Turkish Journal of Electrical Engineering and Computer Sciences

Volume 25 | Number 3 Article 69

1-1-2017

Edge distance graph kernel and its application to small molecule classification

MEHMET TAN

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 2479 – 2490

doi:10.3906/elk-1603-323

Edge distance graph kernel and its application to small molecule classification

Mehmet TAN*
Department of Computer Engineering, TOBB University of Economics and Technology, Ankara, Turkey

Received: 31.03.2016 • **Accepted/Published Online:** 20.09.2016 • **Final Version:** 29.05.2017

Abstract: Graph classification is an important problem in graph mining with various applications in different fields. Kernel methods have been successfully applied to this problem, recently producing promising results. A graph kernel that mostly specifies classification performance has to be defined in order to apply kernel methods to a graph classification problem. Although there are various previously proposed graph kernels, the problem is still worth investigating, as the available kernels are far from perfect. In this paper, we propose a new graph kernel based on a recently proposed concept called edge distance-k graphs. These new graphs are derived from the original graph and have the potential to be used as novel graph descriptors. We propose a method to convert these graphs into a multiset of strings that is further used to compute a kernel for graphs. The proposed graph kernel is then evaluated on various data sets in comparison to a recently proposed group of graph kernels. The results are promising, both in terms of performance and computational requirements.

Key words: Graph kernels, graph classification, chemical compound classification

1. Introduction

Graphs gain more importance with the continuous growth of structured data, including the World Wide Web, social networks, biological networks, program flows, and biological molecules. All these data can be represented as graphs composed of nodes, individual entities of interest, and edges, with the connections depicting the interactions or relationships among the entities. Devising fast and accurate algorithms for this kind of data is essential for correctly interpreting a highly connected society, which is usually modeled as a social network, and for predicting a specific outcome (or behavior) of a network (or graph) by exploiting the previously known outcome of similarly structured networks. The research that extracts such useful information from graphs is known as graph mining. One of the most important topics in graph mining is the classification of graphs. Grouping a given set of graphs into a set of predefined categories is called graph classification and is of utmost importance in several fields such as drug discovery [1], software verification, extraction of similar code [2], or clustering of people in a social network [3]. Kernel methods [4], such as support vector machines (SVMs), have been widely and successfully applied to classification problems including graph classification [5]. One of the most active fields in this area is the construction of efficient, reliable, and easy-to-implement graph kernels that can be used effectively in classification. Despite the variety of the proposed kernels, current kernels suffer either from performance issues or computational requirements. There is no single best kernel that can be applied to all domains and problems.

A large portion of the graph classification problems include moderately sized graphs. Among other

*Correspondence: mtan@etu.edu.tr

problems, the classification of small molecules, such as chemical compounds, is important, as the prediction of some properties, such as solubility and toxicity of drug candidate compounds, is essential for drug discovery. Additionally, in silico determination of the functions of some small proteins is needed, as they play important roles in various biological processes [6]. In this paper, we propose a new graph kernel based on the edge distance-k graphs introduced in [7]. An edge distance-k graph depicts the edges that are of distance k to each vertex, depending on a vertex-edge distance measure. We propose a descriptor based on a set of edge distance-k graphs corresponding to a set of k values. Then, based on this representation of graphs, we propose a graph kernel that performs better than certain state-of-the-art kernels, evaluated in terms of both performance and computational requirements. The results demonstrate the applicability and effectiveness of the kernel.

In the next section, we give a brief overview of notation and background. Section 3 describes the proposed graph kernel and Section 4 reports and discusses the experimental results. We conclude in Section 5 and give some future research directions.

2. Background

2.1. Related work

Various different graph kernels have been proposed, exploiting different features of graphs such as random walks or paths [8–10], subtrees [11,12], and subgraphs [13,14]. One study [9] proposed a random walk kernel that computes the similarity by considering all the random walks in the given graph pair. Another [11] proposed a subtree kernel with limited height by extending the work from [8]. The proposed kernel is successful for some of the data sets, but it generally suffers from scalability. Scalability is the main concern of several proposed kernels, as they are designed for fast analysis of large graphs [15]. A kernel based on walks on shortest path graphs, which is a transformation of a given graph into another that includes the shortest path information, was proposed in [10]. Shervashidze et al. in [12] proposed a family of Weisfeiller–Lehman (WL) kernels that are members of the propagation kernel family. Propagation kernels compute the similarity by iteratively propagating the structural information through nodes and edges. Recently, the authors of [16] discussed the general definition of propagation kernels and extended them by applying locally sensitive hashing to decrease the computational requirements.

Enumeration of all subgraphs of a given graph is known to be NP-complete [8]. Therefore, the authors of [13] proposed using only frequent subgraphs as descriptors. The authors of [14] also restricted enumeration by comparing maximum common subgraphs only. A subgraph matching kernel that, given a radius, computes the similarity by comparing the neighborhood of each vertex was proposed in [17]. Recently, graph classification over streams was investigated in [18], where the main problem addressed is the difficulty of constructing a single feature space over streams where the graphs should be mapped as the data set expands continuously. Finally, the authors of [19] proposed a kernel based on subgraph matchings for small-sized attributed graphs.

We selected five of the mentioned kernels for comparison to the proposed kernel. Among these, the WL kernel [12] propagates the label information for each node to the others and therefore embeds the substructure information into the labels of the nodes. Iteratively propagated labels are then relabeled to shorter ones for efficiency, and each graph is then described as a count vector of these labels. The kernel is computed by performing a dot product of these vectors. The neighborhood subgraph pairwise distance kernel [17] decomposes the graph into small subgraphs of increasing radius and compares them to compute the kernel value. It compares the graphs by computing fast graph invariants for subgraphs and then adding up all pairwise subgraph similarity values. The optimal assignment kernel [20] works by finding an optimal assignment (or matching) among the

atoms of a molecular graph. It does this by considering the neighborhood information, certain structural elements (a ring, for instance), and other chemical properties of each atom. The marginalized kernel [9] is based on comparing label sequences of random walks that can be up to infinite length. A method was proposed for computing this by solving a system of linear simultaneous equations. The shortest path kernel [10] defines a similarity among the graphs based on the polynomially computable shortest paths on graphs.

This paper describes a new kernel that is based on a novel descriptor that exploits edge distance-k graphs. This is the first time that edge distance-k graphs are used as descriptors for labeled graphs. The descriptors mentioned in [7] are based on B-matrices and do not consider vertex labels. It is important to consider the labels of nodes, especially for molecular graphs. Therefore, this paper proposes using the edge distance-k graphs as descriptors for the labeled graphs in an easily parallelizable manner. The previously proposed kernels are not directly convertible to a parallel version, and, to the best of our knowledge, parallel versions are not available for most of them. The EDk kernel that we propose is suitable for parallelization, which is also described in this paper.

2.2. Definitions and notation

An undirected graph G consists of a nonempty set of vertices (or nodes), $V(G)$, and a set of edges, $E(G)$. An edge {u,w} is said to connect two vertices uw and is usually abbreviated (or labeled) as uw. If $V(G)$ and/or $E(G)$ are labeled using symbols from a finite alphabet, then G is said to be a labeled graph. If $V(G)$ can be decomposed into two disjoint sets $V_1(G)$ and $V_2(G)$, where $V(G) = V_1(G) \cup V_2(G)$, such that $uw \in E(G)$ implies $u \in V_1(G)$ and $w \in V_2(G)$, then G is called a bipartite graph. A path is a sequence of distinct vertices (except start and end vertices) connected by an edge. The distance $d(uw)$ between nodes u and w on a graph is the length of the shortest path starting from u and ending on w. The maximum distance between nodes is the diameter of the graph, which will be shown as $diam(G)$. Edge distance d_e is defined as the distance between a node and an edge: $d_e(vuw) = (d(vu) + d(vw))/2$. An invariant is a graph property that does not change under different graph representations such as different labelings or drawings. When defining the graph kernel, we will use the $< ., . >$ notation for a dot-product.

For a given graph G, a distance-k graph is a graph with vertex set $V(G)$, where an edge exists between vertices $uv \in V(G)$ if $d(uv) = k$ [21]. Note that, for such graphs, as k grows, the information about G goes from local to global. Therefore, invariants for a set of distance-k graphs can be good candidates for graph descriptors. Based on these, a matrix representation of graphs is proposed called the B-matrix representation, where each element of the matrix includes the following information [22]:

$$B_{k,l} = \text{number of nodes that have } l\text{members in their } k - \text{shells}$$

where a k-shell for a node v is defined as the set of nodes that are of distance k to v.

Recently, in [7], Czech extended distance-k graphs to edge distance-k graphs and proposed the edge B-matrix representation for graphs:

$$B^E_{i,l} = \text{number of nodes that have } l \text{ edges in their } \frac{1}{2}i - \text{edge} - \text{shells}$$

where an i-edge-shell for a node v is the set of edges that are of distance i to v. More detail is given for edge distance-k graphs in the next section.

Kernel methods are a set of algorithms for pattern analysis [4]. The kernel, which is the core of these sets of methods, can roughly be considered as a similarity function among the objects of interest. A kernel usually

computes the similarity by mapping the objects to a higher dimensional Hilbert space. It is said to be valid if it is positive, semidefinite, and symmetric, where validity is a necessary property for the kernel to be safely used in classification. SVMs are one of the most widely used kernel methods among others such as kernel regression and kernel PCA [23].

3. Edge distance kernel

In this section, we first define the new proposed descriptor, and then we give the formal definition of the edge distance kernel. In addition, we analyze the time complexity and prove that the proposed kernel is valid.

3.1. A new graph descriptor based on edge distance-k graphs

An edge distance-k graph ($ED_k(G)$) is a bipartite graph between sets $V(G)$ and $E(G)$. An edge exists between node $v \in V(G)$ and $e \in E(G)$ if $d_e(ve) = k$ (see Figures 1a–1c). An edge in $ED_k(G)$ will be named in the same way as the convention mentioned above. For instance, the edge connecting node 4 and node {3,5} in the ED_1 graph in Figure 1b is edge 435. All connections for a node will be concatenated. Therefore, label 43556 corresponds to the connections of node 4, as 4 is connected to 35 and 56. Each such string will be called the adjacency string of a specific node (4 in this case) for the given $ED_k(G)$.

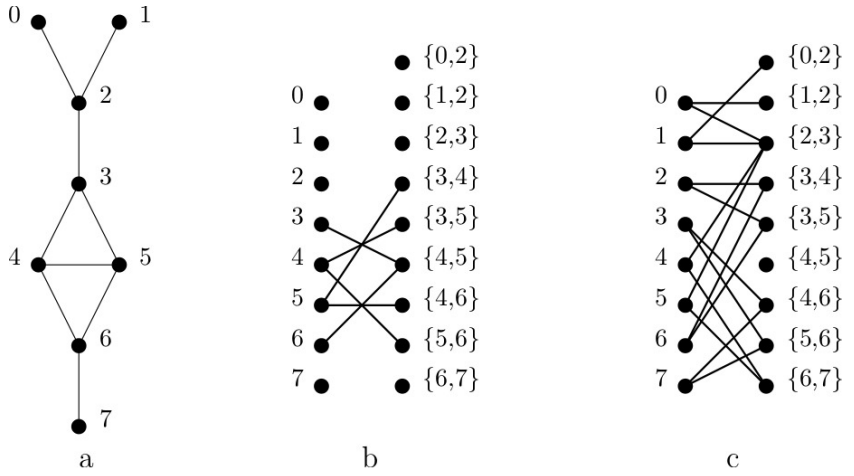


Figure 1. a) Sample graph, b) edge distance-1 graph, c) edge distance-1.5 graph (adapted from [7]).

Among other representations, $ED_k(G)$ can be represented as a list of edges. For example, the list representing the ED_1 graph in Figure 1b is {345, 43556, 53446, 645}, corresponding to node labels concatenated with the lexicographically sorted list of connected edges for each node. As the edges are considered, the nodes with no connections are not included in the list. We will denote this list of edges representing $ED_k(G)$ as $\overline{ED}_k(G)$.

The nodes of the graph in Figure 1 have unique labels. Hence, in this case, $\overline{ED}_k(G)$ is a set. Some graphs can have nodes with identical labels. For such graphs, $\overline{ED}_k(G)$ can include the same elements more than once; thus, it is a multiset for the general case. Therefore, we will depict $\overline{ED}_k(G)$ as $\{s_0 : l_0,\ s_1 : l_1,\ s_2 : l_2,...\}$, where s_i corresponds to the adjacency string of node i and l_i is the number of occurences of s_i in $\overline{ED}_k(G)$.

This representation may include some ambiguities. Given $d_e(vuw) = k$, there is no way to uniquely identify the pairwise distances of the nodes u and w to v. Distance information may provide the ability to distinguish between some cases that are otherwise indistinguishable. For example, when $d_e(vuw) = 1.5$, it is impossible without further information to derive which of the nodes u or w is of distance 1 and which is of distance 2 to v. Therefore, one final extension for $\overline{ED}_k(G)$ is the addition of distance information to each edge label: $s_i = vud(vu)wd(vw)$. Figure 2 depicts the insertion of distance information to adjacency strings of nodes 0, 1, and 3 for $ED_{1.5}(G)$ of the graph in Figure 1.

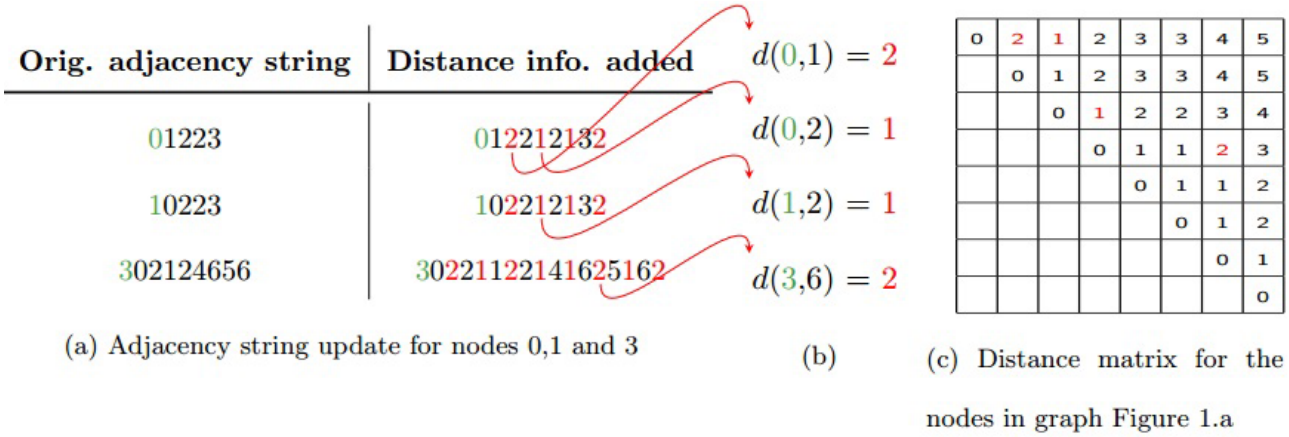


Figure 2. Insertion of distance information to adjacency strings.

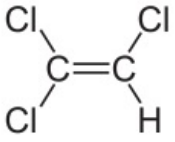
Figure 3 gives an example of deriving ED_k for the graph of a real chemical compound, trichloroethylene. The labels here are not unique; there are 3 chloride and 2 carbon atoms in the compound. Therefore, there are three edges labeled 'ClC1Cl0' in $\overline{ED}_{0.5}(G)$. Note that $ED_k(G)$ for integer k has edges only when G is cyclic; otherwise, it is a graph composed only of nodes. As trichloroethylene has no cycles, $\overline{ED}_k(G) = \{\}$ for $k = \{1, 2\}$. We will utilize this information in computing the kernel. Additionally, note that $\overline{ED}_k(G) = \{\}$ for $k > diam(G)$. Consequently, for any graph, $ED_k(G)$ is computed for $k = \{0.5, 1, 1.5, \ldots, diam(G)\}$ (diameter is 3 in Figure 3).

3.2. Edge distance graph kernel

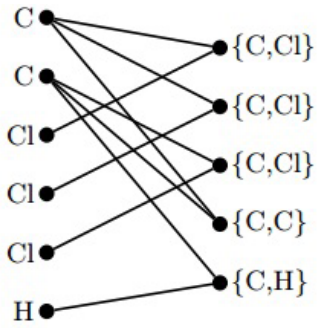
In order to define a kernel based on edge distances, we first define a similarity metric between $\overline{ED}_k(G_1)$ and $\overline{ED}_k(G_2)$. Note that the adjacency strings in $\overline{ED}_k(G)$ can also be seen as a hash table, where adjacency strings are keys and their number of occurrences are values. Let $\omega_k(G)$ be the set of keys of hash table $\overline{ED}_k(G)$ and let $G^k[s]$ denote the value of key s in $\overline{ED}_k(G)$. Then we define the similarity metric $S_k(G_1G_2)$ as follows:

$$S_k(G_1G_2) = T * \delta(\omega_k(G_1), \omega_k(G_2), \emptyset) + \sum_{s \in \omega_k(G_1) \cap \omega_k(G_2)} \delta(G_1^k[s], G_2^k[s]) * |s|,$$

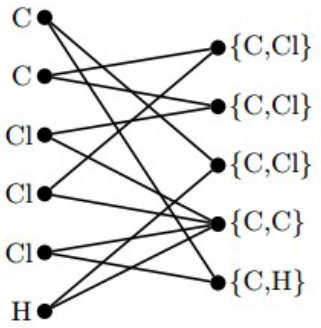
where $\delta(ijk)$ is the ternary Kronecker delta that returns 1 if $i = j = k$ and 0 otherwise, and $|s|$ is the length of s. T is the constant that represents the case where both $\overline{ED}_k(G_1)$ and $\overline{ED}_k(G_2)$ are empty, i.e. if, for a given k, two graphs have empty $\overline{ED}_k(G)$, then their S_k value is T. Based on our experiments, we suggest using $T = 10$, which is the value we used in the experimental results. In summary, $S_k(G_1G_2)$ returns the total



(a) Trichloroethylene



(b) edge distance-0.5 graph



(c) edge distance-1.5 graph

k	Multisets for each ED_k
0.5	{CC0C1C0Cl1C0Cl1 : 1, CC0C1C0Cl1C0H1 : 1, ClC1Cl0 : 3, HC1H0 : 1}
1	{}
1.5	{CC1Cl2C1Cl2 : 1, CC1Cl2C1H2 : 1, ClC1C2C1Cl2 : 2, ClC1C2C1H2 : 1, HC1C2C1Cl2 : 1}
2	{}
2.5	{ClC2Cl3C2Cl3 : 1, ClC2Cl3C2H3 : 2, HC2Cl3C2Cl3 : 1}

(d)

Figure 3. Edge distance-k multisets for the compound trichloroethylene.

number of common elements with an equal number of occurrences in $\overline{ED}_k(G_1)$ and $\overline{ED}_k(G_2)$ weighted by the length of the element.

Definition 1 *The edge distance-k (EDk) graph kernel between graphs* G_1 *and* G_2 *is defined as follows:*

$$EDk(G_1G_2) = \sum_{k}^{D} S_k(G_1G_2),$$

where $D = min(diam(G_1) - 0.5, diam(G_2) - 0.5)$.

The EDk kernel is obviously symmetric, as $S_k(G_1G_2) = S_k(G_2G_1)$. Although it does not include an explicit dot-product operation, we will show that it is actually a dot-product kernel. Let each $s_i \in \overline{ED}_k(G)$ be concatenated with $G^k[s]$ producing a new concatenated adjacency string $s_i\_G^k[s_i]$. Additionally, let $\omega_k(G) = \_k\_$ if $\overline{ED}_k(G) = \{\}$. Without loss of generality, let Σ^* be the combination of the set of all possible concatenated adjacency strings and a set of special strings $\forall k\_k\_\}$ denoting the case of empty $\overline{ED}_k(G)$. Since k is limited and the node labeling alphabet is finite, Σ^* is also finite. A vector of length $|\Sigma^*|$ can then be defined as the descriptor for each graph, where these vectors are the parameters of the dot-product computing EDk kernel. The following theorem gives the definition of these vectors and formally states the validity.

Theorem 1 *The EDk kernel is positive definite.*

Proof The EDk kernel defines a kernel with the corresponding feature map $\phi^{(k)}_{EDk}$,

$$\phi^{(k)}_{EDk}(G) = \forall k \in \{0.5, 1, 1.5, ..., diam(G)\}\{g(s)|s \in \Sigma^*,$$

where

$$g(s) = \begin{cases} \sqrt{|s^{\text{-}}|} * \mathbf{1}_{\omega_k(G)}(s), & \text{if} \overline{ED}_k(G) \neq \{\} \\ \sqrt{T}, & \text{if} \overline{ED}_k(G) = \{\} \end{cases},$$

where $s^{\text{-}}$ is the prefix of s up to the _ character and $\mathbf{1}$ is the indicator function. Then the EDk kernel is defined as the following dot-product:

$$EDk(G_1G_2) =< \phi^{(k)}_{EDk}(G_1), \phi^{(k)}_{EDk}(G_2) > .$$

□

3.3. Time complexity

For the ease of notation, in this section we will denote $|V(G)|$ as n and $|E(G)|$ as m. To compute edge distances, first all shortest paths have to be computed. The single source shortest path problem can be solved in $O(m + nlog(n))$ by using Dijkstra's algorithm. Thus, all shortest paths for a graph can be solved in $O(mn + n^2log(n))$. The second step in computing the kernel is the computation of the edge distances. This step can be performed at $O(mn)$ in the worst case; however, it can be performed efficiently in practice, since only distances up to a maximum of h have to be computed. Third, building edge distance representations from these requires $O(hmn)$ time, as each $ED_k(G)$ can have at most mn edges. Hence, all the descriptors for N graphs can be computed in $O(Nhmn + Nn^2log(n))$ time. Finally, the Gram matrix computation can be performed in $O(N^2)$. Therefore, the complexity of computing the EDk graph kernel for N graphs is $O(Nhmn + Nn^2log(n) + N^2)$. There can be different methods or heuristics to achieve speed-ups in practice. First, there are various successful shortest path approximation algorithms (such as [24]) that bring significant computational gains in exchange for a small error. These can be applied for finding the shortest paths faster. Second, parallel implementations and GPU algorithms are possible to be applied, as the methods here are easily divided into independent pieces of execution. Table 1 gives the time complexities of various kernels compared to EDk.

Table 1. Time complexities of the kernels (h: number of iterations, n: |V(G)|, m: |E(G)|, N: number of graphs).

EDk	**WL**	**NSPD**	**OA**	**MG**	**SP**
O(Nhmn + Nn^{2}log(N) + N^2)	O(Nhm + N^{2}hn)	O(N^{2}n)	O(N^{2}n^3)	O(N^{2}n^6)	O(N^{2}n^4)

3.4. Parallel EDk (EDkP)

The EDk kernel is based on computation of a descriptor for each graph in a given data set. The operations performed to compute this are highly independent, which means that they can be performed in a totally distributed manner. Therefore, we also propose a distributed version of the algorithm that greatly improves the computation time of the kernel.

The distribution of the computation is based on the independence of the computation from the descriptors for each graph. Each $\overline{ED}_k(G)$ can be distributed to a different core or machine on a cluster, and then the results can be collected back to construct the descriptors for the whole data set. The parallelization of the gram matrix computation is straightforward, as it is composed of N^2 independent computations. However, parallelization for descriptor computations cannot be performed for every previously proposed kernel, as they include sequential computations. The simple scheme that we follow for parallel computation is summarized in Figure 4.

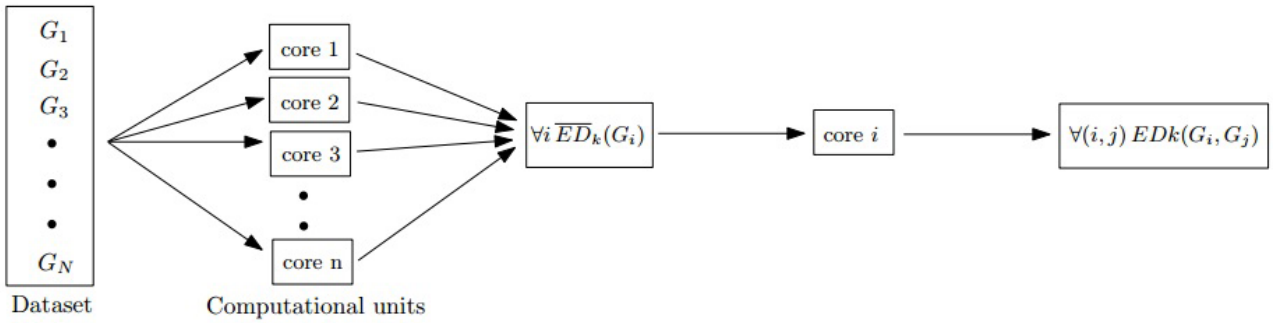


Figure 4. Parallel computation of the **EDk** kernel.

4. Empirical evaluation

4.1. Data sets and benchmarking kernels

We evaluated the proposed kernel on 9 data sets with different characteristics. The descriptions of the data sets are given below, and some of their statistical properties are given in Table 2.

Table 2. Data set statistics. N is the total number of compounds, +% is the percentage of positive samples, N_V is the average number of vertices, and N_E is the average number of edges.

	N	+%	$\mathbf{N_V}$	$\mathbf{N_E}$
FM	349	40.9	25.2	25.6
MM	336	38.3	25.0	25.4
FR	351	34.5	26.0	26.5
MR	344	44.1	25.6	25.9
NCI109	4154	50.0	57.47	59.71
HIV	1503	28.1	59.03	61.55
MUTAG	188	66.5	17.9	19.8

FM, FR, MM, MR: The Predictive Toxicology Challenge (PTC) [25] data sets consist of four data sets describing the toxic effect of a number of compounds on four different animals: female mice (FM), male mice (MM), female rats (FR), and male rats (MR). Each compound is labeled as toxic or nontoxic.

MUTAG: A set of 188 compounds with class labels set according to whether they have mutagenic effects on the bacterium *Salmonella typhimurium* or not [26].

NCI109: NCI109 is a balanced subset of the anticancer screening data set available from PubChem.

HIV: A set of compounds tested against HIV. The compounds are grouped as active and moderately active (http://dtp.nci.nih.gov/docs/aids/aids_data.html).

We compared the performance of the EDk kernel to 5 previously proposed kernels. The WL kernel [12] and the neighborhood subgraph pairwise distance (NSPD) kernel [17] are recent kernels that can be considered as state-of-the-art. There are also other well-known kernels that we used in comparison, including the optimal assignment (OA) kernel [20], the marginalized (MG) kernel [9], and the shortest path (SP) kernel [10].

4.2. Experimental setup

We report the classification performance of the kernels by using scikit-learn [27], the machine learning library for Python. The results were obtained by using 10-fold cross-validation with fixed folds for all algorithms. All experiments were executed on a computer with Intel Core-i5 3.2 GHz CPU and 6 GB of memory running Linux.

We used 10-fold cross-validation to divide the data into training and testing sets. Then the values of the critical parameters of each kernel were chosen by another 10-fold cross-validation on training data only. The model with the selected parameters is then constructed for the training data and applied to the testing data. The reported average accuracies are for testing data. For the WL kernel, h, the level parameter, is chosen from the set $h = \{1, 2, 3, ..., 10\}$. For the NSPD kernel, there are two critical parameters, maximum radius and distance. As in the original NSPD paper [17], we chose the radius (r) from $r = \{0, 1, 2, 3, 4, 5\}$ and distance (d) from $d = \{3, 4, 5, 6, 8, 10, 12, 14, 20\}$ separately for each fold. The OA kernel uses the default parameters from the original paper, as suggested by the authors. For the MG kernel, termination probabilities were optimized among {0.1, 0.3, 0.5, 0.7}. Though k in the EDk kernel can obtain values similar to 0.5, we chose from integer values $k = \{1, 2, 3, .., 10\}$, as the other values do not significantly change the result. The C parameter of the SVM is also optimized among $10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3$.

All the Gram matrices of kernels were normalized (unless they were obtained in $[0, 1]$) by the following standard equation, where κ and κ^N stand for the kernel and normalized kernel, respectively:

$$\kappa^N(G_1 G_2) = \frac{\kappa(G_1, G_2)}{\sqrt{\kappa(G_1, G_1) * \kappa(G_2, G_2)}}.$$

We used the Parallel Python (http://www.parallelpython.com) package, which is an open source Python module, to execute parallel Python code in systems with multiple cores. The 2 cores on the computer on which we ran the experiments were used in parallel execution.

5. Results and discussion

The experimental results are given in Tables 3 and 4. Table 3 reports the mean accuracies of each algorithm for 10-fold cross-validation. The execution times of the kernels are given in Table 4. The EDk kernel performs competitively with other kernels for most of the data sets; for FM and FR, it is the best performing kernel. For MM, it performs worse than WL, NSPD, and MG, but it performs comparably to OA and SP kernels. For MR, although WL is the worst performing kernel, EDk and NSPD reach almost equal accuracies. MG is the best performing kernel for MR. For relatively larger data sets, such as NCI109 and HIV, EDk shows the same performance as the recent kernels WL and NSPD, and it outperforms OA, MG, and SP. OA could not output a result in less than 12 h for NCI109. For MUTAG, EDk is better than WL, NSPD, and MG, where OA performs best this time.

When we look at the execution timetable, we see that EDk is faster than most of the kernels. In addition, note that the results for the EDk and WL kernels are the cumulative durations for computing all 10-Gram matrices corresponding to the different parameter values given in the previous section, not the Gram matrix

Table 3. Comparison in terms of accuracy (±standard error). – : took more than 12 h.

	FM	**FR**	**MM**	**MR**	**NCI109**	**HIV**	**MUTAG**
EDk	64.15 ± 1.8	67.83 ± 1.92	64.88 ± 1.44	61.01 ± 2.25	87.17 ± 0.53	83.57 ± 0.9	87.69 ± 2.16
WL	63.0 ± 2.42	63.21 ± 2.26	69.41 ± 2.3	58.68 ± 1.61	88.01 ± 0.62	83.24 ± 0.91	86.11 ± 2.46
NSPD	63.91 ± 2.05	64.08 ± 1.65	69.35 ± 3.06	61.86 ± 2.8	87.94 ± 0.57	83.57 ± 0.94	84.53 ± 3.37
OA	62.72 ± 1.68	66.65 ± 1.47	64.56 ± 2.48	63.07 ± 2.2	-	82.1 ± 0.71	89.36 ± 1.59
MG	63.601.61	65.21 ± 1.66	67.31 ± 2.04	64.55 ± 2.83	76.82 ± 0.69	77.85 ± 0.65	85.64 ± 2.86
SP	62.45 ± 1.28	65.23 ± 2.02	65.46 ± 1.63	63.37 ± 1.79	79.27 ± 0.33	78.91 ± 0.63	87.78 ± 2.54

Table 4. Running time (in seconds, unless otherwise stated).

	FM	**FR**	**MM**	**MR**	**NCI109**	**HIV**	**MUTAG**
EDk	2.95	3.07	2.73	2.97	282.06	58.10	1.17
EDkP	1.65	1.81	1.59	1.58	142.64	30.11	0.83
WL	8.93	9.12	8.40	8.81	290.77	93.90	3.72
NSPD	9.40	9.47	9.11	9.56	1175.25	185.44	2.83
OA	82.88	100.77	76.06	80.51	$>$ 12 h	648.08 min	28.50
MG	40.50	44.33	37.60	41.51	534.08 min	75.91 min	7.83
SP	2.50	2.40	1.75	1.86	86.78	37.87	0.54

for a single parameter. However, for NSPD, the result is given for only one parameter setting (r = 4, d = 5). The OA kernel's results are according to the default parameter values. It could not compute the Gram matrix for the NCI109 data set in less than 12 h, and generally it is the slowest kernel. The MG kernel is the second slowest kernel, and it also took a long time to compute the Gram matrix for NCI109. The SP kernel is fast, but its performance is not very reliable, as it performs well for small-sized data sets, but for NCI109 and HIV, SP performs 9% and 5% worse than the EDk kernel. From these results, it seems that when the data set and its graphs become larger, the performance of SP drops. EDkP (the parallel version of EDk) computes the kernel faster than EDk. Note that these results are achieved with an ordinary multicore desktop computer.

In summation, the EDk kernel is competitive with the state-of-the-art kernels in terms of performance. It is significantly faster than WL, NSPD, OA, and MG. Only SP is faster, but its performance is very poor for some data sets. Therefore, EDk is either faster or performs better than the kernels in our comparison set. Its parallel version is easy to implement and has the potential to be executed on larger scale distributed architectures, such as a cloud, using the MapReduce framework.

6. Conclusion

In this paper, we presented a new graph kernel, the edge distance-k graph kernel. The kernel exploits the fact that the edge distance-k graphs actually include valuable topological information for graphs to be used as a graph descriptor. By producing several edge distance-k graphs up to a given k, the kernel computes the similarity among edge distance-k graphs, based on a similarity metric. Both this similarity metric and the kernel itself are novel, built upon the previously proposed distance-k and edge distance-k graphs. This is the first time that edge-distance-k graphs are used as graph descriptors for labeled graphs, and a parallel version of the kernel is also proposed and implemented. The kernel is promising in the sense that it is fast and accurate. The parallelization of the kernel is straightforward and its use can be extended to a cluster or a cloud as well as a MapReduce framework.

Notwithstanding its promising results, several of the kernel's aspects can be improved. Feature selection can be applied to the kernel, as feature selection has been previously shown to increase performance for graph classification for different kernels [28,29]. Moreover, it would be interesting to investigate the performance of the kernel in domains requiring pairwise kernels such as protein–ligand or protein–protein interaction predictions.

References

[1] Swamidass SJ, Chen JH, Bruand J, Phung P, Ralaivola L, Baldi P. Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. Bioinformatics 2005; 21: 359-368.

[2] Vujošević-Janičić M, Nikolić M, Tošić D, Kuncak V. Software verification and graph similarity for automated evaluation of students' assignments. Inform Software Tech 2013; 55: 1004-1016.

[3] Henderson K, Gallagher B, Li L, Akoglu L, Eliassi-Rad T, Tong H, Faloutsos C. It's who you know: graph mining using recursive structural features. In: SIGKDD 2011 International Conference on Knowledge Discovery and Data Mining; 21–24 August 2011; San Diego, CA, USA. New York, NY, USA: ACM. pp. 663-671.

[4] Schölkopf B, Smola AJ. Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. Cambridge, MA, USA: MIT Press, 2002.

[5] Vishwanatan SVN, Schraudolph NN, Kondor R, Borgwardt KM. Graph kernels. J Mach Learn Res 2010; 11: 1201-1242.

[6] Yang X, Tschaplinski TJ, Hurst GB, Jawdy S, Abraham PE, Lankford PK, Adams RM, Shah MB, Hettich RL, Lindquist E et al. Discovery and annotation of small proteins using genomics, proteomics, and computational approaches. Genome Res 2011; 21: 634-641.

[7] Czech W. Invariants of distance k-graphs for graph embedding. Pattern Recogn Lett 2012; 33: 1968-1979.

[8] Gärtner T, Flach P, Wrobel S. On graph kernels: hardness results and efficient alternatives. In: Schölkopf B, Warmuth MK, editors. Learning Theory and Kernel Machines. Berlin, Germany: Springer-Verlag, 2003. pp. 129-143.

[9] Kashima H, Tsuda K, Inokuchi A. Marginalized kernels between labeled graphs. In: AAAI 2003 International Conference on Machine Learning; 21–24 August 2003; Washington, DC, USA. Palo Alto, CA, USA: AAAI. pp. 321-328.

[10] Borgwardt KM, Kriegel HP. Shortest-path kernels on graphs. In: IEEE 2005 International Conference on Data Mining; 27–30 November 2005; Washington, DC, USA. New York, NY, USA: IEEE. pp. 74-81.

[11] Mahe P, Vert JP. Graph kernels based on tree patterns for molecules. Mach Learn 2009; 75: 3-35.

[12] Shervashidze N, Schweitzer P, van Leeuwen EJ, Mehlhorn K, Borgwardt KM. Weisfeiler-Lehman graph kernels. J Mach Learn Res 2011; 12: 2539-2561.

[13] Deshpande M, Kuramochi M, Wale N, Karypis G. Frequent substructure-based approaches for classifying chemical compounds. IEEE T Knowl Data En 2005; 17: 1036-1050.

[14] Schietgat L, Costa F, Ramon J, Raedt LD. Effective feature construction by maximum common subgraph sampling. Mach Learn 2011; 83: 137-161.

[15] Shervashidze N, Vishwanathan SVN, Petri T, Mehlhorn K, Borgwardt KM. Efficient graphlet kernels for large graph comparison. In: International Conference on Artificial Intelligence and Statistics; 16–18 April 2009; Clearwater Beach, FL, USA. Harrisburg, PA, USA: ASAIP. pp. 488-495.

[16] Neumann M, Patricia N, Garnett R, Kersting K. Efficient graph kernels by randomization. In: ACM 2012 European Conference on Machine Learning and Knowledge Discovery in Databases; 24–28 September 2012; Bristol, UK. New York, NY, USA: ACM. pp. 378-393.

[17] Costa F, Grave KD. Fast neighborhood subgraph pairwise distance kernel. In: ACM 2010 International Conference on Machine Learning; 21–24 June 2010; Haifa, Israel. New York, NY, USA: ACM. pp. 255-262.

[18] Li B, Zhu X, Chi L, Zhang C. Nested subtree hash kernels for large-scale graph classification over streams. In: IEEE 2012 International Conference on Data Mining; 10–13 December 2012; Brussels, Belgium. New York, NY, USA: IEEE. pp. 399-408.

[19] Kriege N, Mutzel P. Subgraph matching kernels for attributed graphs. In: IEEE 2012 International Conference on Machine Learning; 15–17 July 2012; Xian, China. New York, NY, USA: IEEE. pp. 1-8.

[20] Fröhlich H, Wegner JK, Sieker F, Zell A. Optimal assignment kernels for attributed molecular graphs. In: ACM 2005 International Conference on Machine Learning; 7–11 August 2005; Bonn, Germany. New York, NY, USA: ACM. pp. 225-232.

[21] Brouwer A, Cohen A, Neumaier A. Distance Regular Graphs. Berlin, Germany: Springer-Verlag, 1989.

[22] Bagrow JP, Bollt EM, Skufca JD, Ben-Avraham D. Portraits of complex networks. Europhys Lett 2008; 81: 68004.

[23] Hoffman T, Schölkopf B, Smola AJ. Kernel methods in machine learning. Ann Stat 2008; 36: 1171-1220.

[24] Gubichev A, Bedathur S, Seufert S, Weikum G. Fast and accurate estimation of shortest paths in large graphs. In: ACM 2010 International Conference on Information and Knowledge Management; 26–30 October 2010; Toronto, ON, Canada. New York, NY, USA: ACM. pp. 499-508.

[25] Helma C, King R, Kramer R, Srinivasan A. A survey of the predictive toxicology challenge 2000–2001. Bioinformatics 2001; 19: 107-108.

[26] Debnath AK, de Compadre RLL, Debnath G, Shusterman AJ, Hansch C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compouns correlation with molecular orbital energies and hydrophobicity. J Med Chem 1991; 34: 786-797.

[27] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al. Scikit-learn: machine learning in Python. J Mach Learn Res 2011; 12: 2825-2830.

[28] Tan M. Information theoretic feature selection for Weisfeiler-Lehman graph kernels. In: ACM 2013 International Conference on Bioinformatics and Computational Biology; 22–25 September 2013; Washington, DC, USA. New York, NY, USA: ACM.

[29] Fei H, Huan J. Structure feature selection for graph classification. In: ACM 2008 International Conference on Information and Knowledge Management; 26–30 October 2008; Napa Valley, CA, USA. New York, NY, USA: ACM. pp. 991-1000.