Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Graph-based pattern recognition on spectral reduced graphs[☆]

Anthony Gillioz[*], Kaspar Riesen

Institute of Computer Science, University of Bern, Neubrückstrasse 10, Bern, 3012, Switzerland

ARTICLE INFO

Keywords:
Graph matching
Graph classification
Graph reduction

ABSTRACT

Graph-based pattern recognition – in particular in conjunction with large graphs – is often computationally expensive. This hampers, or makes it at least challenging, to employ graph-based representations for real-world data. To address this issue, we propose a method for reducing the size of the underlying graphs to their most important substructures using spectral graph clustering. The proposed method partitions the nodes of the graphs into clusters and then merges each cluster into supernodes. The motivation of this procedure is to reduce the computational cost of any graph comparison algorithm while maintaining the accuracy of the final classification. To assess the benefits and limitations of our method, we conduct thorough experiments on nine real-world datasets with different levels of graph reductions. The classification is obtained by four different graph classifiers (viz. a KNN based on graph edit distance, two SVMs based on a shortest path graph and a Weisfeiler–Lehman graph kernel, as well as a graph neural network). The results indicate that we can reduce computation time by up to two orders of magnitude without substantially degrading the classification accuracy.

1. Introduction

Graphs are one of the most fundamental and powerful data structures in computer science and are thus often used to model complex systems or patterns [1,2]. Basically, a graph consists of finite sets of nodes and edges. The nodes are generally employed to represent the attributes of the basic entities of a pattern and the edges are used to represent the relationships between pairs of those basic entities.

Graph matching is a common and essential task in graph-based pattern recognition [3,4], with applications in many domains (e.g., in computer vision [5], text processing [6], or bioinformatics [7], to name three prominent examples). The goal of any graph matching algorithm is to identify common substructures in the underlying graphs and derive a similarity or dissimilarity score upon the found matching. The widespread use of both graphs and graph matching in pattern recognition motivates the research, development and study of efficient methods for this task. Several methods have been proposed in the literature to perform graph matching [3,4]. Noteworthy instances include the *Graph Edit Distance* [8], *Graph Kernels* [9], and *Graph Neural Networks* [10].

A major limitation of graph-based pattern recognition is its high computational cost. Graph edit distance, for instance, is known to be $\mathcal{NP}$-complete in its general form, making it unfeasible for large real-world graphs. Many graph kernels also have rather high time complexity. To address the computational problems of graph matching, approximation techniques have been proposed for both graph edit distance [11] and graph kernels [12].

An alternative approach for improving the efficiency of graph-based pattern recognition is to work with reduced versions of the original graphs [13]. This idea is used, for instance, in [14] where the graphs are decomposed using the Fiedler vector in a hierarchical graph simplification process. The authors of [15] introduce another graph reduction technique using centrality measures to select essential nodes. The resulting graphs substantially improve the computation time in graph-based pattern recognition scenarios. In [16], a reduction method is proposed based on a novel spectral coarsening algorithm that uses the head and tail eigenvalues to obtain a multilevel graph representation.

In the present paper, we propose a novel two-step approach with the aim of substantially reducing the time required for graph classification. In a first step, the original graphs are reduced to a given level using spectral clustering. In a second step, graph matching and classification is performed on the reduced graphs (using both graph edit distance in conjunction with a KNN, two graph kernels with an SVM, and a graph neural network). The main contribution of this paper is to investigate the effects of this graph reduction process in a typical graph-based pattern recognition scenario. In particular, we evaluate the effectiveness of the proposed method by comparing the

☆ This material is based upon work supported by the Swiss National Science Foundation (SNSF) under Grant Nr. 200021_188496.

* Corresponding author.
E-mail addresses: anthony.gillioz@unibe.ch (A. Gillioz), kaspar.riesen@unibe.ch (K. Riesen).

https://doi.org/10.1016/j.patcog.2023.109859
Received 20 February 2023; Received in revised form 10 July 2023; Accepted 28 July 2023
Available online 1 August 2023

matching time and classification accuracy achieved on the reduced graphs with the corresponding metrics observed in the original graph domain. This experimental setup allows us to study the impact of the proposed spectral graph reduction on graph-based pattern recognition computation. To the best of our knowledge, this is the first contribution that is based on spectral graph clustering for reducing the size of a graph for pattern recognition purposes.

The remainder of the present paper is organized as follows. In Section 2, we provide a brief overview of the basic definitions of graphs, graph edit distance, the graph kernels and the graph neural network actually used. In Section 3, we describe the spectral graph clustering employed to find the graph partitioning and then introduce the novel method for graph reduction. In Section 4, we describe the experimental setup and present the main results of the empirical investigation. Finally, in the last section, we summarize our findings and suggest directions for future work.

2. Background

In this section, we formally introduce and review several basic concepts necessary in our novel framework. In particular, we formally introduce the concept of a graph in Section 2.1 and then review, in Sections 2.2, 2.3, and 2.4, the graph matching approaches actually used in our empirical evaluation.

2.1. Basic graph theory

A *graph* $G = (V, E)$ is a tuple of finite sets V and E, where V is a non-empty set of n *nodes* (sometimes called *vertices*) and $E \subseteq V \times V$ is a set of m *edges*. In a *simple, undirected graph* an edge $e \in E$ is an unordered pair of distinct nodes $(v_i, v_j) \in V \times V$ that represents any kind of relationship that may exist between nodes v_i and v_j. The nodes v_i and v_j are also called *endpoints* of an edge $e = (v_i, v_j)$, and the adjacency between node v_i and v_j via edge e is denoted by $v_i \sim v_j$. The *degree* of a node $v \in V$ (denoted by $\deg(v)$) is the number of incident edges to node v.

An n-dimensional feature vector $\mathbf{x}_i \in \mathbb{R}^n$, termed *label*, might be attached to any node $v_i \in V$ containing real-valued node attributes. The same accounts for the edges. In the case that there are labels on the nodes or edges (or both), one speaks of *labeled*, or *attributed*, graphs. The graph domain $\mathcal{G} = \{G^{(1)}, \dots, G^{(N)}\}$ is defined as a set of N graphs contained in a given dataset.

A graph $G = (V, E)$ is termed *connected*, if there exists at least one *path* connecting every pair of nodes $v_i, v_j \in V \times V$. Otherwise, if there exists at least one pair of nodes $v_i, v_j \in V \times V$ that is not connected by a path, graph G is referred to as *disconnected*. The structure of a graph $G = (V, E)$ is often encoded by means of the adjacency $\mathbf{A}$, degree $\mathbf{D}$, or Laplacian $\mathbf{L}$ matrix.

Given N i.i.d. classified training graphs $\mathcal{D} = \{G_i, y_i\} \subseteq (\mathcal{G} \times \mathcal{Y})$, where $\mathcal{G}$ represents the graph domain and $\mathcal{Y}$ the corresponding class label alphabet. The *graph classification task* consists of finding a model $f : \mathcal{G} \to \mathcal{Y}$ that assigns a class label $y \in \mathcal{Y}$ to any input graph $G \in \mathcal{G}$. In this paper, we make use of graph edit distance coupled with a KNN classifier, two graph kernels jointly used with an SVM classifier, and a graph neural network as graph classification model f. Hence, we cover three of the most commonly used families of graph classifiers, and we briefly review these concepts in the next three subsections.

2.2. Distance-based graph classification

The traditional approach for graph classification is based on the *K-Nearest Neighbor* (KNN) algorithm. The KNN algorithm assigns a class label to a new sample based on the majority class of its nearest neighbors in the feature space. As this classifier depends on a dissimilarity measure only, it is particularly well suited for graph-based pattern classification.

Dissimilarity measures for graphs are often defined upon graph matching. *Graph matching* [3,4] corresponds to the general task of comparing two graphs $G = (V, E)$ and $G' = (V', E')$ with each other and trying to find similar (sub-)structures in G and G'.

A popular error-tolerant graph matching method is *Graph Edit Distance* [8]. Its high flexibility and adaptability allows graph edit distance to be applied to a wide range of problems [17,18]. One of the major benefits of graph edit distance, compared to other graph dissimilarity measures, is that it provides information about how the substructures of the graphs actually match with each other. To this end, graph edit distance computes an *edit path* $\lambda = \{e_1, \dots, e_k\}$ which represents the k *edit operations* actually necessary to transform graph G into graph G'. Three edit operations are commonly employed, namely *insertion*, *deletion*, and *substitution* which are defined on both nodes and edges. A cost function $c(e_i)$ associated with each edit operation e_i is generally used to formalize the severity of operation e_i. The graph edit distance $\mathrm{GED}(G, G')$ between two graphs G and G' is then defined by the overall cost of the minimum cost edit path between G and G'.

The overall complexity of graph edit distance optimization is known to be $\mathcal{NP}$-complete for general graphs [19], which hinders its application to large-scale problems. However, many graph edit distance approximations have been proposed in the last decade [11,20]. Note that, these sub-optimal algorithms do not guarantee to find the global minimum edit path. In the present paper, we use a sub-optimal graph edit distance computation that offers cubic time complexity in the number of nodes [20].

2.3. Kernel based graph classification

Graph Kernels [9] constitute another prominent family of graph classification algorithms. Roughly speaking, a graph kernel is a measure of similarity between graphs that compares their underlying structures. More formally, a graph kernel is a symmetric, positive semi-definite function $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ defined on the graph domain $\mathcal{G}$.

The vast majority of graph kernels proposed are instances of so-called *convolution kernels*. Given two graphs G and G', the idea of the convolution framework is to decompose G and G' into substructures and evaluate a kernel between each pair of such substructures. Using a convolution operation, these similarities are then turned into a kernel function on the complete graphs. Prominent examples are, for instance, walk kernels [12], cycle kernels [21], or subgraph kernels [22], to name just three examples. In the present paper we make use of two widely applied graph kernels, viz. the shortest path kernel [23] and the Weisfeiler–Lehman kernel [24] in conjunction with *Support Vector Machines* (SVMs).

The main concept of the *Shortest-Path Kernel* [23] is to derive a kernel $k_{\mathrm{SP}} : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ based on both attributes and length of the shortest paths between pairs of nodes $(v_i, v_j) \in V \times V$. The similarities between shortest paths in both graphs G and G' are then aggregated to obtain an overall similarity score between G and G'.

The *Weisfeiler–Lehman kernel* is another popular graph kernel with efficient classification performance [24]. This kernel works on top of a well-known graph isomorphism algorithm, namely the Weisfeiler–Lehman graph isomorphism test. This test consists of an iterative method that produces a canonical form for each graph. In each iteration, the current feature label l of a given node v_i is aggregated with the labels of all adjacent nodes and replaced with a new compressed label.

The Weisfeiler–Lehman graph kernel repeats the above-described procedure $h \geq 0$ times and then compares the final sets of node labels between the two graphs. The similarity between the node label sets is then used as a measure of similarity between the graphs.

2.4. Graph neural network based classification

Graph Neural Networks (GNNs) [10] are a type of deep learning method that is specifically designed to work with graph-based data. At a high level, GNNs learn representations for each node in a graph based on their local neighborhood structure. This is done by propagating information between neighboring nodes in the graph, in a way that is similar to how information spreads in a social network. This allows GNNs to capture important local patterns and relationships in the graph, while also leveraging the overall structure of the graph to make predictions.

The typical architecture of a GNN consists of several layers of computation, each of which involves two main steps, viz. message passing and node updating. In the message-passing step, each node in the graph sends a message to its neighboring nodes, based on its current feature representation. These messages are then aggregated and transformed into a new representation for each node. In the node updating step, each node combines its new representation with its old representation, using a neural network to compute a new feature vector. This new feature vector is then passed on to the next layer in the GNN, or used for making predictions about the graph.

There are different variations of GNNs available. In the present paper, we make use of the *Deep Graph Convolutional Neural Network* (DGCNN) [25]. This architecture consists of three consecutive stages. First, graph convolutional layers are used to extract local substructure features of the nodes and establish a consistent node ordering. Second, a SortPooling layer arranges the node features in the established order and standardizes input sizes. Third, traditional convolutional and dense layers are utilized to process the sorted graph representations and generate the final classification.

3. Graph reduction method

The major objective of the present paper is to introduce and research a novel method for substantially reducing the size of graphs while preserving their essential properties. These properties can vary depending on the specific problem at hand. In this work, we aim to maintain the classification accuracy achieved on the reduced graphs as close as possible to the one obtained on the original graphs.

Specifically, given a graph $G = (V, E)$ with n nodes and m edges, we create a reduced graph $G_\rho = (V_\rho, E_\rho)$ with $n_\rho < n$ nodes and $m_\rho < m$ edges such that G_ρ is a good approximation of G in some sense [26]. Parameter $\rho \in \mathbb{N}$ is a user-defined reduction factor that controls the extent to which the original graph is reduced. For example, a value of $\rho = 2$ results in a reduction of the graph size (i.e., the number of nodes) by approximately 50%. The reduced graph domain $\mathcal{G}_\rho = \{G_\rho^{(1)}, \dots, G_\rho^{(N)}\}$ is obtained from the original graph domain $\mathcal{G}$ by reducing all graphs $G \in \mathcal{G}$ according to the defined reduction procedure.

The aim of the following two subsections is twofold. First, in Section 3.1, we elaborate on the principles of spectral graph clustering, which builds the basis of our reduction method. Second, in Section 3.2, we demonstrate how one can employ the node partitioning resulting from spectral clustering to achieve a substantial and meaningful reduction of the underlying graphs.

3.1. Graph clustering

Graph Clustering [27] is a technique for dividing the nodes of a graph into groups, called *clusters*, such that the nodes within each cluster are closely related in some pre-defined sense. The clustering of the nodes is typically based on the underlying structure of the graph so that nodes belonging to the same cluster must exhibit "similar behavior".

In the present paper, spectral graph clustering [28] is used as the basis for graph reduction, leveraging key properties of the graph Laplacian $\mathbf{L}$ [29].

The basic idea behind spectral graph clustering is to compute a node embedding based on the k smallest eigenvectors of the graph Laplacian matrix (where k is the number of clusters). This embedding encodes information about the connections between the nodes and can thus be used to identify clusters of densely connected nodes within the graph. Once the embedding has been computed, standard clustering methods (such as k-means [30] or others) can be applied to find the node partitioning of the graph.

For our specific purpose of fast graph reduction, we apply a slight modification to the standard spectral clustering algorithm to improve the computational efficiency of the eigendecomposition (line 2). Instead of embedding the nodes in a k-dimensional space, where k is the number of clusters to be identified in the graph, we embed the nodes in a d-dimensional space with $d < k$. This reduction in the number of dimensions speeds up the computation of the eigenvalues and eigenvectors, and therefore the overall clustering method. The value of d is treated as a free parameter and is optimized for each dataset individually.

3.2. Graph reduction

Based on the spectral graph clustering described above, we propose to coarsen the underlying graphs in a novel graph reduction approach, which is potentially able to preserve important structural features of the graphs. We are aware that graph reduction has been largely investigated in the literature under different formalisms and different names, such as *graph summarization*, *graph coarsening*, and *graph clustering* [13,26]. Moreover, connections between graph reduction and spectral algorithms were also explored in [16,31]. The major contribution of the present paper is that we thoroughly evaluate the effects of reduced graphs on a wide range of graph classifiers.

Before applying the spectral graph clustering algorithm, we first conduct a pre-processing step to all of the graphs in the underlying dataset. In this pre-processing step, we consider all connected components of the graphs as distinct graphs. In other words, in case a certain graph $G \in \mathcal{G}$ consists of more than one connected component, we individually apply the spectral graph clustering to each connected component. This pre-processing step turns out to be necessary because the spectral clustering algorithm may not produce meaningful results when applied to graphs that consist of more than one connected component (due to the multiplicity of zero eigenvalues, which can result in problematic node embeddings).

The basic idea of the proposed graph reduction approach is to condense the nodes of one cluster into supernodes and simplify the edge structure between these supernodes. To create the supernodes, we first obtain the graph partitioning by means of the spectral clustering algorithm and then condense all the nodes in a given partition into a single entity without considering the intra-cluster edges. The computation of the feature vector attached to the supernode is achieved by summing up the n feature vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$ of the n nodes that belong to the same cluster.[1] This process effectively reduces the size of the graph by replacing multiple nodes with a single supernode. Additionally, we condense all inter-cluster edges into a single edge between two corresponding supernodes in order to simplify the edge structure while preserving the connections between the clusters.

In Fig. 1, we illustrate how our supernode creation process operates. In this example, we show two levels of reduction, with $\rho = 2$ and $\rho = 4$, and how nodes are merged at each level once the node clustering is determined. The clusterings obtained with $\rho = 2$ and $\rho = 4$ are represented by blue and red circles, respectively. In the reduced graphs $G_{\rho=2}$ and $G_{\rho=4}$, the node feature vectors correspond to the sum of the feature vectors of the nodes belonging to the blue and red clusters, respectively. Note also how the inter-cluster edges are simplified during the reduction process.

[1] Note that other, in particular more elaborated, methods for the labeling of supernodes could be defined — see future work in Section 5.

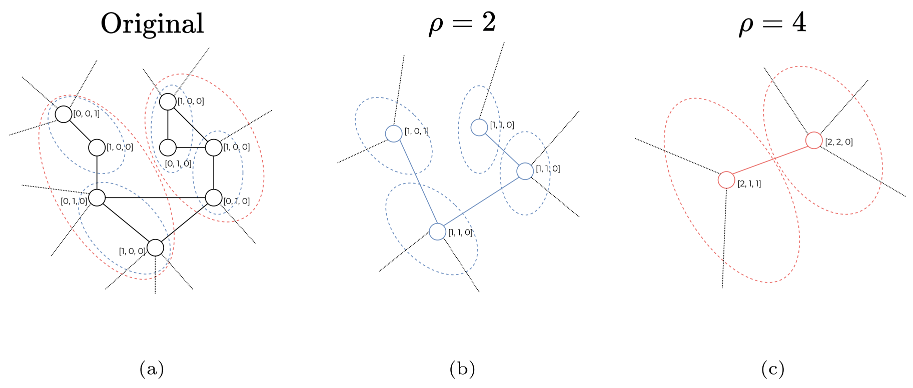

Fig. 1. Example of the supernode creation process. We show parts of the original graph G in (a) and two corresponding graph reductions in (b) and (c) with $\rho = 2$ and $\rho = 4$, respectively.

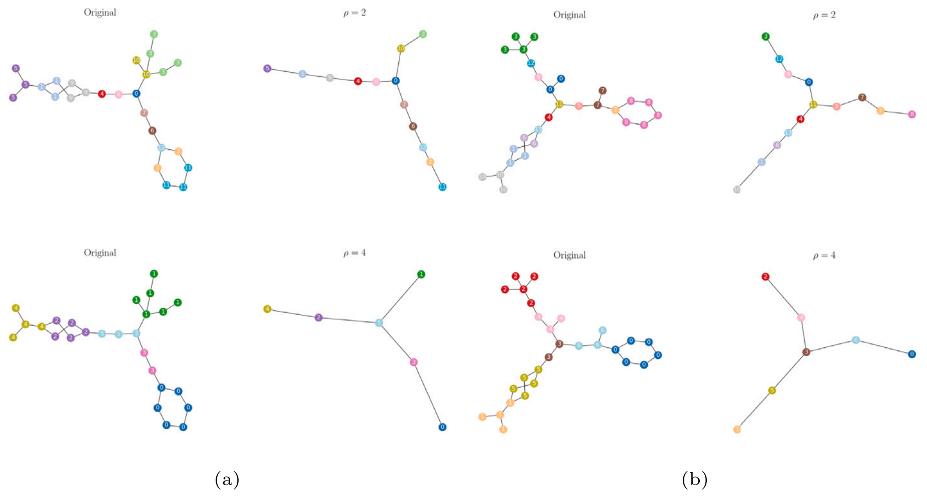

Fig. 2. Two examples of the graph reduction algorithm on two different graphs (a) and (b) from the NCI1 dataset using two reduction factors $\rho = 2$ and $\rho = 4$. Each color corresponds to a cluster in the original graph and the corresponding color in the reduced graph represents the condensation of all nodes in the respective cluster into a supernode. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In our reduction scheme, the number of clusters k corresponds to the number of supernodes obtained upon merging the nodes (and vice versa). Hence, value k determines the size of the reduced graph that is finally created. The number of nodes in each cluster is proportional to the reduction factor ρ, which specifies the amount by which the size of a graph must be divided to obtain the desired reduction. In particular, the number of clusters can be calculated as $k = \frac{|V|}{\rho}$, where $|V|$ is the total number of nodes in the graph. For example, if we have a graph with $|V| = 1000$ nodes and a reduction factor of $\rho = 2$, the size of the graph would be reduced by a factor of two, resulting in $k = \frac{1000}{2} = 500$ clusters.

In Fig. 2, we provide two real-world examples that illustrate the graph reduction process. We use graphs from the NCI1 dataset (see Section 4.1 for details on this particular dataset). We apply a graph reduction with parameters $\rho = 2$ and $\rho = 4$. As can be seen in the figure, both clustering and reduction reflect the communities present in the graphs. In the original graphs, each cluster found is represented by a unique color, and in the reduced graphs, the color corresponds to the reduction of nodes within that cluster into a single supernode. An interesting aspect of spectral partitioning is that it might produce unequally sized clusters, with some clusters containing only a single node while others containing multiple nodes. This behavior is noteworthy because we typically want to keep the main communities within the same cluster.

4. Experimental evaluation

This section is organized as follows. First, in Section 4.1, we briefly outline the datasets used in our evaluation. Next, in Section 4.2, we summarize the experimental setup. In Section 4.3, we present the major results obtained on the datasets, and finally, in Section 4.4, we present a qualitative evaluation of the effects of the proposed reduction.

4.1. Datasets

We conduct empirical evaluations of the novel reduction approach using nine datasets from the TUDataset graph repository [32]. Table 1 provides details on the number of graphs, and classes, as well as the average, minimum, and maximum number of nodes and edges per graph for each dataset.

The first six datasets (BZR, DHFR, Enzymes, Mutagenicity, NCI1, and NCI109) are composed of graphs that represent real-world molecules and their potential effects or activities. The BZR dataset consists of graphs that represent ligands for the benzodiazepine receptor, while the DHFR dataset contains inhibitors of dihydrofolate reductase. The Enzymes dataset is divided into six classes and includes graphs that encode protein structure elements. The Mutagenicity dataset includes graphs representing chemical compounds and their potential to cause mutation. The NCI1 and NCI109 datasets, which are sourced from the National Cancer Institute, consist of graphs representing molecular compounds and their ability to inhibit the growth of cancerous or noncancerous cells. The last three datasets (COLLAB, REDDIT-MULTI-5K,

Table 1
Statistics of the graph datasets. We show the number of graphs ($|G|$), the number of classes ($|\Omega|$), and the average ($\emptyset|V|$, $\emptyset|E|$), minimum ($\min|V|$, $\min|E|$), and maximum ($\max|V|$, $\max|E|$) number of nodes and edges per dataset.

Dataset			∅		min		max	
	$\|G\|$	$\|\Omega\|$	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$
BZR	405	2	35.7	38.4	13	13	57	60
DHFR	467	2	42.4	44.5	20	21	71	73
Enzymes	600	6	32.6	62.1	2	1	126	149
Mutagenicity	4,337	2	30.3	30.8	4	3	417	112
NCI1	4,110	2	29.9	32.3	3	2	111	119
NCI109	4,127	2	29.7	32.1	4	3	111	119
COLLAB	5,000	3	74.5	2,457.8	32	60	492	40,119
REDDIT-MULTI-5K	4,999	5	508.5	594.9	22	21	3,648	4,783
REDDIT-MULTI-12K	11,929	11	391.4	456.9	2	1	3,782	5,171

Table 2
Mean of the graph densities of the original graphs and their reduced counterparts (for $\rho = 4$ and $\rho = 16$) for all datasets.

Dataset	Original	$\rho = 4$	$\rho = 16$
BZR	0.06	0.27	0.70
DHFR	0.05	0.21	0.81
Enzymes	0.16	0.32	0.57
Mutagenicity	0.09	0.37	0.50
NCI1	0.09	0.33	0.50
NCI109	0.09	0.34	0.49
COLLAB	0.51	0.54	0.79
REDDIT-MULTI-5K	0.01	0.04	0.15
REDDIT-MULTI-12K	0.02	0.08	0.21

and REDDIT-MULTI-12K) contain graphs that represent different social media networks. COLLAB, for instance, consists of graphs that represent the collaboration networks of researchers from various fields of physics. Each graph in this dataset is labeled with one of three fields of the corresponding researcher. The REDDIT-MULTI-5K and REDDIT-MULTI-12K datasets consist of graphs that represent discussions from various subreddits, labeled with the subreddit it belongs to.

Table 1 reveals that on most of the graph datasets, the number of nodes are quite similar to the number of edges. This implies that the graphs are sparse, which is positive in our scenario. Spectral graph clustering is particularly efficient on sparse graphs because it involves computing the eigendecomposition of the Laplacian matrix $\mathbf{L}$ (which is computationally efficient on sparse matrices).

We are aware that there are numerous additional datasets available in the TUDataset graph repository that could potentially be used in our evaluation. However, rather simple baseline approaches, such as global sum pooling of node features, which reduce graphs to a single feature vector (completely neglecting the edge structure), perform very well and even outperform elaborated graph kernels in some cases [33]. Therefore, we only use datasets in our analysis for which the classification accuracy using this naïve baseline approach is lower than the accuracy obtained by an SVM using a 4-Weisfeiler–Lehman kernel (i.e., a Weisfeiler–Lehman kernel with $h = 4$).[2]

An interesting question is whether or not the graph reduction process has a substantial influence on the graph density. To find this out, we show in Table 2 the average graph density for both the original graphs and reduced graphs (with reduction levels of $\rho = 4$ and $\rho = 16$). We observe that the original graphs are generally sparse (except the COLLAB dataset where a mean density of 0.51 is observed). On the other datasets, the mean densities range from 0.01 (on REDDIT-MULTI-5K) to 0.16 (on Enzymes). We observe a trend towards increasing graph densities as the reduction factor is increased. On some data sets the increase in density is substantial. For instance, on BZR the density is increased from 0.06 to 0.27 and 0.70 for $\rho = 4$ and $\rho = 16$, respectively. However, it is also observed that even with the strongest reduction, complete graphs are not obtained (the highest density is obtained on the DFHR data with 0.81).

4.2. Experimental setup

For each dataset described in the previous subsection, we create reduced graph domains $\mathcal{G}_\rho$ by reducing the original graphs with reduction factors $\rho \in \{2, 4, 8, 16\}$. These reduction factors lead to slightly reduced graphs (when $\rho = 2$) to quite strongly reduced graphs (when $\rho = 16$). The reduction process is not applied on graphs that have a number of nodes $|V|$ already smaller than, or equal to, ρ. For each reduction factor ρ, we generate reduced graphs using different dimensions for node embedding during spectral clustering. That is, the dimension of the node embedding space is varied in $d \in \{2, 3, 4, 5, 8\}$. Note that d is treated as an additional free hyperparameter and is chosen during the optimization phase.

For each experiment, we produce stratified splits of the datasets into training, validation, and test sets using a 60%, 20%, and 20% split size, respectively. For each dataset, we optimize the classifiers and hyperparameters five times with different data splits and random initialization by means of the validation sets. Finally, the mean and standard deviation of the classification results for the five runs obtained on the test sets are reported.

For the computation of graph edit distance, we use unit costs for both node and edge insertions/deletions. To calculate the node substitution cost $c(u_i \rightarrow v_i)$, we utilize the Euclidean distance between the node features $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively, with a cost limit of 2.0. Formally, the substitution cost is $c(u_i \rightarrow v_i) = \min(\|\mathbf{x}_i - \mathbf{x}_j\|, 2.0)$.

This definition ensures that the substitution cost is never greater than the sum of cost of a deletion and a subsequent insertion. Parameter $\alpha \in]0,1[$ represents the relative importance of node and edge edit operation costs and is varied from 0.1 to 0.9 in increments of 0.1 in our evaluation (with the exception of both REDDIT datasets on the original graphs and REDDIT-MULTI-12K with reduction level $\rho = 2$. Here, α is fixed to 0.5 due to the high computational cost).

The only parameter that needs to be optimized for the KNN classifier is the number of neighbors k considered in the classification process. We optimize this parameter in the range $k \in \{3, 5, 7\}$. For the SVM we optimize parameter $C \in 10^{\{-2.0,-1.5,\ldots,2.0\}}$, which serves as a regularization parameter to control the trade-off between the requirements of large margins and few misclassifications. In other words, C controls what is more important, the minimization of the structural risk or the minimization of the empirical risk. For the training process of the GNN experiments, we use the hyperparameters as proposed in [25].

The experimental evaluation is divided into two parts. In the first part, described in Section 4.3, we evaluate the classification performance of the four graph classification algorithms reviewed in Sections Section 2.2, 2.3, and 2.4. By comparing the classification accuracy of these four systems on both the original graphs and the reduced counterparts we can examine the power of the proposed reduction mechanism. We also compare the runtime of the matching and kernel algorithms on both the original and reduced graphs in order to observe the time gain attributable to our novel approach.

In the second part of the evaluation, in Section 4.4, we present scatter plots that visualize the correlations between the similarity/dissimilarity values obtained on the original graphs and the similarity/dissimilarity values obtained on the reduced graphs.

4.3. Classification accuracy and computation time

In this subsection, we address the following two research questions:

[2] We report the results of this naïve baseline approach together with the results of the novel method in Section 4.3.

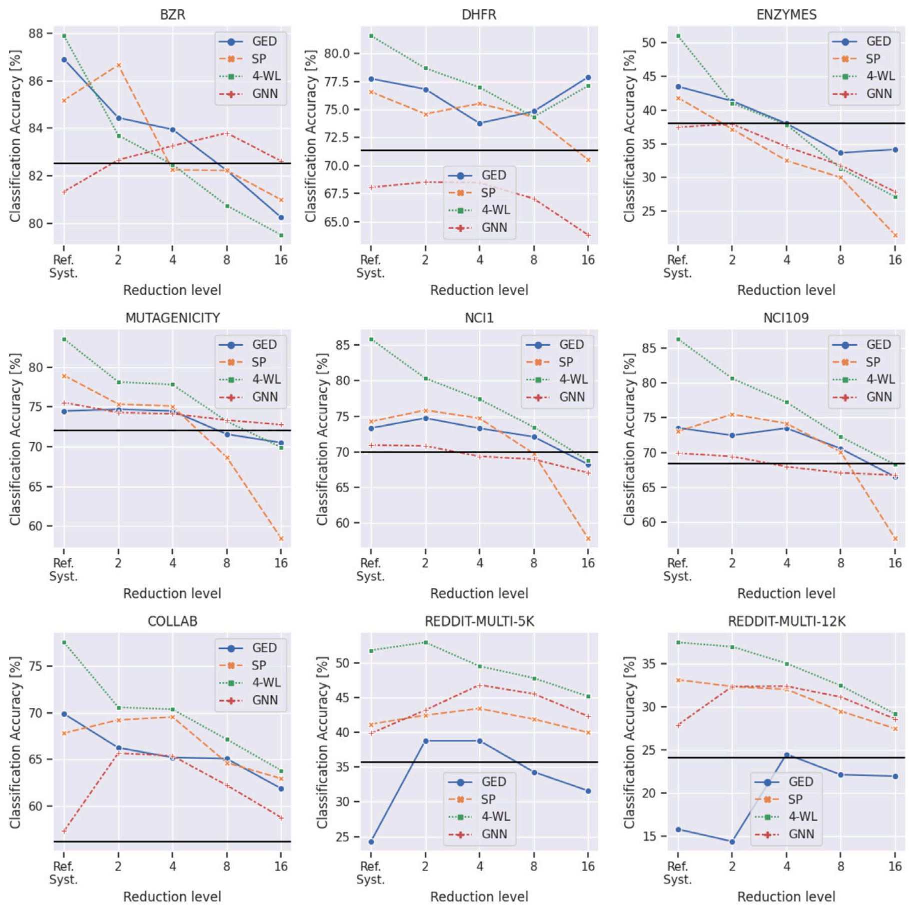

Fig. 3. Graph classification accuracies for all datasets and all reduction levels using GED (Graph Edit Distance and KNN), SP (Shortest Path Kernel and SVM), 4-WL (4-Weisfeiler–Lehman Kernel and SVM), and GNN (DGCNN graph neural network), including the corresponding reference systems that rely on the original graphs ($\rho = 1$). The black horizontal lines represent the results obtained with the baseline, where all the graphs are reduced to a single feature vector.

Q.1. Does the proposed reduction lead to graphs on which a significant decline in classification accuracy is observed (compared to using the original graphs)? Are there substantial differences among the four classifiers employed on the reduced graphs?

Q.2. How large is the runtime improvement that can be achieved by performing graph classification on the reduced rather than on the original graphs?

In order to answer question **Q.1**, we present the classification accuracies for each reduced graph domain across all datasets and all classifiers in Fig. 3. Additionally, in each figure, the black horizontal lines represent the results obtained with the baseline where all graphs are reduced to a single feature vector by means of a global sum pooling. The classification of these vectors is then performed with an SVM based on a RBF kernel. Overall, we observe a general, yet relatively slight, decrease in classification accuracy as the size of the graphs is reduced. However, the classification accuracy remains relatively stable even with strongly reduced graphs. This is particularly noteworthy as it indicates that the reduced graphs still retain sufficient information for accurate classification. For example, the classification accuracies of the KNN using graph edit distance obtained on the datasets BZR, DHFR, MUTAGENICITY, NCI1, and NCI109 remain relatively consistent even with strongly reduced graphs. This observation is also valid for the two kernel classifiers. However, on some datasets, we also observe rather strong reductions of the classification accuracies — in particular when strong graph reductions are applied (see for instance the ENZYMES dataset where the accuracy drop is clearly visible for all classifiers).

Note that on some datasets – especially on the unlabeled datasets REDDIT-MULTI-5K and REDDIT-MULTI-12K – we can actually improve the classification accuracy when the classification is performed on the reduced rather than the original graphs. This phenomenon might be attributed to the fact that we fixed parameter α to 0.5 for the reference system (with $\rho = 1$) in order to avoid computational expenses and thus the results shown here are somehow sub-optimal.

In general, we observe that the black horizontal line, representing the results obtained with the baseline where all graphs are reduced to a single feature vector, is only crossed when the graphs are strongly reduced (with reduction levels of $\rho = 8$ or $\rho = 16$). However, on the DHFR dataset and both REDDIT datasets, we observe that even for the original graphs and slightly reduced graphs with $\rho = 2$ and $\rho = 4$, the accuracies obtained with GNN and GED, struggle to surpass the accuracy of the naïve baseline. This indicates that neither GNN nor GED are well-suited methods for solving those tasks.

Overall the results suggest that our approach is effective in improving the accuracy of pattern recognition systems based on graph representations, although, there may be some instances where the baseline outperforms the novel approach (particularly for graphs that have undergone significant reduction).

To provide a more precise analysis of the relative differences among the different systems, we present the classification accuracies for all datasets, classification methods, and reduction levels in Table 3. This table allows us to compare the number of instances where the classification accuracy achieved on the reduced graphs is statistically significantly worse than the accuracy achieved on the original graphs. We employ a t-test using the classification accuracy of the five runs to determine if there is a statistically significant difference in accuracy between the reference system and the systems that use the reduced graphs.

Table 3
Classification accuracies obtained by all classifiers, viz. a KNN using graph edit distance (GED), as well as the shortest path graph kernel (SP), the 4-Weisfeiler–Lehman graph kernel (4-WL) in conjunction with an SVM and the DGCNN graph neural network (GNN). We present results on all datasets and reduction levels, i.e., Ref. System ($\rho = 1$) and $\rho \in \{2, 4, 8, 16\}$. Using symbols ∘/•, we indicate results that are statistically significantly better or worse than those achieved with the reference system, respectively.

Dataset	Classifier	Ref. System	$\rho = 2$	$\rho = 4$	$\rho = 8$	$\rho = 16$
BZR	GED	86.9 ± 4.0	84.4 ± 5.4	84.0 ± 6.1	82.2 ± 3.5	80.2 ± 5.5
	SP	85.2 ± 3.7	86.7 ± 3.3	80.2 ± 3.7	82.2 ± 3.5	81.0 ± 3.5
	4-WL	87.9 ± 1.6	83.7 ± 3.0•	82.5 ± 2.7•	80.7 ± 2.8•	79.5 ± 3.2•
	GNN	81.3 ± 0.8	82.7 ± 1.0	83.3 ± 0.5∘	83.8 ± 0.7∘	82.6 ± 0.7∘
DHFR	GED	77.7 ± 1.0	76.8 ± 2.4	73.8 ± 3.1•	74.8 ± 1.1•	77.9 ± 3.4
	SP	76.6 ± 3.1	74.6 ± 1.9	75.5 ± 3.9	74.3 ± 2.9	70.5 ± 3.3•
	4-WL	81.6 ± 2.9	78.7 ± 2.0	77.0 ± 1.8•	74.3 ± 2.8•	77.1 ± 2.8•
	GNN	68.0 ± 1.4	68.5 ± 1.1	68.5 ± 1.1	67.0 ± 0.5	63.8 ± 0.7•
ENZYMES	GED	43.5 ± 5.5	41.3 ± 3.1	38.0 ± 3.9	33.7 ± 3.6	34.2 ± 5.1
	SP	41.8 ± 4.0	37.2 ± 1.9	32.5 ± 2.4•	30.0 ± 2.8•	21.5 ± 2.7•
	4-WL	51.0 ± 3.5	41.0 ± 3.9•	37.8 ± 4.3•	31.3 ± 2.9•	27.2 ± 1.9•
	GNN	37.5 ± 1.6	37.9 ± 1.6	34.6 ± 0.4•	31.8 ± 1.1•	27.9 ± 0.5•
MUTAGENICITY	GED	74.5 ± 1.6	74.7 ± 1.7	74.5 ± 1.2	71.6 ± 1.5•	70.5 ± 1.4•
	SP	79.0 ± 1.2	75.4 ± 1.5•	75.1 ± 1.5•	68.7 ± 1.3•	58.5 ± 1.4•
	4-WL	83.5 ± 1.3	78.2 ± 1.1•	77.8 ± 1.2•	73.2 ± 1.2•	69.9 ± 1.4•
	GNN	75.6 ± 0.6	74.3 ± 0.3•	74.1 ± 0.4•	73.3 ± 0.2•	72.8 ± 0.2•
NCI1	GED	73.3 ± 1.1	74.7 ± 1.4	73.3 ± 0.9	72.1 ± 0.9•	68.2 ± 1.6•
	SP	74.3 ± 1.0	75.9 ± 0.9	74.7 ± 1.3	69.8 ± 0.9•	57.9 ± 1.2•
	4-WL	85.8 ± 1.1	80.3 ± 1.1•	77.4 ± 0.5•	73.5 ± 1.1•	68.8 ± 1.2•
	GNN	70.9 ± 0.9	70.8 ± 0.7	69.4 ± 0.4•	69.0 ± 0.2•	67.1 ± 0.3•
NCI109	GED	73.5 ± 2.1	72.4 ± 1.3	73.5 ± 2.3	70.6 ± 0.9•	66.5 ± 0.7•
	SP	73.0 ± 0.8	75.5 ± 1.5∘	74.2 ± 2.0	70.1 ± 1.5•	57.7 ± 1.9•
	4-WL	86.2 ± 1.3	80.6 ± 1.0•	77.2 ± 1.9•	72.3 ± 0.8•	68.3 ± 1.3•
	GNN	69.9 ± 0.8	69.4 ± 0.5	68.0 ± 0.4•	67.1 ± 0.2•	66.8 ± 0.1•
COLLAB	GED	69.9 ± 1.2	66.3 ± 1.6•	65.2 ± 1.3•	65.1 ± 1.8•	61.9 ± 2.2•
	SP	67.8 ± 1.7	69.2 ± 1.0	69.6 ± 1.0	64.7 ± 2.2•	63.0 ± 1.9•
	4-WL	77.6 ± 0.5	70.6 ± 1.6•	70.4 ± 1.7•	67.2 ± 2.1•	63.8 ± 2.0•
	GNN	57.3 ± 0.3	65.9 ± 0.4∘	65.4 ± 0.2∘	62.2 ± 0.5∘	58.8 ± 0.1∘
REDDIT-MULTI-5K	GED	24.9 ± 1.1	38.3 ± 1.7∘	38.1 ± 1.2∘	34.2 ± 1.9∘	31.1 ± 2.1∘
	SP	41.2 ± 0.9	42.5 ± 0.5∘	43.4 ± 1.8∘	41.9 ± 1.6	40.0 ± 0.8•
	4-WL	52.3 ± 1.1	52.9 ± 1.2	49.5 ± 0.9•	47.8 ± 1.0•	45.2 ± 0.9•
	GNN	39.9 ± 0.2	43.2 ± 0.3∘	46.8 ± 0.3∘	45.4 ± 0.4∘	42.3 ± 0.7∘
REDDIT-MULTI-12K	GED	15.3 ± 0.7	14.1 ± 0.5•	24.3 ± 1.2∘	21.8 ± 1.6∘	21.0 ± 1.7∘
	SP	33.8 ± 0.4	32.4 ± 0.7	32.1 ± 0.8	29.5 ± 0.9•	27.5 ± 0.2•
	4-WL	37.0 ± 0.8	37.0 ± 1.2	35.1 ± 0.7i•	32.5 ± 0.4•	29.2 ± 0.4•
	GNN	27.9 ± 0.9	32.3 ± 0.3∘	32.4 ± 0.3∘	31.2 ± 0.4∘	28.6 ± 0.4

We first analyze the effects of the different reduction levels by comparing all datasets and classifiers simultaneously. When the reduction levels are small, we observe no statistically significant difference to the original graphs in 26 out of 36 cases and 19 out of 36 cases for $\rho = 2$ and $\rho = 4$, respectively. However, as the reduction levels increase to $\rho = 8$ and $\rho = 16$, the results degrade and we find that only 12 out of 36 and 9 out of 36 cases are statistically equivalent to the original system, respectively. These observations, together with the insights from Fig. 3, suggest that it is possible to obtain reasonable results on the reduced graphs, but that it is considerably more difficult to do so (especially with strongly reduced graphs).

Next, we compare the four different classifiers for all datasets and reduction levels simultaneously. We observe that when using graph edit distance, we obtain results that are statistically equivalent to those obtained on the original graphs in 23 out of 36 cases. When using the graph neural network (DGCNN), we obtain results that are statistically equivalent to the original system also in 22 out of 36 cases. When using the shortest path graph kernel, we obtain results that are statistically equivalent to the original system in 19 out of 36 cases, when using the 4-Weisfeiler–Lehman graph kernel, we obtain results that are statistically equivalent in only 3 out of 36 cases. This analysis clearly shows that the 4-Weisfeiler–Lehman graph kernel does not cope well with the reduced graphs. One possible explanation for these rather bad results is that the similarity matrix obtained with the 4-Weisfeiler–Lehman kernel on the reduced graphs tends to shrink towards zero, as it will be explained in further detail in the following subsection.

To investigate the potential of the proposed graph reduction with respect to computation time, that is answering question **Q.2**, we present the runtimes of all classifiers[3] and reduction levels. In particular, Fig. 4 illustrates the average runtime, calculated over five runs for all datasets and classification methods.

In summary, our method demonstrates a clear improvement in runtime for all tested configurations. Already with the first reduction level (i.e., $\rho = 2$), the results indicate a substantial decrease in computation time. That is, we observe an average reduction of the total runtime of about a factor of two for all datasets and classifiers. As the reduction factor is increased, we further observe a consistent decrease in runtime. When the graphs are strongly reduced (with $\rho = 16$), we see a reduction in the runtime of approximately an order of magnitude on all datasets and classification methods. This improvement is even more pronounced for the large graphs stemming from the REDDIT datasets, where we see a reduction of two orders of magnitude.

In general, we observe that computationally demanding classifiers (i.e., graph edit distance and the shortest path kernel) benefit to a greater extent from using the reduced graphs. However, also the 4-Weisfeiler–Lehman kernel, which is computationally more efficient

[3] We omit the analysis of the runtime achieved using the GNN classifier, as the classification runtime is negligible once the GNN has been trained, and no clear difference appears in the runtime between the reference system and the systems that operate on the reduced graphs.

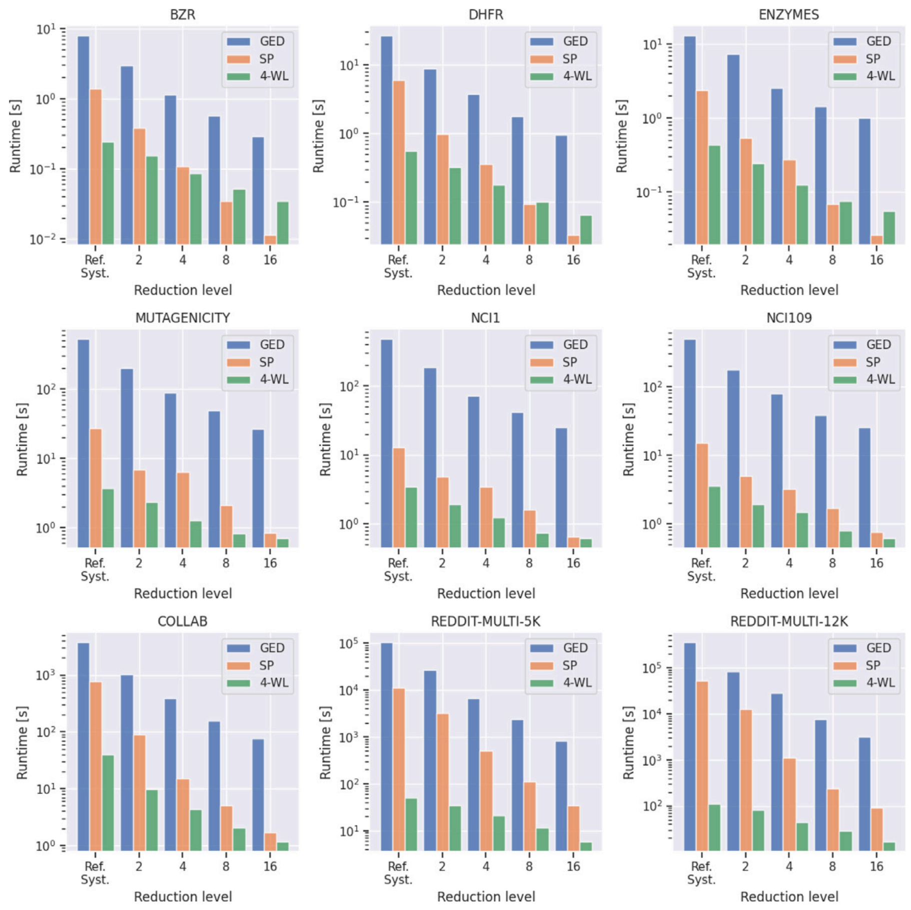

Fig. 4. Runtime in seconds on a logarithmic scale for all reduction levels and all datasets using all three classification systems, viz. a KNN using graph edit distance (GED), as well as the shortest path graph kernel (SP) and the 4-Weisfeiler–Lehman graph kernel (4-WL) in conjunction with an SVM.

than both graph edit distance and shortest path kernel, shows clear improvements through the use of reduced graphs. It is worth noting that these reductions in runtime are achieved while maintaining, or even improving, the classification accuracy (as we have seen before).

4.4. Similarity/dissimilarity quality measure

Graph edit distance and graph kernels are often used in conjunction with distance-based classifiers and SVMs, respectively. Thus, it is important to determine whether the similarities/dissimilarities obtained on reduced graphs are reliable. To validate this, we visually compare the similarities/dissimilarities obtained on the original graphs to those obtained in the reduced graph domains using scatter plots (see Fig. 5). Each point in these plots represents a similarity/dissimilarity in the original graph domain (x-axis) and the corresponding distance on the reduced graph (y-axis). The Pearson Correlation Coefficient (PCC) is also shown to indicate the linear correlation between the distances obtained on the original and reduced graphs. The black diagonal represents the one-to-one correspondence between reduced and original graphs. Due to the lack of space, we show results on two datasets only (BZR and NCI1). Note, however, that similar behavior is observed for the other datasets as well.

The graph edit distances obtained in the original graph domain and in the reduced ones appear to be quite correlated. In general, when the distance between two graphs is large in the original domain, it is also relatively large in the reduced domain. Conversely, when the distance is small in the original graph domain, it is also small in the reduced graph domain. Yet, the similarities obtained with both graph kernels are reduced when performed on the weakly reduced graphs (i.e., with $\rho = 2$) and shrink towards zero when the graphs are strongly reduced.

Overall, we observe that graph edit distances on the reduced graphs retain a coherent correlation and are still suitable for classification, while it may be more difficult to utilize the graph kernel similarities, as they tend to approach zero when the graphs are reduced in their sizes. This accounts for the lower classification accuracy obtained with the reduced graphs, as compared to the original graphs, for both graph kernels.

Upon deeper analysis, we observe that diverse graph similarities/dissimilarities in the original graph domain are mapped to the same value in the reduction space. The large number of equal similarities/dissimilarities between different pairs of graphs makes it difficult to discern any pattern in the data. Thus, it is no longer possible to extract trends from the intra-class similarities or dissimilarities (shown with red dots) and inter-class similarities/dissimilarities (shown with blue dots).

In Table 4, we show the PCCs between similarities/dissimilarities obtained in the original and reduced graph domain (we show the correlations between the original domain and two reduction levels only ($\rho = 2$ and $\rho = 16$)). As already seen in Fig. 5, the PCCs of graph edit distance remain stable when the graphs are strongly reduced. However, the PCCs tend to decrease when the reduction is increased and approaches zero when using both graph kernels. This trend is consistently visible across all labeled datasets. Yet, on the three unlabeled datasets (i.e., COLLAB, and both REDDIT datasets), we observe that the PCCs remain stable when the reduction is increased. This stability is actually also visible in the corresponding scatter plots. In Fig. 6, for instance, we show as an example a comparison of the Weisfeiler–Lehman kernel similarities in the original and in the reduced graph domain on the COLLAB dataset.

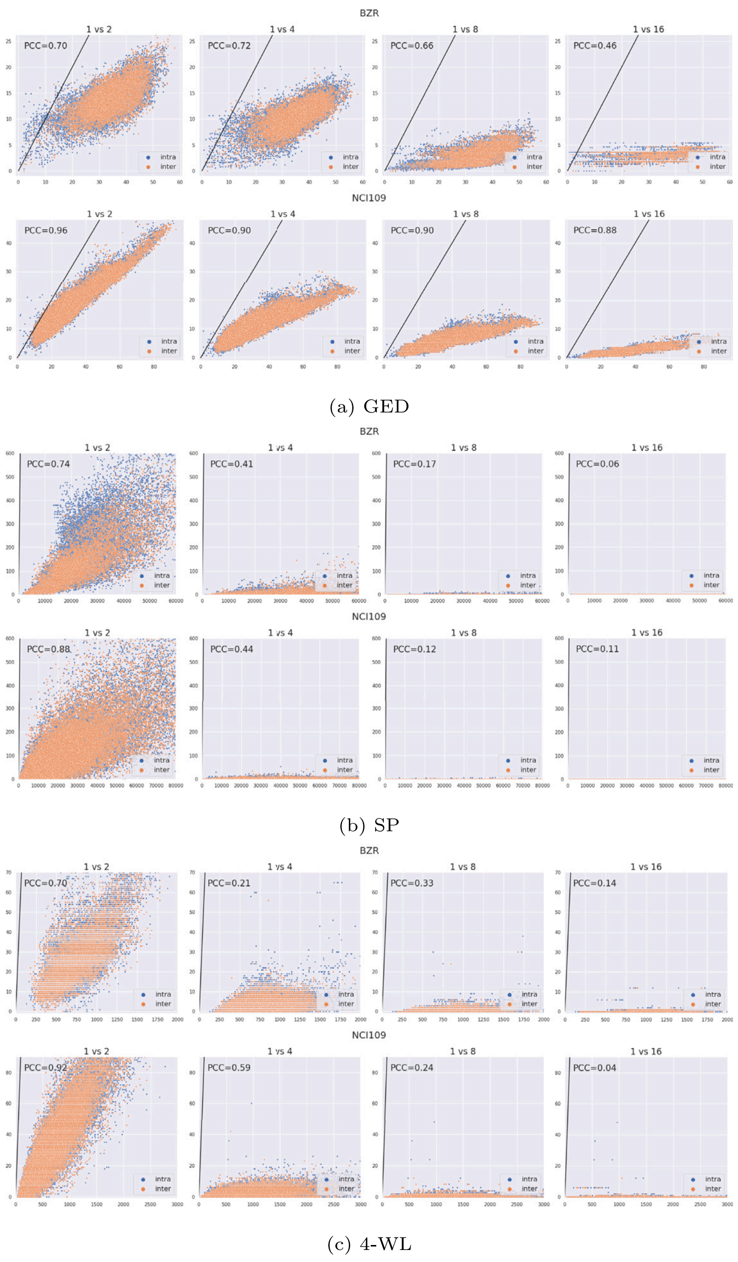

Fig. 5. Comparison of pairwise similarities/dissimilarities between graphs in the original and the reduced graph domains for both BZR and NCI109 datasets. Three classification systems are used to make the comparison including a KNN using graph edit distance (GED), as well as the shortest path graph kernel (SP), and the 4-Weisfeiler–Lehman graph kernel (4-WL) combined with an SVM.

5. Conclusions and future work

Graphs are powerful and flexible data structures and are thus widely used as representation formalism in pattern recognition and related fields. A prominent task in graph-based pattern recognition is graph classification. In order to tackle this particular task, many algorithms, such as the combination of graph edit distance with a KNN or the use of graph kernels with SVMs, have been proposed. However, it is generally admitted that these standard algorithms are computationally expensive, hindering their application on problems where large graphs are required. Throughout the years, various sub-optimal approximations have been proposed to speed up the computation time of those methods. Also, the use of size-reduced graphs has been proposed as a possible solution to the high complexity of algorithms expecting graphs as input.

In the present paper, we propose and research spectral graph clustering as the basis for a novel graph reduction framework. In particular, we use spectral graph clustering to first partition the nodes of a graph and then condense each partition of the nodes into supernodes. The benefit of this reduction process is that it can efficiently discover significant communities in the underlying graphs, thus offering accurate clusters for reducing the graphs to their most significant structures.

The proposed procedure to reduce the size of the graphs can be easily controlled by a parameter that defines the size of the resulting graphs (basically by the number of clusters to be found in the graph). The general goal of the proposed reduction framework is to speed up the computation of standard graph classification algorithms while maintaining satisfactory classification accuracy. In order to demonstrate the effectiveness of the proposed graph reduction technique, we compare the classification accuracy as well as the computation time on the original and reduced graphs with four different classifiers (graph edit distance with a KNN as well as shortest path, 4-Weisfeiler–Lehman kernel with SVMs, and graph neural network).

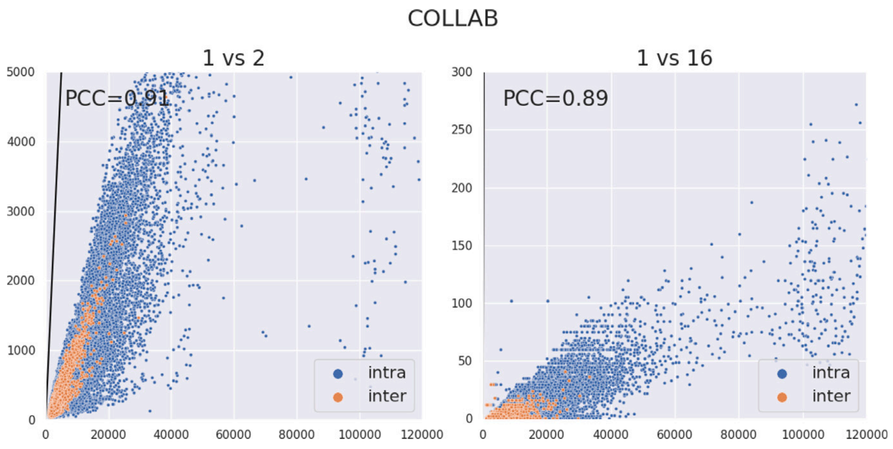

Fig. 6. Comparison of pairwise similarities, obtained with the 4-Weisfeiler–Lehman graph kernel, between the original graphs and their reduced counterparts (with $\rho = 2$ and $\rho = 16$) for the COLLAB dataset.

Table 4
Pearson correlation coefficient (PCC) between similarities (i.e., shortest path (SP) and the 4-Weisfeiler–Lehman graph kernel (4-WL))/dissimilarities (i.e., graph edit distance (GED)) obtained in the original and reduced graph domain on all datasets. We show the correlations between the original domain and two reduction levels ($\rho = 2$ and $\rho = 16$).

	GED		SP		4-WL	
Dataset	$\rho = 2$	$\rho = 16$	$\rho = 2$	$\rho = 16$	$\rho = 2$	$\rho = 16$
BZR	0.70	0.46	0.74	0.06	0.70	0.14
DHFR	0.67	0.71	0.69	0.09	0.76	0.11
ENZYMES	0.91	0.79	0.84	0.02	0.87	0.03
MUTAGENICITY	0.90	0.82	0.87	0.05	0.86	0.04
NCI1	0.97	0.90	0.89	0.07	0.92	0.03
NCI109	0.96	0.88	0.88	0.11	0.92	0.04
COLLAB	0.80	0.90	0.77	0.73	0.91	0.89
REDDIT-MULTI-5K	0.95	0.99	0.94	0.93	0.96	0.91
REDDIT-MULTI-12K	0.94	0.99	0.92	0.90	0.93	0.92

We conduct a comprehensive experimental evaluation on nine real-world graph datasets. Our experimental evaluation shows that while the classification accuracy decreases with the use of reduced graphs in general, the resulting accuracies are still comparable to those obtained with the original-sized graphs. In more detail, we are able to draw the following conclusions regarding the classification accuracy.

1. We find that the use of graph edit distance in conjunction with a KNN on reduced graphs achieves comparable classification accuracies to that of the original graphs in the majority of the cases.
2. We also show that the shortest path kernel works quite well on the weakly reduced graphs (i.e., $\rho \in \{2, 4\}$), as the classification accuracy remains statistically similar to those achieved with the reference system on the majority of datasets.
3. The reduced graphs have the least beneficial effect when used in conjunction with the 4-Weisfeiler–Lehman kernel, as the classification accuracy drops statistically significantly for almost all the reduction levels and datasets.
4. We observe that graph edit distances obtained in the original and the reduced graph domain remain in the same order of magnitude, while the similarities (obtained with the shortest path and the 4-Weisfeiler–Lehman graph kernel) shrink towards zero as the reduction level is increased.

The empirical evaluation also shows a significant decrease in computation time across all datasets and graph classifiers. That is, we observe a reduction of the runtime of about a factor of two when using weakly reduced graphs (i.e., $\rho = 2$) and at least one order of magnitude (and in some cases more than two orders of magnitude) when the graphs are strongly reduced (i.e., $\rho = 16$). These empirical results suggest that our approach may provide significant time savings compared to existing methods, which could be particularly useful in contexts where the time required for classification is a major limiting factor.

Regarding future research activities, we see several rewarding avenues to be pursued. Currently, the process of aggregating nodes of one cluster into supernodes consists of summing up the node features. As a first direction for future research, more advanced methods for merging nodes into supernodes could be explored. Specialized GNNs may be utilized in this process, with the goal of identifying optimal feature vectors for the supernodes. This could further improve the efficiency and effectiveness of the proposed graph reduction methods. A second line of research involves other fast and high-performing graph clustering algorithms – rather than spectral clustering – for graph partitioning [34]. A third line of research involves using the reduced graphs in a novel graph-based pattern recognition paradigm that addresses the permutation problem. The general idea is to fix the number of nodes for all graphs to a global value and arrange the nodes of each reduced graph based on their spectral information. This approach would resolve the issue of node permutation, potentially enabling faster graph comparisons in subsequent steps. Another possibility for future research involves the incorporation of variants in our general framework. In some preliminary experiments, for instance, we explore two modifications of the method described in the present paper. The first modification involves a continuous node hashing approach, which exchanges node features within a cluster through intra-cluster edges and iteratively updates each node's feature according to the received features. The second modification consists of applying alternative clustering methods on the spectral node embeddings (rather than k-means). With both modifications, no significant differences in classification accuracy can be observed when compared with the present configuration. It is worth noting, however, that these results are based on limited experimentation and further research may be necessary to fully evaluate the relative performance of these methods.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] F.B. Silva, R.d.O. Werneck, S. Goldenstein, S. Tabbone, R.d.S. Torres, Graph-based bag-of-words for classification, Pattern Recognit. 74 (2018) 266–285, http://dx.doi.org/10.1016/j.patcog.2017.09.018.

[2] Z. Gharaee, S. Kowshik, O. Stromann, M. Felsberg, Graph representation learning for road type classification, Pattern Recognit. 120 (2021) 108174, http://dx.doi.org/10.1016/j.patcog.2021.108174.

[3] D. Conte, P. Foggia, C. Sansone, M. Vento, Thirty Years Of Graph Matching In Pattern Recognition, Int. J. Pattern Recognit. Artif. Intell. 18 (3) (2004) 265–298, http://dx.doi.org/10.1142/S0218001404003228.

[4] P. Foggia, G. Percannella, M. Vento, Graph Matching and Learning in Pattern Recognition in the Last 10 Years, Int. J. Pattern Recognit. Artif. Intell. 28 (1) (2014) http://dx.doi.org/10.1142/S0218001414500013.

[5] S. Wan, S. Pan, S. Zhong, J. Yang, J. Yang, Y. Zhan, C. Gong, Multi-level graph learning network for hyperspectral image classification, Pattern Recognit. 129 (2022) 108705, http://dx.doi.org/10.1016/j.patcog.2022.108705.

[6] X. Li, B. Wu, J. Song, L. Gao, P. Zeng, C. Gan, Text-instance graph: Exploring the relational semantics for text-based visual question answering, Pattern Recognit. 124 (2022) 108455, http://dx.doi.org/10.1016/j.patcog.2021.108455.

[7] A. Mrzic, P. Meysman, W. Bittremieux, P. Moris, B. Cule, B. Goethals, K. Laukens, Grasping frequent subgraph mining for bioinformatics applications, BioData Min. 11 (1) (2018) 20:1–20:24, http://dx.doi.org/10.1186/s13040-018-0181-9.

[8] H. Bunke, G. Allermann, Inexact graph matching for structural pattern recognition, Pattern Recognit. Lett. 1 (4) (1983) 245–253, http://dx.doi.org/10.1016/0167-8655(83)90033-8.

[9] N.M. Kriege, F.D. Johansson, C. Morris, A survey on graph kernels, Appl. Netw. Sci. 5 (1) (2020) 6, http://dx.doi.org/10.1007/s41109-019-0195-3.

[10] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A Comprehensive Survey on Graph Neural Networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2021) 4–24, http://dx.doi.org/10.1109/TNNLS.2020.2978386.

[11] A. Fischer, C.Y. Suen, V. Frinken, K. Riesen, H. Bunke, Approximation of graph edit distance based on Hausdorff matching, Pattern Recognit. 48 (2) (2015) 331–343, http://dx.doi.org/10.1016/j.patcog.2014.07.015.

[12] U. Kang, H. Tong, J. Sun, Fast Random Walk Graph Kernel, in: Proceedings of the Twelfth SIAM International Conference on Data Mining, Anaheim, California, USA, April 26-28, 2012, SIAM / Omni Press, 2012, pp. 828–838, http://dx.doi.org/10.1137/1.9781611972825.71.

[13] Y. Liu, T. Safavi, A. Dighe, D. Koutra, Graph Summarization Methods and Applications: A Survey, ACM Comput. Surv. 51 (3) (2018) 62:1–62:34, http://dx.doi.org/10.1145/3186727.

[14] H. Qiu, E.R. Hancock, Graph matching and clustering using spectral partitions, Pattern Recognit. 39 (1) (2006) 22–34, http://dx.doi.org/10.1016/j.patcog.2005.06.014.

[15] A. Gillioz, K. Riesen, Speeding up Graph Matching by Means of Systematic Graph Reductions Using Centrality Measures, in: 2022 12th International Conference on Pattern Recognition Systems, ICPRS, 2022, pp. 1–7, http://dx.doi.org/10.1109/ICPRS54038.2022.9854062.

[16] Y. Jin, A. Loukas, J.F. JáJá, Graph Coarsening with Preserved Spectral Properties, in: S. Chiappa, R. Calandra (Eds.), The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy], in: Proceedings of Machine Learning Research, 108, PMLR, 2020, pp. 4452–4462, URL http://proceedings.mlr.press/v108/jin20a.html.

[17] P. Maergner, N.R. Howe, K. Riesen, R. Ingold, A. Fischer, Offline Signature Verification Via Structural Methods: Graph Edit Distance and Inkball Models, in: 16th International Conference on Frontiers in Handwriting Recognition, ICFHR 2018, Niagara Falls, NY, USA, August 5-8, 2018, IEEE Computer Society, 2018, pp. 163–168, http://dx.doi.org/10.1109/ICFHR-2018.2018.00037.

[18] C. Garcia-Hernandez, A. Fernández, F. Serratosa, Ligand-Based Virtual Screening Using Graph Edit Distance as Molecular Similarity Measure, J. Chem. Inf. Model. 59 (4) (2019) 1410–1421, http://dx.doi.org/10.1021/acs.jcim.8b00820.

[19] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.

[20] K. Riesen, H. Bunke, Approximate graph edit distance computation by means of bipartite graph matching, Image Vis. Comput. 27 (7) (2009) 950–959, http://dx.doi.org/10.1016/j.imavis.2008.04.004.

[21] T. Horváth, T. Gärtner, S. Wrobel, Cyclic pattern kernels for predictive graph mining, in: W. Kim, R. Kohavi, J. Gehrke, W. DuMouchel (Eds.), Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004, ACM, 2004, pp. 158–167, http://dx.doi.org/10.1145/1014052.1014072.

[22] N.M. Kriege, P. Mutzel, Subgraph Matching Kernels for Attributed Graphs, in: Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012, icml.cc / Omni Press, 2012, pp. 1–20, URL http://icml.cc/2012/papers/542.pdf.

[23] K.M. Borgwardt, H.-P. Kriegel, Shortest-Path Kernels on Graphs, in: Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA, IEEE Computer Society, 2005, pp. 74–81, http://dx.doi.org/10.1109/ICDM.2005.132.

[24] N. Shervashidze, P. Schweitzer, E.J.v. Leeuwen, K. Mehlhorn, K.M. Borgwardt, Weisfeiler-Lehman Graph Kernels, J. Mach. Learn. Res. 12 (2011) 2539–2561, http://dx.doi.org/10.5555/1953048.2078187, URL https://dl.acm.org/doi/10.5555/1953048.2078187.

[25] M. Zhang, Z. Cui, M. Neumann, Y. Chen, An End-to-End Deep Learning Architecture for Graph Classification, in: S.A. McIlraith, K.Q. Weinberger (Eds.), Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, AAAI Press, 2018, pp. 4438–4445.

[26] J. Chen, Y. Saad, Z. Zhang, Graph coarsening: From scientific computing to machine learning, CoRR (2021) arXiv:2106.11863, URL https://arxiv.org/abs/2106.11863, arXiv:2106.11863.

[27] S.E. Schaeffer, Graph clustering, Comput. Sci. Rev. 1 (1) (2007) 27–64, http://dx.doi.org/10.1016/j.cosrev.2007.05.001.

[28] U.v. Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (4) (2007) 395–416, http://dx.doi.org/10.1007/s11222-007-9033-z.

[29] J. Lurie, Review of Spectral Graph Theory: by Fan R.K. Chung, SIGACT News 30 (2) (1999) 14–16, http://dx.doi.org/10.1145/568547.568553.

[30] N. Shi, X. Liu, Y. Guan, Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm, in: Third International Symposium on Intelligent Information Technology and Security Informatics, IITSI 2010, Jinggangshan, China, April 2-4, 2010, IEEE Computer Society, 2010, pp. 63–67, http://dx.doi.org/10.1109/IITSI.2010.74.

[31] A. Merchant, M. Mathioudakis, Y. Wang, Graph Summarization via Node Grouping: A Spectral Algorithm, in: T.-S. Chua, H.W. Lauw, L. Si, E. Terzi, P. Tsaparas (Eds.), Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023, ACM, 2023, pp. 742–750, http://dx.doi.org/10.1145/3539597.3570441.

[32] C. Morris, N.M. Kriege, F. Bause, K. Kersting, P. Mutzel, M. Neumann, TUDataset: A collection of benchmark datasets for learning with graphs, CoRR (2020) arXiv:2007.08663, URL https://arxiv.org/abs/2007.08663, arXiv:2007.08663.

[33] K.M. Borgwardt, M.E. Ghisu, F. Llinares-López, L. O'Bray, B. Rieck, Graph Kernels: State-of-the-Art and Future Challenges, Found. Trends Mach. Learn. 13 (5–6) (2020) http://dx.doi.org/10.1561/2200000076.

[34] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, C. Schulz, Recent Advances in Graph Partitioning, in: L. Kliemann, P. Sanders (Eds.), Algorithm Engineering - Selected Results and Surveys, in: Lecture Notes in Computer Science, vol. 9220, 2016, pp. 117–158, http://dx.doi.org/10.1007/978-3-319-49487-6_4.

Anthony Gillioz has received a M.Sc. in Computer Science from the University of Bern in 2020. He is currently a Ph.D. student in the Pattern Recognition Group at the University of Bern. His research interests are pattern recognition with a special focus on graph matching and graph reduction.

Kaspar Riesen received his M.Sc. and Ph.D. degrees in Computer Science from the University of Bern in 2006 and 2009, respectively. His Ph.D. thesis received the Alumni price for an outstanding work at the Institute of Computer Science and Applied Mathematics of the University of Bern. His research interests cover the fields of artificial intelligence, pattern recognition, machine learning and data mining. In particular, he is working on the development of novel algorithms for solving graph matching problems in various domains of intelligent information processing.