A Degeneracy Framework for Graph Similarity

Giannis Nikolentzos[1], **Polykarpos Meladianos**[2], **Stratis Limnios**[1] and **Michalis Vazirgiannis**[1]
[1] École Polytechnique, France
[2] Athens University of Economics and Business, Greece
{nikolentzos, mvazirg}@lix.polytechnique.fr, pmeladianos@aueb.gr, stratis.limnios@inria.fr

Abstract

The problem of accurately measuring the similarity between graphs is at the core of many applications in a variety of disciplines. Most existing methods for graph similarity focus either on local or on global properties of graphs. However, even if graphs seem very similar from a local or a global perspective, they may exhibit different structure at different scales. In this paper, we present a general framework for graph similarity which takes into account structure at multiple different scales. The proposed framework capitalizes on the well-known k-core decomposition of graphs in order to build a hierarchy of nested subgraphs. We apply the framework to derive variants of four graph kernels, namely graphlet kernel, shortest-path kernel, Weisfeiler-Lehman subtree kernel, and pyramid match graph kernel. The framework is not limited to graph kernels, but can be applied to any graph comparison algorithm. The proposed framework is evaluated on several benchmark datasets for graph classification. In most cases, the core-based kernels achieve significant improvements in terms of classification accuracy over the base kernels, while their time complexity remains very attractive.

1 Introduction

Graphs are well-studied structures which are utilized to model entities and their relationships. In recent years, graph-based representations have become ubiquitous in many application domains. For instance, social networks, protein and gene regulatory networks, and textual documents are commonly represented as graphs. Furthermore, in the past years, graph classification has arisen as an important topic in many domains such as in Computational Biology [Schölkopf *et al.*, 2004], in Chemistry [Mahé and Vert, 2009] and in Natural Language Processing [Nikolentzos *et al.*, 2017a]. For example, in Chemistry, we are often interested in predicting the mutagenicity of a chemical compound by comparing its graph representation with other compounds of known functionality.

So far, kernel methods have emerged as one of the most effective tools for graph classification, and have achieved state-of-the-art results on many graph datasets [Shervashidze *et al.*, 2011]. Once we define a positive semidefinite kernel function for the input data, a large family of learning algorithms called kernel methods [Smola and Schölkopf, 1998] become available. In more details, kernels are functions that correspond to a dot product in a reproducing kernel Hilbert space, and which measure the similarity between two objects. Kernel functions do not require their inputs to be represented as fixed-length feature vectors, and they can also be defined on structured data such as graphs, trees and strings. Hence, kernel methods provide a flexible framework for performing graph classification.

Most graph kernels in the literature are instances of the R-convolution framework [Haussler, 1999]. These kernels decompose graphs into their substructures and add up the pairwise similarities between these substructures. Specifically, there are kernels that compare graphs based on random walks [Gärtner *et al.*, 2003; Vishwanathan *et al.*, 2010; Sugiyama and Borgwardt, 2015], subtrees [Gärtner *et al.*, 2003; Mahé and Vert, 2009], cycles [Horváth *et al.*, 2004], shortest paths [Borgwardt and Kriegel, 2005], and small subgraphs [Shervashidze *et al.*, 2009; Kriege and Mutzel, 2012]. Recently, there was a surge of interest in kernels that are built upon global properties of graphs [Johansson *et al.*, 2014; Johansson and Dubhashi, 2015; Nikolentzos *et al.*, 2017b]. In general, these approaches embed the vertices of each graph in a vector space, and then compare graphs based on these embeddings.

Most existing graph kernels can thus be divided into two classes. The first class consists of kernels that compare local substructures of graphs (i. e. trees, cycles, graphlets), while the second class includes kernels that capture global properties of graphs and are sensitive to the large scale structure of graphs. Some examples of the second class are the random walk based kernels, and the kernels that compare graphs based on the embeddings of their vertices. Therefore, existing graph kernels focus mainly on either local or global properties of graphs. In practice, it would be desirable to have a kernel that can take structure into account at multiple different scales [Kondor and Pan, 2016]. Two well-known kernels that account for that are the Weisfeiler–Lehman subtree kernel [Shervashidze *et al.*, 2011] and the propagation kernel [Neumann *et al.*, 2016]. However, both approaches assume node-labeled graphs. Recently, the multiscale Laplacian kernel was introduced to effectively compare structure at different scales [Kondor and Pan, 2016], while some neural network architectures were also designed to address the same problem [Dai *et al.*, 2016].

In this paper, we propose a framework for comparing structure in graphs at a range of different scales. Our framework is based on the k-core decomposition which is capable of uncovering topological and hierarchical properties of graphs. Specifically, the k-core decomposition builds a hierarchy of nested subgraphs, each having stronger connectedness properties compared to the previous. By measuring the similarity between the corresponding according to the hierarchy subgraphs and combining the results, we can build more accurate measures of graph similarity. More specifically, the contributions of this paper are threefold:

- We propose a general framework that allows existing graph similarity algorithms to compare structure in graphs at multiple different scales. The framework is based on the k-core decomposition of graphs and is applicable to any graph comparison algorithm.
- We demonstrate our framework on four graph kernels, namely the graphlet kernel, the shortest path kernel, the Weisfeiler-Lehman subtree kernel, and the pyramid match kernel.
- We evaluate the proposed framework on several benchmark datasets from bioinformatics, chemoinformatics and social networks. In most cases, the variants obtained from our framework achieve significant improvements over the base kernels.

The rest of this paper is organized as follows. Section 2 introduces some preliminary concepts and gives details about graph degeneracy and the k-core decomposition. Section 3 provides a detailed description of our proposed framework for graph similarity. Section 4 evaluates the proposed framework on several standard datasets. Finally, Section 5 concludes.

2 Preliminaries

In this section, we first define our notation, and we then introduce the concepts of k-core and degeneracy. We also give details about the algorithm that extracts the k-cores of a graph.

2.1 Definitions and Notations

Let $G = (V, E)$ be an undirected and unweighted graph consisting of a set V of vertices and a set E of edges between them. We will denote by n the number of vertices and by m the number of edges. The neighbourhood $\mathcal{N}(v)$ of vertex v is the set of all vertices adjacent to v. Hence, $\mathcal{N}(v) = \{u : (v, u) \in E\}$ where (v, u) is an edge between vertices v and u of V. We denote the degree of vertex v by $d(v) = |\mathcal{N}(v)|$. Given a subset of vertices $S \subseteq V$, let $E(S)$ be the set of edges that have both end-points in S. Then, $G' = (S, E(S))$ is the subgraph induced by S. We use $G' \subseteq G$ to denote that G' is a subgraph of G. The degree of a vertex $v \in S$, $d_{G'}(v)$, is equal to the number of vertices that are adjacent to v in G'. A labeled graph is a graph with labels on vertices and/or edges. In this paper, we will consider two types of graphs: (1) unlabeled graphs and (2) graphs with labeled vertices. For the second type of graphs, given a set of labels $\mathcal{L}$, $\ell : V \rightarrow \mathcal{L}$ is a function that assigns labels to the vertices of the graph.

2.2 Degeneracy and k-core Decomposition

The k-core decomposition of graphs is a powerful tool for network analysis and it is commonly used as a measure of importance and well connectedness for vertices in a broad spectrum of applications. The study of k-core decomposition and degeneracy goes back to the 60s. More specifically, the first definition of a concept related to k-core (coloring number) was given by Erdős and Hajnal [1966]. The degeneracy of a graph was later defined by Lick and White [1970]. The notion of k-core was first introduced by Seidman [1983] to study the cohesion of social networks. In recent years, the k-core decomposition has been established as a standard tool in many application domains such as in network visualization [Alvarez-Hamelin *et al.*, 2006], in protein function prediction [Wuchty and Almaas, 2005] and in graph clustering [Giatsidis *et al.*, 2014].

More formally, let G be a graph and G' a subgraph of G induced by a set of vertices S. Then, G' is defined to be a k-core of G, denoted by C_k, if it is a maximal subgraph of G in which all vertices have degree at least k. Hence, if G' is a k-core of G, then $\forall v \in S$, $d_{G'}(v) \geq k$. Each k-core is a unique subgraph of G, and it is not necessarily connected. The core number $c(v)$ of a vertex v is equal to the highest-order core that v belongs to. In other words, v has core number $c(v) = k$, if it belongs to a k-core but not to any $(k+1)$-core. The degeneracy $\delta^*(G)$ of a graph G is defined as the maximum k for which graph G contains a non-empty k-core subgraph, $\delta^*(G) = \max_{v \in V} c(v)$. Furthermore, assuming that $\mathcal{C} = \{C_0, C_1, \ldots, C_{\delta^*(G)}\}$ is the set of all k-cores, then $\mathcal{C}$ forms a nested chain:

$$C_{\delta^*(G)} \subseteq \ldots \subseteq C_1 \subseteq C_0 = G$$

Since the k-cores of a graph form a nested chain of subgraphs, the k-core decomposition is a very useful tool for discovering the hierarchical structure of graphs. Figure 1 depicts an example of a graph and its corresponding k-core decomposition. As we observe, the degeneracy of this graph is $\delta^*(G) = 3$; thus, the decomposition creates four nested k-core subgraphs, with the 3-core being the maximal one. The nested structure of the k-core subgraphs is indicated by the dashed lines. Furthermore, the color on the nodes indicates the core number c of each vertex.

The popularity of the k-core decomposition stems mainly from the fact that it can be computed in linear time [Matula and Beck, 1983; Batagelj and Zaveršnik, 2011]. The algorithm for performing the k-core decomposition of a graph is illustrated in Algorithm 1. The algorithm runs in $\mathcal{O}(n + m)$ time. The underlying idea is that we can obtain the i-core of a graph if we recursively remove all vertices with degree less than i and their incident edges from the graph until no other vertex can be removed. Since higher-order cores are nested within lower-order cores, we compute k-cores sequentially from $k = 0$ to $k = \delta^*(G)$. Therefore, at each iteration, the algorithm removes the lowest degree vertex and sets its core number accordingly.

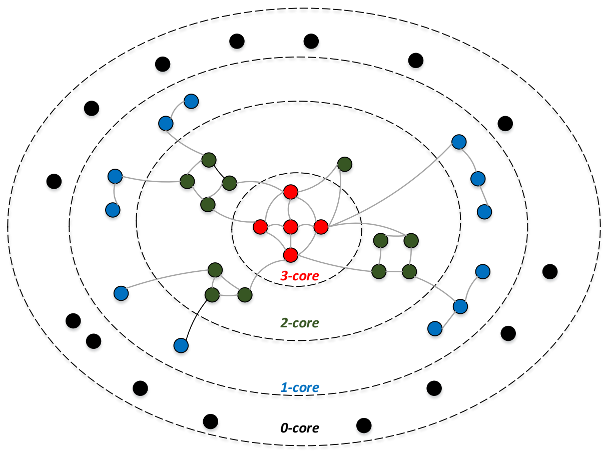


Figure 1: Example of core decomposition of graph.

Algorithm 1 k-core Decomposition

```
Input: A graph $G = (V, E)$
Output: A set of $k$-cores $\mathcal{C}$
$\mathcal{C} = \{V\}$
$k = \min_{v \in V} d(v)$
for $i = 1$ to $n$ do
  Let $v$ be the vertex with the smallest degree in $G$
  if $d(v) > k$ then
    add $V$ to $\mathcal{C}$
    $k = d(v)$
  end if
  $V = V \setminus \{v\}$
end for
```

3 Degeneracy Framework

In this Section, we propose a new framework for graph similarity that is based on the concept of k-core, and we show how existing graph kernels can be plugged into the framework to produce more powerful kernels.

3.1 Core-based Graph Kernels

We next propose a framework for obtaining variants of existing graph kernels. Since the framework utilizes the k-core decomposition, we call the emerging kernels *core variants* of the base kernels. The proposed framework allows the comparison of the structure of graphs at multiple different scales as these are expressed by the graphs' k-cores.

The intuition of using the k-core algorithm to decompose a graph is that internal cores are more important compared to external cores. Hence, they are more likely to reveal information about the class label of the graph compared to external cores. This is by no means implausible since internal cores correspond to subgraphs of high density. As mentioned above, the k-core decomposition is typically used to identify areas of increasing connectedness inside the graph. In almost all graphs, density is an indication of importance [Lee *et al.*, 2010]. For example, in protein-protein interaction networks, dense subgraphs may correspond to protein complexes. Hence, we expect that by decomposing graphs into subgraphs of increasing importance, we will be able to capture their underlying structure, and compare them effectively.

Algorithm 2 Core-based Kernel

```
Input: A pair of graphs $G$ and $G'$
Output: Result of the kernel function $val$
$val = 0$
$\delta^*_{min} = \min\big(\delta^*(G), \delta^*(G')\big)$
Let $C_i, C'_i$ be the $i$-cores of $G, G'$, for $i = 0, \ldots, \delta^*_{min}$
for $i = \delta^*_{min}$ to $0$ do
  $val = val + kernel(C_i, C'_i)$
end for
```

We next introduce the degeneracy framework for deriving core variants of existing kernels.

Definition 1. *Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. Let also k be any kernel for graphs. Then, the core variant of the base kernel k is defined as*

$$k_c(G, G') = k(C_0, C'_0) + k(C_1, C'_1) + \ldots + k(C_{\delta^*_{min}}, C'_{\delta^*_{min}}) \tag{1}$$

where δ^*_{min} is the minimum of the degeneracies of the two graphs, and $C_0, C_1, \ldots, C_{\delta^*_{min}}$ and $C'_0, C'_1, \ldots, C'_{\delta^*_{min}}$ are the 0-core, 1-core,…, δ^*_{min}-core subgraphs of G and G' respectively.

In the following, we will prove the validity of the core variants produced by our framework.

Theorem 1. *Let the base kernel k be any positive semidefinite kernel on graphs. Then, the corresponding core variant k_c of the base kernel k is positive semidefinite.*

Proof. Let ϕ be the feature mapping corresponding to the base kernel k

$$k(G, G') = \langle \phi(G), \phi(G') \rangle$$

Let $g_i(\cdot)$ be a function that removes from the input graph all vertices with core number less than i and their incident edges. Then, we have

$$k(C_i, C'_i) = \big\langle \phi(g_i(G)), \phi(g_i(G')) \big\rangle$$

Let us define the feature mapping $\psi(\cdot)$ as $\phi(g_i(\cdot))$. Then we have

$$k(C_i, C'_i) = \langle \psi(G), \psi(G') \rangle$$

hence k is a kernel on G and G' and k_c is positive semidefinite as a sum of positive semidefinite kernels. □

Given two graphs G, G' and a base kernel k, the steps of computing the core variant of k are given in Algorithm 2.

The above definition provides a framework for increasing the expressive power of existing graph kernels. In contrast to other existing frameworks, the proposed framework is not limited to R-convolution kernels [Yanardag and Vishwanathan, 2015] or to node-labeled graphs [Shervashidze *et al.*, 2011]. Furthermore, it should be mentioned that the proposed framework is not even restricted to graph kernels, but can be applied to any algorithm that compares graphs. Hence, it can serve as a generic tool applicable to the vast literature of graph matching algorithms [Conte *et al.*, 2004].

3.2 Computational Complexity

The proposed framework takes into account structure at different scales, yet it remains an interesting question how it compares to base kernels in terms of runtime complexity. Its computational complexity depends on the complexity of the base kernel and the degeneracy of the graphs under comparison. More specifically, given a pair of graphs G, G' and an algorithm A for comparing the two graphs, let $\mathcal{O}_A$ be the time complexity of algorithm A. Let also $\delta^*_{min} = \min\big(\delta^*(G), \delta^*(G')\big)$ be the minimum of the degeneracies of the two graphs. Then, the complexity of computing the core variant of algorithm A is $\mathcal{O}_c = \delta^*_{min}\mathcal{O}_A$. It is well-known that the degeneracy of a graph is upper bounded by the maximum of the degrees of its vertices and by the largest eigenvalue of its adjacency matrix λ_1. Since in most real-world graphs it holds that $\lambda_1 \ll n$, it also holds that $\delta^*_{max} \ll n$, and hence, the time complexity added by the proposed framework is relatively low.

3.3 Dimensionality Reduction Perspective

The k-core decomposition can also be seen as a method for performing dimensionality reduction on graphs. Given the i-cores C_i, $i = 1, \ldots, \delta^*(G)$ of a graph G, each core C_i can be considered as an approximation of the graph where features of low importance (i. e. vertices belonging to low-order cores and their incident edges) have been removed from the graph. The approximation error can be computed by the Frobenius norm of the difference of the adjacency matrices of the two graphs $er = ||A - A_i||_F$ where A, A_i are the adjacency matrices of graph G and its i-core respectively.

In cases where the input graphs are very large, the running time of high-complexity algorithms is prohibitive. For example, computing the shortest path kernel on the D&D dataset takes almost 1 hour. In such cases, we can take advantage of the k-core decomposition to effectively prune a large number of vertices from the input graphs by retaining only their high-order cores. Then, it may be possible to employ a high-complexity algorithm. For example, by replacing the graphs contained in the D&D dataset with their 3-cores, we managed to compute the core variant of the shortest path kernel in less than 5 minutes and to achieve accuracy comparable to the best performing algorithms ($avg.\ acc = 77.92$).

3.4 Base Kernels

We apply the proposed framework to the following four graph kernels:

(1) **graphlet kernel** (GR) [Shervashidze *et al.*, 2009]: The graphlet kernel counts identical pairs of graphlets (i. e. subgraphs with k nodes where $k \in 3, 4, 5$) in two graphs.

(2) **shortest path kernel** (SP) [Borgwardt and Kriegel, 2005]: The shortest path kernel counts pairs of shortest paths in two graphs having the same source and sink labels and identical length.

(3) **Weisfeiler-Lehman subtree kernel** (WL) [Shervashidze *et al.*, 2011]: The Weisfeiler-Lehman subtree kernel for a number of iterations counts pairs of matching subtree patterns in two graphs, while at each iteration updates the labels of the vertices of the two graphs.

(4) **pyramid match graph kernel** (PM) [Nikolentzos *et al.*, 2017b]: The pyramid match graph kernel first embeds the vertices of the input graphs in a vector space. It then partitions the feature space into regions of increasingly larger size and takes a weighted sum of the matches that occur at each level.

For some base kernels, one might be able to exploit the fact that high-order cores are contained into lower-order cores in order to perform some computations only once instead of repeating them for all cores. One example of such a base kernel is the graphlet kernel. Given two cores of a graph C_i and C_j with $i < j$, all the graphlets found in C_j will also be present in C_i.

4 Experiments and Evaluation

In this section, we first describe the datasets that we used for our experiments. We next give details about the experimental settings. We last report on the performance of the base kernels and the core variants.

4.1 Datasets

We evaluated the proposed framework on standard graph classification datasets derived from bioinformatics and chemoinformatics (MUTAG, ENZYMES, NCI1, PTC-MR, D&D), and from social networks (IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, REDDIT-MULTI-5K, REDDIT-MULTI-12K)[1]. Note that the social network graphs are unlabeled, while all other graph datasets come with vertex labels.

4.2 Experimental Setup

To perform graph classification, we employed a C-Support Vector Machine (SVM) classifier and performed 10-fold cross-validation. The whole process was repeated 10 times for each dataset and each method. The parameter C of the SVM was optimized on the training set only.

All kernels were written in Python[2]. The parameters of the base kernels and their corresponding core variants were selected using cross-validation on the training dataset. We chose parameters for the graph kernels as follows. For the graphlet kernel, on labeled graphs, we count all connected graphlets of size 3 taking labels into account, while on unlabeled graphs, we sample 500 graphlets of size up to 6. For the Weisfeiler-Lehman subtree kernel, we chose the number of iterations h from $\{4, 5, 6, 7\}$. For the pyramid match kernel, the dimensionality of the embeddings d was chosen from $\{4, 6, 8, 10\}$, while the number of levels L was chosen from $\{2, 4, 6\}$.

We report in Table 1 average prediction accuracies and standard deviations. Core variants with statistically significant improvements over the base kernels are shown in bold as measured by a t-test with a p value of ≤ 0.05. We also report in Table 2 the time required for computing the kernel matrix of each core variant relative to the time required for

[1]The datasets and statistics are available at https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets

[2]Code available at https://www.lix.polytechnique.fr/~nikolentzos/code/core_framework.zip

Method \ Dataset	MUTAG	ENZYMES	NCI1	PTC-MR	D&D
GR	69.97 (± 2.22)	33.08 (± 0.93)	65.47 (± 0.14)	56.63 (± 1.61)	77.77 (± 0.47)
Core GR	**82.34** (± 1.29)	33.66 (± 0.65)	**66.85** (± 0.20)	57.68 (± 1.26)	78.05 (± 0.56)
SP	84.03 (± 1.49)	40.75 (± 0.81)	72.85 (± 0.24)	60.14 (± 1.80)	77.14 (± 0.77)
Core SP	**88.29** (± 1.55)	41.20 (± 1.21)	**73.46** (± 0.32)	59.06 (± 0.93)	77.30 (± 0.80)
WL	83.63 (± 1.57)	51.56 (± 2.75)	84.42 (± 0.25)	61.93 (± 2.35)	79.19 (± 0.39)
Core WL	**87.47** (± 1.08)	47.82 (± 4.62)	**85.01** (± 0.19)	59.43 (± 1.20)	79.24 (± 0.34)
PM	80.66 (± 0.90)	42.17 (± 2.02)	72.27 (± 0.59)	56.41 (± 1.45)	77.34 (± 0.97)
Core PM	**87.19** (± 1.47)	42.42 (± 1.06)	**74.90** (± 0.45)	**61.13** (± 1.44)	77.72 (± 0.71)

Method \ Dataset	IMDB BINARY	IMDB MULTI	REDDIT BINARY	REDDIT MULTI-5K	REDDIT MULTI-12K
GR	59.85 (± 0.41)	35.28 (± 0.14)	76.82 (± 0.15)	35.32 (± 0.09)	22.68 (± 0.18)
Core GR	**69.91** (± 0.19)	**47.34** (± 0.84)	**80.67** (± 0.16)	**46.77** (± 0.09)	**32.41** (± 0.08)
SP	60.65 (± 0.34)	40.10 (± 0.71)	83.10 (± 0.22)	49.48 (± 0.14)	35.79 (± 0.09)
Core SP	**72.62** (± 0.59)	**49.43** (± 0.42)	**90.84** (± 0.14)	**54.35** (± 0.11)	**43.30** (± 0.04)
WL	72.44 (± 0.77)	51.19 (± 0.43)	74.99 (± 0.57)	49.69 (± 0.27)	33.44 (± 0.08)
Core WL	**74.02** (± 0.42)	51.35 (± 0.48)	**78.02** (± 0.23)	50.14 (± 0.21)	**35.23** (± 0.17)
PM	68.53 (± 0.61)	45.75 (± 0.66)	82.70 (± 0.68)	42.91 (± 0.42)	38.16 (± 0.19)
Core PM	**71.04** (± 0.64)	**48.30** (± 1.01)	**87.39** (± 0.55)	**50.63** (± 0.50)	**42.89** (± 0.14)

Table 1: Classification accuracy (± standard deviation) of the graphlet kernel (GR), shortest path kernel (SP), Weisfeiler-Lehman subtree kernel (WL), pyramid match kernel (PM) and their core variants on the 10 graph classification datasets. Core variants with statistically significant improvements over the base kernels are shown in bold as measured by a t-test with a p value of ≤ 0.05.

computing the kernel matrix of its base kernel as measured on a 3.4GHz Intel Core i7 with 16Gb of RAM.

4.3 Results

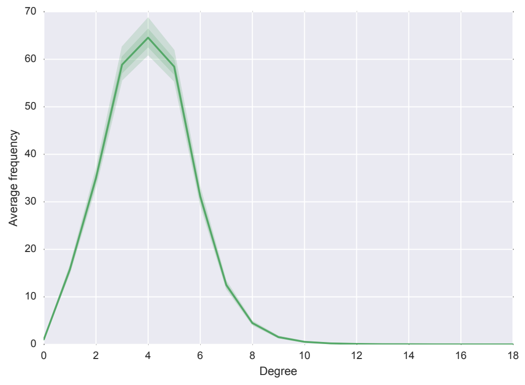
We begin our experiments by comparing the base kernels with their core variants. Table 1 demonstrates that the proposed framework improves the classification accuracy of every base kernel on almost all datasets. More specifically, the core variants outperformed their base kernels on 37 out of the 40 experiments. It should be mentioned that the difference in performance between the core variants and their base kernels was larger on the social interaction datasets compared to the bioinformatics and chemoinformatics datasets. The obtained results confirm our intuition that the densest areas of graphs are the most important. Furthermore, the results show that the hierarchy of nested subgraphs generated by the k-core decomposition allows existing algorithms to compare structure in graphs at multiple different scales. On most datasets, the increase in performance of the GR, SP and PM kernels due to the use of the proposed framework is very large. Specifically, core GR improved by more than 10% the accuracy attained by the GR kernel on 4 datasets. Conversely, core WL yielded in general only slightly better accuracies compared to its base kernel. The WL kernel builds a summary of the neighborhood of each vertex. Our intuition is that the local neighborhood of a vertex in a k-core is not dramatically different from its neighbourhood in the graph. Hence, for small values of the parameter h of WL, the summaries that are generated in a k-core are very similar to those generated in the whole graph and do not thus provide much additional information.

In terms of runtime, we can observe that in most cases, the extra computational cost required to compute the core variant of a kernel is negligible. We computed the average degeneracy δ^*_{ave} of the graphs contained in each dataset (shown in Table 2), and we observed that the running time is very related to its value. On the IMDB-BINARY and IMDB-MULTI datasets, computing the core variant requires more than 6 times the time of computing the base kernels. However, even that increase in running time is by no means prohibitive. It is also interesting to note that the extra computational cost comes with a significant improvement in accuracy.

We next investigate why the core variants lead to greater improvements on the social interaction datasets compared to the bioinformatics and chemoinformatics datasets. We attribute this difference in the behavior of the core variants to the underlying structure of the two types of graphs. Figure 2 illustrates the degree distribution of the D&D and REDDIT-BINARY datasets. We observe that the latter follows the well-known power-law distribution while the former does not. We should mention that we have observed almost identical behavior on the other bioinformatics/chemoinformatics and social interaction datasets, and the plots were omitted for illustration purposes. We can safely assume that the higher-order cores of the graphs of the REDDIT-BINARY dataset capture the most informative areas of the graph. Conversely, in graphs with structure similar to that of the graphs of the bioinformatics datasets, many nodes may end up sharing the exact same core number due to the coarse granularity of the k-core decompo-

	MUTAG	ENZYMES	NCI1	PTC-MR	D&D	IMDB BINARY	IMDB MULTI	REDDIT BINARY	REDDIT MULTI-5K	REDDIT MULTI-12K
SP	1.69x	2.52x	1.62x	1.65x	3.00x	12.42x	17.34x	1.04x	1.05x	1.18x
GR	1.85x	2.94x	1.75x	1.50x	3.44x	7.95x	8.20x	2.24x	2.37x	2.80x
WL	1.76x	2.77x	1.68x	1.62x	3.34x	7.13x	6.84x	1.52x	1.58x	1.54x
PM	1.87x	2.79x	1.68x	1.50x	3.67x	6.92x	6.33x	1.90x	1.98x	1.96x
δ^*_{ave}	2.00	2.98	1.98	1.73	3.96	9.15	8.15	2.33	2.27	2.24

Table 2: Comparison of running times of base kernels vs their core variants. The values indicate the relative increase in running time when compared to the corresponding base kernel.



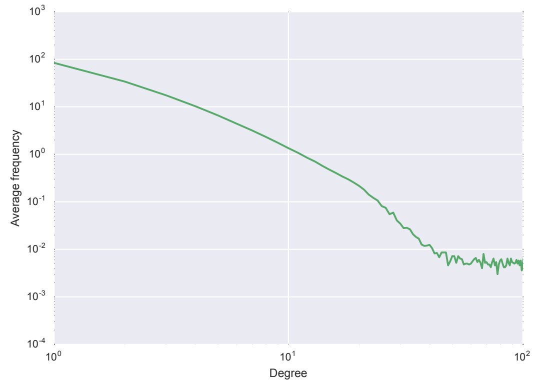


Figure 2: Degree distribution of D&D (left) and REDDIT-BINARY (right) datasets. Both axis of the right figure are logarithmic.

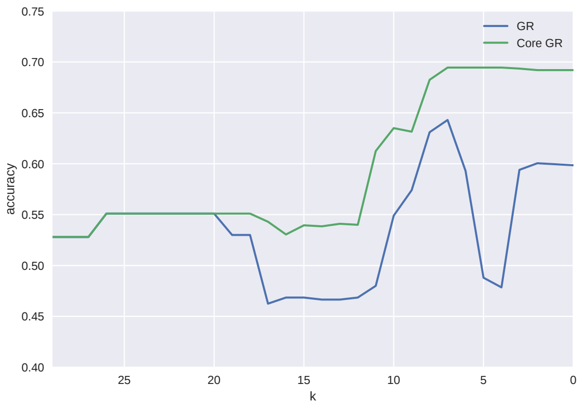


Figure 3: Classification accuracy of the graphlet kernel (GR) and its core variant (core GR) on the IMDB-BINARY dataset for the whole range of k-cores.

sition (leading to small degeneracies).

Finally, we compare the core GR kernel with its base kernel on the whole range of k-cores on the IMDB-BINARY dataset. For $k \in \{0, \ldots, 29\}$, we compute the GR kernel and its core variant, perform graph classification, and compare the achieved classification accuracies. The obtained results are shown in Figure 3. We can see that for $k < 20$, core GR systematically leads to better accuracies compared to its base kernel. The same behavior was also observed on most of the remaining datasets. An interesting observation is that for some k, by retaining only the internal k-cores of the graphs, we can get better classification accuracies compared to the 0-cores (i. e. the input graphs).

5 Conclusion

In this paper, we defined a general framework for improving the performance of graph comparison algorithms. The proposed framework allows existing algorithms to compare structure in graphs at multiple different scales. The conducted experiments highlight the superiority in terms of accuracy of the core variants over their base kernels at the expense of only a slight increase in computational time.

Acknowledgments

The authors would like to thank the anonymous IJCAI reviewers for their insightful comments. Giannis Nikolentzos is supported by the project "ESIGMA" (ANR-17-CE40-0028).

References

[Alvarez-Hamelin *et al.*, 2006] I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. *Advances in Neural Information Processing Systems*, 18:41–50, 2006.

[Batagelj and Zaveršnik, 2011] V. Batagelj and M. Zaveršnik. Fast algorithms for determining (generalized) core groups in social networks. *Advances in Data Analysis and Classification*, 5(2):129–145, 2011.

[Borgwardt and Kriegel, 2005] K.M. Borgwardt and H. Kriegel. Shortest-path kernels on graphs. In *Proceedings of the 5th International Conference on Data Mining*, pages 74–81, 2005.

[Conte *et al.*, 2004] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(03):265–298, 2004.

[Dai *et al.*, 2016] H. Dai, B. Dai, and L. Song. Discriminative Embeddings of Latent Variable Models for Structured Data. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2702–2711, 2016.

[Erdős and Hajnal, 1966] P. Erdős and A. Hajnal. On chromatic number of graphs and set-systems. *Acta Mathematica Hungarica*, 17(1-2):61–99, 1966.

[Gärtner *et al.*, 2003] T. Gärtner, P. Flach, and S. Wrobel. On Graph Kernels: Hardness Results and Efficient Alternatives. In *Learning Theory and Kernel Machines*, pages 129–143. 2003.

[Giatsidis *et al.*, 2014] C. Giatsidis, F. Malliaros, D. Thilikos, and M. Vazirgiannis. CORECLUSTER: A Degeneracy Based Graph Clustering Framework. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 44–50, 2014.

[Haussler, 1999] D. Haussler. Convolution kernels on discrete structures. *Technical Report*, 1999.

[Horváth *et al.*, 2004] T. Horváth, T. Gärtner, and S. Wrobel. Cyclic Pattern Kernels for Predictive Graph Mining. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 158–167, 2004.

[Johansson and Dubhashi, 2015] F. Johansson and D. Dubhashi. Learning with Similarity Functions on Graphs using Matchings of Geometric Embeddings. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 467–476, 2015.

[Johansson *et al.*, 2014] F. Johansson, V. Jethava, D. Dubhashi, and C. Bhattacharyya. Global graph kernels using geometric embeddings. In *Proceedings of the 31st International Conference on Machine Learning*, pages 694–702, 2014.

[Kondor and Pan, 2016] R. Kondor and H. Pan. The Multiscale Laplacian Graph Kernel. In *Advances in Neural Information Processing Systems*, pages 2982–2990, 2016.

[Kriege and Mutzel, 2012] N. Kriege and P. Mutzel. Subgraph Matching Kernels for Attributed Graphs. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1015–1022, 2012.

[Lee *et al.*, 2010] V. Lee, N. Ruan, R. Jin, and C. Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*, pages 303–336. 2010.

[Lick and White, 1970] D. Lick and A. White. k-degenerate graphs. *Canadian J. of Mathematics*, 22:1082–1096, 1970.

[Mahé and Vert, 2009] P. Mahé and J. Vert. Graph kernels based on tree patterns for molecules. *Machine Learning*, 75(1):3–35, 2009.

[Matula and Beck, 1983] D. Matula and L. Beck. Smallest-last Ordering and Clustering and Graph Coloring Algorithms. *Journal of the ACM*, 30(3):417–427, 1983.

[Neumann *et al.*, 2016] M. Neumann, R. Garnett, C. Bauckhage, and K. Kersting. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning*, 102(2):209–245, 2016.

[Nikolentzos *et al.*, 2017a] G. Nikolentzos, P. Meladianos, F. Rousseau, Y. Stavrakas, and M. Vazirgiannis. Shortest-Path Graph Kernels for Document Similarity. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1890–1900, 2017.

[Nikolentzos *et al.*, 2017b] G. Nikolentzos, P. Meladianos, and M. Vazirgiannis. Matching Node Embeddings for Graph Similarity. In *Proceedings of the 31st AAAI Conference in Artificial Intelligence*, pages 2429–2435, 2017.

[Schölkopf *et al.*, 2004] B. Schölkopf, K. Tsuda, and J.P. Vert. *Kernel Methods in Computational Biology*. MIT press, 2004.

[Seidman, 1983] S. Seidman. Network Structure and Minimum Degree. *Social networks*, 5(3):269–287, 1983.

[Shervashidze *et al.*, 2009] N. Shervashidze, T. Petri, K. Mehlhorn, K.M. Borgwardt, and S.V.N. Vishwanathan. Efficient Graphlet Kernels for Large Graph Comparison. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 488–495, 2009.

[Shervashidze *et al.*, 2011] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K.M. Borgwardt. Weisfeiler-Lehman Graph Kernels. *The Journal of Machine Learning Research*, 12:2539–2561, 2011.

[Smola and Schölkopf, 1998] A. Smola and B. Schölkopf. *Learning with kernels*. Forschungszentrum Informationstechnik, 1998.

[Sugiyama and Borgwardt, 2015] M. Sugiyama and K.M. Borgwardt. Halting in random walk kernels. In *Advances in Neural Information Processing Systems*, pages 1639–1647, 2015.

[Vishwanathan *et al.*, 2010] S.V.N. Vishwanathan, N. Schraudolph, R. Kondor, and K.M. Borgwardt. Graph Kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010.

[Wuchty and Almaas, 2005] S. Wuchty and E. Almaas. Peeling the yeast protein network. *Proteomics*, 5(2):444–449, 2005.

[Yanardag and Vishwanathan, 2015] P. Yanardag and S.V.N. Vishwanathan. Deep Graph Kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374, 2015.