Effective Graph Classification Based on Topological and Label Attributes

Geng Li[1], Murat Semerci[2], Bülent Yener[1] and Mohammed J. Zaki[1]*

[1]*Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY 12180, USA*

[2]*Department of Computer Engineering, Bogazici University, Istanbul, Turkey*

Received 23 September 2011; revised 30 March 2012; accepted 25 April 2012
DOI:10.1002/sam.11153
Published online 12 June 2012 in Wiley Online Library (wileyonlinelibrary.com).

Abstract: Graph classification is an important data mining task, and various graph kernel methods have been proposed recently for this task. These methods have proven to be effective, but they tend to have high computational overhead. In this paper, we propose an alternative approach to graph classification that is based on feature vectors constructed from different global topological attributes, as well as global label features. The main idea is that the graphs from the same class should have similar topological and label attributes. Our method is simple and easy to implement, and via a detailed comparison on real benchmark datasets, we show that our topological and label feature-based approach delivers competitive classification accuracy, with significantly better results on those datasets that have large unlabeled graph instances. Our method is also substantially faster than most other graph kernels. © 2012 Wiley Periodicals, Inc. Statistical Analysis and Data Mining 5: 265–283, 2012

Keywords: graph classification; graph kernels; graph mining; topological attributes

1. INTRODUCTION

With the proliferation of graph data, there has been a lot of interest in recent years to develop effective methods for classifying graph objects [1]. Applications range from chem-informatics [2,3] (e.g., compounds that are active or inactive for some target) and bioinformatics [4,5] (e.g., classifying proteins into different families, classifying tissue samples), to telecommunication networks (e.g., classifying customers based on their calling behavior) and social networks (e.g., classifying users based on their feeds on Twitter, Facebook, etc.).

The graph classification problem can be stated as follows: There is a dataset of graphs $G_i \in \mathcal{D}$, with $i = 1, \ldots, N$. Each graph $G_i = (V_i, E_i)$ is given as a collection of vertices $V_i = \{v_{i1}, \ldots, v_{in}\}$ and edges $E_i = \{(v_a, v_b)|v_a, v_b \in V_i\}$. The graph G_i may have labels on the nodes and edges, drawn from some common set of labels Σ for the entire dataset $\mathcal{D}$. Finally, each graph G_i has a corresponding class $y_i \in C$, where C is the set of k categorical class labels, given as $C = \{1, \ldots, k\}$. The goal of graph classification is to learn a model $f : \mathcal{D} \rightarrow C$ that predicts the class label for any graph. Typically the model is learned from a *training set* of graphs with known class labels. The model is then evaluated on a *testing set* of graphs. The accuracy of the classification model can be tested by comparing the predicted output $\hat{y}_i = f(G_i)$ with the true class label y_i (provided it is known).

The main challenge in classifying graphs is how to convert the discrete graph objects into numeric features or similarities for effective classification. Graph kernel methods have attracted a lot of attention because of their ability to represent the graph data as a $N \times N$ symmetric, positive semi-definite *kernel matrix* $\mathbf{K} = \{\kappa(G_i, G_j)\}_{i,j=1}^{N}$ that records the pair-wise similarities between graphs in $\mathcal{D}$. Conceptually, the kernel function $\kappa(G_i, G_j)$ represents an inner-product between the vectors corresponding to the two graphs G_i and G_j in some N-dimensional *feature space*; see ref. 6 for more details on kernel methods. Once the kernel matrix has been constructed, it is possible to classify the graphs with a support vector machine (SVM) [7], using the supplied kernel matrix $\mathbf{K}$. There has been a lot of research activity in trying to develop more effective and efficient graph kernel functions κ. These methods can broadly be classified into methods based on random walks [8,9], shortest paths (SPs) [10], cycles [11] subtrees [2,12,13], and subgraphs [14,15,16]. Despite the research above, it is fair to say that efficient and effective

Correspondence to: Mohammed J. Zaki (zaki@cs.rpi.edu)

graph classification still remains a challenge, especially for large graphs.

In this paper we propose an alternative approach to constructing a feature vector for graph classification. Instead of relying on 'patterns' like path, cycles, subtrees, and subgraphs, we compute several global topological and label attributes from each graph $G_i \in \mathcal{D}$. The values for these attributes yield a numeric feature vector $F_i = (f_{i1}, \ldots, f_{ip})$. The set of feature vectors F_i and the corresponding class labels y_i are then used to construct an SVM classifier. We show that our approach is both effective and scalable compared with state-of-the-art graph kernel methods. We conduct an extensive set of experiments over several real graphs, representing chemical compounds, proteins, and cell-graph datasets. We demonstrate that our approach yields better or competitive accuracy in a fraction of the time taken by other kernels. Our method is particularly effective in classifying large unlabeled graphs, since it is able to effectively capture the structural differences among the classes.

2. RELATED WORK

Graph kernels compute the similarity between pairs of graphs in $\mathcal{D}$, based on the common patterns they share. The patterns can range from the simple to the complex. Specifically the kernels are designed to exploit random walks [4,8,9,17], shortest paths [10], cyclic patterns [11], subtrees [2,3,12,13], and subgraphs [14–16]. Another class of graph kernels, e.g. the diffusion kernel [18], deal with the similarity between nodes of a single graph. However, our focus in this paper is on kernels between different graphs[1], which we discuss in more detail below.

Random Walk Kernels: The similarity of two graphs $G_i, G_j \in \mathcal{D}$ can be quantified by counting labeled walks that are common to both of them. The random walk kernel [8], one of the first graph kernels, is based on this idea. The kernels in refs [4,9] are also based on random walks over labeled graphs. Computing the pair-wise kernel values has worst case $O(n^6)$ complexity, where n denotes the number of nodes in G_i and G_j. A more efficient version of the random walk kernel was proposed in ref. 17, reducing the complexity to $O(n^3)$ per pair of graphs. One potential problem with these kernels is that artificially high kernel values may be obtained by repeatedly visiting same nodes and edges multiple times [20]. We refer to ref. 21 for a recent overview of random-walk (RW)-based graph kernels.

Shortest Path Kernels: The SP graph kernel [10] first computes the SP graph $S = (V_S, E_S)$ for each graph $G = (V, E) \in \mathcal{D}$. Here $V_S = V$, and a weighted edge (v_a, v_b) exists in E_S if v_a and v_b are connected by a path in G, with the edge weight representing the SP length between v_a and v_b (infinity if they are not reachable). Given the SP graphs S_i and S_j for two input graph G_i and G_j the kernel is defined as the sum over all pairs of edges from S_i and S_j, using any suitable positive definite kernel on the edges. The all-pairs SP graphs can be computed in $O(n^3)$ time, and the kernel can then be computed in $O(n^4)$ time, since S_i and S_j each have $O(n^2)$ edges. Other variants of the SP kernel include equal length SPs, k SP, k shortest disjoint paths, and so on [10].

Cyclic Pattern Kernels: The cyclic pattern kernel [11] is based on counting the number of common cycles that occur in both graphs. Since there is no known polynomial time algorithm to find all the cycles in a graph, sampling, and time-bounded enumeration of cycles are used to measure the similarity of the graphs.

Subtree Kernels: Subtree kernels are based on common subtrees in the graphs [12]. The main idea is to consider pairs of nodes from G_i and G_j and see if they share common tree-like neighborhoods, i.e. to count the pairs of identical subtrees of height h rooted at vertex $v_a \in G_i$ and $v_b \in G_j$. The kernel is defined as the sum over all pairs of vertices of a suitably defined vertex pair kernel. The complexity of this approach is $O(n^2 h 4^d)$, where d denotes the maximum degree. Another subtree kernel was proposed in ref. 2, based on a path representation of the trees obtained via a depth-first search on the input graphs. The kernel function is computed on these paths (e.g., the ratio of the longest common path).

The recently proposed Weisfeiler-Lehman (WL) kernel [13] is a fast subtree kernel that scales up to large, labeled graphs. It uses the WL isomorphism test, which uses iterative multiset-label determination, label compression, and relabeling steps. The isomorphism test terminates after a prespecified number of iterations h. If the sets of labels for nodes are not identical, then two graphs are considered as non-isomorphic, otherwise, they are isomorphic. The WL graph kernel counts the matching multiset labels for the two graphs G_i and G_j in each iteration of the WL isomorphism test. The WL kernel has $O(mh)$ complexity, where m is the number of edges in the graphs.

Graphlet and Subgraph Kernels: Similar graphs should have similar subgraphs. Graphlet kernels measure the similarity of two graphs by the dot product of count vectors of all possible connected subgraphs of order k (i.e., the graphlets, also called as k-minors) [14,15]. For any k (usually set to 3, 4, or 5), there are $2^{\binom{k}{2}}$ possible graphlets of size k, but many of them are isomorphic. Usually, to avoid the dependence on the size, the count vector is normalized into a probability vector, and the graphlet kernel is redefined

[1] A preliminary version of this paper appeared in the MLG'11 Workshop [19], which does not have formal proceedings.

as the dot product of the normalized count vectors for two graphs. Exhaustive enumeration of all graphlets has complexity $O(n^k)$. For a graph with bounded degree d, the connected graphlets can be enumerated in $O(nd^{(k-1)})$ [14].

Frequent subgraph mining can also be used to define a kernel between two graphs [16]. Let $\mathcal{F} = \{s_1, \ldots, s_p\}$ denote the set of p frequent and discriminative subgraph patterns mined from $\mathcal{D}$. Each graph $G_i \in \mathcal{D}$ is then represented as a binary feature vector $\{0, 1\}^p$ where feature j is set to 1 if and only if s_j is isomorphic to a subgraph in G_i. The kernel between G_i and G_j can be defined over their binary feature vectors. CORK [16] implements this approach; it uses gSpan [22] to mine the subgraphs, and selects near-optimal features (subgraphs) from that set, that are most discriminative for classification.

In our experiments in Section 4, we compare with the following graph kernel methods: fast geometric RW kernel [17], SP kernel [10], graphlet (GK) kernel [14], Ramon-Gärtner (RG) subtree kernel [12], and WL subtree kernel [13]. We also compare with CORK [16].

3. GRAPH ATTRIBUTES FOR CLASSIFICATION

As we have seen above, while many sophisticated graph kernels have been proposed, efficiency and scalability remain as challenges, for large graph datasets. Our basic idea is to compute several topological and label attributes for each graph in the dataset, and to use the derived feature-vector attributes for classification. Like most of the graph kernel work, we use the SVM as the classifier of choice. The graph attributes we use are listed below.

f-1. **Average degree:** The degree of a node is defined as the number of its neighboring edges. Average degree is the average value of the degree of all nodes in the graph, i.e. $\overline{d}(G) = \sum_i^n d(u_i)/n$, where $d(u_i)$ denotes the degree of node u_i.

f-2. **Average clustering coefficient:** For a node u, the clustering coefficient $c(u)$ represents the likelihood that any two neighbors of u are connected. More formally, the clustering coefficient of a node u is defined as: $c(u) = \frac{\lambda(u)}{\tau(u)}$, where $\lambda(u)$ is the number of triangles (complete graph with three nodes) of a node u and $\tau(u) = \frac{d(u)^2 - d(u)}{2}$, the number of triples a node u has. Figure 1 shows a triangle and its three triples. Alternatively, the clustering coefficient for node u can be defined as the ratio of the number of actual edges between the neighbors of u to the number of possible edges between them. The clustering coefficient $C(G)$ of a graph is the average of $c(u)$ taken over all the nodes in the graph, i.e. $C(G) = \frac{1}{n}\sum_{i=1}^n c(u_i)$. Here we use $C(G)$ as one of our global graph features. Generally, average clustering coefficient is a very popular metric in network analysis, but in some specific graph datasets, such as chemical compounds, there do not exist many triangles in any graph instance, which results in the clustering coefficient taking on value close to 0.

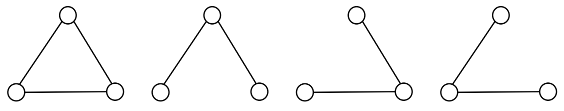
Fig. 1 A triangle with its three triples.

f-3. **Average effective eccentricity:** The eccentricity of a node u is defined as $e(u) = max\{d(u, v) : v \in V\}$, where the distance $d(u, v)$ is the length of the shortest path from u to v. For *effective eccentricity* we take the maximum length of the SP from u, so that u can reach at least 90% of nodes in the graph. Effectiveness is a more robust measure if we take noise into consideration. The average effective eccentricity is the average of effective eccentricities of all nodes in the graph.

f-4. **Maximum effective eccentricity (effective diameter)**: Maximum effective eccentricity is defined as the maximum value of effective eccentricity over all nodes in the graph. Note that the maximum eccentricity is the graph diameter, i.e. $diam(G) = max\{e(u)|u \in V\} = max\{d(u, v)|u, v \in V\}$. Maximum effective eccentricity is thus the same as effective diameter.

f-5. **Minimum effective eccentricity (effective radius):** Minimum effective eccentricity is defined as the minimum value of effective eccentricity over all nodes in the graph. Note that minimum eccentricity is called the graph radius, i.e. $rad(G) = min\{e(u)|u \in V\}$, thus minimum effective eccentricity is the effective radius.

f-6. **Average path length (closeness centrality):** The closeness centrality of a node u is defined as the reciprocal of the averaged total path length between node u and every other node that is reachable from node u, where $u \in V$, i.e. $close(u) = \frac{n-1}{\sum_{v \in V, v \neq u} d(u,v)}$. We take the average of closeness centrality of all nodes as a global metric for a graph.

f-7. **Percentage of central points:** We define a point u to be a central point if it has an eccentricity equal to the effective radius of the graph, i.e. it satisfies: $\{u \in V : \textit{effective-rad}(G) = e(u)\}$. The ratio of the number of central points to the total number of points in the graph is selected as a feature.

f-8. **Giant connected ratio:** A giant component is a subgraph that is connected and has the maximum number of nodes. We take the ratio of the number of nodes of the giant connected component to the total number of nodes in the entire graph as a global metric for a graph. Note that if the entire graph is connected, the ratio is 1, thus in those datasets that are comprised of connected graphs, this attribute will not be a good graph descriptor. However, not all graphs in our experimental study are Connected; thus, this ratio may be a meaningful attribute to use.

f-9. **Percentage of isolated points:** We define a isolated point in a graph to be a node with degree zero. The ratio of isolated points to the total number of nodes in the entire graph is considered as a feature. For graphs that are connected, this feature will not be meaningful, but there are datasets we used in our study that do have isolated points.

f-10. **Percentage of end points:** A node which has a degree of one is defined as an end point. The ratio of the number of end points to the total number of nodes in the entire graph is selected as a feature.

f-11. **Number of nodes:** Total number of nodes in the graph.

f-12. **Number of edges:** Total number of edges in the graph.

f-13. **Spectral radius:** The spectral radius is defined as the largest magnitude eigenvalue of the adjacency matrix of the graph. More formally, let $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_s|$ be the distinct eigenvalues of the adjacency matrix A of the graph, sorted by their magnitude. The spectral radius of the graph, $\rho(G)$, is defined as: $\rho(G) = |\lambda_1|$.

f-14. **Second largest eigenvalue:** The value of the second largest eigenvalue of the adjacency matrix A, i.e. $|\lambda_2|$.

f-15. **Trace:** Sum of the eigenvalues of the adjacency matrix, i.e. $\sum_i^n \lambda_i$. This is in fact equivalent to the trace of the adjacency matrix A, i.e. $Tr(A) = \sum_{i=1}^n a_{ii}$. This feature is useful only if the graph has several loops, i.e. an edge joining a vertex to itself. For a simple graph, which is loop-free, the trace equals 0 because the elements on the main diagonal of A are all zeros.

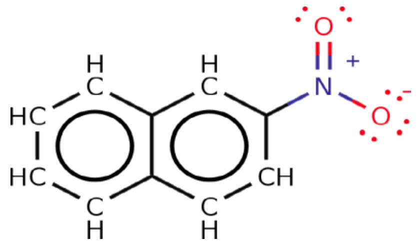

Fig. 2 A chemical compound from PTC dataset (with implicit hydrogens). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

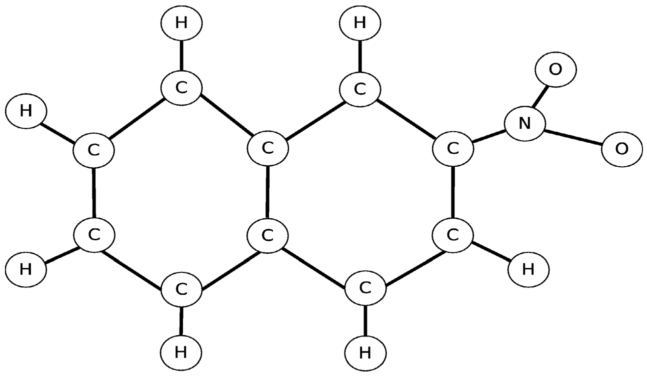

Fig. 3 The graph representation for Fig. 2. Labels on node/atoms: O = oxygen, H = hydrogens, N = nitrogen, C = carbon. Labels on edges/bonds: A = aromatic, S = single, D = double.

f-16. **Energy:** Squared sum of the eigenvalues of the adjacency matrix A. More formally, the energy of a graph G is: $E(G) = \sum_i^n \lambda_i^2$.

f-17. **Number of eigenvalues:** Number of distinct eigenvalues, $s \le n$, of the adjacency matrix A of the graph. The adjacency matrix A of an undirected graph has n eigenvalues; however, they are not necessarily distinct.

f-18. **Label Entropy:** We employ label entropy to measure the uncertainty of labels. Suppose a graph G has q different labels: $l_1,\ldots, l_q$, then the label entropy is given as $H(G) = -\sum_{i=1}^q p(l_i) \log p(l_i)$.

f-19. **Neighborhood Impurity:** We define the impurity degree of a node u as:

$$\textit{ImpurityDeg}(u) = |L(v) : v \in N(u), L(u) \neq L(v)|$$

where $L(u)$ is the label, and $N(u)$ is the neighborhood of (the nodes adjacent to) node u. If all nodes in the neighborhood of u have the same node label, the impurity degree is 0. For the whole graph, we are only interested in the nodes that have impurity degree larger than 0, i.e. the nodes which have at least one neighbor whose label is different. The neighborhood impurity of a graph G is defined as the average impurity degree over nodes with positive impurity.

f-20. **Link Impurity:** An edge (u, v) is defined to be impure if $L(u) \neq L(v)$. The link impurity of a graph G is defined as: $\frac{|(u,v) \in E : L(u) \neq L(v)|}{m}$, where m is the number of total edges in graph G.

Example: Figure 2 illustrates an example from PTC (the Predictive Toxicology Challenge) chemical compound dataset (see Section 4.2 for description). Figure 3 is a graph representation for the molecule in Fig. 2. Nodes/edges are assigned a label based on their types, properties, etc. Table 1 gives the graph feature vector based on the 20 global graph attributes that are listed in the order given above. For instance, there are 20 nodes in the graph, with 9 nodes having degree 1, and 11 nodes having degree 3. The average degree (f_1) is therefore $\overline{d} = \frac{9\times1+11\times3}{20} = \frac{42}{20} = 2.1$. Since there are no triangles in the graph the clustering coefficient (f_2) is 0. As another example, the graph has $m = 21$ edges, but there are 11 impure links ($L(u) \neq L(v)$), thus the link impurity is given as $f_{20} = 10/21 = 0.48$. The other features can be computed based on their definition. □

Graph Classification: In computing the feature values, if a certain graph in the dataset is disconnected and contains several components, we compute the average value for a given graph metric over all the components. Each graph G_i in the database is finally represented by its corresponding feature vector F_i over the 20 topological and label attributes. However, using the raw or unnormalized feature values does not perform well. This is mainly because the original feature have different range of values (see Table 1 for example), which would give more importance to features with larger values than those with smaller values. Instead we normalize the feature values via range and z-normalization.

In range normalization each value x of a feature f_i is transformed into $r(x) = \frac{x - \min}{\max - \min}$, where min and max denote the minimum and maximum value for f_i. In z-normalization x is replaced by $z\text{-}score(x) = \frac{x-\mu}{\sigma}$, where μ and σ are the mean and standard deviation for f_i. By normalizing the values all features are considered on equal footing, which helps improve the classification accuracy. Once each graph G_i in the dataset $\mathcal{D}$ has been transformed into its corresponding normalized feature vector of length 20, $F_i = (f_{i1}, \ldots, f_{i20})$, we use the SVM classifier over the new feature-vector dataset, using the Gaussian or radial basis kernel (RBF) (we tried a linear kernel too, but RBF gave better results).

Computational Complexity: The various graph attributes range from the simple to the complex, with higher computational times for the more complex features. In the analysis below, we use n to denote the number of nodes $|V|$ (also called the graph order), and m to denote the number of edges $|E|$ (also called the graph size).

For each graph in the database, the number of nodes (f_{11}) and edges (f_{12}) are already known, so their cost is $O(1)$. If they are not known beforehand, we can compute them in one pass over the entire graph in time $O(n + m)$. The degree-based attributes such as the percentage of isolated or end nodes (f_9, f_{10}) and average degree (f_1) can be computed in time linear in the graph order and size, i.e. in $O(n + m)$ time. The giant connected component (f_8) can also be found via breadth-/depth-first search in $O(n + m)$ time.

The clustering coefficient (f_2) can be computed in time $O(nd_{\max}^2)$, where $d_{\max}$ is the maximum degree for the graph. A better approximation is to use the average degree $d = \frac{2m}{n}$. To compute the clustering coefficient for each node takes on average $O(d^2) = \frac{2m^2}{n^2}$. The time to compute the average clustering coefficient over all nodes is then $O(\frac{m^2}{n})$.

The eccentricity-based attributes (f_3, f_4, f_5, f_7) and the average path length (f_6) can be easily computed from the all-pairs SP matrix. The all-pairs matrix can be computed in time $O(n^2 + nm)$ via n calls of single-source SPs, each of which can be computed in breadth-/depth-first search in time $O(n + m)$, since we assume that each edge has weight one. From the SP matrix, the attributes can be computed in $O(n^2)$ time.

The spectral attributes (f_{13} to f_{17}) depend on the eigen-decomposition of G, which can be computed in $O(n^3)$ time in the worst case. However, typically real-world graphs are

Table 1. Global graph feature vector for the example.

F	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
V	2.10	0.00	5.75	8	4	0.29	0.15	1.00	0.00	0.45
F	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
V	20	21	2.56	2.15	0.0	42.00	20	1.09	1.11	0.48

F = feature, V = value

sparse, which can be exploited to reduce the complexity to $O(n^2)$ [23]. Also note that when the input graphs are very large, we compute only the top $k \geq 2$ eigenvalues. For sparse graphs, the top k eigenvalues can be computed in $O(mkt + nk^2t + k^3t)$ time (e.g., using the implicitly restarted Lanczos method [24]), where t is the number of iterations until convergence. For sparse graphs, with $m = O(n)$, if $k \ll n$, the time reduces to $O(nt)$. The trace (f_{15}) and energy (f_{16}) are computed only over these k eigenvalues. The number of eigenvalues (f_{17}) is not very informative in this case.

Finally, label entropy (f_{18}) can be computed in $O(n)$, neighborhood impurity (f_{19}) can be computed in $O(nd_{\max})$ and link impurity (f_{20}) can be computed in $O(n + m)$ time.

4. EXPERIMENTS

4.1. Experimental Setup

We compare our graph-feature-based classification approach with state-of-the-art graph kernel classifiers. More specifically, we compare with the following graph kernel methods: fast geometric RW kernel [17], SP kernel [10], graphlet (GK) kernel [14], RG subtree kernel [12], and WL subtree kernel [13]. We relied on a Matlab implementation of all of these kernels[2]. As suggested in ref. 13, we used the tuned parameter settings for each of the kernels as follows. For RW, the decay weight is chosen in the range $\lambda \in \{10^{-6}, \ldots, 10^{-2}\}$. For RG we set $\lambda_r = \lambda_s = 1$. For the WL kernel, we choose $h = \{1, \ldots, 10\}$, which means that ten different kernel matrices are computed. For SP, we use equal length SPs, and for GK we use connected three minors. We also compare with the subgraph-feature-based approach of CORK [16][3], which is implemented in C++. For CORK, we use 10% minimum support to mine the frequent discriminative subgraphs. Recall that CORK uses the efficient gSpan [22] frequent graph mining algorithm, and then does feature selection. We also compared with a direct approach that uses all mined frequent subgraphs as binary features for classification; we used the Gaston [25] subgraph mining method for this.

In the discussion below, our graph feature approach is denoted as GF. GF was written in Python, with NumPy [26] and Networkx [27] modules for linear algebra and graph support. Note that both NumPy and Matlab use low-level C implementations for most matrix operations; therefore, the timing results are comparable[4], although there might be slight differences. We present results for three variants of the GF approach: GF(no) denotes the method using a raw or unnormalized feature vector, whereas GF(r) and GF(z) use range and z-score normalized feature vectors, respectively.

We use the libsvm [28] (support vector machine library) for all of the kernels and our method. The graph kernels use the kernel matrix computed via the particular graph kernel, whereas we use the default Gaussian or RBF kernel in libsvm: $\kappa(G_i, G_j) = \exp\left\{-\gamma \|F_i - F_j\|^2\right\}$, where $\gamma = \frac{1}{p}$, where p equals the number of features, which is 20 for all datasets, except those that do not have any labels on the nodes and edges. The latter include the cell-graph datasets, and the non-label versions of the other datasets; for these the number of features is $n' = 17$, since we do not use the label-based features (f_{18} to f_{20}). We also use a RBF kernel for CORK as that gave us the best results.

For each method, we perform ten runs of tenfold cross-validation, and we tune the C parameter, for C-SVM, using only the training folds. We will report the graph kernel matrix computation (for other methods), or for the graph feature computation (for GF). We will plot SVM training times for each method for some selected datasets as well. All the experiments were performed on MAC OS $\times$ 10.5 with two 2.66GHz Dual Core Intel Xeon processors, with 4GB 667MHz DDR2 memory.

For performance assessment, we report the average accuracy and standard deviation over the tenfold cross-validation run ten times. We also assess whether the accuracy of our method is significantly better or worse than the accuracy of the best previous method. For this we use the paired t-test for the difference between the accuracies in each fold. We report the value of the t-statistic, given as: $t = \frac{\sqrt{N}\mu}{\sigma}$, where μ and σ^2 denote the mean and variance of the difference in accuracy between the two methods, and $N = 100$ is the total sample size (ten runs times tenfolds). We fail to reject the null hypothesis, that there is no significant difference, if $t \in (-t_{\alpha/2,N-1}, t_{\alpha/2,N-1})$, where α is the significance level for a two-tailed t-test with $N - 1$ degrees of freedom. We use $\alpha = 0.05$ for which the interval is $(-1.98, 1.98)$. If the value of t-statistic is outside this interval, the two methods are statistically significantly different in performance.

4.2. Datasets

We used three different types of graph datasets: chemical compounds, proteins, and cell graphs. See Table 2 for statistics on the different graphs. The table shows the total size of each dataset, including the number of classes, the number of points in each class, the average and maximum number of vertices and edges for the graphs, and the average degree. Note that the chemical compounds and protein datasets have two classes, whose sizes are shown under the positive and negative column labels. The

[2] obtained from Prof. K. Borgwardt and N. Shervashide

[3] obtained from Marisa Thoma

[4] For performance comparison of NumPy and Matlab, see http://www.scipy.org/PerformancePython for instance.

Table 2. Benchmark datasets.

(a) Chemical compound datasets

Dataset	size (N)	Classes	Positive	Negative	Avg. $\|V\|$	avg.$\|E\|$	Max. $\|V\|$	Max. $\|E\|$	Avg. deg
MUTAG	188	2	125	63	17.7	38.9	28	33	2.19
NCI1	4 110	2	2 057	2 053	29.9	32.3	111	119	2.16
NCI109	4 127	2	2 079	2 048	29.7	32.1	111	119	2.16
PTC(MM)	336	2	129	207	25.0	25.4	109	108	1.98
PTC(FM)	349	2	143	206	25.2	25.6	109	108	1.99
PTC(MR)	344	2	152	192	25.6	26.0	109	108	1.99
PTC(FR)	351	2	121	230	26.1	26.5	109	108	1.99

(b) Protein datasets

Class	Size (N)	Classes	Positive	Negative	Avg. $\|V\|$	Avg.$\|E\|$	Max. $\|V\|$	Max. $\|E\|$	Avg. deg
D & D	1 178	2	691	487	284.3	715.7	5 748	14 267	4.98
CATH1	712	2	384	328	205.7	819.8	568	2356	7.79
CATH2	190	2	109	81	308.0	1 254.8	568	2 220	8.14

(c) Cell-graph datasets.

Tissue	Class	Size (N)	Avg. $\|V\|$	Avg.$\|E\|$	Max. $\|V\|$	Max. $\|E\|$	Avg. deg
Breast	Invasive (EI)	202	966.9	12 503.6	1956	51454	22.07
	Noninvasive (EN)	93	889.7	13 459.4	1 940	48 750	25.47
	Benign (EB)	151	829.4	15 677.7	1 885	49 165	34.72
Bone	Ost(OO)	49	532.2	2 324.7	2 855	18 790	5.42
	Frac(OF)	39	497.7	1 599.2	1 913	13 564	4.25
	Normal(ON)	20	174.2	1 174.0	612	7 309	8.14
Brain	Glioma(AG)	329	4 550.2	43 400.5	7 311	98 572	18.04
	Inflamation(AI)	107	42 44.1	39 457.7	7 113	90 029	17.06
	Benign(AB)	210	789.0	3 988.9	1 755	9 309	9.65

E = breast, O = bone, A = brain.

cell-graphs datasets have three classes, and their sizes are shown under the 'size(N)' column.

Chemical Compounds: The chemical compound datasets include MUTAG [29], NCI1 and NCI109 [30], and PTC[5], which have been employed as benchmark datasets in previous graph kernel papers. MUTAG is a dataset of mutagenic aromatic and heteroaromatic nitro compounds assayed for mutagenicity on bacterium *Salmonella typhimurium*. We used two balanced subsets of the NCI (National Cancer Institute) datasets. The class labels are based on an anti-cancer screen, as active or inactive. The PTC datasets record the carcinogenicity of several hundred chemical compounds for male rats (MR), female rats (FR), male mice (MM), and female mice (FM). As one can in Table 2(a), these graphs are very small (20–30 nodes and 25–40 edges) and sparse, with average degree around 2.

Proteins: The D&D dataset [31], which has also been used by previous studies, consists of 1178 proteins, with 691 enzymes and 487 nonenzymes. In addition, we created two new datasets from CATH[6], a manually curated database of protein domain structures. CATH1 consists of proteins in the same class (mixed alpha–beta), but having different architectures (alpha–beta barrel vs. two-layer sandwich). CATH2 has proteins in the same class (mixed alpha–beta), architecture (alpha–beta barrel), and topology (TIM barrel), but in different homology classes (aldolase vs. glycosidases). CATH2 is harder to classify, because proteins in the same topology class are structurally similar. The protein graphs are ten times larger in size than chemical compounds, with 200–300 nodes and 700–1250 edges (Table 2(b)), and average degree is 5–8. We use another variant of the CATH datasets, without the node labels (which correspond to the amino acids). We denote these as CATH1 (w/o L) and CATH2 (w/o L).

Cell-graphs: We also performed experiments on cancer cell-graph datasets [5,32,33]. These graphs were constructed from the histopathological samples from three different types of tissues: breast, bone, and brain. Within each type of tissue we consider three classes: healthy

[5] www.predictive-toxicology.org/ptc

[6] www.cathdb.info

Table 3. Accuracy (±Standard Deviation): Chemical Compound Datasets.

	MUTAG	NCI1	NCI109	PTC(MM)	PTC(FM)	PTC(MR)	PTC(FR)
GF(no.)	86.75 ± 6.38	67.18 ± 2.99	66.30 ± 1.67	60.99 ± 4.05	55.90 ± 6.40	51.15 ± 5.87	58.12 ± 6.52
GF(r)	87.11 ± 8.59	69.64 ± 1.69	69.44 ± 2.26	62.78 ± 6.83	63.90 ± 3.97	57.52 ± 6.63	**66.71 ± 6.47**
GF(z)	**91.37 ± 4.77**	75.62 ± 2.05	74.22 ± 1.66	63.38 ± 5.43	59.87 ± 6.13	**63.94 ± 7.05**	62.96 ± 6.65
RW	84.01 ± 6.61	–	–	60.58 ± 8.92	58.98 ± 9.72	51.40 ± 5.77	64.63 ± 8.74
SP	88.13 ± 7.15	73.82 ± 1.61	72.89 ± 2.17	57.52 ± 9.98	52.41 ± 9.79	58.46 ± 6.08	63.67 ± 5.27
GK	83.93 ± 6.48	69.18 ± 2.62	69.82 ± 1.89	58.04 ± 8.22	55.86 ± 8.95	55.19 ± 5.66	59.41 ± 5.36
RG	86.23 ± 4.41	–	–	64.30 ± 7.89	58.45 ± 7.01	57.61 ± 8.32	63.52 ± 6.40
WL	86.89 ± 6.33	**84.11 ± 1.61**	**83.50 ± 2.34**	**67.23 ± 5.87**	**64.37 ± 6.57**	58.10 ± 7.18	65.22 ± 5.34
CORK	86.19 ± 7.82	78.12 ± 1.61	77.76 ± 1.48	61.85 ± 8.04	57.90 ± 6.53	60.75 ± 7.31	65.51 ± 9.82
t-statistic	**4.02**	**−28.61**	**−29.27**	**−3.91**	−0.88	**3.26**	1.40

Notes: bold *t*-statistic means statistically significant.

(normal/benign), cancerous (invasive, osteo-sarcoma:Ost, Glioma), and damaged (noninvasive, fracture:frac, inflammation). We perform binary classification for each pair of classes within a tissue type. Usually, it is much easier to distinguish between normal and cancerous classes, but harder to classify cancerous versus damaged classes, for each tissue type. For example, for the breast tissue, classifying EB versus EN and EB versus EI is easier than classifying EI versus EB. In contrast to the chemical compounds and proteins, the cell graphs are even larger and denser (see Table 2(c)). Average graph size is five to ten times larger than proteins, with 200−4500 nodes and 100−43 000 edges. Average degree is 4−8 for bone, 10−18 for brain, and 22−35 for breast tissue. The cell graphs are unlabeled (i.e., no labels on nodes or edges).

4.3. Graph Kernel Comparison

4.3.1. Chemical compound datasets

Table 3 shows the accuracy comparison for our GF approach versus other graph kernels on the chemical compound datasets. Each cell records the average classification accuracy, as well as the standard deviation, over the tenfold cross-validation over ten different runs. Table 4 reports the wall clock running times for each method on the different datasets. A '–' in any cell means that the computation of the kernel matrix did not finish in one day, and thus the run was aborted. We can observe from the results that graph features with unnormalized/raw values, denoted GF(no), do not perform well in terms of accuracy. Furthermore, for these graphs, the z-normalized GF(z) approach delivers the best results, except for PTC(FM) and PTC(FR).

On the MUTAG dataset, GF(z) is the best overall method. Looking at the *t*-statistic, it is significantly better than the next best method, SP kernel, at a significance level of $\alpha = 0.05$, since $t \notin (-1.98, 1.98)$. Considering the time, we can see that GF takes only a fraction of the time compared to other methods. It is six times faster than SP.

On the NCI1 and NCI109 datasets, the WL subtree kernel has the best accuracy. The NCI datasets are quite tree-like; we can see in Table 2 that for both the average and maximum number of edges, we have $|E| \approx |V|$. Given that the WL kernel is a subtree kernel, it is well suited for such tree-like datasets. However, in terms of time, GF is over 25 times faster.

The PTC datasets are also tree-like, and thus the WL kernel performs well. However, while WL is the best method for MM, GF(z) is the best method for MR. These differences are statistically significant. On FM, the WL kernel has a slight advantage, whereas on FR data, GF(r) is slightly better, although there is no significant difference between them. In terms of computational time, GF method is vastly superior, being six times faster than WL. GK is

Table 4. Running times on chemical compound datasets.

	MUTAG	NCI1	NCI109	PTC(MM)	PTC(FM)	PTC(MR)	PTC(FR)
GF	**0.78s**	**36.48s**	**36.77s**	**2.30s**	**2.56s**	**2.66s**	**2.50s**
RW	5m3s	–	–	2h3m42s	2h16m11s	2h12m7s	2h17m28s
SP	4.61s	16m56s	21m2s	35.82s	35.79s	36.14s	37.72s
GK	1.42s	3m21s	3m25s	4.88s	5.05s	5.04s	5.22s
RG	42m54s	–	–	2h11m1s	2h16m54s	2h14m17s	2h20m6s
WL	5.88s	15m30s	16m1s	14.86s	16.22s	15.51s	16.10s
CORK	1m1s	33m19s	35m44s	19.92s	23.90s	23.03s	27.04s

h = hours, m = minutes, s = seconds.
Notes: Bold values indicate best performance.

Table 5. Accuracy (±standard deviation): chemical compound datasets without labels.

	MUTAG w/o L	NCI1 w/o L	NCI109 w/o L	PTC(MM) w/o L	PTC(FM) w/o L	PTC(MR) w/o L	PTC(FR) w/o L
GF(no.)	82.43 ± 7.51	65.72 ± 2.37	64.14 ± 1.83	55.03 ± 9.95	51.85 ± 6.46	51.99 ± 7.89	59.24 ± 6.87
GF(r)	87.28 ± 6.46	67.25 ± 2.25	66.63 ± 1.50	62.44 ± 8.37	63.29 ± 7.27	56.45 ± 8.44	**66.70 ± 6.02**
GF(z)	**87.78 ± 6.35**	71.36 ± 2.21	71.53 ± 1.33	63.39 ± 10.36	61.29 ± 4.87	**63.42 ± 8.55**	62.15 ± 7.68
GK	87.39 ± 6.96	62.75 ± 2.16	62.03 ± 2.50	61.02 ± 6.38	59.84 ± 8.66	56.61 ± 9.01	65.80 ± 6.17
WL	86.75 ± 5.82	**78.69 ± 3.33**	**77.03 ± 5.47**	**66.99 ± 7.62**	**64.73 ± 9.63**	57.01 ± 7.83	62.10 ± 7.40
CORK	86.78 ± 7.84	–	–	–	–	–	–

Notes: Bold values indicate best performance.

the second fastest method, but its accuracy is not very high. Compared with RW, SP, RG, and CORK methods, our GF approach is one to three orders of magnitude faster.

To study the effect of graph topology versus graph attributes/labels, we also compared the performance of the GF method versus the GK, WL, and CORK methods on unlabeled versions of the chemical compound datasets, as shown in Table 5. Unfortunately, CORK could not run on the unlabeled datasets (except for MUTAG) since too many frequent topological subgraphs are mined once labels are omitted, and the (sub)graph isomorphism checks become expensive. The main observation from these experiments is that GF's performance remains essentially the same between the labeled and unlabeled datasets, which indicates that the label information is not that helpful for GF; it relies primarily on the topological features. Also, WL does suffer when labels are omitted, especially for the NCI datasets. This suggests that for the NCI datasets labels are important to improve performance, and thus GF may also benefit if we can incorporate more effective label features (which is part of future work).

4.3.2. *Protein datasets*

The protein graphs are much larger compared with the chemical compound datasets. The accuracy and timing results are shown in Tables 6 and 7. In terms of accuracy, among the GF variants, the unnormalized version is the worst, whereas GF(r) gives the best results, except on the unlabeled D&D (w/o L) and CATH2(w/o L) datasets.

For the D&D enzyme dataset, the only kernel methods that finished within 24 h were GF, GK, WL, and CORK. Here WL has the best accuracy, but GF is about three times faster. Although CORK is four times faster, since it uses C++ and GF uses python, the timing comparison is not entirely fair. Also, it is worth noting that on the unlabeled versions of the protein datasets, CORK could not be run.

For the CATH datasets, we can see that GF is 2–5 times faster than WL, about 20 times faster than SP, 4–8 times faster than GK, and 10–20 times faster than CORK. RW and RG were not able to finish in the allotted time (1 day). For CATH1, we find that GF(r) has a slight (although not significant) advantage over other approaches in terms of accuracy. CATH1 is an easier dataset to classify, since the proteins in the two classes differ at the architecture level, and thus are structurally different. All the methods do well on this dataset. On CATH2 data, CORK gave the best overall results, and GF was significantly worse. CATH2 is a much harder dataset to classify from a structural viewpoint. This is because the two classes have significant structural similarity. On the other hand, there are possibly significant differences in the protein sequences between the two classes. GF mainly relies on topological graph features, and the three label features (f_{18} to f_{20}) are not able to capture the subsequence similarity (since they a re local and are not

Table 6. Accuracy (±standard deviation): (a) protein datasets, (b) protein datasets without labels.

	D&D	CATH1	CATH2
(a) Original Datasets			
GF(no)	62.99 ± 4.49	83.29 ± 4.98	61.58 ± 10.27
GF(r)	76.32 ± 2.72	**99.02 ± 0.90**	81.57 ± 5.39
GF(z)	75.95 ± 2.66	98.46 ± 1.32	79.27 ± 9.46
RW	–	–	–
SP	–	98.88 ± 1.37	96.32 ± 3.37
GK	75.13 ± 2.71	98.32 ± 0.84	94.74 ± 4.71
RG	–	–	–
WL	**78.29 ± 3.05**	98.59 ± 1.09	94.21 ± 4.97
CORK	71.22 ± 4.56	94.24 ± 2.77	**97.89 ± 2.58**
t-statistic	**−3.75**	0.07	**−26.75**

(b) Datasets without Node/Edge Labels

	D&D (w/o L)	CATH1(w/o L)	CATH2(w/o L)
GF(no)	62.48 ± 3.35	82.86 ± 5.43	61.05 ± 8.87
GF(r)	76.06 ± 3.31	**98.60 ± 1.54**	77.89± 7.74
GF(z)	**77.51 ± 5.08**	97.90 ± 1.57	**81.05 ± 3.49**
RW	–	–	–
SP	–	97.89 ± 1.70	76.32 ± 8.57
GK	69.27 ± 4.78	97.61 ± 2.52	66.84 ± 11.05
RG	–	–	–
WL	74.19 ± 2.60	98.59 ± 1.26	76.84 ± 8.22
CORK	–	–	–
t-statistic	**4.74**	0.15	**4.67**

Notes: Bold values indicate best performance, and bold t-statistic means statistically significant.

Table 7. Running times on protein datasets.

	D&D	CATH1	CATH2
GF	52m35s	**4m36s**	**2m15s**
RW	–	–	–
SP	–	1h42m12s	42m26s
GK	23h14m53s	37m8s	15m54s
RG	–	–	–
WL	2h12m57s	22m33s	4m45s
CORK	**14m10s**	41m28s	43m6s

h = hours, m = minutes, s = seconds.

designed to look at subsequences). GF is thus not able to distinguish between the two classes as well as the other kernels that can effectively utilize label information. To verify this hypothesis, we removed the node (amino acid) labels from the CATH datasets; thus, all methods have to rely only on topological information. We also include a comparison the unlabeled D&D dataset for completeness. CORK aborted on all the unlabeled proteins datasets, since the (sub)-graph isomorphism checks in this case become too expensive. We now find that GF(z) is significantly superior to other methods on CATH2(w/o L), and also on the D&D dataset. While the accuracy of GF(z) remains close to the labeled case, the other methods suffer a significant drop in accuracy. For example, WL has an accuracy of 94.21 on CATH2, but only 76.85 on CATH2(w/o L). This confirms two things: (i) there is significant sequence similarity between sequences in the same class, exploiting which helps the other methods and (ii) even though the CATH2 classes are topologically similar, there are enough structural differences that GF is still able to exploit. Likewise, GF can leverage the topological differences between the enzymes and nonenzymes for D&D.

4.3.3. *Cell-graph datasets*

Table 8 shows the accuracy comparison for GF versus other graph kernels on the cell-graph datasets. The corresponding timing results are shown in Table 9. One of the differences between cell-graph and the other datasets is that cell graphs are much larger (e.g., 4550 nodes and 43 400 edges for brain graphs). Furthermore, while the other datasets are labeled, the cell-graph data does not have any labels.

We show the results separately for each tissue type, and we show results for binary classification. Among the GF variants, GF(z) is usually better than GF(r), or is close to it. The unnormalized version is significantly worse. For GF, we use only the top $k = 2$ eigenvalues for bone (O), and $k = 100$ for brain (A). This is because computing all the eigenvalues for these large graphs is expensive.

For the cell-graph datasets, GF significantly outperforms all other methods in terms of accuracy, and also has an advantage in terms of time. For the largest graphs, from brain tissues, even GK and WL were not able to complete within a day of computation time. We also do not report the accuracies for CORK, since it aborted on the cell-graphs datasets. Because of the large graph size and the lack of labels, the graph mining step in CORK fails to enumerate any discriminative subgraphs.

Table 8. Accuracy (±standard deviation) on cell-graph datasets.

	EB vs. EI	EN vs. EB	EN vs. EI	OF vs. ON	ON vs. OO	OF vs. OO
GF(no)	57.78 ± 7.25	61.92 ± 5.21	64.48 ± 4.96	65.67 ± 20.76	71.19 ± 14.06	58.83 ± 16.59
GF(r)	87.70 ± 5.30	**86.35 ± 7.02**	82.76 ± 5.35	**98.33 ± 5.00**	**94.29 ± 7.00**	53.47 ± 22.08
GF(z)	**88.05 ± 4.27**	84.58 ± 6.96	**83.84 ± 4.41**	97.67 ± 6.67	92.86 ± 7.14	**63.75 ± 14.09**
RW	–	–	–	90.00 ± 11.06	76.43 ± 14.30	49.72 ± 14.28
SP	76.27 ± 5.16	77.61 ± 6.29	73.22 ± 6.14	94.67 ± 8.19	78.33 ± 14.57	60.67 ± 12.19
GK	66.01 ± 9.57	75.19 ± 8.13	62.38 ± 6.91	56.00 ± 21,12	60.71 ± 13.27	51.39 ± 13.57
RG	–	–	–	72.33 ± 12.52	67.04 ± 15.54	48.87 ± 14.07
WL	87.23 ± 4.57	74.81 ± 6.09	71.22 ± 8.15	**98.33 ± 5.00**	63.57 ± 17.63	61.25 ± 13.60
t-statistic	0.23	**9.31**	**12.74**	0	**12.28**	1.29

	AG vs AI	AG vs AB	AB vs AI
GF(no)	75.36 ± 5.06	61.04 ± 3.98	66.20 ± 8.36
GF(r)	**88.28 ± 3.40**	98.70 ± 1.45	99.06 ± 2.81
GF(z)	87.91 ± 2.86	**99.26 ± 0.91**	98.74 ± 2.87
RW	–	–	–
SP	–	–	–
GK	–	–	97.79 ± 2.01
RG	–	–	–
WL	–	–	**99.38 ± 1.88**
t-statistic	–	–	−0.14

Notes: Bold t-statistic means statistically Significant. E = breast, O = bone, A = brain.

Table 9. Running times on cell-graph datasets.

	EB vs. EI	EN vs. EB	EN vs. EI	OF vs. ON	ON vs. OO	OF vs OO
GF	**1h11m13s**	47m11s	**1h15m40s**	**23.7s**	1m26s	1m43s
RW	–	–	–	11m16s	18m14s	40m8s
SP	8h21m36s	5h44m23s	9h40m4s	19m8s	1h11m58s	1h27m2s
GK	14h12m4s	9h51m37s	11h46m28s	5m33s	9h47s	10m47s
RG	–	–	–	24.15s	**43.70s**	**1m29s**
WL	1h55m42s	**44m29s**	1h23m35s	55.30s	2m56s	2m32s
	AG vs AI	AG vs AB	AB vs AI			
GF	**17h50m5s**	**13h53m38s**	**5h15m44s**			
RW	–	–	–			
SP	–	–	–			
GK	–	–	9h11m12s			
RG	–	–	–			
WL	–	–	7h24m29s			

h = hours, m = minutes, s = seconds.

The accuracy of the competing methods depends on the two classes being compared. WL has comparable accuracy only on the easier to classify pairs, namely benign and cancerous breast graphs (EB vs. EI), normal and fractured bone graphs (OF vs. ON), and benign and inflamed brain graphs (AB vs. AI). On the other hand, on the other hard pairs, such as inflamed/noninvasive/damaged versus cancer (EN vs. EI, OF vs. OO, and AI vs. AG), our graph feature approach is significantly superior. For example, on EN vs. EI, GF(z) has an accuracy of 83.84, whereas SP has an accuracy of 73.22, WL has an accuracy of 71.22 and GK has an even lower value (62.38).

It is interesting to note that since cell-graph datasets do not have labels, all the kernels can only use structural information for computing the kernel matrix. The fact that GF has the best accuracies implies that the topological attributes we compute are well suited to extract discriminating features among the graphs from different classes. In fact, these attributes are much better at capturing the topological differences than the corresponding kernels based on subtrees, graphlets, SPs, and random walks, especially on large, unlabeled graphs, such as the cell graphs.

4.3.4. *SVM training times*

Figure 4 plots SVM training times for the different methods on four selected datasets: MUTAG, PTC(MR), NCI1, and CATH1. They all use the same libsvm package. For MUTAG, the training times for GF and CORK are about five times faster than other graph kernel methods, which are all relatively close to each other. For PTC(MR), there are significant differences in the training times. GF still has the least computational time, whereas the Graphlet kernel had the longest time. GF is over four times faster than the WL kernel. For NCI1, since the computation of RW and RG kernel matrix did not finish within a day, we excluded them from the plot. GF retains the fastest SVM training speed. Finally on the CATH1 dataset, GF(z) and CORK are the fastest, whereas the WL kernel is the slowest, being over ten times slower than GF(z). Figure 4 shows that GF methods with normalization give rise to easily optimized SVM classifiers compared with other graph kernel methods.

4.3.5. *Scalability study*

To study the scalability of our GF approach, we selected some datasets from each of the three groups: NCI1, PTC(FR), CATH2, and OF-OO. Next, we replicate the instances 10, 50, and 100 times, and we compare with the original replication factor of 1. For instance, NCI1 dataset with replication factor 100 has $100 \times 4110 = 411\,000$ instances. Figure 5 shows the time it takes to extract the GF topological features for the four selected datasets. As expected, the runtime of GF is linear in the number of graph instances.

4.4. Frequent Subgraph Features

We evaluated the effectiveness of using frequent subgraph features for graph classification. Recall that the CORK method first uses the gSpan algorithm [22] to mine frequent subgraphs, and then uses feature selection to create the final classifier. CORK's performance on chemical compound datasets appears in Table 3. We also employed the Gaston [25] frequent subgraph miner to obtain all frequent subgraphs at 10% minimum support (same as that used for CORK/gSpan). Next, we convert each subgraph into a binary feature, noting its presence or absence in

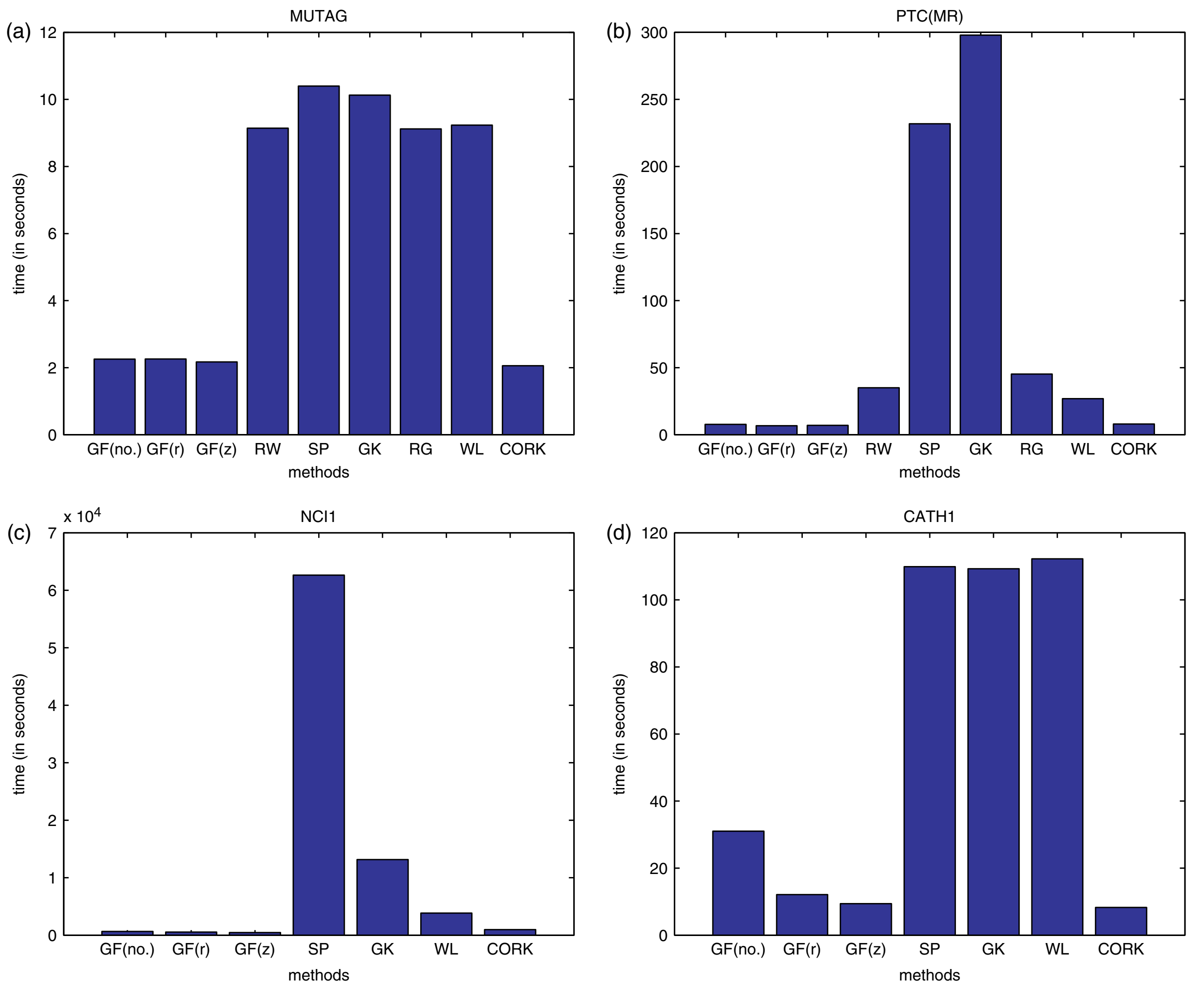

Fig. 4 SVM training times. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

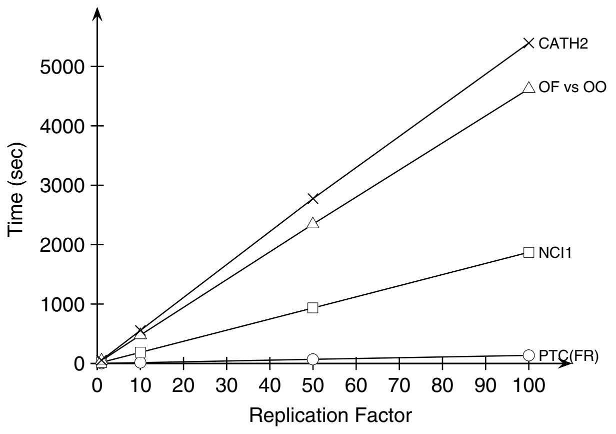

Fig. 5 Scalability study: GF(z) time for various replication factors.

each training/testing example, without doing any feature selection. Table 10 shows the accuracy of this approach on the chemical compound datasets. We find that except for PTC(FR), this approach is not as good as the WL or GF methods. Interestingly, the Gaston approach is typically better than CORK, even though it does not do any feature selection. It also is the fastest among all approaches in terms of time; its efficient C++ implementation finishes in under 2s for all the chemical compound datasets (although it is not fair to compare it with the Matlab/Python implementations). Unfortunately, Gaston was not able to mine all the frequent subgraphs for the protein and cell-graph datasets (at 10% minimum support threshold) within the 24 h time threshold. For some higher support values (e.g., 30%) Gaston was able to run, but it outputs so many patterns that the SVM classification step failed (it exceeded the 4 GB memory). Combined with the fact that even CORK cannot be run on the cell-graph and other unlabeled datasets, the whereas subgraph features can help for labeled graphs, they are not effective for unlabeled graphs, where GF is particularly strong.

Table 10. Frequent subgraphs as binary features.

	MUTAG	NCI1	NCI109	PTC(MM)	PTC(FM)	PTC(MR)	PTC(FR)
GASTON	81.87 ± 5.58	80.92 ± 1.45	81.00 ± 1.37	61.35 ± 6.91	59.30 ± 9.24	63.14 ± 8.69	**68.64 ± 8.90**

Table 11. Feature rankings-range normalization.

F	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
MUTAG	4				3							2	1			5				
NCI1								5	2				4				1		3	
NCI109									2				4	5			1		3	
PTC(MM)	4	1		3		2				5										
PTC(FM)		1				3	5					4		2						
PTC(MR)		1					3						4	2				5		
PTC(FR)		1								5	2	3				4				
D&D						2				4				3			5	1		
CATH1		5									4	1				2	3			
CATH2												4				2	5	1		3
CATH1(w/o L)		5									4	1				2	3	–	–	–
CATH2(w/o L)										4	2	5				3	1	–	–	–
EB vs EI	1		5								4			3			2	–	–	–
EN vs EB	3	1					4				5						2	–	–	–
EN vs EI		3	5								2		1	4				–	–	–
OF vs ON		1						3		5	2						4	–	–	–
ON vs OO		1							4	5	2						3	–	–	–
OF vs OO	3					4	2	1			5							–	–	–
AG vs AI	5	2							1		3					4		–	–	–
AG vs AB		4	1			3		5			2							–	–	–
AB vs AI		4	1			5		2			3							–	–	–
counts	6	**13**	4	1	1	6	4	5	4	6	**13**	**7**	5	6	0	**7**	**11**	3	2	1

4.5. Feature Importance Study

We carried out a detailed study to rank the different graph features, with the goal to identify which features carry the most information. We used the SVM-wrapper feature selection method [34], which ranks the features via recursive feature elimination using SVM. In Tables 12 and 11 we record the ranks of the top-k effective features, for $k = 5$. The last row of the table notes the number of occurrences of each feature in the top five list. While it is clear that the most informative graph attributes are dataset dependent, some general conclusions can still be made. For instance, for GF with range normalization, the top five features (shown in bold in Table 11) based on the number of occurrences include: (i) average clustering coefficient (f_2), (ii) number of nodes (f_{11}), (iii) number of eigenvalues (f_{17}), (iv) number of edges (f_{12}), and (v) energy (f_{16}). For GF with z-normalization, the top-five-occurrences-based features (shown in bold in Table 12) are: (i) number of nodes (f_{11}), (ii) average degree (f_1), (iii) average clustering coefficient (f_2), (iv) number of edges (f_{12}), and (v) number of eigenvalues (f_{17}). Thus, good discriminating features depend on both the graph datasets and different normalization methods. We still observe some high level trends in the rankings from the tables. The spectral attributes (f_{13}, f_{14}, f_{16}, f_{17}) are generally quite effective. The SP-based attributes (f_3, f_4, f_5, f_6, f_7) do not appear to be that effective compared to the spectral attributes, especially for range normalization. f_1 (average degree) and f_2 (average clustering coefficient) are also generally good features. f_8 (giant connected ratio), f_9 (percentage of isolated points), f_{10} (percentage of end points) might be effective if the graphs in the dataset contain several components, e.g. bone tissues dataset. As for the label-based features, for half of the datasets (five of ten) in which graphs have node labels, the feature f_{18} (label entropy) is in the top five. Note that none of them select f_{15} (trace) as an effective discriminating feature, since none of the graphs in our experiments contain loops. A detailed dataset specific analysis of feature importance is given next.

Chemical Compounds Datasets: We first take a look at range normalization in Table 11, for the chemical compound datasets: MUTAG, NCI1 and NCI109, and PTC.

For MUTAG, the top five features are f_{13} (spectral radius), f_{12} (number of edges), f_5 (effective radius), f_1 (average degree), and f_{16} (energy).

The NCI datasets all have similar top five feature rankings among themselves, i.e., the first four features are the same, f_{17} (number of eigenvalues), f_9 (percentage

Table 12. Feature rankings–z-normalization.

F	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
MUTAG	3				2								1					4		5
NCI1								3			2						1	4		5
NCI109				3							2	4					1		5	
PTC(MM)		1	5			4					3	2								
PTC(FM)		5	2			1	4										3			
PTC(MR)	2												4	1				5		3
PTC(FR)		1								3	4					5	2			
D&D	3				5						4	1		2						
CATH1											2	3				4	1	5		
CATH2			4	3		1					5							2		
CATH1(w/o L)				5							2	3				4	1	–	–	–
CATH2(w/o L)			5	2		1				4	3							–	–	–
EB vs EI	1										2	3				4	5	–	–	–
EN vs EB	1	3											4	2			5	–	–	–
EN vs EI	1	5									2	3	4					–	–	–
OF vs ON	3	1						5			2			4				–	–	–
ON vs OO	3	1		5							2			4				–	–	–
OF vs OO	4		5				2	1				3						–	–	–
AG vs AI	5	1						3			2			4				–	–	–
AG vs AB	4	3					2		5		1							–	–	–
AB vs AI	4	2	1						3		5							–	–	–
counts	**12**	**10**	6	5	2	4	3	4	2	2	**16**	**8**	4	6	0	4	**8**	5	1	3

of isolated points), f_{19} (neighborhood impurity), and f_{13} (spectral radius), except for the fifth feature: for NCI1 it is f_8 (giant connected ratio) and for NCI109 it is f_{14} (second largest eigenvalue). Note that both range and z-normalization choose f_{17} (number of eigenvalues) as the top feature for these tree-like datasets. There is rich literature showing that the eigenvalues of a graph capture many topological properties. For example, multiplicity of eigenvalues usually corresponds to symmetries in the graph [35] (although the correspondence is not exact). The possible explanation for f_{17} being selected by the NCI graphs is that one class has more balanced tree-like structures than the other class.

However, the PTC graphs have quite different feature rankings with each other. It is interesting to see that all of them put f_2, average clustering coefficient, as the first rank. But since PTCs are chemical compounds, most graphs have zero triangles. Thus most graphs will have f_2 feature value 0. Using only average clustering coefficient as the feature for classification, the accuracies for GF(r) are: PTCMM (61.68 ± 6.83), PTCFM (59.02 ± 6.05), PTCMR (55.83 ± 7.83), and PTCFR (65.53 ± 9.32), which are close to the accuracies by using the full 20 features for classification (see Table 3). One possible explanation is that since positive or negative graphs have f_2 value 0, an SVM classifier will not be able to discriminate between them. Rather, the SVM will classify all of them as either positive or negative. Since the PTC datasets are balanced datasets to some extent (with approximately 40% positives and 60% negatives), the classifier accuracies are expected to around 60%. The SVM-wrapper feature selection method also shows that f_2 is an informative graph attribute. Other attributes are less informative or even redundant. Nevertheless, considering them together still improves the accuracies by 1–4%, compared to using f_2 alone (see Table 3).

For the z-normalization results in Table 12, we see that for MUTAG, the top five features are f_{13} (spectral radius), f_2 (average clustering coefficient), f_1 (average degree), f_{18} (label entropy), and f_{20} (link impurity). For the two NCI graphs, there are two features selected by both: f_{11} (number of nodes) and f_{17} (number of eigenvalues). For the four PTC graphs, f_2 (average clustering coefficient) is selected by three datasets. Note that f_5 (effective radius), f_8 (giant connected ratio), f_9 (percentage of isolated points), f_{15} (trace), and f_{19} (neighborhood impurity) are not selected either by range or z-normalization. Since chemical compound usually contain only one component and form a simple graph, some features such as isolated points, giant connected ratio and trace have little discriminating power.

Proteins: The protein datasets include D&D and CATH1 and CATH2 (with and without labels). In Table 11, these five datasets all put feature f_{17} (number of eigenvalues) in the top five, whereas four of them have f_{16} (energy) and f_{12} (number of edges) in the top five, and three choose f_{11} (number of nodes). Compared with chemical compounds, protein datasets (graphs) are relatively large. Thus some simple statistics, e.g. number of nodes and number of edges might be more effective and can have more discriminating power. The dimension of the adjacency matrix for each graph is larger compared to chemical compound datasets, which increases the value of the energy feature.

In Table 12, for z-normalization, besides all of the protein datasets choosing f_{11} (number of nodes) as an important feature, three in five consider f_4 (effective diameter) and f_{12} (number of edges) important for classification. For D&D dataset, both the range and z-normalization methods select f_{14} (the second largest eigenvalue).

Cell graphs: Cell graphs are the largest graphs used in our experiments. The datasets have three different types of cancerous tissues: breast, bone, and brain. Each has three binary classification tasks. With range normalization (Table 11) all of them consider f_{11} (number of nodes) and nearly all of them (seven in nine) put f_2 (average clustering coefficient) as important discriminating features.

For breast-cancer datasets, besides the features mentioned above, two in three select f_1 (average degree), f_3 (average effective eccentricity), f_{14} (second largest eigenvalue), and f_{17} (number of eigenvalues) among the top five features. For bone-cancer datasets, two in three select f_8 (giant connected ratio), f_{10} (percentage of end points), and f_{17} (number of eigenvalues) in the top five. Note that each graph in bone has multiple connected components and the largest number of components of one graph is 158 (including isolated points). Thus, f_8 and f_{10} become important discriminating features here. For brain-cancer datasets, besides the features mentioned above (f_{11}, f_2), two in three select f_3 (average effective eccentricity), f_6 (closeness centrality), and f_8 (giant connected ratio). It is known that different types of tissues in brain have different cellular density levels [33]. The cancerous tissues (glioma) have higher cellular density while the healthy tissues (benign) have lower cellular density. At the same time, cancerous tissues and damaged tissues (inflammation) have equally high cellular density. Hence, on AG versus AB and AB versus AI, except for the node-based features (f_{11} and f_2), some path-based features such as f_3 (average effective eccentricity) and f_6 (closeness centrality) also show discriminating power.

For z-normalization (Table 12), all of the cell-graph datasets choose f_1 (average degree) and almost all of them (seven in nine) choose f_2 (average clustering coefficient) and f_{11} (number of nodes), as important features. Figure 6 plots the frequency distributions for the average degree (f_1) and average clustering coefficient (f_2) for the three breast cancer cell-graph classes (invasive: EI, noninvasive: EN, and benigh: EB). One can observe that there are significant differences among the class-specific distributions, which help discriminate them in our GF approach.

It is worth remarking that while f_{11} (number of nodes) and f_{12} (number of edges) are good discriminating features (they are usually in the top five), the accuracy is not significantly different even without these features, as shown in Table 13. The table shows the performance of GF with and without these two features: GF is with the original set of 17 features (discounting the label-based features $f_{18} - f_{20}$, since Cell-graphs are unlabeled), whereas GF* is with 15 features (with f_{11} and f_{12} removed). We can see that the differences in accuracies are not significant, except for ON versus OO, where the drop in accuracy is approximately 6%. Note that in over half the cases there is even a slight increase in accuracy for GF*, with the reduced set of features. For OF versus OO there is a 3.5% increase for GF*. One possible explanation is that features are not mutually independent and some missing features could be compensated to some extent by other features. In any case, even without the two simplest features, namely number of nodes and edges, GF* method is superior to other graph kernels on the cell-graph datasets (see Table 8).

4.6. Augmented Topological Features

To further examine the effect of additional topological features, we augmented the initial 17 features (f_1 to f_{17}, mentioned in Section 3), with an additional 10 global graph features, for a total of 30 global features, including the 3 label-based features (f_{18} to f_{20}). These augmented features are described below.

f-21. **Eigen-exponent:** It is defined as the slope of the best-fitting line for the decreasing eigenvalues, i.e. λ_i versus i, in a log–log plot.

f-22. **Hop-plot exponent:** The hop-plot value reflects the size of the neighborhood between any two nodes. Let $R(h)$ denote the number of node pairs that are reachable within h hops. The hop-plot exponent is the slope of best-fitting line in a log–log plot of the number of reachable pairs $R(h)$ as a function of h.

f-23. **Averaged current-flow closeness centrality:** A variant of closeness centrality based on effective resistance between nodes in a network [36].

f-24. **Degree assortativity coefficient:** Assortativity measures the similarity of connections in the graph with respect to the node degree. It is essentially the same as the Pearson correlation coefficient of degrees between pairs of adjacent nodes [37].

f-25. **Number of cliques:** This is the number of maximal cliques in each graph.

f-26. **Average neighbor degree:** First, we compute the average degree of the neighborhood of each node.

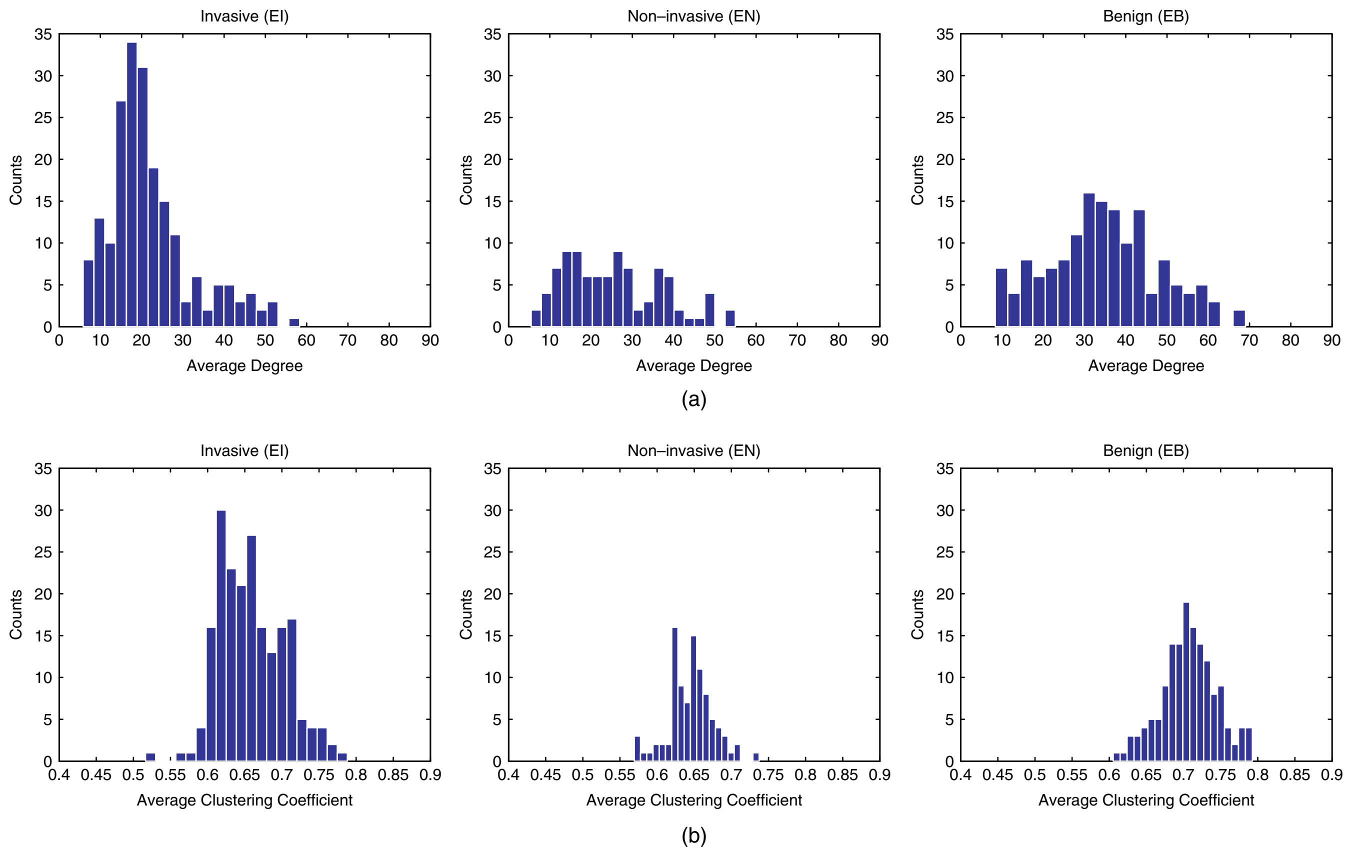

Fig. 6 Breast cell-graphs: class-0specific average degree (a) and clustering coefficient distribution (b). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Then, for the whole graph we take the average of these values over all nodes.

f-27. **Transitivity:** Defined as the fraction of all possible triangles in each graph. A possible triangle is a triple which has two edges sharing one vertex.

f-28. **Periphery:** The periphery is the set of nodes with eccentricity equal to the effective diameter. We compute the fraction of nodes that comprise the periphery in each graph.

f-29. **Cycle basis:** A graph may have multiple cycles. However, each cycle in the graph can be decomposed into the sum (defined as 'exclusive or' of the edges) of several cycles. We call a minimal collection of such cycles as a cycle basis. The cardinality of cycle basis is selected as a global feature.

f-210. **Square clustering coefficient:** While clustering coefficient by triangles give the likelihood that any two neighbors of u are connected, the square clustering coefficient gives the probability that two neighbors of node v share a common neighbor different from v.

Tables 14 and 15 give the accuracy and running time, respectively, for the GF methods, with and without the augmented features. GF(20) denotes the use of the original 20 features, and GF(30) includes the augmented features. The results are shown on selected datasets from each group, i.e. MUTAG and PTC(MM) from chemical compounds, D&D and CATH2(w/o L) from proteins, and EN versu EI from cell graphs. We observe that the augmented features improve the performance for CATH2 (w/o L) to some extent, with not much difference for the other datasets. However, interestingly, on the cell-graph dataset EN−EI, the use of the additional features significantly reduces the accuracy. This may be because the other features already capture most of the global topological properties for the cell-graph dataset. Also, these augmented features can be expensive to compute, being typically three times slower than the 20 initial features. Combined with the feature importance study above, we conclude that the simpler

Table 13. Accuracy (±standard deviation) on cell-graph datasets.

	EB vs. EI	EN vs. EB	EN vs. EI	OF vs. ON	ON vs. OO	OF vs. OO
GF(no)	57.78 ± 7.25	61.92 ± 5.21	64.48 ± 4.96	65.67 ± 20.76	71.19 ± 14.06	58.83 ± 16.59
GF(r)	87.70 ± 5.30	**86.35 ± 7.02**	82.76 ± 5.35	**98.33 ± 5.00**	**94.29 ± 7.00**	53.47 ± 22.08
GF(z)	**88.05 ± 4.27**	84.58 ± 6.96	**83.84 ± 4.41**	97.67 ± 6.67	92.86 ± 7.14	**63.75 ± 14.09**
GF*(no)	57.79 ± 4.92	61.90 ± 7.02	68.40 ± 7.36	62.67 ± 12.45	66.43 ± 15.00	55.33 ± 16.38
GF*(r)	**86.99 ± 6.60**	**86.87 ± 8.14**	**84.40 ± 5.31**	**95.00 ± 7.64**	87.14 ± 11.87	56.81 ± 16.30
GF*(z)	86.41 ± 5.19	81.97 ± 5.48	84.10 ± 6.75	**95.00 ± 10.67**	**88.57 ± 12.45**	**67.22 ± 15.17**

	AG vs. AI	AG vs. AB	AB vs. AI
GF(no)	75.36 ± 5.06	61.04 ± 3.98	66.20 ± 8.36
GF(r)	**88.28 ± 3.40**	98.70 ± 1.45	**99.06 ± 2.81**
GF(z)	87.91 ± 2.86	**99.26 ± 0.91**	98.74 ± 2.87
GF*(no)	75.02 ± 4.55	81.63 ± 4.10	67.56 ± 10.67
GF*(r)	**87.83 ± 4.76**	98.70 ± 1.86	99.05 ± 1.45
GF*(z)	86.93 ± 4.42	**99.44 ± 0.85**	**99.38 ± 1.25**

E = breast, O = bone, A = brain, GF* = features f_{11}, f_{12} removed. The best results for GF and GF* are shown in bold.

Table 14. Accuracy (± standard deviation) on selected datasets.

	MUTAG	PTC(MM)	D&D	CATH2(w/o L)	EN vs EI
GF(r)(20)	87.11 ± 8.59	62.78 ± 6.83	76.32 ± 2.72	77.89 ± 7.74	87.70 ± 5.30
GF(z)(20)	**91.37 ± 4.77**	**63.38 ± 5.43**	75.95 ± 2.66	81.05 ± 3.49	**88.05 ± 4.27**
GF(r)(30)	87.22 ± 9.20	63.37 ± 5.11	**76.74 ± 2.60**	80.00 ± 5.67	80.38 ± 5.76
GF(z)(30)	89.91 ± 5.48	61.60 ± 5.87	75.97 ± 3.42	**83.16 ± 7.74**	82.74 ± 3.40

topological features are typically sufficient to capture most of the important structural properties, and it is not that beneficial to include too many complex topological attributes such as cliques, square clustering, cycles.

5. CONCLUSIONS

We propose a simple yet effective and efficient graph classification approach that is based on topological and label graph attributes. The graph dataset is converted into a feature-vector dataset, which can be classified easily using any classifier. Our main idea is that graphs from the same class should have similar attribute values. On the basis of an extensive comparison with state-of-the-art graph kernel classifiers, we show that our approach yields competitive or better accuracies and has typically much lower computational times. Our conclusion is that graph attributes are effective in capturing discriminating structural information from different classes. While no method is uniformly the best, our approach is particularly effective for unlabeled graphs. Combining our graph features, with the best features from other approaches, such as the WL kernel, has the potential to yield even better methods, especially for labeled graphs.

This work opens up fruitful directions for future research. First, we would like to consider features that are more local, instead of the mainly global ones we consider in this paper. One approach to achieve this is to use the complete local distribution (e.g., degree distribution), as a complex feature in classification, instead of computing the average (e.g., average degree) like we do in the global approach. Second, we would like to construct better label attributes. We have exploited only three labeled features so far, and it is clear that some datasets such as the anticancer chemical compounds (NCI) and protein structural classes (CATH2) can benefit from richer label-based attributes. Third, we would like to exploit additional features from the other graph kernels (e.g., WL kernel). Finally, we would like to understand which graph and kernel features are the most

Table 15. Running times on selected datasets.

	MUTAG	PTC(MM)	D&D	CATH2 (w/o L)	EN vs EI
GF(20)	0.78s	2.30s	52m35s	2m15s	1h15m40s
GF(30)	3.04s	8.52s	1h26m35s	6m18s	3h24m59s

h = hours, m = minutes, s = seconds.

informative in terms of classification, and eventually, even for clustering.

ACKNOWLEDGMENTS

We thank Prof. Karsten Borgwardt and Nino Shervashidze for providing the Matlab implementation for the various graph kernel methods, and Marisa Thoma for the CORK implementation. This work was supported in part by NSF grant EMT-0829835, and NIH grant 1R01EB0080161-01A1.

REFERENCES

[1] T. Joachims, T. Hofmann, Y. Yue, and C.-N. Yu, Predicting structured objects with support vector machines, Commun ACM 52(11) (2009), 97–104.

[2] L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi, Graph kernels for chemical informatics, Neural Netw 18 (2005), 1093–1110.

[3] P. Mahè and J.-P. Vert, Graph kernels based on tree patterns for molecules, Mach Learn 75(1) (2009), 3–35.

[4] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel, Protein function prediction via graph kernels, In ISMB (Supplement of Bioinformatics), 2005, 47–56.

[5] C. Bilgin, C. Demir, C. Nagi, and B. Yener, Cell-graph mining for breast tissue modeling and classification, In 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2007, 5311–5314.

[6] B. Scholkopf and A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, Cambridge, MA, MIT Press, 2001.

[7] V. N. Vapnik, The Nature of Statistical Learning Theory, New York, Springer-Verlag, 1995.

[8] T. Gärtner, P. Flach, and S. Wrobel, On graph kernels: hardness results and efficient alternatives, In 16th Annual Conference on Computational Learning Theory, 2003, 129–143.

[9] H. Kashima, K. Tsuda, and A. Inokuchi, Marginalized kernels between labeled graphs, In International Conference on Machine Learning, 2003, 321–328.

[10] K. M. Borgwardt and H.-P. Kriegel, Shortest-path kernels on graphs, In 5th IEEE International Conference on Data Mining, Washington, DC, 2005, 74–81.

[11] T. Horvàth, T. Gärtner, and S. Wrobel, Cyclic pattern kernels for predictive graph mining, In 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, 158–167.

[12] J. Ramon and T. Gärtner, Expressivity versus efficiency of graphs kernels, In 1st International Workshop on Mining Graphs, Trees and Sequences, 2003.

[13] N. Shervashidze and K. M. Borgwardt, Fast subtree kernels on graphs, In Advances in Neural Information Processing Systems, 2009, 1660–1668.

[14] N. Shervashidze, S. V. Vishwanathan, T. H. Petri, K. Mehlhorn, and K. M. Borgwardt, Efficient graphlet kernels for large graph comparison, In 12th International Conference on Artificial Intelligence and Statistics, 2009, 488–495.

[15] I. R. Kondor, N. Shervashidze, and K. M. Borgwardt, The graphlet spectrum, In International Conference on Machine Learning, Vol. 382, A. P. Danyluk, L. Bottou, and M. L. Littman, eds., 2009, 67.

[16] M. Thoma, H. Cheng, A. Gretton, J. Han, H.-P. Kriegel, A. Smola, L. Song, P. S. Yu, X. Yan, and K. M. Borgwardt, Discriminative frequent subgraph mining with optimality guarantees, Stat Anal Data Mining 3(5) (2010), 302–318.

[17] S. V. N. Vishwanathan, K. M. Borgwardt, and N. N. Schraudolph, Fast computation of graph kernels, In Neural Information Processing Systems, B. Schölkopf, J. Platt, and T. Hoffman, eds., Cambridge, MA, MIT Press, 2006, 1449–1456.

[18] I. R. Kondor and J. Lafferty, Diffusion kernels on graphs and other discrete structures, In International Conference on Machine Learning, 2002, 315–322.

[19] G. Li, M. Semerci, B. Yener, and M. J. Zaki, Graph classification via topological and label attributes, In 9th Workshop on Mining and Learning with Graphs (with SIGKDD), 2011.

[20] P. Mahè, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert, Extensions of marginalized graph kernels, In International Conference on Machine Learning, 2004.

[21] S. V. N. Vishwanathan, N. N. Schraudolph, I. R. Kondor, and K. M. Borgwardt, Graph kernels, J Mach Learn Res 11 (2010), 1201–1242.

[22] X. Yan and J. Han, gspan: Graph-based substructure pattern mining, In IEEE International Conference on Data Mining, 2002.

[23] C. Meyer, Matrix Analysis and Applied Linear Algebra, Philadelphia, PA, SIAM, 2000.

[24] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. Vorst, Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Philadelphia, SIAM, 2000.

[25] S. Nijssen and J. Kok, A quickstart in frequent structure mining can make a difference, Proceedings of the 2004 ACM SIGKDD international conference o n Knowledge discovery and data mining, 2004, 647–652.

[26] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001. Available at http://www.scipy.org.

[27] A. A. Hagberg, D. A. Schult, and P. J. Swart, Exploring network structure, dynamics, and function using NetworkX, In Proceedings of the 7th Python in Science Conference (SciPy2008), Pasadena, CA,2008, 11–15.

[28] C.-C. Chang and C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans Intell Syst Technol 2 (2011), 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[29] A. Debnath, R. L. de Compadre, G. Debnath, A. Shusterman, and C. Hansch, The structure-activity relationship of mutagenic aromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity, J Med Chem 34 (1991), 786–797.

[30] N. Wale and G. Karypis, Comparison of descriptor spaces for chemical compound retrieval and classification, In IEEE International Conference on Data Mining, 2006.

[31] P. Dobson and A. J. Doig, Distinguishing enzyme structures from non-enzymes without alignments, J Mol Biol 330(4) (2003), 771–783.

[32] C. C. Bilgin, P. Bullough, G. E. Plopper, and B. Yener, Ecm-aware cell-graph mining for bone tissue modeling and classification, Data Mining Knowl Discov 20 (2010), 416–438.

[33] C. Demir, S. H. Gultekin, and B. Yener, Learning the topological properties of brain tumors, IEEE/ACM Trans Comput Biol Bioinform 2(3) (2005), 262–270.

[34] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, Gene selection for cancer classification using support vector machines, Mach Learn 46(1) (2002), 389–422.

[35] L. Lovász and K. Vesztergombi, Geometric representations of graphs, Paul Erdos and his Mathematics, 1999.

[36] U. Brandes and D. Fleischer, Centrality measures based on current flow, STACS 2005, 2005, 533–544.

[37] M. Newman, Networks: An Introduction, New York, NY, Oxford University Press, 2010.