Efficient graphlet kernels for large graph comparison

Nino Shervashidze
MPI for Biological Cybernetics
MPI for Developmental Biology
Tübingen, Germany

S.V.N. Vishwanathan
Department of Statistics
Purdue University
West Lafayette, IN, USA

Tobias H. Petri
Institute for Computer Science
LMU München
München, Germany

Kurt Mehlhorn
MPI for Informatics
Saarbrücken, Germany

Karsten M. Borgwardt
MPI for Biological Cybernetics
MPI for Developmental Biology
Tübingen, Germany

Abstract

State-of-the-art graph kernels do not scale to large graphs with hundreds of nodes and thousands of edges. In this article we propose to compare graphs by counting *graphlets*, *i.e.,* subgraphs with k nodes where $k \in \{3, 4, 5\}$. Exhaustive enumeration of all graphlets being prohibitively expensive, we introduce two theoretically grounded speedup schemes, one based on sampling and the second one specifically designed for bounded degree graphs. In our experimental evaluation, our novel kernels allow us to efficiently compare large graphs that cannot be tackled by existing graph kernels.

1 Introduction

Graph comparison is an important problem in application areas as disparate as bioinformatics, chemoinformatics, chemistry, sociology and telecommunication. Broadly speaking, existing graph comparison algorithms may be classified into three categories: *set* based, *frequent subgraph* based, and *kernel* based. Set based approaches represent a graph as a set of edges, or as a set of nodes, or both. Graphs are then compared by measuring similarity between pairs of edges or pairs of nodes between two graphs. While these approaches are computationally feasible – typically scaling linearly or quadratically in the number of nodes and edges – they are rather naive, as they neglect the structure of the graphs, *i.e.,* their topology.

Appearing in Proceedings of the 12^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, Florida, USA. Volume 5 of JMLR: W&CP 5.

Frequent subgraph mining algorithms, on the other hand, aim to detect subgraphs that are frequent in a given dataset of graphs. Afterwards, feature selection is applied to select the most discriminative subgraphs. Efficient methods such as gSpan (Yan & Han, 2003) have been developed for this task, which use elegant data structures and efficient branch-and-bound search strategies. Unfortunately, their computational complexity scales exponentially with graph size in the worst case.

Graph kernels represent an attractive middle ground. They respect and exploit graph topology, but restrict themselves to comparing substructures of graphs that are computable in polynomial time. Many different graph kernels have been defined, which focus on different types of substructures in graphs, such as random walks (Gärtner et al., 2003; Kashima et al., 2004), shortest paths (Borgwardt & Kriegel, 2005), subtrees (Ramon & Gärtner, 2003), and cycles (Horvath et al., 2004). Several studies have recently shown that these graph kernels can achieve results competitive with the state of the art on benchmark datasets from bioinformatics and chemistry (Borgwardt et al., 2005; Ralaivola et al., 2005).

When using graph kernels, a practitioner is faced with three questions: Out of the numerous variants, which graph kernel should I choose for a particular application? Does this kernel capture graph similarity better than others? Is it cheap to compute? Unfortunately, these questions are far from being answered. Almost all existing approaches are ad-hoc and are generally motivated by runtime considerations. To make matters worse, there is no theoretical justification on why certain types of subgraphs are better than the others. The other extreme of comparing all possible subgraphs has been shown to be NP-hard (Gärtner et al., 2003), thus making it practically infeasible. Even the fastest polynomial-runtime graph kernels scale as

$O(n^3)$ (Vishwanathan et al., 2007) or $O(n^4)$ (Borgwardt & Kriegel, 2005). These difficulties limit the practical applicability of these kernels to graphs with a few hundreds of nodes.

In essence, this problem could be solved by a rich enough, still efficiently computable representation that adequately captures the topology of the input graphs. Drawing analogy from probability theory, we seek to efficiently compute the sufficient statistics of a graph.

Paper contributions In this work we define a graph kernel based on the distribution of subgraphs of size k, $k \in \{3, 4, 5\}$, which we refer to as graphlets (Section 2).

This distribution, we argue, is similar to a sufficient statistic of the graph, especially when the graph is large. Exhaustive enumeration of these subgraphs is prohibitively expensive, scaling as $O(n^k)$ where n is the number of nodes in the graph and $k \in \{3, 4, 5\}$ the size of the subgraphs. We propose two theoretically grounded speedups. First, we show that sampling a fixed number of graphlets suffices to bound the deviation of the empirical estimates of the graphlet distribution from the true distribution. Second, we show that for graphs of degree bounded by d, the exact number of all graphlets of size k can be determined in time $O(nd^{k-1})$. Large real world graphs are often sparse with $d \ll n$. Finally, we experimentally show that our novel sampling scheme allows us to compute graph kernels on graphs of sizes that are beyond the scope of the state-of-the-art (Section 5).

It is clear that isomorphic graphs have the same graphlet distributions. But the question whether in a general setting the equality of graphlet distributions implies isomorphism is still unanswered. It is closely related to the graph reconstruction problem, which is a classic open problem in graph theory. Nonetheless, it is interesting whether similarity of graphlet distributions leads to a meaningful measure of similarity between corresponding graphs. We provide an experimental answer to this question in Section 5.

2 Graphlet kernels

2.1 Notation

Before we define our novel kernel, we here summarize key concepts and clarify our notation.

By definition, a *graph* is a pair $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is an ordered set of n *vertices*, and $E \subseteq V \times V$ is the set of *edges*. The size of a graph is the size of the set V, here n. All edges of the form $(v, v_i) \in E$ are said to be *incident* on vertex v. Given $G = (V, E)$ and $H = (V_H, E_H)$, we say H is a *subgraph* of G (or H has an embedding in G), denoted by $H \sqsubseteq G$, iff there is an injective mapping $\alpha : V_H \to V$ such that $(v, w) \in E_H$ iff $(\alpha(v), \alpha(w)) \in E$. Furthermore, $\#(H \sqsubseteq G)$ denotes the number of embeddings of H in G.

A graph is called *undirected* if $(v_i, v_j) \in E$ iff $(v_j, v_i) \in E$; otherwise it is referred to as *directed*. Although many of our techniques are applicable to both directed and undirected graphs, for ease of exposition, we will exclusively deal with undirected graphs in the remainder of this paper.

Any edge (v_i, v_i) is called a *self loop*. In a general graph two vertices v_i and v_j may be connected by more than one edge. A *simple graph* is a graph with no self loops or multiple edges. Here we always work with simple, connected graphs.

A simple graph can equivalently be represented by an *adjacency matrix* A of size $n \times n$. The (i, j)-th entry of A is 1 if an edge (v_i, v_j) exists and zero otherwise. The adjacency matrix of an undirected graph is symmetric.

Two graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* (denoted by $G \cong G'$) if there exists a bijective mapping $g : V \to V'$ (called the isomorphism function) such that $(v_i, v_j) \in E$ iff $(g(v_i), g(v_j)) \in E'$.

2.2 The graphlet kernel

Let $\mathcal{G} = \{graphlet(1), \ldots, graphlet(N_k)\}$ be the set of size-k graphlets and G be a graph of size n. Define a vector f_G of length N_k whose i-th component corresponds to the frequency of occurrence of $graphlet(i)$ in G, $\#(graphlet(i) \sqsubseteq G)$. We will call f_G the k-spectrum of G. This statistic is the foundation of our novel graph kernel.

Definition 1 (Graphlet kernel) *Given two graphs G and G' of size $n \geq k$, the graphlet kernel k_g is defined as*

$$k_g(G, G') := f_G^\top f_{G'}. \tag{1}$$

As our goal is to develop scalable graph kernels, we study graphlet kernels based on the 3-, 4- and 5-spectra of graphs here. In order to account for differences in the sizes of the graphs, which can greatly skew the frequency counts f_G, we normalize the counts to probability vectors:

$$D_G = \frac{1}{\#all\ graphlets\ in\ G}\ f_G,$$

and work with the following normalized variant of (1):

$$k_g(G, G') = D_G^\top D_{G'}. \tag{2}$$

Since there are $\binom{n}{k}$ size-k subgraphs in a graph, computing D_G for each graph of size n requires $O(n^k)$

effort. Once all the D_G vectors are computed, then computing (2) requires essentially $O(1)$ work. In the sequel we will show how the $O(n^k)$ pre-processing step can be made efficient.

Clearly, if $G \cong G'$, then $f_G = f_{G'}$. But is the reverse true? It has been shown that when $n = k + 1$ and $n \leq 11$, equality of k-spectra implies isomorphism. For $n > 11$, it is still a conjecture whether a graph of size n can be reconstructed from its subgraphs of size $n - 1$.

2.3 Reconstruction of graphs

In fact, this problem of graph reconstruction is a classic open problem in graph theory (Kelley, 1957; Hemminger, 1969): Let $G = (V, E)$ be a undirected graph of size n. For each $v \in V$, let G_v denote a node-deleted subgraph of G, *i.e.,* the graph obtained by deleting node v and all the edges incident on it from G. Can G be reconstructed, up to an isomorphism, from its set of node-deleted subgraphs $\{G_v\}_{v \in V}$?

Kelley (1957) proved the following theorem: Let $G = (V, E)$ and $G' = (V', E')$ be trees and $g : V \to V'$ be an isomorphism function such that G_v is isomorphic to $G'_{g(v)}$ for all $v \in V$, then G is isomorphic to G'. He conjectured that the following theorem is true for arbitrary graphs:

Theorem 2 (Graph reconstruction conjecture) *Let G and G' be graphs of size greater than 2 and $g : V \to V'$ be an isomorphism function such that G_v is isomorphic to $G'_{g(v)}$ for all $v \in V$. Then G is isomorphic to G'.*

Kelley (1957) verified his conjecture by enumeration of all possible graphs for $2 < n \leq 6$, which was later extended to $2 < n \leq 11$ by (McKay, 1997). Special classes of graphs such as regular graphs, and disconnected graphs have also been shown to be reconstructible (Kelley, 1957). The general case, however, remains a conjecture. It is widely believed though, that if a counterexample to the graph reconstruction problem exists, then it will be of size $n \gg 11$ (McKay, 1997).

2.4 A recursive definition of the graphlet kernel

Here we show that the graphlet kernel (1) can also be defined based on the decomposition of a graph of size n recursively into its subgraphs of size k. This decomposition is similar to the one in the graph reconstruction setting, but here we are interested in the similarity of graphs rather than in their isomorphism. We will refer to subgraphs of size k as k minors, as formalized in the following definition.

Definition 3 (k minors) *Let M be a $n \times n$ matrix. The set of all size-k sub-matrices of M obtained by deleting $n-k$ rows and corresponding columns of M is called the k minors of M. Analogously, given a graph G of size n, the set of all size-k graphs obtained by deleting $n - k$ nodes from G is called the k minors of G.*

We now study some properties of k minors. For simple undirected unweighted graphs, the entries in the upper triangular submatrix of the adjacency matrix completely determine the graph. This submatrix contains $\binom{k}{2}$ entries, each of which could either be 1 or 0 depending on the presence or absence of the corresponding edge. Therefore, there are $2^{\binom{k}{2}}$ different k minors. Following (Przulj, 2007) we refer to them as size k *graphlets*, and denote them as $\mathcal{G} = \{graphlet(1), \ldots, graphlet(2^{\binom{k}{2}})\}$. Modulo isomorphism, there will be only $N_k < 2^{\binom{k}{2}}$ distinct graphlets. This is denoted by a matrix $P \in \{0,1\}^{2^{\binom{k}{2}} \times 2^{\binom{k}{2}}}$ with entries

$$P_{ij} = \begin{cases} 1 & \text{if } graphlet(i) \cong graphlet(j), \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

It is easy to see that the matrix P corresponds to the isomorphism kernel between graphs of size k, hence is a symmetric positive semi-definite (PSD) matrix.

Definition 4 (Recursive graphlet kernel) *Given two graphs G and G' of size $n \geq k$, let $\mathcal{M}$ and $\mathcal{M}'$ denote the set of principal minors of G and G' respectively. The recursive graph kernel, k_n, based on principal minors is defined as*

$$k_n(G, G') = \\ = \begin{cases} \frac{1}{(n-k)^2} \sum_{S \in \mathcal{M}, S' \in \mathcal{M}'} k_{n-1}(S, S') & \text{if } n > k, \\ \delta(G \cong G') & \text{if } n = k \end{cases} \quad (4)$$

where $\delta(G \cong G')$ is 1 if G and G' are isomorphic, and 0 otherwise. The graphlet kernel is defined as $k_g(G, G') := k_n(G, G')$.

Since the above kernel compares the k minors in both G and G', it can be computed non-recursively. We state the following result without proof.

Lemma 5 *Let $\mathcal{M}_k$ and $\mathcal{M}'_k$ denote the set of k minors of G and G' respectively. The graphlet kernel can be computed without recursion via*

$$k_g(G, G') = k_n(G, G') = \sum_{S \in \mathcal{M}_k} \sum_{S' \in \mathcal{M}'_k} \delta(S \cong S'). \quad (5)$$

Equivalently, $k_g(G, G')$ *equals*

$$\sum_{S,S' \in \mathcal{G}} \#(S \sqsubseteq G)\, \#(S' \sqsubseteq G')\, \delta(S \cong S'). \tag{6}$$

Recall that there are $2^{\binom{k}{2}}$ graphlets of size k, and the matrix P encodes isomorphism relations between them. Suppose we define a $2^{\binom{k}{2}}$-dimensional vector f_G whose i-th component corresponds to the frequency of occurrence of *graphlet*(i) in a graph G, then one can rewrite (6) as $k_g(G, G') = f_G^\top P f_{G'}$. As before, we normalize the counts to probability vectors:

$$D_G = \frac{1}{\#\textit{all graphlets in } G}\, f_G,$$

and work with the normalized variant of (6):

$$k_g(G, G') = D_G^\top P D_{G'}. \tag{7}$$

Since P is a PSD matrix, it immediately follows that (7) is a valid positive semi-definite kernel.

3 Sampling from graphs

In order to compute our kernel exactly one needs to count all graphlets of size k in the input graphs. A graph with n nodes has $\binom{n}{k}$ or equivalently $O(n^k)$ graphlets, which is prohibitively expensive to enumerate. Therefore we resort to sampling. The hope is that if a sufficient number of random samples is drawn, then the empirical distribution is *close* to the actual distribution of graphlets in the graph. The number of samples needed to achieve a given confidence with a small probability of error is called the sample complexity.

This approach is not new; the problem of sampling subgraphs from graphs has been widely studied in the bio-informatics literature (Przulj, 2007; Kashtan et al., 2004; Wernicke, 2005). Unfortunately, the algorithms proposed there are ad-hoc and do not provide any bounds on sample complexity. Recently, (Weissman et al., 2003) proved distribution dependent bounds for the L_1 deviation between the true and the empirical distributions. We adapt their results to derive strong sample complexity bounds.

3.1 Sample complexity bound

Let $\mathcal{A} = \{1, 2, \ldots, a\}$ denote a finite set of elements. For two probability distributions P and Q on $\mathcal{A}$, the L_1 distance between P and Q is defined as

$$||P - Q||_1 := \sum_{i=1}^{a} |P(i) - Q(i)|. \tag{8}$$

Given a multiset $X := \{X_j\}_{j=1}^m$ of independent identically distributed (iid) random variables X_j drawn from some distribution D (denoted as $X_j \sim D$), the empirical estimate of D is defined as

$$\hat{D}^m(i) = \frac{1}{m} \sum_{j=1}^{m} \delta(X_j = i), \tag{9}$$

where $i \in \mathcal{A}$, and $\delta(\cdot)$ denotes the indicator function; $\delta(X_j = i) = 1$ if $X_j = i$ and zero otherwise.

Theorem 6 *Let D be a probability distribution on the finite set $\mathcal{A} = \{1, \ldots, a\}$. Let $X := \{X_j\}_{j=1}^m$, with $X_j \sim D$. For a given $\epsilon > 0$ and $\delta > 0$,*

$$m = \left\lceil \frac{2\left(\log 2 \cdot a + \log\left(\frac{1}{\delta}\right)\right)}{\epsilon^2} \right\rceil \tag{10}$$

samples suffice to ensure that $P\left\{||D - \hat{D}^m||_1 \geq \epsilon\right\} \leq \delta$.

We postpone the proof of this theorem to a longer version of the paper, and note in the passing that it follows from the results of (Weissman et al., 2003).

3.2 Implications of the bound

In order to apply Corollary 6 to our problem we set $\mathcal{A}$ to be the set of all size-k graphlets and assume that they are distributed according to an unknown distribution D. Furthermore, let m be the number of graphlets randomly sampled from the graph. Then (10) gives the number of samples needed to ensure that the empirical distribution $\hat{D}^m$ is at most ϵ distance away from the true distribution D with confidence $1 - \delta$.

When dealing with unlabeled graphs, there are a total of $2^{\binom{k}{2}}$ possible graphlets of size k. But, modulo isomorphism, there are only $N_k < 2^{\binom{k}{2}}$ distinct graphlets. For example, consider the case $k = 4$: We have $a = N_k = 11$, while $2^{\binom{4}{2}} = 64$. If we set $\epsilon = 0.05$ and $\delta = 0.05$, then our bound implies that we only need to sample $8,497$ graphlets from a graph. If we decrease ϵ to 0.01 and δ to 0.01, then this number increases to $244,596$.

4 Bounded degree graphs

While sampling allows us to deal with graphs on which the exhaustive enumeration of all graphlets is infeasible, in pratice, there is a large fraction of graphs on which complete counting can be performed efficiently: the class of graphs with bounded degree d. We present two algorithms for efficiently counting graphlets in graphs of low degree: one for counting *all connected graphlets*, and one for counting *all graphlets*.

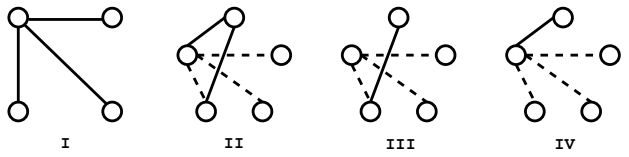

Figure 1: Connected graphlets of size k which do not contain a length-$k-1$ path (I is a size-4 graphlet, while II-IV are size-5 graphlets)

4.1 Enumerating all connected graphlets

We assume that our graphs are given in standard adjacency list representation. As a preprocessing step, we construct a data structure that supports checking the existence of edges in time $O(1)$. Given vertices u and v, the data structure checks whether $(u, v) \in E$. We may either use the adjacency matrix or a hashing scheme (Mehlhorn & Sanders, 2008, Chapter 4). Observe that the adjacency matrix can be constructed in time $O(|E|)$ if one uses an implicit initialization scheme (Mehlhorn & Sanders, 2008, Exercise 3.16). With such a data structure one can determine in time $O(k^2)$ which graphlet is induced by a path of length k.

Theorem 7 *Let G be a bounded degree graph, and let d denote the maximum degree. Then all connected graphlets of G with size $k \in \{3, 4, 5\}$ can be enumerated in $O(nd^{k-1})$ time.*

Proof Graphlets of size k can be divided into two classes: graphlets that contain a simple path of length $k-1$, and graphlets that do not contain such a path.

By a simple depth first search (DFS) all paths of length $k-1$ originating from a node v can be computed in $O(d^{k-1})$ time. Counting the size-k graphlets induced by these paths requires $O(d^{k-1})$ effort; the overall complexity for a graph with n nodes is therefore $O(nd^{k-1})$. One caveat is that the same graphlet might be induced by more than one length-$k-1$ paths, and hence might be counted multiple times. To account for this, we need to divide the final counts by the number of length-$k-1$ paths per graphlet.

The second class can be computed efficiently for $k \in \{3, 4, 5\}$. For $k = 3$, there is no connected graphlet that does not contain at least one path of length 2.

For $k = 4$, there is only one connected graphlet that does not contain a length-3 path (see I in Figure 1). Let us call this graphlet a 3 star. Let d_i denote the degree of node v_i. We look up the $\binom{d_i}{3}$ neighbor triplets of v_i, and check if they induce the graphlet we are interested in. The time complexity per node is $O(d^3)$, and for the entire graph is $O(nd^3)$.

For $k = 5$, there are 3 connected graphlets with no length-4 path (see II to IV in Figure 1). To compute the frequency of their occurrence we note that all contain the 3 star as a subgraph. So we first enumerate all occurrences of 3 star, and then check the neighbors of each node in 3 star to see if they induce the graphlets in question, an $O(d)$ operation per graphlet. This brings the overall complexity of the method to $O(nd^4)$, as claimed. ■

4.2 Counting all graphlets

Here we show that all size 3 and 4 graphlets can be counted efficiently in bounded degree graphs. We relegate the proof for size 5 graphlets to an extended version of this article.

Theorem 8 *For a fixed node v_1, we can compute the distribution of subgraphs of size 3 and 4 in time $O(d^2)$ and $O(d^3)$ respectively, where d is the maximal degree of any node.*

Proof Let us first consider counting graphlets of size 3. Modulo isomorphism there are 4 types of such graphlets. Let F_i and $|F_i|$ denote the graphlet with i edges and the number of its occurrences in a graph respectively, $i \in \{0, 1, 2, 3\}$, and $N(v)$ the set of neighbors of a node v. We first count subgraphs with at least one edge and then obtain the number of graphlets of type F_0 by subtracting $|F_1|+|F_2|+|F_3|$ from C_n^3,which is the number of all triplets of nodes in the graph.

For each pair of nodes (v_1, v_2) connected by an edge, we have to distinguish four cases for the third node v_3: $v_3 \in N(v_1) \cap N(v_2)$, $v_3 \in N(v_1) \setminus (N(v_2) \cup \{v_2\})$, $v_3 \in N(v_2) \setminus (N(v_1) \cup \{v_1\})$, $v_3 \notin N(v_1) \cup N(v_2)$.

It is easy to see that the subgraph spanned by v_1, v_2 and v_3 in the first case corresponds to the graphlet with 3 edges, in the second and third cases it corresponds to the graphlet with 2 edges, and in the fourth to the graphlet with 1 edge.

Enumerating all pairs of edges originating at v_1 is a $O(d)$ effort. For each pair (v_1, v_2), determining the cardinality of the sets $N(v_1) \cap N(v_2)$, $N(v_1) \setminus (N(v_2) \cup \{v_2\})$ and $N(v_2) \setminus (N(v_1) \cup \{v_1\})$ has $O(d)$ time complexity as well. As to the cardinality of the set $V \setminus (N(v_1) \cup N(v_2))$, it can be easily computed by observing that it is equal to $n - |N(v_1) \cup N(v_2)|$. This leads to the overall complexity of $O(d^2)$.

Note that counting graphlets in the proposed way would imply counting them twice as many times as the number of edges they contain. To deal with this, we need to divide the final counts by twice the number of edges per graphlet.

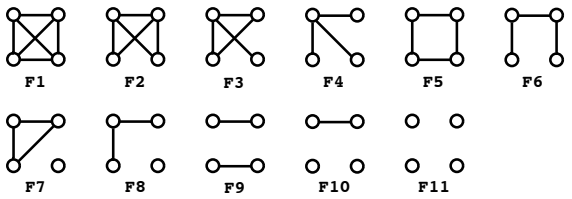

Figure 2: All graphlets of size 4

We now consider size 4 graphlets.

Modulo isomorphism there are 11 graphlets of size 4 (see Figure 2). Let us denote these graphlets F_i and their counts $|F_i|$, $i \in 1, 2, \ldots, 11$. As in the previous case, we will first count all graphlets which contain at least one edge.

Assume we want to count subgraphs containing edge (v_1, v_2). As before, for v_2 there are $|N(v_1)|$ choices and for each pair (v_1, v_2) we have 4 cases for the third node v_3: $v_3 \in N(v_1) \cap N(v_2)$, $v_3 \in N(v_1) \setminus N(v_2)$, $v_3 \in N(v_2) \setminus N(v_1)$ and $v_3 \notin N(v_1) \cup N(v_2)$.

v_3 from the first three cases can be enumerated in $O(d)$. And for each triplet (v_1, v_2, v_3) we can count size-4 subgraphs containing this triplet in $O(d)$, as we can compute cardinalities of all intersections of $N(v_1)$, $N(v_2)$ and $N(v_3)$ in $O(d)$.

As to v_3 in the fourth case, there are 2 types of graphlets which arise in this case and do not arise in previous cases: graphlets of type F_9 and F_{10}. For fixed v_1 and v_2 it is possible to obtain the number of graphlets of type F_9 by counting the number of edges not adjacent to any of the nodes $v \in N_1(v_1) \cup N_1(v_2)$. This quantity is equal to $m+1-|N(v_1)|-|N(v_2)|-K$, where m is the number of edges in the graph which can be precomputed, $|N(v_1)| + |N(v_2)| - 1$ is the number of edges adjacent to v_1 or v_2 and K is the number of edges adjacent to nodes in $(N(v_1) \cup N(v_2)) \setminus \{v_1, v_2\}$. The latter equals to the number of previously counted graphlets with the same v_1 and v_2, where v_3 and v_4 are connected (i.e. where $v_4 \in N(v_3)$). Once $|F_9|$ is computed, we obtain $|F_{10}|$ by subtracting $|F_9|$ from $\binom{n-|N(v_1)\cup N(v_2)|}{2}$, which is the number of pairs of nodes outside $N(v_1) \cup N(v_2)$. The fourth case does not increase the runtime complexity of previous cases and remains $O(d^3)$ per v_1.

At last, $|F_{11}| = \binom{n}{4} - \sum_{i=1}^{10} |F_i|$.

Note that, as in the previous case, we will count each graphlet at least twice as many times as the number of edges it contains. Additionally, in case if v_3 and v_4 are both neighbors of v_1 or v_2 (*i.e.,* all configurations except $v_4 \in N(v_3) \setminus (N(v_1) \cup N(v_2))$), the graphlet spanned by these four nodes will be counted twice per fixed (v_1, v_2). To avoid this, we divide the counts of these graphlets by 2. ■

dataset	size	classes	# nodes	# edges
MUTAG	188	2 (125 vs. 63)	17.7	38.9
PTC	344	2 (192 vs. 152)	26.7	50.7
Enzyme	600	6 (100 each)	32.6	124.3
D & D	1178	2 (691 vs. 587)	284.4	1921.6

Table 1: Statistics on classification datasets.

5 Experiments

In this section, we evaluate the performance of our kernel and compare it with state of the art graph kernels in terms of runtime, scalability, and prediction accuracy. Our baseline comparators are the classic random walk kernel of (Gärtner et al., 2003; Kashima et al., 2004; Vishwanathan et al., 2007), that counts common walks in two graphs, and the shortest path kernel of (Borgwardt & Kriegel, 2005), that compares shortest path lengths in two graphs. Both these kernels work on generic graphs, and are shown to perform competitively in their respective publications. For the random walk kernel we uniformly set the decay factor $\lambda = 10^{-4}$. For the shortest path kernel we used the delta kernel to compare shortest-path distances.

Datasets We perform experiments on three different well known, publicly available datasets namely MUTAG, PTC, Enzyme. We also test our kernels with a large protein function prediction dataset from (Dobson & Doig, 2003), which we will refer to as D & D. Table 1 provides a summary.

Experimental settings We test different variants of our graphlet kernels: by varying the graphlet sizes $k \in \{3, 4, 5\}$, the types of graphlets we consider (fully connected vs all), and sample size (different values of precision, ϵ, and confidence, δ).

To compute size-3 graphlet kernel based on sampling, we drew samples of 1016 graphlets (corresponding to $\epsilon = 0.1$, $\delta = 0.1$), 1154 graphlets ($\epsilon = 0.1$, $\delta = 0.05$), 4061 graphlets ($\epsilon = 0.05$, $\delta = 0.1$) and 4615 graphlets ($\epsilon = 0.05$, $\delta = 0.05$). Analogously, for size-4 and size-5 graphlet kernels we used four sample sizes: {1986, 2125, 7942, 8497} and {5174, 5313, 20696, 21251} respectively.

We use a binary C-Support Vector Machine (SVM) to test the efficacy of our kernels. We perform 10-fold cross validation, and for each fold we independently tune the value of C, the SVM regularizer constant, by considering the training data from that fold. This process is averaged over 10 random splits of the data. We report classification accuracies in Table 2 and runtimes for kernel matrix computation in Table 3.

Kernel	MUTAG	PTC	Enzymes	D & D
RW	71.89 ± 0.66	55.44 ± 0.15	14.97 ± 0.28	> 1 day
SP	81.28 ± 0.45	55.44 ± 0.61	27.53 ± 0.29	> 1 day
GK A3 1016	79.70 ± 0.43	55.34 ± 0.33	23.07 ± 0.38	74.92 ± 0.12
GK A3 1154	79.54 ± 0.41	54.90 ± 0.42	24.22 ± 0.37	75.05 ± 0.10
GK A3 4061	80.91 ± 0.40	55.21 ± 0.34	25.45 ± 0.70	75.23 ± 0.13
GK A3 4615	80.21 ± 0.37	55.13 ± 0.37	24.85 ± 0.79	75.44 ± 0.13
GK A3 all	82.11 ± 0.62	55.26 ± 0.39	25.80 ± 0.23	75.41 ± 0.13
GK C3	66.55 ± 0.83	57.68 ± 0.43	19.80 ± 0.21	74.14 ± 0.12
GK A4 1986	79.80 ± 0.38	58.88 ± 0.50	26.93 ± 0.57	74.47 ± 0.17
GK A4 2125	80.69 ± 0.31	58.86 ± 0.53	27.25 ± 0.44	74.50 ± 0.14
GK A4 7942	81.74 ± 0.44	59.25 ± 0.50	27.96 ± 0.40	74.53 ± 0.16
GK A4 8497	81.48 ± 0.38	59.39 ± 0.57	28.06 ± 0.45	74.59 ± 0.16
GK A4 all	82.17 ± 0.58	59.65 ± 0.31	28.95 ± 0.50	74.62 ± 0.12
GK C4	69.00 ± 0.74	58.62 ± 0.41	23.61 ± 0.22	75.90 ± 0.10
GK A5 5174	81.62 ± 0.69	58.26 ± 0.47	29.54 ± 0.42	75.11 ± 0.14
GK A5 5313	81.93 ± 0.71	58.29 ± 0.42	29.52 ± 0.25	75.14 ± 0.14
GK A5 20696	82.64 ± 0.66	57.88 ± 0.54	29.96 ± 0.52	75.52 ± 0.12
GK A5 21251	83.27 ± 0.79	58.03 ± 0.42	30.48 ± 0.51	75.35 ± 0.10
GK A5 all	83.50 ± 0.60	58.65 ± 0.40	30.64 ± 0.26	> 1 day
GK C5	70.94 ± 0.76	56.06 ± 0.46	26.66 ± 0.23	> 1 day

Table 2: Classification accuracy (in % ± standard error) on unlabeled graph benchmark datasets. RW is the random walk kernel, while SP is the shortest-path kernel. GK Ak m denotes our graphlet kernel computed using m samples of size k graphlets. GK Cn denotes the graphlet kernel computed using *all* connected graphlets of size k. '> 1 day' means computation did not finish within 24 hours.

Kernel	MUTAG	PTC	Enzymes	D & D
RW	42.3"	2' 39"	10' 45"	> 1 day
SP	23.2"	2' 35"	5' 1"	> 1 day
GK A3 1016	21.5"	29.7"	39"	2' 9"
GK A3 1154	23.1"	42.6"	48.7"	2' 19"
GK A3 4061	1' 18"	2' 39"	1' 51"	6' 35"
GK A3 4615	1' 38"	3' 1"	2' 51"	5' 58"
GK A3 all	0.35"	0.9"	3.34"	2' 34"
GK C3	0.14"	0.36"	1.3"	2' 14"
GK A4 1986	1' 39"	3' 2"	4' 20"	11' 35"
GK A4 2125	1' 46"	3' 16"	4' 36"	12' 21"
GK A4 7942	6' 33"	12' 3"	16' 35"	42' 45"
GK A4 8497	6' 57"	12' 49"	17' 38"	45' 36"
GK A4 all	4.38"	10.8"	49.3"	2h 44' 59"
GK C4	0.26"	0.9"	4.1"	35' 22"
GK A5 5174	3' 14"	8' 1"	16' 57"	1h 29' 54"
GK A5 5313	3' 18"	8' 6"	17' 3"	1h 1' 54"
GK A5 20696	8' 56"	18' 28"	42' 2"	1h 30' 18"
GK A5 21251	9' 5"	18' 4"	27'	2h 6' 45"
GK A5 all	7' 17"	16h 2' 16"	20h 26' 8"	> 1 day
GK C5	0.79"	2.1"	40.7"	> 1 day

Table 3: Runtime on unlabeled graph benchmark datasets (implemented in Matlab, for abbreviations see Table 2)

Results On MUTAG, PTC and Enzymes, modeled as unlabeled graphs, graphlet kernels counting all graphlets reached the highest accuracy. Graphlet kernels based on sampling also yield similarly good results. The classification accuracies they reach are comparable to that of the shortest path kernel on MUTAG and Enzyme, while on PTC they are better. In all cases they comprehensively outperform random walk kernels. Note that our sampling technique is independent of the size of the graph, and yet yields comparable results to expensive kernels which depend on the size of the graph.

In terms of runtime, 4 and 5-node graphlet sampling and 5-node graphlet counting are expensive and slower than the shortest path and the random walk kernels on small datasets such as MUTAG and PTC. As graph size increases (Enzyme), graphlet sampling gets more competitive: Sampling 1986 and 2125 size-4 graphlets on Enzyme is already faster than computing shortest path and random walk kernels. On D & D, none of the latter kernels finishes computation within 24 hours, nor does the counting of all 5-node graphlets. The graphlet kernels based on sampling manage to compute kernel matrices on D & D in less than 2 hours and 7 minutes for 5-node graphlets.

Kernels based on counting graphlets in bounded degree graphs are fast to compute for MUTAG, PTC, Enzymes, but less so for D & D. This is due to the fact that the first three datasets have a low maximum degree (4, 4 and 9 respectively), whereas for D & D it is 52. In terms of accuracy, on PTC and D & D they are comparable with other kernels, while on MUTAG and Enzymes they perform worse. Disconnected graphlets seem to be essential for correct classification on these datasets.

We note in the passing that even though our graphlet kernels do not exploit any domain knowledge and operate on simple unlabeled graph models of proteins, on the D & D dataset the classification accuracy they obtain is comparable with published work that uses heavily annotated vector or graph models of proteins (Dobson & Doig, 2003; Borgwardt et al., 2005).

6 Conclusions

In this paper, we have proposed efficient graph kernels based on counting or sampling limited size subgraphs in a graph. Our approaches are highly efficient for unlabeled graphs, yet a central challenge of future research will be to make our results memory and runtime efficient on graphs with discrete node labels, and even more so on graphs with continuous node labels. Our methods for efficient counting of graph features are not limited to being used in graph kernels, but can be applied in a variety of problems in graph mining.

References

Borgwardt, K. M., & Kriegel, H.-P. (2005). Shortest-path kernels on graphs. *Proc. Intl. Conf. Data Mining* (pp. 74–81).

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

Dobson, P. D., & Doig, A. J. (2003). Distinguishing enzyme structures from non-enzymes without alignments. *J. Mol. Biol.*, *330*, 771–783.

Gärtner, T., Flach, P., & Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. *COLT* (pp. 129–143). Springer.

Hemminger, R. L. (1969). On reconstructing a graph. *Proceedings of the American Mathematical Society*, *20*, 185–187.

Horvath, T., Gärtner, T., & Wrobel, S. (2004). Cyclic pattern kernels for predictive graph mining. *KDD* (pp. 158–167).

Kashima, H., Tsuda, K., & Inokuchi, A. (2004). Kernels on graphs. *Kernels and Bioinformatics* (pp. 155–170). Cambridge, MA: MIT Press.

Kashtan, N., Itzkovitz, S., Milo, R., & Alon, U. (2004). Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, *20*, 1746–1758.

Kelley, P. (1957). A congruence theorem for trees. *Pacific J. Math.*, *7*, MR 19:442.

McKay, B. (1997). Small graphs are reconstructible. *Australas. J. Combin.*, *15*, 123–126.

Mehlhorn, K., & Sanders, P. (2008). *Algorithms and data structures: The basic toolbox*. Springer.

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

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

Ramon, J., & Gärtner, T. (2003). Expressivity versus efficiency of graph kernels. *First International Workshop on Mining Graphs, Trees and Sequences*.

Vishwanathan, S. V. N., Borgwardt, K., & Schraudolph, N. N. (2007). Fast computation of graph kernels. *NIPS*. Cambridge MA: MIT Press.

Weissman, T., Ordentlich, E., Seroussi, G., Verdu, S., & Weinberger, M. J. (2003). *Inequalities for the l_1 deviation of the empirical distribution* (Technical Report HPL-2003-97(R.1)). HP Labs, Palo Alto.

Wernicke, S. (2005). A faster algorithm for detecting network motifs. *WABI* (pp. 165–177).

Yan, X., & Han, J. (2003). Closegraph: mining closed frequent graph patterns. *KDD* (pp. 286–295).