14. P.J. Moreno, P. Ho, and N. Vasconcelos. A Kullback-Leibler Divergence Based Kernel for SVM Classification in Multimedia Applications. In *Proc. of Neural Information Processing Systems (NIPS-2003)*, 2003.
15. M. Neumann, K. Kersting, and B. Ahmadi. Markov logic sets: Towards lifted information retrieval using pagerank and label propagation. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI–11)*, 2011.
16. J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In *Proceedings of the 1st International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74, 2003.
17. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.
18. N. Shervashidze, P. Schweitzer, E.J. van Leeuwen, K. Mehlhorn, and K.M. Borgwardt. Weisfeiler–Lehman Graph Kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.
19. N. Shervashidze, S.V.N. Vishwanathan, T. Petri, K. Mehlhorn, and K.M. Borgwardt. Efficient graphlet kernels for large graph comparison. *Journal of Machine Learning Research - Proceedings Track*, 5:488–495, 2009.
20. P. Singla and P. Domingos. Lifted First-Order Belief Propagation. In *Proc. of the 23rd AAAI Conf. on Artificial Intelligence (AAAI-2008)*, pages 1094–1099, 2008.
21. K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.-R. Müller. A new discriminative kernel from probabilistic models. *Neural Computation*, 14(10):2397–2414, 2002.
22. S.V.N. Vishwanathan, N.N. Schraudolph, R.I. Kondor, and K.M. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.
23. J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Proc. of Neural Information Processing Systems (NIPS-2000)*, pages 689–695, 2000.
24. D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Proc. of Neural Information Processing Systems (NIPS-2009)*, 2003.
25. Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, CMU-CALD-02-107, Carnegie Mellon University, 2002.