Network Science **2** (2): 254–276, 2014. © Cambridge University Press 2014
doi:10.1017/nws.2014.14

Generalized graphlet kernels for probabilistic inference in sparse graphs

JOSE LUGO-MARTINEZ and PREDRAG RADIVOJAC
Department of Computer Science and Informatics, Indiana University, Bloomington, Indiana 47405, U.S.A.
(*e-mail:* {jlugomar,predrag}@indiana.edu)

Abstract

Graph kernels for learning and inference on sparse graphs have been widely studied. However, the problem of designing robust kernel functions that can effectively compare graph neighborhoods in the presence of noisy and complex data remains less explored. Here we propose a novel graph-based kernel method referred to as an edit distance graphlet kernel. The method was designed to add flexibility in capturing similarities between local graph neighborhoods as a means of probabilistically annotating vertices in sparse and labeled graphs. We report experiments on nine real-life data sets from molecular biology and social sciences and provide evidence that the new kernels perform favorably compared to established approaches. However, when both performance accuracy and run time are considered, we suggest that edit distance kernels are best suited for inference on graphs derived from protein structures. Finally, we demonstrate that the new approach facilitates simple and principled ways of integrating domain knowledge into classification and point out that our methodology extends beyond classification; e.g. to applications such as kernel-based clustering of graphs or approximate motif finding. Availability: www.sourceforge.net/projects/graphletkernels/

Keywords: *graph kernels, vertex classification, graph edit distance*

1 Introduction

Over the past decade, the importance of graphs (networks) in modeling domain-specific phenomena and their influence in numerous scientific disciplines has grown considerably. A wide variety of such graphs, including link graphs in computer security, gene regulatory networks in computational biology, or social networks in behavioral and social sciences, have been studied with the goal of identifying interesting patterns and performing probabilistic inference. In a supervised learning scenario, typical problems involve assigning class labels to entire graphs, i.e. graph classification (Tsuda & Saigo, 2010); assigning class labels to vertices and/or edges in a single graph, i.e. vertex and/or edge classification (Koller & Friedman, 2009); or predicting the existence of edges in graphs, i.e. link prediction (Liben-Nowell & Kleinberg, 2007). Similarly, frequent subgraph mining (Jiang *et al*., 2013), motif finding (Milo *et al*., 2002), and community detection (Fortunato, 2010) are common unsupervised approaches. Although the methodology in most of these subfields has matured, further advances in network science and graph algorithms are still necessary to address practical challenges presented by large, noisy, complex, or time-varying data, as well as to design scalable algorithms that exploit domain knowledge in a principled and well-understood manner.

Here we focus on the problem of vertex classification in a supervised learning framework. Given a sparse and generally disconnected graph, where only a small fraction of vertices have class label assignments, the task is to predict class assignments for each unlabeled vertex. For instance, an important problem in computer security is the detection of phishing web sites from graphs where vertices represent web sites and edges correspond to links between these web sites. Similar situations arise in gene regulatory networks or protein-protein interaction networks in computational biology where protein function prediction (Sharan *et al.*, 2007) or disease gene prioritization (Moreau & Tranchevent, 2012) have emerged as important tasks. Finally, questions related to marketing (Leskovec *et al.*, 2007), political affiliation (Conover *et al.*, 2011), or influence (Wasserman & Faust, 1994; Aral & Walker, 2012) in social networks have gained significant attention in computational social sciences. An intuitive approach in all these problems is to devise methods that tend to assign the same class labels to vertices with similar neighborhoods. Therefore, effective techniques are needed for capturing local interconnectivity patterns between the vertex of interest and its neighbors as well as for incorporating domain knowledge and auxiliary data into classification.

There are three principal ways to address vertex classification. A straightforward approach is to provide a vector space representation of vertices in the graph and then apply machine learning algorithms to learn the target function (Xu & Li, 2006). Various local or global statistics such as vertex degree, clustering coefficient, or betweenness centrality can be computed to map each vertex into a vector of fixed length. Another approach involves the use of kernel functions on graphs combined with machine learning methods that operate with pairs of objects from the input space (Ralaivola *et al.*, 2005; Vishwanathan *et al.*, 2010). Kernel approaches are advantageous if domain knowledge provides clues on how to measure similarity between objects, rather than about which object properties may be useful for classification. Finally, vertex classification can be addressed by using probabilistic graphical models, e.g. Markov Random Fields (Koller & Friedman, 2009), or related label-propagation (Zhu & Ghahramani, 2002) and flow-based algorithms (Nabieva *et al.*, 2005). Such approaches, at least in their standard formulation, largely rely on the assumption that neighboring vertices should generally be assigned the same class labels. For this reason, probabilistic graphical models are generally preferred for learning smooth functions and may not be effective in situations where vertices with similar neighborhoods that are far away in a graph or in a different connected component need to be assigned the same class label.

In this article, we are primarily interested in kernel-based classification. Graph kernels were introduced by Kondor and Lafferty (2002) in the form of diffusion kernels. Several other methods have been subsequently proposed, including random walk kernels (Gärtner *et al.*, 2003; Kashima *et al.*, 2003), cycle pattern kernels (Horváth *et al.*, 2004), shortest path kernels (Borgwardt & Kriegel, 2005), weighted decomposition kernels (Menchetti *et al.*, 2005), subtree pattern kernels (Ramon & Gärtner, 2003; Hido & Kashima, 2009; Shervashidze *et al.*, 2011), commute time kernels (Fouss *et al.*, 2007), and graphlet kernels (Shervashidze *et al.*, 2009; Vacic *et al.*, 2010). While these methods exploit a variety of ways to compare graphs, most of them are tailored towards the problem of graph classification. Moreover, these methods are not explicitly designed to account for noise in graph data as well as to naturally operate on graphs associated with particular labeling (coloring) functions, e.g. those that map vertices to finite alphabets.

Here we describe the development of *edit distance graphlet kernels*. Inspired by string kernel approaches (Leslie *et al.*, 2002; Leslie & Kuang, 2004; Kuang *et al.*, 2005), we extend the concept of labeled graphlet kernels to directed graphs and introduce flexibility in the graphlet matching algorithm. We use complex data from molecular biology and social sciences to provide evidence that the new kernels compare favorably to the previously published methods on both undirected and directed graphs. Our main contribution and emphasis in this work is on conceptual generalization of graphlet kernels for vertex classification, characterization of the feature space, and identification of learning problems in which one can expect an increase in performance accuracy, even if traded for decreased scalability. The edit distance kernels were implemented in C/C++. The source code and documentation are available free of charge under the MIT Open Source license agreement.

The remainder of this paper is organized as follows. In Section 2, we review the basic concepts related to graphs and then describe our methodology aimed at providing robustness and flexibility of graph kernels through calculating edit distances on graphs. In Section 3, we describe the construction of several new data sets, show how domain knowledge can be incorporated into the graph analysis, and provide comprehensive performance evaluation. In Section 4, we summarize the work and discuss other areas where our methodology may be applicable.

2 Methods

In the following subsections, we briefly review the definitions of graphs, graphlets, and edit distance between graphs. We then describe the development of a novel class of graphlet kernels and analyze their properties.

2.1 Graphs and graphlets

A graph G is a pair (V, E), where V is a set of vertices and $E \subseteq V \times V$ is a set of edges. In a vertex-labeled graph, we consider a labeling function $f : V \rightarrow \Sigma$, where Σ is a finite alphabet. In an edge-labeled graph, we can consider a similar mapping $g : E \rightarrow \Xi$, but due to the nature of our data (Section 3) we largely ignore edge labeling (note that vertex and edge labeling are also referred to as coloring). A graph is said to be simple if its vertices have no self-loops, and undirected if E is symmetric. A rooted graph G is a graph together with one vertex distinguished from the others. We denote such graphs as $G = (V, v, E)$, where $v \in V$ is the root.

We define a *graphlet* to be a small, simple, connected, rooted graph (Figure 1). A graphlet with n vertices is referred to as an n-graphlet. In this framework, it is sufficient to work with graphlets up to isomorphism (root- and/or label-preserving in the case graphs are rooted and/or labeled). We shall always enumerate graphlets up to isomorphism and use n_i to denote the i-th graphlet of order n. For example, when $|\Sigma| = 1$, there are six rooted graphs with $n = 3$ vertices, but only three up to isomorphism (Figures 1–2). We refer to the non-isomorphic unlabeled graphlets as *base graphlets*. When $|\Sigma| > 1$, automorphic structures with respect to the same

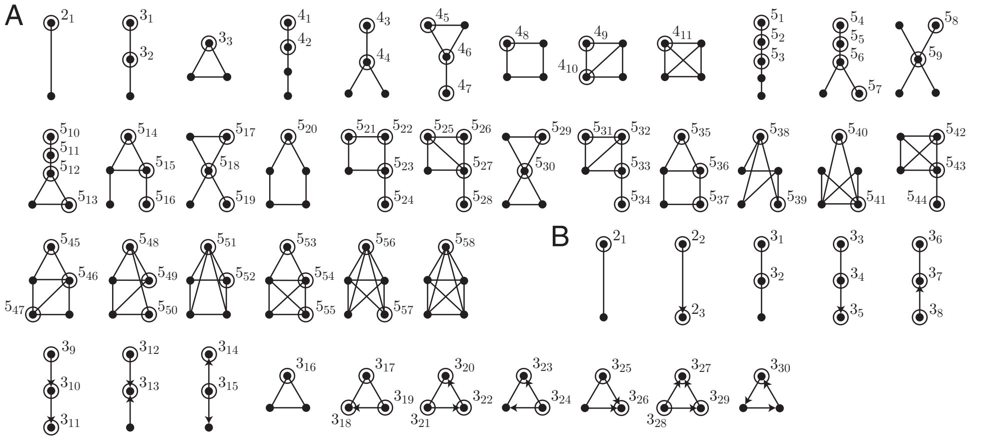

Fig. 1. (A) All undirected graphlets with 2 to 5 vertices and (B) all directed graphlets with 2 and 3 vertices. The root node of each graphlet is circled. All graphlets are presented in a compressed notation; e.g., the two non-isomorphic graphlets 4_1 and 4_2 are shown in one drawing, etc. The bidirectional edges in part B are shown without arrows.

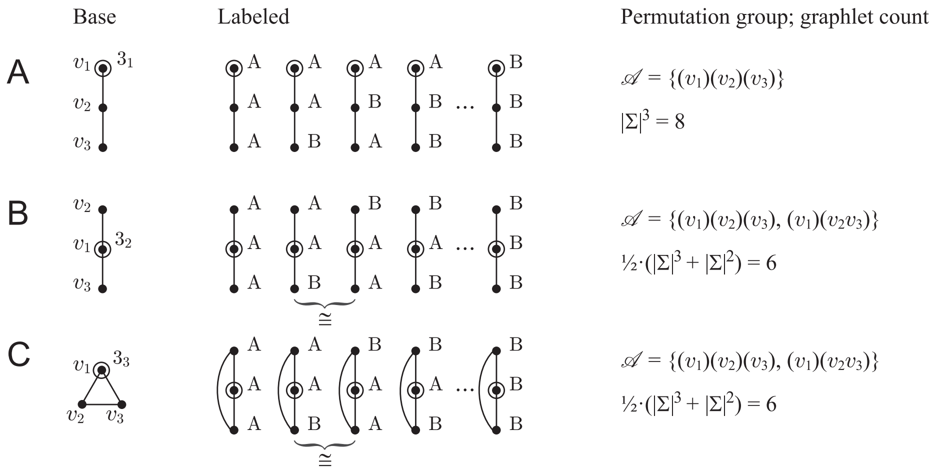
Base
Labeled
Permutation group; graphlet count
$\mathscr{A} = \{(v_1)(v_2)(v_3)\}$
$|\Sigma|^3 = 8$
$\mathscr{A} = \{(v_1)(v_2)(v_3), (v_1)(v_2v_3)\}$
$½\cdot(|\Sigma|^3 + |\Sigma|^2) = 6$
$\mathscr{A} = \{(v_1)(v_2)(v_3), (v_1)(v_2v_3)\}$
$½\cdot(|\Sigma|^3 + |\Sigma|^2) = 6$

Fig. 2. Labeled graphlets for $\Sigma = \{A, B\}$, permutation group $\mathscr{A}$, and graphlet count for each base 3-graphlet. Isomorphic labeled graphlets are indicated with $\cong$. Note that base graphlet 3_1 (A) is asymmetric whereas base graphlets 3_2 (B) and 3_3 (C) have equivalent symmetries which result in identical permutation groups and graphlet counts.

base graphlet may exist; thus, the number of labeled graphlets per base structure is generally less than $|\Sigma|^n$. As an example, consider 3-graphlets; here, there are $|\Sigma|^3$ graphlets corresponding to the asymmetric base graphlet 3_1 but only $\frac{1}{2}(|\Sigma|^3 + |\Sigma|^2)$ corresponding to the base graphlets 3_2 and 3_3 (Figure 2). This is a result of symmetries in the base graphlets that give rise to automorphisms among labeled structures.

There is a long history of using small graphs to probe large graphs (Lovász, 1967; Bondy & Hemminger, 1977; Borgs *et al.*, 2006). In computational sciences, graphlets have been recently used to capture global or local graph topology (Przulj *et al.*, 2004; Przulj, 2007; Shervashidze *et al.*, 2009; Vacic *et al.*, 2010). In this work we exploit rooted graphs and note that they are different from the notion of k-subgraphs used in the network science literature (Frank, 1988). Graphlets are connected and rooted, whereas k-subgraphs need not be either. Technically, our definition of graphlets also differs from the one by Przulj and colleagues (2004; 2007), which relies on the notion of an orbit. Finally, the concept of a graphlet shall be distinguished from a network motif. Typically, a motif is a subgraph whose frequent occurrence is statistically significant according to an assumed background model (Milo *et al.*, 2002).

Graph edit distance is a generalization of the concept of edit distance for strings. Given two vertex- and/or edge-labeled graphs G and H, the edit distance between these graphs corresponds to the minimum number of edit operations necessary to transform G into H. The allowed edit operations typically comprise insertion or deletion of vertices and edges, or in the case of labeled graphs, substitutions of vertex and edge labels. Any sequence of edit operations that transforms G into H is referred to as an *edit path*. Thus, the *graph edit distance* between G and H corresponds to the length of the shortest edit path between them. In a more general case where each edit operation is assigned a weight (cost), the graph edit distance corresponds to the edit path of minimum weight (cost).

2.2 Problem formulation

We consider a binary vertex classification problem. Given a simple vertex-labeled graph $G = (V, E, f, \Sigma)$, another labeling function $t : V \to \{+1, -1\}$ is introduced and referred to as a *target function*, where $t(v) = +1$ indicates that vertex $v \in V$ belongs to the positive class and $t(v) = -1$ indicates that vertex v belongs to the negative class. In a real-life scenario, only a small fraction of vertices in V are expected to be annotated by t. Thus, the goal is to learn t from the graph and infer class posterior probabilities for every vertex not annotated by t.

We briefly digress to emphasize the difference between graph classification and vertex classification. In graph classification, one is given a set of graphs $\mathscr{G} = \{G_i\}_{i=1}^{n}$ with some training data $\{(G_i, t_i)\}_{i=1}^{m}$, where $m \leqslant n$, and the objective is to learn the target function $t : \mathscr{G} \to \{+1, -1\}$. Graph classification can be seen in chemical informatics where one may desire to classify small compounds as toxic vs. non-toxic or in computational biology where the task may be to classify protein structures into different functional categories. Vertex classification can be similarly formulated as a learning problem of some target function $t : \mathscr{H} \to \{+1, -1\}$, where $\mathscr{H} = \{H_i\}_{i=1}^{n}$ is a set of rooted graphs and each $H_i = (V, v_i, E)$ corresponds to the same graph $G = (V, E)$ rooted at a different vertex $v_i \in V$. Therefore, vertex classification can be considered to be a special case of graph classification on rooted graphs.

Given an input space $\mathscr{X}$, a kernel function is any symmetric positive semi-definite mapping $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ (Shawe-Taylor & Cristianini, 2004). Positive semi-definiteness guarantees the existence of a possibly infinite-dimensional Hilbert space such that the kernel function can be computed as the inner product of the images of input objects. It also guarantees a globally optimal solution (unique if positive definite) by a support vector machine solver (Haussler, 1999). A graph kernel is a user-specified kernel function that measures the similarity between two graphs. Therefore, designing a good kernel function between vertex neighborhoods is key to accurate inference.

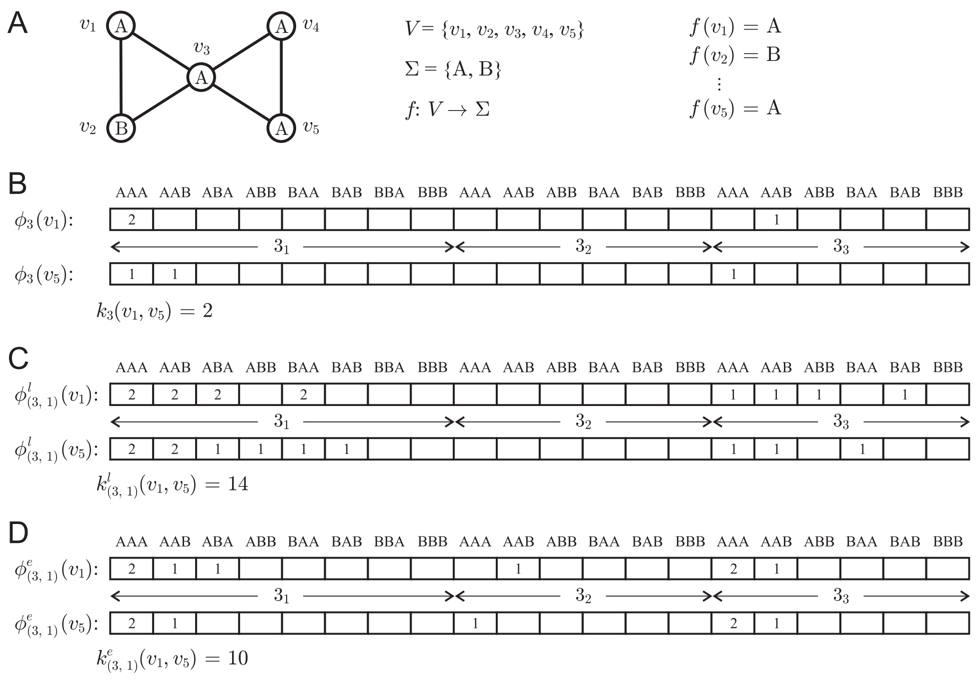

Fig. 3. An example of calculating kernel values. Panel (A) shows a vertex-labeled graph with $V = \{v_1, v_2, v_3, v_4, v_5\}$ and $\Sigma = \{\text{A, B}\}$. Panel (B) depicts count vectors for all labeled 3-graphlets rooted at vertices v_1 and v_5. Panels (C)-(D) depict count vectors and kernel values for edit distance kernels in which the allowed edit operations are one label substitution and one edge insertion/deletion, respectively. In panels (B)-(D), all labeled graphlets are associated with corresponding unlabeled graphlets 3_1, 3_2 and 3_3. The count vector corresponding to the kernel that incorporates both one label substitution and one edge indel can be computed as $\phi^l_{(n,1)}(v) + \phi^e_{(n,1)}(v) - \phi_n(v)$.

2.3 Graphlet kernels for vertex classification

Consider a graph $G = (V, E, f, \Sigma)$, where $|\Sigma| \geqslant 1$. We define the n-graphlet count vector for any vertex $v \in V$ as

$$\phi_n(v) = (\varphi_{n_1}(v), \varphi_{n_2}(v), \ldots, \varphi_{n_{\kappa(n,\Sigma)}}(v)),$$

where $\varphi_{n_i}(v)$ is the count of the i-th labeled n-graphlet and $\kappa(n, \Sigma)$ is the total number of labeled n-graphlets. A kernel function between the n-graphlets for vertices u and v can be defined as

$$k_n(u, v) = \langle \phi_n(u), \phi_n(v) \rangle ,$$

where $\langle x, y \rangle$ represents the inner product of vectors x and y. As an example, consider vertices v_1 and v_5 in the graph depicted in Figure 3(a) where $\Sigma = \{\text{A, B}\}$. The resulting 3-graphlet count vectors $\phi_3(v_1)$ and $\phi_3(v_5)$ are shown in Figure 3(a). In this case, we count $|\Sigma|^3 = 8$ graphlets corresponding to the base graphlet 3_1 and $\frac{1}{2}(|\Sigma|^3 + |\Sigma|^2) = 6$ graphlets corresponding to each of the 3_2 and 3_3 base graphlets (see also Figure 2). Figure 3(b) also shows the calculation for $k_3(v_1, v_5)$.

The graphlet kernel function can now be defined as

$$k(u,v) = \sum_{n=1}^{N} k_n(u,v),$$

where N is a small integer (Vacic *et al.*, 2010). Note that given the count vectors, there are alternative ways to define both k_n and k. For example, the Jaccard similarity coefficient (Ralaivola *et al.*, 2005) can be used to define k_n, while a product of kernels (Haussler, 1999) or a hyperkernel (Borgwardt *et al.*, 2005) can be used to combine different kernel functions. The particular representation in this work was chosen for its simplicity and the fact that more sophisticated multiple kernel learning schemes usually result in only small improvements over linear combinations (Gönen & Alpaydin, 2011). We also consider a normalized kernel calculated as

$$k'(u,v) = \frac{k(u,v)}{\sqrt{k(u,u)\,k(v,v)}},$$

which corresponds to the cosine similarity function between feature representations of u and v. It is not difficult to show that both $k(u,v)$ and $k'(u,v)$ are kernel functions (Haussler, 1999).

2.4 The edit distance graphlet kernels

In this section we introduce kernel functions which incorporate inexact matches between graphlets. As before, we formalize the notion of an edit distance kernel by considering a set of n-graphlets. Let us define a vector of counts for an m-edit distance representation of vertex v as

$$\phi_{(n,m)}(v) = (\psi_{(n_1,m)}(v), \psi_{(n_2,m)}(v), \ldots, \psi_{(n_{\kappa(n,\Sigma)},m)}(v)),$$

where

$$\psi_{(n_i,m)}(v) = \sum_{n_j \in E(n_i,m)} w(n_i,n_j) \cdot \varphi_{n_j}(v).$$

As defined previously, $\varphi_{n_j}(v)$ is the count of the j-th labeled n-graphlet rooted at v and $E(n_i,m)$ is a set of n-graphlets such that for each $n_j \in E(n_i,m)$ there exists an edit path of length at most m that transforms n_i into n_j. When $m = 0$, $E(n_i,m)$ contains only element n_i and $\phi_{(n,m)} = \phi_n$. Weights $w(n_i,n_j) \geqslant 0$ can be used to adjust the influence of pseudo counts and control computational complexity; in this work, we set $w(n_i,n_j) = 1$ if $n_j \in E(n_i,m)$ and $w(n_i,n_j) = 0$ otherwise.

The length-m edit distance n-graphlet kernel $k_{(n,m)}(u,v)$ between vertices u and v can be computed as an inner product between the respective count vectors $\phi_{(n,m)}(u)$ and $\phi_{(n,m)}(v)$. Hence, the length-m edit distance graphlet kernel function can be expressed as

$$k_m(u,v) = \sum_{n=1}^{N} k_{(n,m)}(u,v).$$

Because of the nature of our data, we only allow substitutions of vertex labels and insertions or deletions (*indels* for short) of edges as edit operations. Hence, we can define two subclasses of edit distance kernels referred to as (vertex) label-substitution $k^l_{(n,m)}$ and edge-indel kernels $k^e_{(n,m)}$. As an example, in Figure 3(c) we show the calculation of the label-substitution 3-graphlet kernel ($m = 1$) between vertices v_1 and v_5. We obtain that $k^l_{(3,1)}(v_1, v_5) = 14$, whereas the similarity between v_1 and v_5 when $m = 0$ is $k_3(v_1, v_5) = 2$ (Figure 3(b)). To provide more detail, let us consider vertex v_1 and base graphlet 3_3. By allowing for a substitution of one vertex label, the graphlet with labels AAB will lead to incremented counts for graphlets AAA, ABB and BAB. Each of these graphlets requires a single edit operation to form a root-preserving label-isomorphic graphlet to the original structure. In Figure 3(d), we show the computation of the edge-indel 3-graphlet kernel ($m = 1$) between vertices v_1 and v_5 for the graph from Figure 3(a). In contrast to label substitutions, edge indels lead to different base graphlets; thus, it may be beneficial to pre-compute the mapping between such base graphlets in order to speed up the generation of count vectors. Note that we ignore operations where removal of an edge produces a disconnected graph. For instance, base graphlet 3_1 can be mapped to 3_3 by adding one edge (Figure 2); however, removal of any edge results in a disconnected graph that is excluded from counting.

It is important to emphasize that edit distance graphlet kernels can be further generalized to address more complex situations. Those involve weighted graphs and/or graphs where one can define similarity functions on the elements of alphabets used for vertex or edge labels. Weighted graphs are typically defined by providing a mapping function $E \to \mathbb{R}^+$, whereas similarity between the elements of a vertex-labeling alphabet can be introduced as $\Sigma \times \Sigma \to [0, \infty)$. These mappings can be combined to obtain weighted edit paths and used to generalize the definition of edit distance kernels as

$$\phi_{(n,\tau)}(v) = (\psi_{(n_1,\tau)}(v), \psi_{(n_2,\tau)}(v), \ldots, \psi_{(n_{\kappa(n,\Sigma)},\tau)}(v)),$$

where

$$\psi_{(n,\tau)}(v) = \sum_{n_j \in E(n_i,\tau)} w(n_i, n_j) \cdot \varphi_{n_j}(v).$$

Here, $E(n_i, \tau)$ is the set of all n-graphlets such that for each $n_j \in E(n_i, \tau)$ there exists an edit path of total weight at most τ that transforms n_i into n_j and $w(n_i, n_j) \geqslant 0$ is the minimum weight (cost) among all edit paths between n_i and n_j (potentially also adjusted for pseudo counts).

2.5 Growth pattern of vertex-labeled graphlets

To understand the importance and properties of the proposed kernel-based classification, it is necessary to investigate the dimensionality of count vectors $\phi_{(n,m)}(v)$. In particular, we are interested in the order of growth of $\kappa(n, \Sigma)$ with n and Σ.

Suppose that G and H are base graphlets of order n. We say that G and H belong to the same symmetry group if and only if the total number of (non-isomorphic) vertex-labeled graphlets corresponding to the base cases G and H are equal for any Σ. The total counts of labeled graphlets over all alphabet sizes induce a partition of base graphlets into symmetry groups (equivalence classes). We denote the set of all symmetry groups over the graphlets of order n as $S(n) = \{S_1(n), S_2(n), \ldots\}$. For example, the set of 3-graphlets can be partitioned into two symmetry groups: $S_1(3) = \{3_1\}$ and $S_2(3) = \{3_2, 3_3\}$. Table 1 lists symmetry groups induced by partitioning base graphlets up to the order of 5 along with the cardinality of each set.

Table 1. *Symmetry groups and their cardinality produced by partitioning the set of undirected and directed base graphlets for* $n \in \{1,2,3,4,5\}$. *The last column lists the number of labeled graphlets over an alphabet* Σ.

	Undirected	**Directed**	
$S_i(n)$	$\lvert S_i(n)\rvert$	$\lvert S_i(n)\rvert$	$m_i(n,\Sigma)$
$S_1(1)$	1	1	$\lvert\Sigma\rvert$
$S_1(2)$	1	3	$\lvert\Sigma\rvert^2$
$S_1(3)$	1	24	$\lvert\Sigma\rvert^3$
$S_2(3)$	2	6	$\frac{1}{2}(\lvert\Sigma\rvert^3 + \lvert\Sigma\rvert^2)$
$S_1(4)$	3	586	$\lvert\Sigma\rvert^4$
$S_2(4)$	6	102	$\frac{1}{2}(\lvert\Sigma\rvert^4 + \lvert\Sigma\rvert^3)$
$S_3(4)$	2	6	$\frac{1}{6}(\lvert\Sigma\rvert^4 + 3 \cdot \lvert\Sigma\rvert^3 + 2\lvert\Sigma\rvert^2)$
$S_4(4)$	–	3	$\frac{1}{3}(\lvert\Sigma\rvert^4 + 2 \cdot \lvert\Sigma\rvert^2)$
$S_1(5)$	13	40,833	$\lvert\Sigma\rvert^5$
$S_2(5)$	6	102	$\frac{1}{4}(\lvert\Sigma\rvert^5 + 2 \cdot \lvert\Sigma\rvert^4 + \lvert\Sigma\rvert^3)$
$S_3(5)$	24	3,624	$\frac{1}{2}(\lvert\Sigma\rvert^5 + \lvert\Sigma\rvert^4)$
$S_4(5)$	2	6	$\frac{1}{24}(\lvert\Sigma\rvert^5 + 6 \cdot \lvert\Sigma\rvert^4 + 11 \cdot \lvert\Sigma\rvert^3 + 6 \cdot \lvert\Sigma\rvert^2)$
$S_5(5)$	6	102	$\frac{1}{6}(\lvert\Sigma\rvert^5 + 3 \cdot \lvert\Sigma\rvert^4 + 2 \cdot \lvert\Sigma\rvert^3)$
$S_6(5)$	5	174	$\frac{1}{2}(\lvert\Sigma\rvert^5 + \lvert\Sigma\rvert^3)$
$S_7(5)$	2	6	$\frac{1}{8}(\lvert\Sigma\rvert^5 + 2 \cdot \lvert\Sigma\rvert^4 + 3 \cdot \lvert\Sigma\rvert^3 + 2 \cdot \lvert\Sigma\rvert^2)$
$S_8(5)$	–	6	$\frac{1}{4}(\lvert\Sigma\rvert^5 + \lvert\Sigma\rvert^3 + 2 \cdot \lvert\Sigma\rvert^2)$
$S_9(5)$	–	54	$\frac{1}{3}(\lvert\Sigma\rvert^5 + 2 \cdot \lvert\Sigma\rvert^3)$

Generalizing this approach to graphlets labeled by any alphabet Σ, it holds that

$$\kappa(n,\Sigma) = \sum_{i=1}^{|S(n)|} m_i(n,\Sigma) \cdot |S_i(n)|,$$

where $m_i(n,\Sigma)$ is the number of (non-isomorphic) labeled graphlets corresponding to any base graphlet from the symmetry group $S_i(n)$ and $|\cdot|$ is a set cardinality operator. This decomposition also suggests that the total dimensionality of the count vectors can be calculated by first finding the symmetry groups corresponding to the base graphlets and then counting the number of labeled graphlets for any one member of the group.

In the cases of both undirected and directed graphlets, $m_i(n,\Sigma)$ can be computed by applying Pólya's enumeration theorem (Pólya, 1937). In particular, Harary and Palmer (1973) showed the derivation of the complete generating function for each symmetry group $S_i(n)$. The key insight in their work was that the set of automorphisms $\mathscr{A}$ of a specific graph $G = (V,E)$ defines a permutation group. Each permutation $\alpha \in \mathscr{A}$ can be written uniquely as the product of disjoint cycles such that for each integer $k \in \{1,\ldots,n\}$, we define $j_k(\alpha)$ as the number of cycles of length k in the disjoint cycle expansion of α. The cycle index of $\mathscr{A}$, denoted as $Z(\mathscr{A})$, is a polynomial in $s_1, s_2, \ldots, s_n$ given by

$$Z(\mathscr{A}; s_1, s_2, \ldots, s_n) = |\mathscr{A}|^{-1} \sum_{\alpha \in \mathscr{A}} \prod_{k=1}^{n} s_k^{j_k(\alpha)}.$$

By applying Pólya's enumeration theorem in the context of enumerating labeled graphlets corresponding to any base graphlet in $S_i(n)$, we obtain that $m_i(n,\Sigma)$ is

Table 2. *The number of undirected and directed graphlets $\kappa(n, \Sigma)$ as a function of n and Σ.*

	Undirected		**Directed**	
n	$\|\Sigma\| = 1$	$\|\Sigma\| = 20$	$\|\Sigma\| = 1$	$\|\Sigma\| = 20$
1	1	20	1	20
2	1	400	3	1,200
3	3	16,400	30	217,200
4	11	1,045,600	697	102,673,600
5	58	100,168,400	44,907	137,252,234,400

determined by substituting $|\Sigma|$ for each variable s_k in $Z(\mathscr{A})$. Hence,

$$m_i(n, \Sigma) = Z(\mathscr{A}; |\Sigma|, |\Sigma|, \ldots, |\Sigma|),$$

where $\mathscr{A}$ is the automorphism group of a base graphlet from $S_i(n)$. We illustrate this for the symmetry group $S_2(3)$ and $\Sigma = \{\text{A, B}\}$, as shown in Figure 2. The automorphism group $\mathscr{A} = \{(v_1)(v_2)(v_3), (v_1)(v_2 v_3)\}$; thus, $Z(\mathscr{A}; s_1, s_2, s_3) = \frac{1}{2}(s_1^3 + s_1 \cdot s_2)$. It follows that, $m_2(3, \Sigma) = Z(\mathscr{A}; 2, 2, 2) = \frac{1}{2}(2^3 + 2 \cdot 2) = 6$.

It is not clear to us how to calculate or approximate the order of growth of the number of symmetry groups $|S(n)|$ beyond the obvious observation that it is at least linear with the number of vertices (Table 1). Regardless of that fact, the number of labeled graphlets increases superlinearly with both n and $|\Sigma|$. Since we are not aware of any tight theoretical bounds on the order of growth of $\kappa(n, \Sigma)$, we demonstrate the extent of sparse representation of count vectors by computing $\kappa(n, \Sigma)$ for some typical parameter values for both undirected and directed graphs (Table 2).

2.6 Algorithms and data structures

Here we show that the computation of edit distance kernels is practical for sparse graphs. Let $N_{n-1}(v) = (V(v), E(v))$ be the neighborhood graph of vertex v such that all nodes in $N_{n-1}(v)$ are at distance at most $n - 1$ from v (directionality of edges is ignored in directed graphs when establishing $N_{n-1}(v)$). We assume that $N_{n-1}(v)$ is significantly smaller than the original graph G; otherwise, the analysis must involve G in place of $N_{n-1}(v)$.

The count vector $\phi_n(v)$ can be generated by first discovering $N_{n-1}(v)$ from G and v and then counting graphlets rooted at v. All nodes and edges in $N_{n-1}(v)$ can be found in a breadth-first fashion starting from v. The worst-case time complexity of this operation is $O(|E(v)|)$. Additionally, there are $O(d^n)$ operations that verify the existence of edges between pairs of non-root vertices of each graphlet rooted at v, where d is the maximum vertex degree (in- and out-degree for directed graphs) in $N_{n-1}(v)$. For these reasons, the computational complexity of the counting algorithm is of the order of $O(|E(v)| + d^n)$ for each training and test data point.

To compute $\phi_{(n,m)}(v)$ we start with $\phi_n(v)$. Generating pseudo counts for a single edit operation requires $O(n|\Sigma| + n^2)$ steps for each nonzero entry in $\phi_n(v)$. These include $O(n|\Sigma|)$ steps to generate all label substitutions and $O(n^2)$ to perform edge insertions and deletions. Therefore, for each vertex v an order of $O(\min\{|V(v)|^n, \kappa(n, \Sigma)\}(n|\Sigma| + n^2)^m)$ operations are necessary, where the $|V(v)|^n$ term enumerates possible n-graphlets in $N_{n-1}(v)$. This bound, however, is not tight because the mapping between base graphlets in cases of edge insertions and deletions can be precomputed to avoid creation of disconnected graphs. In summary, owing to the exponential time complexity, edit distance kernels are applicable only to sparse graphs when relatively small values of n and m are used.

The implementation we propose for computing edit distance kernels is an extension of the available solutions for string kernels (Rieck & Laskov, 2008). Because the order of nodes in each graphlet can be pre-specified it is easy to see that each asymmetric graphlet type can be treated exactly as if it were a string. Only minor modifications are necessary for each symmetry group, as the symmetric (groups of) vertices can also be sorted lexicographically (Vacic *et al.*, 2010). The implementation in this work is based on a hash structure, where computing the kernel function is performed in time linear in the number of non-zero elements.

3 Results

In this section we first summarize classification problems and data sets. We then describe simple ways of incorporating domain knowledge into classification and study the impact of inexact graphlet matches. Finally, we compare the proposed kernels to previously published approaches.

3.1 Classification problems and data sets

We evaluate the performance on nine data sets constructed from three groups of vertex-classification problems. All data sets are available on our web site.

Protein structure data: We consider the problem of predicting different types of functional residues in protein structures. In order to obtain vertex-labeled graphs, protein structures assembled by Xin and Radivojac (2011) were modeled as protein contact graphs, where each amino acid residue was represented as a vertex and two spatially close residues (6Å or less between two C_α atoms) were linked by an undirected edge. We created four data sets addressing different biological problems, i.e. prediction of (1) catalytic residues (Cat), (2) phosphorylation sites (Phos), (3) zinc-binding residues (Zn), and (4) DNA-binding residues (DNA). In each case, the training data consisted of one large vertex-labeled disconnected graph in which each connected component corresponded to one protein chain.

Protein-protein interaction data: Given a human protein-protein interaction (PPI) network and the molecular function of each protein (gene), the classification task is to predict gene-disease associations. In the PPI network, proteins are represented as vertices and proteins that physically interact are linked by undirected edges. Starting from a large data set from (Radivojac *et al.*, 2008), we constructed three binary classification problems: prediction of (1) cancer vs. other disease genes (Can), (2) metabolic disorder vs. other disease genes (Met), and (3) metabolic disorder vs. cancer genes (Met/Can). In the Met/Can data set, all genes associated with both cancer and metabolic disorders were removed.

Blog and tweet data: Given a set of political blogs (or tweets), we consider the classification problem of labeling each blog (or user) as liberal or conservative. For each blog, a directed link graph was constructed from the list of links to other political blogs. Each vertex (i.e. political blog URL) was then labeled based on its

Table 3. *Data set summary. For each graph $G = (V, E)$ we show the number of vertices and edges in the entire graph as well as the number of positive (n_+) and negative (n_-) data points.*

	Cat	**Phos**	**Zn**	**DNA**	**Can**	**Met**	**Met/Can**	**Blogs**	**Tweets**
$\|V\|$	104,751	138,174	87,540	22,531	9,590	9,590	9,590	1,490	18,470
$\|E\|$	$2.6 \cdot 10^6$	$2.0 \cdot 10^6$	$2.4 \cdot 10^6$	$4.3 \cdot 10^6$	41,456	41,456	41,456	33,436	61,157
$\|\Sigma\|$	40	40	40	40	16	16	16	7	16
n_+	988	1,157	1,420	2,921	435	314	284	586	4,000
n_-	988	1,157	1,420	2,921	1,082	1,203	405	636	4,000

weblog link directory. We use a single day snapshot of over 1,000 political blogs captured before the 2004 presidential election in the U.S. over seven ($|\Sigma| = 7$) online weblog directories (Adamic & Glance, 2005). This data set was selected because vertex labels are noisy; that is, many of the listing directories were self-reported or automatically categorized. Similarly, for each Twitter user, a directed link graph was constructed by connecting user A to user B if B retweeted any message originally tweeted by A. We use a sample of the retweet network obtained from political communication between users of Twitter in the six weeks prior to the 2010 U.S. congressional midterm elections (Conover *et al.*, 2011).

The basic parameters from all data sets are summarized in Table 3. The graph density, calculated as $|E|/\binom{|V|}{2}$, was examined for each of the data sets. Low density graphs were observed for Phos ($2.1 \cdot 10^{-4}$) and Tweets ($3.6 \cdot 10^{-4}$) data, whereas graphs with a relatively high density were found in DNA ($1.6 \cdot 10^{-2}$) and Blogs ($3.0 \cdot 10^{-2}$) data.

3.2 Integrating domain knowledge

In order to provide fair comparisons between all protein structure-based methods and also demonstrate the integration of domain knowledge, we incorporated evolutionary information into functional residue prediction by extending the amino acid alphabet from 20 to 40 symbols. We first formed a multiple sequence alignment starting from each protein sequence using non-redundant GenBank as a reference database, then calculated the conservation index for each position using AL2CO (Pei & Grishin, 2001), and finally grouped conservation scores into high and low categories. A new label was constructed by combining the old label (one of the 20 amino acids) and the binary conservation score. For instance, the amino acid alanine was split into highly conserved alanines and other alanines.

To incorporate domain knowledge into the PPI network data, we first predicted Gene Ontology (GO) terms for each protein using the FANN-GO algorithm (Clark & Radivojac, 2011) and then applied hierarchical clustering on the predicted term scores to group proteins into $|\Sigma| = 16$ broad functional categories. Given two proteins (i.e. vertices in the graph) u and v, FANN-GO was used to provide vectors of functional annotations $\mathbf{f}$ and $\mathbf{g}$ such that each component was a prediction score for a single GO term from the Molecular Function and Biological Process ontologies ($\mathbf{f}$ and $\mathbf{g}$ were 2,132-dimensional vectors of numbers between 0 and 1). Because the components in these vectors are non-negative, the distance between u and v was calculated as

$$d(u,v) = 1 - \frac{\mathbf{f}^T \mathbf{g}}{\|\mathbf{f}\| \, \|\mathbf{g}\|},$$

where $\|\mathbf{f}\|$ is the Euclidean norm of $\mathbf{f}$. Subsequently, we applied hierarchical clustering with Ward's linkage criterion to produce $|\Sigma| = 16$ functional classes of proteins.

In the case of social network data, we exploited auxiliary data in the Twitter problem as follows: we (1) collected each user's tweets; (2) stemmed, filtered out short and common stop words from all tweets; (3) generated a bag-of-words tweet representation for each user; and finally (4) applied hierarchical clustering on the bag-of-words data to group users into $|\Sigma| = 16$ groups (the distance between vertices in the graph was computed in the same way as in the PPI network). Edit distance kernels become critical in cases of large alphabets because exact matches between labeled graphlets become less likely.

3.3 Evaluation methodology

For each data set, we evaluated edit distance kernels by comparing them to the standard graphlet kernel (Vacic *et al.*, 2010) and two in-house implementations of random walk kernels. The random walk kernels were implemented as follows: given a graph G and two vertices u and v, simultaneous random walks w_u and w_v were generated from u and v using random restarts. In the conventional random walk implementation, a walk was scored as 1 if the entire sequences of labels between w_u and w_v matched; otherwise, a walk was scored as 0. After 100,000 steps, the scores over all walks were summed to produce a kernel value between u and v. In order to construct a random walk similar to the graphlet edit distance approach, a cumulative random walk kernel was also implemented. Here, any match between the labels of vertices u_i and v_i in the i-th step of each walk was scored as 1, while a mismatch was scored as 0. Thus, a walk of length ℓ could contribute between 0 and ℓ to the total count. In each of the random walks, the probability of restart was selected from a set $\{0.1, 0.2, \ldots, 0.5\}$ and the result with the highest accuracy was reported.

In case of the protein structure data, all graph kernels were compared against three domain-specific methods: FEATURE (Halperin *et al.*, 2008), the method developed by Gutteridge *et al.* (2003), and the structure kernel (Xin *et al.*, 2010). We used the default parameters as provided in the original publications. These methods are representative of many computational approaches for the problem of predicting functional residues in protein structures.

The performance of each method was evaluated through a 10-fold cross-validation. An exception was made for the protein structure data, where per-chain cross-validation was applied to emulate a realistic scenario in which a completely new protein chain (i.e. connected graph) is presented to a classifier to predict on. In each iteration of cross-validation, 10% of protein chains were selected for the test set, whereas the remaining 90% were used for training. Support vector machines (SVMs) and K-nearest neighbor (KNN) classifiers were used to construct all predictors and perform comparative evaluation. For support vector machines, we used SVM^{light} with the default value for the capacity parameter (Joachims, 2002). For the KNN classifiers, the best value for K was determined by varying it in the range $\{5, 10, \ldots, 35\}$.

We estimated the area under the ROC curve (AUC), which plots the true positive rate (sensitivity, *sn*) as a function of false positive rate (1 - specificity, $1-sp$). The area under the ROC curve corresponds to the probability that a classifier will correctly discriminate between one randomly selected positive data point and one randomly selected negative data point (Hanley & McNeil, 1982). Statistical significance of the performance differences between a pair of algorithms was formulated as one-sided hypothesis testing and the P-values were estimated using bootstrapping.

3.4 Performance effect of inexact matches

The performance effect of inexact matches on classification in undirected graphs is shown in Table 4. As expected, we observe that flexibility in graphlet matching plays a significant role in improving performance. In isolation, the label and/or edge mismatch kernels improved accuracy on all data sets; in some cases by 15 percentage points ($n = 5$; Cat, Zn; Table 4). Generally, improvements were achieved for larger graphlet sizes. This is expected because inexact matches in small graphlets contribute a large relative change. Tables 5–7 show the performance when all graphlet sizes were combined (Section 3.5). The performance effects of inexact matches in directed graphs are shown in Table 8. Here, the best performance on Blogs and Tweets data was again achieved by edit distance kernels, but with smaller improvements compared to those observed on biological data. These improvements, however, were similar to the ones observed when Blogs and Tweets data were converted to undirected graphs.

3.5 Overall classification performance

The performance of edit distance kernels was evaluated against the standard graphlet kernel and two random walk kernels on each data set. Tables 5 and 6 summarize the AUC estimates for each method and data set, while Figure 4(a)–(c) shows ROC curves for one representative data set from each group of problems. We also compare the performance of edit distance kernels against three representative computational biology tools for the protein structure data (Table 7). Results for Blogs and Tweets data using directed graphs are reported in Table 8.

Although performance accuracies varied among data sets and methods, edit distance kernels outperformed both graphlet kernels and random walk kernels. Edit distance kernels achieved the highest AUCs on all nine data sets with SVM classifiers (Table 5) and eight data sets with KNN classifiers (Table 6). Relative to the standard graphlet kernels, an increase in performance was observed in all nine cases with both SVMs and KNNs. To determine if the performance of edit distance kernels was statistically significant compared to standard graphlet kernels on each data set, we performed one-sided hypothesis testing with the following null and alternative hypotheses: H_0 : two kernel methods have the same performance, and H_1 : edit distance kernel outperforms the graphlet kernel. Bootstrapping was used to obtain the empirical P-values and the Bonferroni correction was used to determine significance thresholds. In each data set, six possible edit distance kernels were compared to a single graphlet kernel; thus, we set $\alpha = \frac{0.05}{6} = 0.0083$. Note that this threshold was somewhat conservative,

Table 4. *Area under the ROC curve estimates for edit distance kernels. The highest performance for each data set is shown in boldface. Standard graphlet kernel is shown using $m = 0$. For each n we allowed m edit operations. The cases where only one type of edit operation was allowed are denoted using words 'label' and 'edge', respectively. Kernel functions for $n = 5$ and $m = 2$ edit operations for functional residue prediction were approximated by using the label-substitution and edge-indel kernel values.*

n	*m*	**Cat**	**Phos**	**Zn**	**DNA**	**Can**	**Met**	**Met/ Can**	**Blogs**	**Tweets**
2	0	0.843	0.578	0.780	0.694	0.670	0.669	0.756	0.912	0.914
	1 (label)	0.835	0.564	0.757	0.715	0.670	0.667	0.763	0.911	0.921
3	0	0.836	0.652	0.784	0.690	0.680	0.689	0.770	0.963	0.978
	1 (label)	0.853	0.617	**0.795**	0.721	0.686	0.688	0.784	0.949	0.979
	1 (edge)	0.842	0.649	0.782	0.701	0.675	0.685	0.771	0.965	0.977
	1 (operation)	0.849	0.616	0.792	0.723	0.684	0.683	0.785	0.949	0.979
	2 (label)	0.852	0.598	0.775	0.733	0.683	0.681	0.780	0.937	0.978
	2 (edge)	0.843	0.647	0.778	0.701	0.672	0.676	0.773	0.962	0.967
	2 (operations)	0.845	0.591	0.783	**0.737**	0.680	0.690	0.782	0.938	0.978
4	0	0.759	0.711	0.759	0.636	0.687	0.678	0.771	0.969	0.984
	1 (label)	0.842	0.680	0.782	0.715	0.695	0.682	0.786	0.967	**0.986**
	1 (edge)	0.827	0.703	0.775	0.678	0.681	0.677	0.770	0.972	0.985
	1 (operation)	0.838	0.682	0.783	0.725	0.684	0.685	0.789	0.969	0.985
	2 (label)	**0.857**	0.652	0.790	0.695	0.697	0.693	0.795	0.962	0.985
	2 (edge)	0.836	0.687	0.777	0.691	0.683	0.673	0.771	**0.974**	0.984
	2 (operations)	0.857	0.655	0.788	0.705	0.695	0.691	0.787	0.963	0.985
5	0	0.674	0.682	0.614	0.502	0.682	0.685	0.775	0.963	0.985
	1 (label)	0.832	**0.720**	0.774	0.536	0.693	0.688	0.789	0.947	0.984
	1 (edge)	0.709	0.694	0.692	0.505	0.686	0.689	0.776	0.965	0.985
	1 (operation)	0.832	0.711	0.778	0.515	0.689	0.687	0.788	0.962	0.985
	2 (label)	0.831	0.703	0.772	0.523	0.697	**0.697**	**0.796**	0.837	0.984
	2 (edge)	0.755	0.704	0.755	0.514	0.685	0.687	0.775	0.969	0.985
	2 (operations)	0.835	0.696	0.776	0.513	**0.699**	0.692	0.788	0.971	0.985

Table 5. *Area under the ROC curve estimates for each method over nine data sets using SVM classifiers. The highest performance for each data set is shown in boldface. Statistically significant AUC values ($P < 0.0083$) between two types of graphlet kernels are marked by an asterisk.*

Method/Dataset	**Cat**	**Phos**	**Zn**	**DNA**	**Can**	**Met**	**Met/ Can**	**Blogs**	**Tweets**
Random walk	0.833	0.574	0.766	0.668	0.600	0.535	0.704	0.705	0.949
Cumulative random walk	0.837	0.606	0.758	0.707	0.548	0.582	0.682	0.775	0.854
Graphlet kernel	0.841	0.693	0.783	0.689	0.668	0.685	0.775	0.968	0.984
Edit distance kernel	**0.861***	**0.724***	**0.795***	**0.727***	**0.689***	**0.699**	**0.800***	**0.973***	**0.986***

because several different edit distance kernels could achieve improved performance over the graphlet kernels. Interestingly, even without parameter optimization, edit distance kernels reached competitive performance against domain-specific methods for predicting functional residues from protein structures. In this group, FEATURE is the only method that does not incorporate evolutionary conservation. On the

Table 6. *Area under the ROC curve estimates for each method over nine data sets using KNN classifiers ($K = 15$). The highest performance for each data set is shown in boldface. Statistically significant AUC values ($P < 0.0083$) between two types of graphlet kernels are marked by an asterisk.*

Method/Dataset	**Cat**	**Phos**	**Zn**	**DNA**	**Can**	**Met**	**Met/Can**	**Blogs**	**Tweets**
Random walk	0.816	0.582	0.761	0.681	0.674	0.668	0.742	0.849	0.968
Cumulative random walk	0.821	0.603	0.776	0.679	0.648	0.609	0.710	0.955	**0.989**
Graphlet kernel	0.831	0.588	0.769	0.693	0.678	0.673	0.763	0.960	0.976
Edit distance kernel	**0.844***	**0.612**	**0.784***	**0.699**	**0.692**	**0.678**	**0.775**	**0.965**	0.979*

Table 7. *Area under the ROC curve estimates for each method over protein structure data sets. The highest performance for each data set is shown in boldface. The performance of the non graph-based methods were reproduced from (Xin & Radivojac 2011).*

Method/Dataset	**Cat**	**Phos**	**Zn**	**DNA**
FEATURE	0.754	0.620	0.767	**0.824**
Gutteridge et al.	0.814	0.559	0.713	0.713
Structure kernel	0.839	0.711	**0.808**	0.800
Graphlet kernel	0.841	0.693	0.783	0.689
Edit distance kernel	**0.861**	**0.724**	0.795	0.727

other hand, it exploits information available for atoms and chemical groups which explains its superior accuracy on DNA-binding sites. The performance of edit distance kernels is consistent with previous biological observations. For example, catalytic residues and zinc-binding sites are highly conserved and spatially compact; hence, a decrease in overall performance was observed as n increased, as shown in Table 4. Furthermore, phosphorylation sites are frequently located in fast-evolving loop regions. Inexact matches were therefore expected to capture high substitution rates and conformational flexibility in such graphs.

Figure 5 illustrates the benefits of edit distance kernels over traditional graphlet kernels. We show local structural neighborhoods of two proteins that share 22% global sequence identity: chain A of 1bf2 and chain A of 1qho. Both enzymes contain the same catalytic triad (D, E, D), but their structural neighborhoods are otherwise different. Using a 1-nearest neighbor classifier with the standard graphlet kernel to predict on residue D374 of 1bf2 results in misclasification. On the other hand, an edit distance kernel correctly classifies D374 as a catalytic residue using position D227 from 1qho as the nearest neighbor. Further examination reveals that the rooted neighborhood of 1bf2 centered at D374 is more similar to a different protein (i.e. chain A of 1pj5) because only exact matches are considered. In contrast, by allowing mismatches in vertex labels, we were able to recover the shared catalytic triad between 1bf2 and 1qho.

We also used the functional residue data sets to evaluate the influence of doubling the alphabet size on classification accuracy. However, except for catalytic residue prediction (Figure 4(d)), the difference in performance was only moderate (Table 9). In fact, the performance on the Phos data set was slightly better without

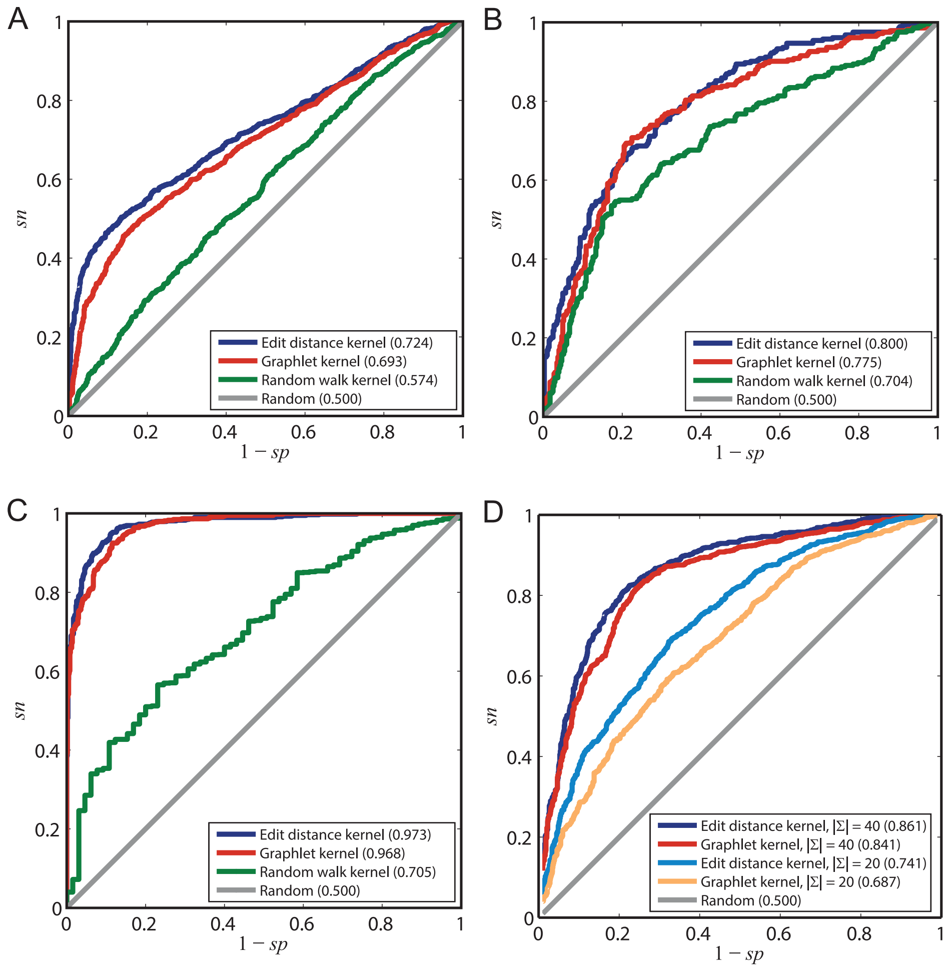

Fig. 4. Comparisons of ROC curves between different kernel methods for four representative data sets. (A) Phos, (B) Met/Can, (C) Blogs, and (D) Cat. AUC values are shown in parentheses for each method. In panel D, we show the influence of increased alphabet size on graphlet kernels and edit distance kernels. (color online)

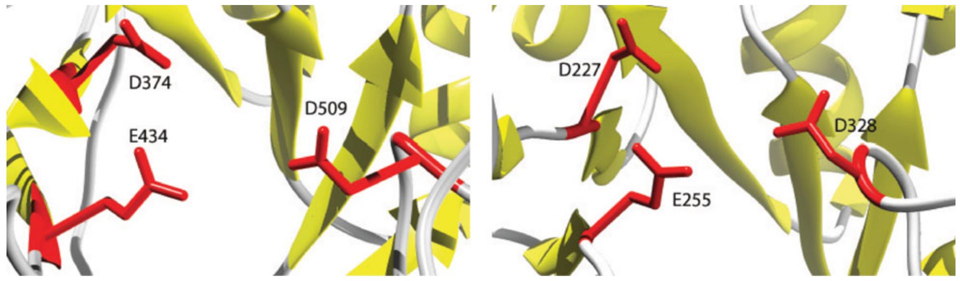

Fig. 5. Structural neighborhoods of a catalytic triad (D, E, D) corresponding to two different protein structures with 22% sequence identity. Left: ISOA protein (PDB id: 1bf2). Right: AMYM protein (PDB id: 1qho). Edit distance kernel based on a substitution of a single vertex label recovers the catalytic triad (D, E, D) between 1bf2 and 1qho which is buried with standard graphlet kernel. (color online)

the expanded alphabet, which reflects an ongoing debate about the evolutionary conservation of phosphorylation sites (Landry *et al.*, 2009; Gray & Kumar, 2011). These results combine to suggest that the 6Å C_α-C_α threshold that was used for constructing protein contact graphs and the choice of the vertex alphabet require separate optimization (e.g., through a validation set) on each data set.

Edit distance kernels also improved performance on the PPI data sets as well as Blogs and Tweets data sets, likely by reducing the impact of noisy vertex labels. Interestingly, our results on Blogs and Tweets data also support findings stating large differences in connectivity patterns between liberal and conservative blogs (Adamic & Glance, 2005) and tweets (Conover *et al.*, 2011). Such differences resulted in very high classification accuracies. The influence of domain knowledge on classification accuracy was significant on the PPI data sets as the performance of all methods increased between six and ten percentage points. The influence of the vertex alphabet on Blogs and Tweets data was even more significant (Table 9).

Finally, the use of edit distance kernels on directed graphs did not yield further accuracy improvements to their undirected counterparts. However, directed graphs provided computational speed-up because they were computed for $N = 3$ as opposed to $N = 5$ used for undirected graphs.

4 Conclusion

In this work, we introduce a framework for the construction of flexible graphlet kernels for comparing vertex neighborhoods in sparse graphs. Flexibility is achieved by allowing approximate graphlet matching, according to the edit distance among them. These new kernels performed competitively with state-of-the-art approaches in the task of predicting functional residues from protein structures, and outperformed random walk kernels on all data sets. Furthermore, by utilizing vertex-labeling we showed a simple way to integrate domain knowledge into classification.

Edit distance kernels can be even further generalized to incorporate similarity functions on the elements of Σ. This can be illustrated in the context of functional residue prediction. In the course of evolution, neutral mutations in DNA are more likely to result in amino acid substitutions between residues with similar physicochemical properties. For example, a positively charged residue is more likely to be replaced by a positively charged, rather than a negatively charged residue. In this work we utilized well-known substitution matrices from computational biology to construct weighted edit paths via an exponential transformation $\exp\{S(A,B)/\max\{S(A,A),S(B,B)\}-1\}$, where $S(A,B)$ is the entry of the BLOSUM62 matrix (Henikoff & Henikoff, 1992) corresponding to amino acids A and B. While these results provided some improvement for $|\Sigma| = 20$ (data not shown), their performance was generally inferior to $|\Sigma| = 40$, a case for which substitution matrices have not been developed. The construction of appropriate substitution matrices for large alphabets has not been explored in this study; however, it may present an attractive and practical approach in the future.

Edit distance graphlet kernels can be seen as a special case of a generalized similarity function of the type $k(\mathbf{u},\mathbf{v}) = \mathbf{u}^T\mathbf{S}^T\mathbf{S}\mathbf{v} = \mathbf{u}^T\mathbf{Q}\mathbf{v}$, where $\mathbf{u}$ and $\mathbf{v}$ are column vectors of counts and $\mathbf{Q}$ is a symmetric positive semi-definite matrix ($\mathbf{Q} = \mathbf{I}$ provides the standard kernel function). Each element $S(i,j)$ of $\mathbf{S}$ is set to 1 if graphlets n_i

Table 8. *Area under the ROC curve estimates for each method on the Blogs and Tweets data using directed graphs. The highest performance for each data set is shown in boldface. The upper part of the table shows kernel performance, using SVMs, for graphlet kernels of particular size. The lower part of the table shows combined kernel performance for SVM and KNN classifiers. Statistically significant AUC values ($P < 0.0083$) between two types of combined graphlet kernels are marked by an asterisk. Both random walk kernels resulted in near-random AUCs, which likely indicates that 100,000 steps that were applied to provide the kernel values were insufficient.*

	Method/Dataset	m	Blogs	Tweets
	2-graphlets	0	0.930	0.920
		1 (label)	0.925	0.925
		1 (edge)	0.925	0.920
		1 (operation)	0.900	0.925
		2 (edge)	0.912	0.914
		2 (operations)	0.901	0.925
	3-graphlets	0	0.964	0.979
		1 (label)	0.965	**0.981**
		1 (edge)	0.967	0.979
		1 (operation)	**0.973**	**0.981**
		2 (label)	0.961	0.980
		2 (edge)	0.969	0.979
		2 (operations)	0.969	0.980
SVM	Graphlet kernel	0	0.963	0.981
	Label substitution	1	0.965	**0.982**
		2	0.962	0.980
	Edge indel	1	0.966	0.980
		2	0.970	0.979
	Edit distance	1	**0.973***	**0.982***
		2	0.969	**0.982**
KNN	Graphlet kernel	0	0.947	0.972
	Label substitution	1	0.933	0.972
		2	0.926	0.967
	Edge indel	1	0.955	**0.974***
		2	**0.960***	**0.974**
	Edit distance	1	0.937	0.973
		2	0.929	0.969

and n_j are within edit distance of m; otherwise, $S(i, j) = 0$. For non-sparse matrices, computing such a kernel may not be generally feasible because of the exceedingly high dimensionality of the count vectors and similarity matrices in many applications (Table 2). Therefore, edit distance graphlet kernels can be seen as practical solutions that approximate full matrix representations yet still efficiently operate in spaces of extremely high dimensionality. It is worth mentioning that a significantly more efficient solution exists when the similarity between $\mathbf{u}$ and $\mathbf{v}$ is computed as $\mathbf{u}^T\mathbf{S}\mathbf{v}$; however, unlike $\mathbf{Q}$, matrix $\mathbf{S}$ is not always positive semi-definite (unless the magnitude of pseudo counts is small) and would require a test for positive semi-definiteness before implementation.

Although the performance accuracy experiments were generally favorable, an important drawback of edit distance kernels is their computational complexity. Specifically, the new kernels, at least in their basic form, are not well suited

Table 9. *The influence of domain knowledge and auxiliary data on the performance of graphlet and edit distance kernels. The numbers show AUC estimates for over nine data sets using SVM and KNN classifiers. In the experiments without domain information, the vertex alphabet sizes were set to* $|\Sigma| = 20$ *for protein structure data and* $|\Sigma| = 1$ *for the remaining data sets. In the experiments with domain information the alphabet sizes were set to* $|\Sigma| = 40$ *for protein structure data,* $|\Sigma| = 16$ *for the PPI data sets,* $|\Sigma| = 7$ *for Blogs, and* $|\Sigma| = 16$ *for Tweets. Statistical significance was calculated between graphlet and edit distance kernels for each learning scenario.*

Method/Dataset	**Cat**	**Phos**	**Zn**	**DNA**	**Can**	**Met**	**Met/ Can**	**Blogs**	**Tweets**
				SVM, without domain information					
Graphlet kernel	0.687	0.730	0.759	0.684	0.630	0.589	0.681	0.758	0.713
Edit distance kernel	0.741*	**0.743**	0.780*	0.718*	0.632	0.604*	0.704*	0.749	0.717*
				SVM, with domain information					
Graphlet kernel	0.841	0.693	0.783	0.689	0.668	0.685	0.775	0.968	0.984
Edit distance kernel	**0.861***	0.724*	**0.795***	**0.727***	0.689*	**0.699**	**0.800***	**0.973***	**0.986***
				KNN, without domain information					
Graphlet kernel	0.655	0.606	0.746	0.676	0.631	0.578	0.688	0.550	0.813
Edit distance kernel	0.668	0.623	0.759*	0.677	0.624	0.590	0.693	0.537	0.815
				KNN, with domain information					
Graphlet kernel	0.831	0.588	0.769	0.693	0.678	0.673	0.763	0.960	0.976
Edit distance kernel	0.844*	0.612	0.784*	0.699	**0.692**	0.678	0.775*	0.965	0.979*

to comparisons of dense graphs and graphs with very large vertex and/or edge alphabets. Our data sets contained a solid range of such situations and provided evidence that, both in terms of accuracy and speed, edit distance kernels seem particularly well suited to problems involving protein structures. For example, a structural neighborhood of depth 4 from the root typically contained about 40 nodes and, thus, neither counting of graphlets with up to 5 nodes nor generating pseudo counts presented a significant computational challenge. On the other end of the spectrum was the Blogs data set that contained more than 1200 relatively densely connected nodes in a typical neighborhood. Here, the running time for the edit distance kernels was significant: execution of a single cross-validation experiment took approximately two weeks on our desktop computers. Although the density of the Tweets data set was significantly lower than the one for Blogs, its run time was even longer, in part due to an order of magnitude larger data set size. As a result, biological networks, especially protein structures, appear to be better application domains for edit distance kernels than social networks.

Finally, edit distance kernels are simply similarity functions between vertices in a graph. Thus, they are applicable beyond vertex classification and supervised learning; for example in kernel-based graph clustering or approximate motif discovery. Another potential use of these kernels is the pre-processing of graph data through graph re-wiring for the subsequent application of probabilistic graphical models, e.g. Markov Random Fields. As such our approach may increase the applicability of probabilistic graphical models to disconnected graphs as well as graphs where the proximity of vertices poorly correlates with the similarity between their class labels.

Acknowledgments

We thank Prof. Esfandiar Haghverdi for important discussions regarding the theory of graph enumeration, Prof. Filippo Menczer for providing us with the Twitter data, Dr. Vladimir Vacic whose original C/C++ code was extended in this work, and Wyatt Clark and Earl Bellinger for proofreading the manuscript. We also thank anonymous reviewers for their comments that led to an improved quality of this study. This work was supported by the Ford Foundation pre-doctoral fellowship (JL-M) and the National Science Foundation grant DBI-0644017 (PR).

References

Adamic, L. A., & Glance, N. (2005). The political blogosphere and the 2004 U.S. election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery* (pp. 36–43). LinkKDD '05.

Aral, S., & Walker, D. (2012). Identifying influential and susceptible members of social networks. *Science*, **337**(6092), 337–341.

Bondy, J. A., & Hemminger, R. L. (1977). Graph reconstruction-a survey. *Journal of Graph Theory*, **1**(3), 227–268.

Borgs, C., Chayes, J., Lovász, L., Sós, V. T., & Vesztergombi, K. (2006). Counting graph homomorphisms. In *Topics discrete math* (pp. 315–371). Algorithms and Combinatorics, vol. 26. Berlin Heidelberg: Springer.

Borgwardt, K. M., & Kriegel, H. P. (2005). Shortest-path kernels on graphs. In *Proceedings of the 5th International Conference on Data Mining* (pp. 74–81). ICDM '05.

Borgwardt, K. M., Ong, C. S., Schonauer, S., Vishwanathan, S. V., Smola, A. J., & Kriegel, H. P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, **21**(Suppl 1), i47–i56.

Clark, W. T., & Radivojac, P. (2011). Analysis of protein function and its prediction from amino acid sequence. *Proteins*, **79**(7), 2086–2096.

Conover, M., Ratkiewicz, J., Francisco, M., Goncalves, B., Flammini, A., & Menczer, F. (2011). Political polarization on twitter. In *Proceedings of the 5th International Conference on Weblogs and Social Media* (pp. 89–96). ICWSM '11.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, **486**(3–5), 75–174.

Fouss, F., Pirotte, A., Renders, J.-M., & Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *Transactions on Knowledge and Data Engineering*, **19**(3), 355–369.

Frank, O. (1988). The triad count statistics. *Journal on Discrete Mathematics*, **72**, 141–149.

Gärtner, T., Flach, P., & Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. *Learning Theory and Kernel Machines*, 129–143.

Gönen, M., & Alpaydin, E. (2011). Multiple kernel learning algorithms. *Journal of Machine Learning Research*, **12**, 2211–2268.

Gray, V. E., & Kumar, S. (2011). Rampant purifying selection conserves positions with posttranslational modifications in human proteins. *Molecular Biology and Evolution*, **28**(5), 1565–1568.

Gutteridge, A., Bartlett, G. J., & Thornton, J. M. (2003). Using a neural network and spatial clustering to predict the location of active sites in enzymes. *Journal of Molecular Biology*, **330**(4), 719–734.

Halperin, I., Glazer, D., Wu, S., & Altman, R. (2008). The FEATURE framework for protein function annotation: modeling new functions, improving performance, and extending to novel applications. *BMC Genomics*, **9**(Suppl 2), S2.

Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, **143**(1), 29–36.

Harary, F., & Palmer, E. M. (1973). *Graphical enumeration.* Academic Press.

Haussler, D. (1999). Convolution kernels on discrete structures. *Technical Report UCSCCRL-99-10, University of California at Santa Cruz.*

Henikoff, S, & Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, **89**(22), 10915–10919.

Hido, S., & Kashima, H. (2009). A linear-time graph kernel. In *Proceedings of the 9th International Conference on Data Mining* (pp. 179–188). ICDM '09.

Horváth, T., Gärtner, T., & Wrobel, S. (2004). Cyclic pattern kernels for predictive graph mining. In *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining* (pp. 158–167). KDD '04.

Jiang, C., Coenen, F., & Zito, M. (2013). A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, **28**(2), 75–105.

Joachims, T. (2002). *Learning to classify text using support vector machines: methods, theory, and algorithms.* Kluwer Academic Publishers.

Kashima, H., Tsuda, K., & Inokuchi, A. (2003). Marginalized kernels between labeled graphs. In *Proceedings of the 20th International Conference on Machine Learning* (pp. 321–328). ICML '03.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques.* MIT Press.

Kondor, R. I., & Lafferty, J. D. (2002). Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th International Conference on Machine Learning* (pp. 315–322). ICML '02.

Kuang, R., Ie, E., Wang, K., Wang, K., Siddiqi, M., Freund, Y., & Leslie, C. (2005). Profile-based string kernels for remote homology detection and motif extraction. *J Bioinform Comput Biol*, **3**, 527–550.

Landry, C. R., Levy, E. D., & Michnick, S. W. (2009). Low functional constraints on phosphoproteomes. *Trends in Genetics*, **25**(5), 193–197.

Leskovec, J., Adamic, L. A., & Huberman, B. A. (2007). The dynamics of viral marketing. *ACM Transactions on the Web* , **1**(1).

Leslie, C., Eskin, E., & Noble, W. S. (2002). The spectrum kernel: A string kernel for SVM protein classification. *Pac Symp Biocomput*, 564–575.

Leslie, C., & Kuang, R. (2004). Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research* , **5**, 1435–1455.

Liben-Nowell, D., & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, **58**(7), 1019–1031.

Lovász, L. (1967). Operations with structures. *Acta Math Hung*, **18**(3-4).

Menchetti, S., Costa, F., & Frasconi, P. (2005). Weighted decomposition kernels. In *Proceedings of the 22nd International Conference on Machine Learning* (pp. 585–592). ICML '05.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., & Alon, U. (2002). Network motifs: simple building blocks of complex networks. *Science*, **298**, 824–827.

Moreau, Y., & Tranchevent, L.C. (2012). Computational tools for prioritizing candidate genes: Boosting disease gene discovery. *Nature Reviews Genetics*, **13**(8), 523–536.

Nabieva, E., Jim, K., Agarwal, A., Chazelle, B., & Singh, M. (2005). Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps. *Bioinformatics*, **21**(Suppl 1), i302–i310.

Pei, J., & Grishin, N. V. (2001). AL2CO: calculation of positional conservation in a protein sequence alignment. *Bioinformatics*, **17**(8), 700–712.

Pólya, G. (1937). Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen. *Acta Mathematica* , **68**, 145–254.

Przulj, N. (2007). Biological network comparison using graphlet degree distribution. *Bioinformatics*, **23**(2), e177–e183.

Przulj, N., Corneil, D. G., & Jurisica, I. (2004). Modeling interactome: scale-free or geometric? *Bioinformatics*, **20**(18), 3508–3515.

Radivojac, P., Peng, K., Clark, W. T., Peters, B. J., Mohan, A., Boyle, S. M., & Mooney, S. D. (2008). An integrated approach to inferring gene-disease associations in humans. *Proteins*, **72**(3), 1030–1037.

Ralaivola, L., Swamidass, S. J., Saigo, H., & Baldi, P. (2005). Graph kernels for chemical informatics. *Neural Networks*, **18**(8), 1093–1110.

Ramon, J., & Gärtner, T. (2003). Expressivity versus efficiency of graph kernels. In *Proceedings of the 1st International Workshop on Mining Graphs, Trees and Sequences*, pp. 65–74.

Rieck, K., & Laskov, P. (2008). Linear-time computation of similarity measures for sequential data. *Journal of Machine Learning Research*, **9**, 23–48.

Sharan, R., Ulitsky, I., & Shamir, R. (2007). Network-based prediction of protein function. *Mol Syst Biol*, **3**, 88.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. New York, NY, USA: Cambridge University Press.

Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K., & Borgwardt, K. M. (2011). Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research* , **12**, 2539–2561.

Shervashidze, N., Vishwanathan, S. V. N., Petri, T. H., Mehlhorn, K., & Borgwardt, K. M. (2009). Efficient graphlet kernels for large graph comparison. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pp. 488–495. AISTATS '09.

Tsuda, K., & Saigo, H. (2010). Graph classification. In *Managing and mining graph data* (pp. 337–363). Advances in Database Systems, vol. 40. Springer.

Vacic, V., Iakoucheva, L. M., Lonardi, S., & Radivojac, P. (2010). Graphlet kernels for prediction of functional residues in protein structures. *Journal of Computational Biology*, **17**(1), 55–72.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R. I., & Borgwardt, K. M. (2010). Graph kernels. *Journal of Machine Learning Research* , **11**, 1201–1242.

Wasserman, S., & Faust, K. (1994). *Social network analysis: Methods and applications*. Cambridge University Press.

Xin, F., Myers, S., Li, Y. F., Cooper, D. N., Mooney, S. D., & Radivojac, P. (2010). Structure-based kernels for the prediction of catalytic residues and their involvement in human inherited disease. *Bioinformatics*, **26**(16), 1975–1982.

Xin, F., & Radivojac, P. (2011). Computational methods for identification of functional residues in protein structures. *Current Protein and Peptide Science*, **12**, 456–469.

Xu, J., & Li, Y. (2006). Discovering disease-genes by topological features in human protein-protein interaction network. *Bioinformatics*, **22**(22), 2800–2805.

Zhu, X., & Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. *Technical Report CMU-CALD-02-107, Carnegie Mellon University*.