# Two-Step Graph Classification on the Basis of Hierarchical Graphs

Anthony Gillioz[a] and Kaspar Riesen[b]
*Institute of Computer Science, University of Bern, Switzerland*

Abstract:     A common method to solve the non-trivial task of classifying general graphs is to employ graph matching in conjunction with a distance- or similarity-based classifier. Unfortunately, optimal graph matching has a high computational complexity hindering its application on large graphs. In order to make matchings also feasible for larger graphs, it has been proposed to work on size-reduced graphs rather than on their original counterparts. In the present paper, we propose a novel method that is based on this idea to further reduce the processing time. In particular, we change the standard classification scheme into a two-step classification method. In the first step, we start with strongly reduced versions of the graphs – having a manageable amount of nodes – in order to prune as many graphs as possible. The second step – the actual classification – is then performed on the remaining graphs only (in their original size). We conduct experimental evaluations on five datasets to research the benefits and limitations of this novel two-step graph classification method. The main finding is that we can substantially speed up the graph matching while preserving satisfying classification accuracy.

## 1 INTRODUCTION

In its most general formulation, a graph is a finite set of basic entities (modeled by means of nodes), together with a set of relations that might exist between pairs of these basic entities (modeled by means of edges). Hence, the power of this data structure relies not only on the properties of the individual nodes but also – and maybe even more importantly – on the relations between the nodes. Graphs are actually universal representational systems and provide a mathematical framework that can be used to study and learn from diverse real-world structures (Conte et al., 2004; Foggia et al., 2014). That is, graph structures can be exploited to represent data in a broad variety of applications, such as *Bioinformatics* (de Ridder et al., 2013), *Social Media Analysis* (Pitas, 2016), and *Network Analysis* (Newman, 2010), to name just three prominent examples.

Graph matching is one of the most fundamental problems in graph-based pattern recognition. It aims at identifying shared substructures in two graphs and evaluating the degree of similarity or dissimilarity between them. Considering the influence of graph matching in miscellaneous pattern recognition applications, it is crucial to develop and research efficient procedures for this task.

A standard method to perform graph matching is *Graph Edit Distance* (GED) (Bunke and Allermann, 1983; Sanfeliu and Fu, 1983). GED quantifies a graph dissimilarity on the basis of the minimum amount of modification required to transform a source graph into a target graph. The authors of (Garey and Johnson, 1979; Abdulrahim, 1998) show that the computation of GED is, in its general formulation, an $\mathcal{NP}$-complete problem. Therefore, matching large graphs by means of GED is often not feasible or at least computationally demanding. This in turn prevents the application of GED in both real-time data analysis and large-scale pattern recognition tasks.

Over the last decade, however, researchers have proposed different approximations to reduce the runtime of GED (Riesen and Bunke, 2009; Blumenthal et al., 2021). These approximations substantially reduce the computational complexity of GED. Yet, even with these polynomial time algorithms, the application of GED remains problematic – in particular for large graphs with many nodes and/or many edges. Another appealing idea to reduce the computation time of graph-based pattern recognition is to work with reduced versions of the graphs rather than their original counterparts. This approach can roughly be split into two categories, viz. *Hierarchical Graph Representations* (Brun et al., 2020) and *Graph Summarizations* (Liu et al., 2018).

[a] https://orcid.org/0000-0001-7352-3708
[b] https://orcid.org/0000-0002-9145-3157

The basic idea of hierarchical graph representations is that the original graphs are more and more compressed as the number of levels increases. Thereby, at the highest level, only an abstraction of the graphs remains and eases the manipulation and the analysis of complex structural data. Graph summarization, on the other hand, is mainly used to discover complex patterns in graphs. This method consists of determining each node's importance or pertinence in the graph structure, and ultimately discarding the nodes with the lowest importance in a sampling strategy.

In the present paper, we propose and research an extension of a recent graph matching framework that makes use of a systematical reduction strategy (Gillioz and Riesen, 2022). The major drawback of this (as well as any other) reduction method is the loss of information when nodes are discarded. Indeed, in (Gillioz and Riesen, 2022) it is shown that strongly reduced graphs result in significant deteriorations of the general classification accuracy. We propose two distinct modifications to the classification procedure to mitigate this effect. The two strategies are embedded in a two-step distance-based classifier where the first step relies on the strongly reduced graphs. The goal of this first step is to get a rough estimate of the pairwise distances as fast as possible and make large parts of the graphs obsolete for the second step. In the second step, we perform both matching and classification only on those original graphs which have actually passed the first step.

The remainder of the present paper is structured as follows. In Section 2, we briefly review the previous work done in the field of graph matching, along with a formal definition of GED. In Section 3, we explain the details of our graph pruning strategy as well as how it can be embedded in a two-step classification scheme. We present the setup of the experiments and discuss the main results in Section 4. In the last section, we conclude the paper and suggest possible extensions for future work.

## 2 RELATED WORK

In this section, we briefly describe and review three related areas to our work, namely the graph matching methods in general, GED (the graph matching method actually employed in this paper), and frameworks relying on reduced graphs.

### 2.1 Graph Matching

Graph matching is the process of finding a mapping between the nodes and edges of two graphs satisfying some constraints in order to find common substructures in both graphs. Graph matching algorithms are commonly categorized into two groups, namely, *exact graph matching* and *inexact graph matching*.

Exact matching methods try to assess whether or not two graphs are strictly identical or share – partially – common subparts. On the other hand, inexact graph matching relaxes the matching constraints and allows a more subtle comparison between completely non-identical graphs. Hence, the latter paradigm is typically more appropriate for real-world problems, where the transcription of the graphs from the observed patterns may contain noise and errors.

Through the years various methods have been proposed to perform inexact graph matching. We briefly review three prominent families of graph matching techniques. For a more thorough review refer to (Conte et al., 2004; Foggia et al., 2014).

*Spectral methods* (Qiu and Hancock, 2006; Caelli and Kosinov, 2004) form an important branch of inexact graph matching techniques. These methods basically use spectral properties of the underlying graph matrices (i.e., the eigenvectors and eigenvalues of the Adjacency and/or Laplacian matrix) to construct a vector space onto which the nodes of the graphs or the complete graphs are projected. This vector space embedding is then used to find potential matches between the underlying graphs.

*Graph kernels* (Kriege et al., 2020) are a second prominent family of graph matching methods. Graph Kernels can be seen as an implicit embedding of the graphs into a feature space. In particular, kernel functions on graphs can be interpreted as an inner product in a feature space and can thus be considered as a similarity metric for graphs. Diverse graph kernels have been proposed in the literature. Many of them rely on certain substructures that can be found in both graphs. For instance, random walk kernels (Kashima et al., 2003) are defined on the basis of similar walks – sequences of nodes with allowed repetition – in two graphs.

*Neural Networks* (Wu et al., 2021) forms the most recent category of graph matching methods. The basic idea is that a neural network learns from the input data how to map graphs toward a compact vector representation and use that representation to compute similarity. In (Riba et al., 2021), for instance, a graph similarity metric is directly computed by means of a graph neural network.

## 2.2 Graph Edit Distance

*Graph Edit Distance* (GED) (Bunke and Allermann, 1983; Sanfeliu and Fu, 1983) is a more traditional graph matching technique. However, due to its great adaptability, several more recent research projects are concerned with this particular dissimilarity measure (Cortés and Serratosa, 2015; Riesen et al., 2016; Blumenthal et al., 2021).

When comparing two graphs $g_1$ and $g_2$, GED computes the minimum amount of *edit operations* necessary to convert $g_1$ to $g_2$. In its original definition, only three edit operations (namely *insertions*, *deletions*, and *substitutions*) are allowed on both nodes and edges. Employing those edit operations, GED computes an *edit path* $\lambda(g_1, g_2)$ between $g_1$ and $g_2$ as a set $\{e_1, ..., e_k\}$ of $k$ edit operations $e_i$ that completely transform $g_1$ into $g_2$. A cost function $c(\cdot)$ is commonly defined to weigh the strength of each edit operation, and GED finds the edit path that minimizes the overall cost.

The combinatorial nature of GED makes it difficult to be applied on large graphs, and thus diverse GED approximations have been proposed. In the present paper, we make use of (Riesen and Bunke, 2009) as a basic graph matching framework.

## 2.3 Hierarchical Graph Matching

The overarching aim of the present paper is to employ GED approximation (Riesen and Bunke, 2009) in conjunction with reduced graphs (Gillioz and Riesen, 2022) in order to define a two-step classification procedure. Hence, closely related to the approach proposed in the present paper is hierarchical graph representations. In the context of image segmentation, hierarchical graph representations consist of computing multilevel representations of images, where each of the levels represents different semantic properties like texture or color (Haxhimusa and Kropatsch, 2004). This representation can then be used, for instance, to find boundaries between regions in an image.

In a pattern recognition context, the authors of (Dutta et al., 2020) propose a hierarchical graphlet embedding. In particular, graphs are embedded into a feature space and a graph clustering method is then employed to find the nodes to combine in order to create a graph hierarchy. Thereby, the resulting representation encodes the abstract information and maintains the relationship with the original graph.

In (Riba et al., 2020), a coarse-to-fine graph matching is proposed which is also similar in spirit to our approach. The idea is to represent graphs as pyramids where each level contains different types of information. The subgraph representations at the lower levels carry coarse information (i.e., the global structure) and those at the higher levels contain more fine information (i.e., the detailed structure). The graph matching is then performed in a coarse-to-fine fashion to prune graphs during the low-level comparisons.

# 3 TWO-STEP GRAPH CLASSIFICATION

One of the standard processes in graph classification is based on computing graph dissimilarities in conjunction with a distance-based classifier (e.g. a *k-Nearest Neighbor (K-NN)* classifier). One of the main contributions of the present paper is to modify this standard classification process into a two-step classification scheme. To this end, we make use of a method that systematically reduces the size of graphs (introduced in (Gillioz and Riesen, 2022)).

More formally, before the classification starts, the underlying graphs are preprocessed as follows. First, we measure the node centrality in order to quantify each node's importance on the overall graph structure. Several node centrality measures could be employed for this task (e.g., PageRank (Brin and Page, 1998) and Betweenness (Freeman, 1977)). For the sake of conciseness, we only use PageRank in the present paper. Eventually, the computed PageRank importance is used to discard the nodes with the least impact on the graph structure.

The relative amount of nodes, that actually remain in the graphs after the reduction process, is controlled via a user-defined parameter. In the present paper, we propose to reduce the size of the graphs by a wide margin and keep only 20% of the total nodes. Hence, for each pattern to recognize we now have two different representations, namely the original graph $g$ and the reduced graph $g'$ (containing the most important nodes only).

In the proposed framework the size-reduced and original graphs are now employed in two separate steps. The basic idea of the first step is to conduct all matchings on the strongly reduced graphs. Then, in the second step, we conduct as few matchings as possible on the original graphs. Hence, our method can also be interpreted as a coarse-to-fine approach that starts on rather coarse representations and eventually continues on the more fine-grained graph representations – similar to the approach presented in (Riba et al., 2020).

However, our approach differs in three major parts with (Riba et al., 2020). First, we use node centrality measures (rather than node clustering) for graph re-

ductions. Second, we employ only two levels of hierarchy (since we primarily aim at speeding up the classification process). Third, we propose two different strategies for the crucial decision on how to proceed after step 1. The first strategy is to select appropriate candidates in the reduced graph space for further processing. The second strategy is to accept the classification obtained in the reduced graph space if the classifier is confident enough. Both strategies are described in greater detail in the next two subsections.

Regardless of the strategy actually employed, the proposed two-step classification scheme allows us to trade off between run-time and classification accuracy. That is, pruning numerous graphs in the first step allows a faster computation time but possibly deteriorates the classifier's accuracy. Contrariwise, keeping many graphs for the second step possibly allows better classification accuracy but might increase the computation time.

## 3.1 Candidate Selection Strategy

The aim of this strategy is as follows. Based on the matching information obtained on the reduced graphs, we keep the nearest training graphs for each test graph. We term these nearest neighbors – actually used in step 2 – as *candidates*. We define a parameter $\omega$ that defines the relative amount of training graphs that are selected as candidates for step 2. With $\omega = 0.1$, for instance, we select 10% of the nearest training graphs as candidates for each test graph.

The second step of the classification process is based on the original graphs. That is, the test graphs are matched with the original graphs that correspond to the selected candidates from step 1. Intuitively, step 1 acts as a filter that pre-selects plausible candidates from the training set for further and more precise investigations during step 2.

The complete process – termed *CandSel($\omega$)* from now on – is formalized in Alg. 1. The algorithm takes as parameters a test graph in two representations $t$ and $t'$ (the original and the size-reduced graph), as well as $N$ original training graphs $G = \{g_{(1)}, \ldots, g_{(N)}\}$ and their corresponding reduced versions $G' = \{g'_{(1)}, \ldots, g'_{(N)}\}$. Note that we use a K-NN classifier on line 9 of the algorithm. However, any other distance or similarity-based classification could be employed as well.

## 3.2 Early Classification Strategy

In this second strategy, we first apply a K-NN classification on the reduced graphs. All graphs for which the classification is — more or less — confident, are

---

**Algorithm 1:** CandSel($\omega$).

1 **STEP 1**
2 $n = \lfloor \omega \cdot N \rfloor$ `// number of candidates`
3 **for** *all $g' \in G'$* **do**
4 $\quad$ compute GED $d(t', g')$
5 **end**
6 $C' = \{g'_{(1)}, \ldots, g'_{(n)}\}$ `// n graphs in G'`
$\quad$ `with smallest distances to t'`
7 $C = \{g_{(1)}, \ldots, g_{(n)}\}$ `// corresponding`
$\quad$ `graphs in G`
8 **STEP 2**
9 Classify $t$ with the aid of the selected candidates $C = \{g_{(1)}, \ldots, g_{(n)}\}$

---

directly classified without further processing in step 2. Formally, we measure the confidence of each decision by means of the number of neighbors $k'$ among the $k$-nearest neighbors that stem from the same class (with $k' \leq k$). If $k'$ is greater than, or equal to, a certain threshold $\delta$ we consider the class prediction as confident enough to be accepted.

Considering a 5-NN, for instance, and we set $\delta = 4$, then at least four of the nearest neighbors have to stem from the same class so that the graph is classified in the first step[1]. Note that this strategy directly depends on the classifier actually employed. That is, when this strategy is used in conjunction with another distance-based classifier, another metric for the confidence has to be defined first.

For any reduced graph $t'$ that is not classified in the first step (due to too-low classification confidence), the classification is conducted on its original counterpart $t$. That is, we have to compute all distances from $t$ to all original training graphs to apply a final K-NN classification. We formalize the proposed procedure – termed *EarlyClass($\delta$)* from now on – in Alg. 2.

## 4 EXPERIMENTAL EVALUATION

## 4.1 Datasets

We evaluate our classification procedure on two datasets from the IAM graph repository (Riesen and Bunke, 2008)[2] (AIDS, Mutagenicity) and on three datasets from TUDataset (Morris et al., 2020)[3] (NCI1, Proteins, IMDB-Binary). Table 1 shows some graph properties such as the number of graphs, the number

---

[1]Note that for a binary classification task with a 5-NN, it makes no sense to set $\delta \leq 3$ because all samples would be immediately classified in the first step.

[2]www.iam.unibe.ch/fki/databases/iam-graph-database
[3]http://www.graphlearning.io/

Algorithm 2: EarlyClass($\delta$).

---

**1  STEP 1**
**2  for** *all $g' \in G'$* **do**
**3**  │  compute GED $d(t', g')$
**4  end**
**5**  $C' = \{g'_{(1)}, \dots, g'_{(k)}\}$ // $k$ graphs in $G'$
         with smallest distances to $t'$
**6  if** *number of graphs of majority class in*
    $C' \geq \delta$ **then**
**7**  │  Accept classification of $t'$
**8  else**
**9**  │  **STEP 2**
**10**  │  Classify $t$ with the aid of the original
      training graphs $G = \{g_{(1)}, \dots, g_{(N)}\}$
**11  end**

---

of classes, and the average number of nodes and edges per dataset.

The AIDS, Mutagenicity, and NCI1 graph datasets represent molecules stemming from two classes. The graphs in the AIDS dataset represent chemical compounds that potentially have an effect against HIV, the graphs in the Mutagenicity dataset represent molecules that may have mutation properties, and the graphs in the NCI1 dataset represent molecules that are able to diminish the expansion of tumorous cells. The graphs in the Proteins dataset correspond to protein structure elements stemming from two classes. Finally, the IMDB-Binary dataset is a social network dataset, where nodes represent actors/actresses and an edge connects two actors/actresses if they appear in the same movie. The classification task on this dataset consists of identifying the genre of the graph representing a certain movie.

## 4.2 Results

The purpose of our experimental evaluation is twofold. First, we aim at investigating the reduction of the computation time actually possible with the proposed framework. Second, we want to evaluate whether or not the classification accuracy can be maintained when applying the two strategies in our two-step classification procedure. Hence, we compare both the run-time and the classification accuracy obtained by our novel methods with a standard classification method, viz. a K-NN classifier that operates on the original graphs only.

### 4.2.1 Candidate Selection

Table 2 shows both the classification accuracy and the run-time for the reference system (i.e., a standard K-NN classifier operating in the original graph

space) and the novel method CandSel($\omega$) with $\omega \in \{0.20, 0.10, 0.05\}$.

On all datasets, we observe that the classification accuracy generally decreases as the parameter $\omega$ is reduced (as expected). Simultaneously, we observe substantial reductions in the run-times with increasing values of $\omega$. The run-time of our novel system with $\omega = 0.10$, for instance, is reduced to about 50% of the run-times of the reference system on all datasets.

On the three datasets AIDS, NCI1 and IMDB-Binary, we observe that the classification accuracy of our novel approach is in general worse than the accuracy obtained by the reference system. We also see that most of these deteriorations are statistically significant (with the exception of the result obtained on the AIDS dataset with $\omega = 0.10$). However, we can also report that at least on AIDS and NCI1 the classification accuracies obtained by means of our novel system are in a fairly similar range as those of the reference system. Moreover, on the other two datasets, viz. Mutagenicity and Proteins, not a single statistically significant deterioration compared to the reference system can be seen – on the contrary, we observe one statistically significant improvement (on the Mutagenicity dataset with $\omega = 0.20$).

Overall these classification results of CandSel($\omega$) are convincing and encouraging especially when considering the substantial decrease in the run-times of our framework compared with the reference system.

### 4.2.2 Early Classification

Table 3 shows the classification accuracy and the run-time achieved with the reference system and our novel method EarlyClass($\delta$) that uses a 5-NN classifier in conjunction with two thresholds $\delta \in \{4, 5\}$[4].

Likewise to the method CandSel($\omega$), the accuracies achieved with EarlyClass($\delta$) are – more or less – comparable to the results of the reference system. For instance, with $\delta = 5$, the classification accuracy remains statistically equivalent to the results obtained with the reference system on all datasets. When the threshold is reduced to $\delta = 4$, the classification accuracy achieved on Mutagenicity, NCI1, and IMDB-Binary is statistically worse than the accuracy of the reference system. Note, however, that our novel system performs in a fairly similar range as the reference system in all cases. Moreover, with $\delta = 4$ we even

---

[4]During the validation of the meta-parameters on the AIDS dataset, we observe that in step 2 the graphs stem from one class only. Hence, rather than performing a second matching on these graphs, we decide to directly classify the few graphs for which the second step is actually necessary. Note that this applies on the AIDS dataset only.

Table 1: Properties of the graph datasets. We show the size of the graph datasets ($|G|$) with the number of graphs in the training, validation, and test set ($|G_{tr}|$, $|G_{va}|$, $|G_{te}|$), the number of classes ($|\Omega|$) and the average number of nodes and edges per graph ($\emptyset|V|$, $\emptyset|E|$).

| Dataset | $|G|$ ($|G_{tr}|$, $|G_{va}|$, $|G_{te}|$) | $|\Omega|$ | $\emptyset|V|$ | $\emptyset|E|$ |
|---|---|---|---|---|
| **AIDS** | 2,000 (250, 250, 1,500) | 2 | 9.5 | 10.0 |
| **Mutagenicity** | 4,337 (1,500, 500, 2,337) | 2 | 30.3 | 30.8 |
| **NCI1** | 4,110 (1,500, 500, 2,110) | 2 | 29.9 | 32.3 |
| **Proteins** | 1,113 (660, 220, 223) | 2 | 39.1 | 72.8 |
| **IMDB-Binary** | 1,000 (600, 200, 200) | 2 | 19.8 | 96.5 |

Table 2: Classification accuracy and run-time (in seconds) obtained with a K-NN classifier on the original graphs (Reference System) and the results obtained with our novel method CandSel($\omega$) with $\omega \in \{0.20, 0.10, 0.05\}$. (○/●: statistically significantly better/worse than the reference system).

| | | | Ref. System | CandSel($\omega$) | | |
|---|---|---|---|---|---|---|
| | | | | $\omega = 0.20$ | $\omega = 0.10$ | $\omega = 0.05$ |
| **Dataset** | **AIDS** | Acc [%] | 98.53 | 97.53● | 98.80 | 96.53● |
| | | Time [s] | 16.93 | 7.56 | 5.62 | 4.99 |
| | **Mutagenicity** | Acc [%] | 71.33 | 72.95○ | 72.02 | 70.60 |
| | | Time [s] | 63.27 | 48.70 | 41.99 | 36.90 |
| | **NCI1** | Acc [%] | 70.33 | 69.24● | 68.96● | 68.15● |
| | | Time [s] | 56.32 | 43.10 | 34.05 | 29.38 |
| | **Proteins** | Acc [%] | 73.82 | 70.82 | 70.82 | 71.24 |
| | | Time [s] | 7.09 | 3.88 | 2.88 | 1.97 |
| | **IMDB-Binary** | Acc [%] | 66.00 | 59.50● | 58.50● | 55.50● |
| | | Time [s] | 4.32 | 3.67 | 2.03 | 2.26 |

observe a statistically significant improvement in the classification accuracy on the Proteins dataset.

Regarding the run-times we also observe substantial speed-ups of our method compared to the reference system. Using threshold $\delta = 4$, for instance, we observe a substantial decrease of the run-time of about 40% on the datasets Mutagenicity, NCI1, and Proteins. On the AIDS dataset, the run-time of our method is even about five times faster than the run-times of the reference system (mainly due to the omitted second step on this dataset). Interestingly, we observe an increase in the run-time of our method on the IMDB-Binary dataset. This result has encouraged us to do some further research and investigations on the behavior of the early classification strategy.

In Table 4 we show the number of actually classified graphs with EarlyClass($\delta$) (we focus on $\delta = 4$) and the corresponding classification accuracy in both steps (step 1 and step 2).

We observe that in general, a large amount of the graphs are classified during the first step. For instance, on the AIDS dataset about 98% of the test graphs are classified during the first step. For Muta-

genicity, NCI1, and Proteins, about 50% of the graphs are instantly classified without any further computation. For IMDB-Binary, we observe that the majority of the graphs are classified during the second step of the algorithm, which might explain why the run-time for this dataset increases. That is, on this particular dataset, the computational overhead of our novel two-step classification method cannot be compensated by many early classifications.

When comparing the classification accuracies achieved in step 1 and step 2 separately, we observe that the classification accuracy decreases in general in the second step (see, for instance, on AIDS, Mutagenicity and Proteins). A possible explanation might be that only the "difficult" graphs remain to be classified during the second step.

Table 3: Classification accuracy and run-time (in seconds) obtained with a K-NN classifier on the original graphs (Reference System) and the results obtained with our novel method EarlyClass($\delta$) with $\delta \in \{5,4\}$. ($\circ$/$\bullet$: statistically significantly better/worse than the reference system).

| | | | Ref. System | EarlyClass($\delta$) | |
| | | | | $\delta = 5$ | $\delta = 4$ |
|---|---|---|---|---|---|
| **Dataset** | **AIDS** | Acc [%] | 98.93 | 98.60 | 98.73 |
| | | Time [s] | 16.93 | 3.48 | 3.31 |
| | **Mutagenicity** | Acc [%] | 71.63 | 71.07 | 68.12$\bullet$ |
| | | Time [s] | 62.83 | 60.07 | 39.00 |
| | **NCI1** | Acc [%] | 70.52 | 70.56 | 68.15$\bullet$ |
| | | Time [s] | 56.32 | 54.22 | 33.78 |
| | **Proteins** | Acc [%] | 73.82 | 75.1 | 75.54$\circ$ |
| | | Time [s] | 7.09 | 7.32 | 4.4 |
| | **IMDB-Binary** | Acc [%] | 66.00 | 64.00 | 62.00$\bullet$ |
| | | Time [s] | 4.32 | 8.14 | 5.63 |

Table 4: Statistics drawn from the method EarlyClass($\delta$) with $\delta = 4$. $|G_{te}|$ is the number of test graphs per dataset. #classified and Acc refer to the number of graphs classified and the classification accuracy obtained, respectively (during that step). Final Acc is the global classification accuracy obtained at the end of the classification process.

| | | $|G_{te}|$ | Step 1 | | Step 2 | | Final Acc [%] |
| | | | #classified | Acc [%] | #classified | Acc [%] | |
|---|---|---|---|---|---|---|---|
| **Dataset** | **AIDS** | 1,500 | 1,479 | 98.78 | 21 | 95.24 | 98.73 |
| | **Mutagenicity** | 2,337 | 1,182 | 70.73 | 1,155 | 65.45 | 68.12 |
| | **NCI1** | 2,100 | 1,004 | 67.43 | 1,096 | 68.81 | 68.15 |
| | **Proteins** | 233 | 133 | 78.95 | 100 | 71.00 | 75.54 |
| | **IMDB-Binary** | 200 | 56 | 55.36 | 144 | 64.58 | 62.00 |

# 5 CONCLUSION AND FUTURE WORK

In the present paper, we investigate the use of a graph reduction method in a two-step classification scheme. In particular, we propose two strategies (candidate selection and early classification) that are applied on reduced graphs in order to speed up the complete classification procedure. Both modifications allow us to control the trade-off between classification accuracy and computation time.

With an empirical evaluation on five graph datasets, we verify the computational advantages of our novel two-step classification technique. That is our pruning strategies substantially reduce the run-time on all datasets. Moreover, we demonstrate that by using strongly reduced graphs in a two-step procedure, it is possible to maintain reasonable classification accuracy in general. Note that our approach is in some cases and on some datasets even capable to improve the classification accuracy of the reference system.

Future investigations involve, for instance, the use of ensemble methods that rely on different levels of graph reductions. That is we plan to create multiple sets of reduced graphs (e.g., in a bootstrapping fashion). The variety created by the reduced graph sets might help to decrease possible overfitting of the classifiers and thus improve the overall classification accuracy.

A natural idea to further reduce the matching time and keep equivalent classification accuracies would be to combine the two strategies proposed in the present paper. That is, the first classification step can be done with the early classification strategy. The graphs with clear classification can then be accepted as classified. For those with an ambiguous classification, only the $\lfloor \omega \cdot N \rfloor$-nearest training graphs are kept as candidates for the second step.

# REFERENCES

Abdulrahim, M. (1998). *Parallel algorithms for labeled graph matching*. PhD thesis, Colorado School of Mines.

Blumenthal, D. B., Gamper, J., Bougleux, S., and Brun, L. (2021). Upper bounding graph edit distance based on rings and machine learning. *Int. J. Pattern Recognit. Artif. Intell.*, 35(8):2151008:1–2151008:32.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Comput. Networks*, 30(1-7):107–117.

Brun, L., Foggia, P., and Vento, M. (2020). Trends in graph-based representations for pattern recognition. *Pattern Recognit. Lett.*, 134:3–9.

Bunke, H. and Allermann, G. (1983). Inexact graph matching for structural pattern recognition. *Pattern Recognit. Lett.*, 1(4):245–253.

Caelli, T. and Kosinov, S. (2004). An eigenspace projection clustering method for inexact graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(4):515–519.

Conte, D., Foggia, P., Sansone, C., and Vento, M. (2004). Thirty years of graph matching in pattern recognition. *Int. J. Pattern Recognit. Artif. Intell.*, 18(3):265–298.

Cortés, X. and Serratosa, F. (2015). Learning graph-matching edit-costs based on the optimality of the oracle's node correspondences. *Pattern Recognit. Lett.*, 56:22–29.

de Ridder, D., de Ridder, J., and Reinders, M. J. T. (2013). Pattern recognition in bioinformatics. *Briefings Bioinform.*, 14(5):633–647.

Dutta, A., Riba, P., Lladós, J., and Fornés, A. (2020). Hierarchical stochastic graphlet embedding for graph-based pattern recognition. *Neural Comput. Appl.*, 32(15):11579–11596.

Foggia, P., Percannella, G., and Vento, M. (2014). Graph matching and learning in pattern recognition in the last 10 years. *Int. J. Pattern Recognit. Artif. Intell.*, 28(1).

Freeman, L. C. (1977). A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41. Publisher: [American Sociological Association, Sage Publications, Inc.].

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

Gillioz, A. and Riesen, K. (2022). Speeding up graph matching by means of systematic graph reductions using centrality measures. In *2022 12th International Conference on Pattern Recognition Systems (ICPRS)*, pages 1–7. IEEE Computer Society.

Haxhimusa, Y. and Kropatsch, W. G. (2004). Segmentation graph hierarchies. In Fred, A. L. N., Caelli, T., Duin, R. P. W., Campilho, A. C., and de Ridder, D., editors, *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops, SSPR 2004 and SPR 2004, Lisbon, Portugal, August 18-20, 2004 Proceedings*, volume 3138 of *Lecture Notes in Computer Science*, pages 343–351. Springer.

Kashima, H., Tsuda, K., and Inokuchi, A. (2003). Marginalized kernels between labeled graphs. In Fawcett, T.

and Mishra, N., editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 321–328. AAAI Press.

Kriege, N. M., Johansson, F. D., and Morris, C. (2020). A survey on graph kernels. *Appl. Netw. Sci.*, 5(1):6.

Liu, Y., Safavi, T., Dighe, A., and Koutra, D. (2018). Graph summarization methods and applications: A survey. *ACM Comput. Surv.*, 51(3):62:1–62:34.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. (2020). Tudataset: A collection of benchmark datasets for learning with graphs. *CoRR*, abs/2007.08663.

Newman, M. E. J. (2010). *Networks: An Introduction*. Oxford University Press.

Pitas, I. (2016). *Graph-Based Social Media Analysis*. CRC Press. Google-Books-ID: BvYYCwAAQBAJ.

Qiu, H. and Hancock, E. R. (2006). Graph matching and clustering using spectral partitions. *Pattern Recognit.*, 39(1):22–34.

Riba, P., Fischer, A., Lladós, J., and Fornés, A. (2021). Learning graph edit distance by graph neural networks. *Pattern Recognit.*, 120:108132.

Riba, P., Lladós, J., and Fornés, A. (2020). Hierarchical graphs for coarse-to-fine error tolerant matching. *Pattern Recognit. Lett.*, 134:116–124.

Riesen, K. and Bunke, H. (2008). IAM graph database repository for graph based pattern recognition and machine learning. In da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J. T., Georgiopoulos, M., Anagnostopoulos, G. C., and Loog, M., editors, *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshop, SSPR & SPR 2008, Orlando, USA, December 4-6, 2008. Proceedings*, volume 5342 of *Lecture Notes in Computer Science*, pages 287–297. Springer.

Riesen, K. and Bunke, H. (2009). Approximate graph edit distance computation by means of bipartite graph matching. *Image Vis. Comput.*, 27(7):950–959.

Riesen, K., Fischer, A., and Bunke, H. (2016). Approximation of graph edit distance by means of a utility matrix. In Schwenker, F., Abbas, H. M., Gayar, N. E., and Trentin, E., editors, *Artificial Neural Networks in Pattern Recognition - 7th IAPR TC3 Workshop, AN-NPR 2016, Ulm, Germany, September 28-30, 2016, Proceedings*, volume 9896 of *Lecture Notes in Computer Science*, pages 185–194. Springer.

Sanfeliu, A. and Fu, K. (1983). A distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. Syst. Man Cybern.*, 13(3):353–362.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1):4–24.