Iterative Creation of Matching-Graphs – Finding Relevant Substructures in Graph Sets*

Mathias Fuchs[1][0000−0002−9482−0442] and Kaspar Riesen[1,2][0000−0002−9145−3157]

[1] Institute of Computer Science, University of Bern, 3012 Bern, Switzerland, mathias.fuchs@inf.unibe.ch
[2] Institute for Informations Systems, University of Appl. Sci. Northwestern Switzerland, 4600 Olten, Switzerland, kaspar.riesen@fhnw.ch

Abstract. Both the amount of data available and the rate at which it is acquired increases rapidly. The underlying data is often complex, making it difficult (or somehow unnatural) to represent it by vectorial data structures. Hence, graphs are a promising alternative for formalizing the data. Actually a large amount of graph-based methods for pattern recognition have been proposed. The vast amount of these methods rely on graph matching procedures. In a recent paper a novel encoding of graph matching information has been proposed. The idea of this encoding is to formalize the stable cores of specific classes by means of graphs (called matching-graphs). In the present paper we aim to further improve the relevance of these matching-graphs by using an iterative creation algorithm. In an empirical evaluation we show that these novel matching-graphs offer a more stable and significant representation of their respective class than the previous version.

Keywords: Graph Matching · Matching-Graphs · Graph Edit Distance.

1 Introduction and Related Work

Pattern recognition emerged to a major field of research which aims at solving diverse problems like signature verification [1], situation recognition [2], or breast cancer detection [3], to name just a few prominent examples. Roughly speaking there are two main approaches for pattern recognition. *Statistical approaches*, which use data structures like *vectors* for data representation and *structural approaches*, which use *strings*, *trees*, or *graphs* for the same task. Graphs provide a powerful alternative to feature vectors and thus, they are widely used in various pattern recognition applications, ranging from protein function/structure prediction [4], over inferring the privacy risk of an image on social media [5], to the detection of Alzheimer's Disease [6]. The main drawback of graphs is, however, the computational complexity of basic operations, which in turn makes graph based algorithms often slower than their statistical counterparts.

A large amount of graph based methods for pattern recognition have been proposed from which many rely on *graph matching* [7, 8]. Graph matching is

* Supported by Swiss National Science Foundation (SNSF) Project Nr. 200021_188496.

typically used for quantifying graph proximity. *Graph edit distance* [9,10], introduced about 40 years ago, is recognized as one of the most flexible graph distance models available. In contrast with many other distance measures (e.g. *graph kernels* [11]), graph edit distance generally offers more information than merely a dissimilarity score, viz. the information which subparts of the underlying graphs actually match with each other (known as *edit path*).

In a recent paper [12], the authors of the present paper propose to explicitly exploit the matching information of graph edit distance. Formally, we encode the matching information derived from graph edit distance into a data structure, called *matching-graph*. The main contribution of the present paper is to further improve the quality of these matching-graphs by means of an iterative process, which selects the best matching-graphs of each iteration and continues to create new matching-graphs from these selected parent graphs. The proposed algorithm is remotely inspired by the idea of genetic algorithms that also aim at emulating the process of natural selection and improvement of a population [13].

Moverover, our approach is similar in spirit to approaches from graph transaction based *Frequent Subgraph Mining (FSM)* [14]. This field also focuses on the identification of frequent subgraphs within a set of graphs (extract all subgraphs that occur more often than a specified threshold). We observe two main categories in FSM, viz. *Apriori-based approaches* and *Pattern-growth approaches* [14]. The apriori-based methods proceed to grow subgraphs by using a *Breadth First Search (BFS)* strategy. Before they continue to graphs of size $k+1$ it first searches for all frequent graphs of size k. Pattern-growth approaches, on the other hand, work by using a *Depth First Search (DFS)* strategy, where one graph is extended until all frequent supergrahs of this graphs are found.

Though the goal of our approach is comparable to that of FSM, our procedure is quite unique. We identify common subgraphs of pairs of graphs by using a graph matching procedure rather than using an algorithm stemming from one of the two main categories discussed above (Apriori or Pattern Growth). We do also not focus on finding comprehensive lists of frequent and large subgraphs but rather we aim at improving our initial set of matching-graphs by iteratively matching these matching-graphs with each other in order to extract more stable and robust graph representatives for each class.

In the present paper we conduct both a quantitative and qualitative experimental evaluation. First, we measure the frequencies of the found matching-graphs in their correct class and second, we visualize and inspect the most frequent subgraphs which in turn enables novel insights into the question which graph substructures actually make up a class of patterns.

The remainder of this paper is organized as follows. Sect. 2 makes the paper self-contained by providing basic definitions and terms used throughout this paper. Next, in Sect. 3 the general procedure for creating a matching-graph is explained together with a description of the novel algorithm that we propose. Eventually, in Sect. 4, we empirically confirm that our algorithm produces indeed a set of highly relevant graph structures. Finally, in Sect. 5, we conclude the paper and discuss some ideas for future work.

2 Graphs and Graph Edit Distance - Basic Definitions

Let L_V and L_E be finite or infinite label sets for nodes and edges, respectively. A *graph* g is a four-tuple $g = (V, E, \mu, \nu)$, where

- V is the finite set of nodes,
- $E \subseteq V \times V$ is the set of edges,
- $\mu : V \to L_V$ is the node labeling function, and
- $\nu : E \to L_E$ is the edge labeling function.

In the present paper we employ *graph edit distance* as basic dissimilarity model for graphs. One of the main advantages of graph edit distance is its high degree of flexibility, which makes it applicable to virtually any kind of graphs.

Given two graphs g_1 and g_2, the basic idea of graph edit distance is to transform g_1 into g_2 using some *edit operations*. A standard set of edit operations is given by *insertions*, *deletions*, and *substitutions* of both nodes and edges. We denote the substitution of two nodes $u \in V_1$ and $v \in V_2$ by $(u \to v)$, the deletion of node $u \in V_1$ by $(u \to \varepsilon)$, and the insertion of node $v \in V_2$ by $(\varepsilon \to v)$, where ε refers to the empty node. For edge edit operations we use a similar notation.

A set $\{e_1, \ldots, e_t\}$ of t edit operations e_i that transform a source graph g_1 completely into a target graph g_2 is called an *edit path* $\lambda(g_1, g_2)$ between g_1 and g_2. Let $\Upsilon(g_1, g_2)$ denote the set of all edit paths transforming g_1 into g_2 while c denotes the cost function measuring the strength $c(e_i)$ of edit operation e_i. The graph edit distance can now be defined as follows.

Let $g_1 = (V_1, E_1, \mu_1, \nu_1)$ be the source and $g_2 = (V_2, E_2, \mu_2, \nu_2)$ the target graph. The *graph edit distance* between g_1 and g_2 is defined by

$$d_{\lambda_{\min}}(g_1, g_2) = \min_{\lambda \in \Upsilon(g_1, g_2)} \sum_{e_i \in \lambda} c(e_i) \quad , \tag{1}$$

Optimal algorithms for computing the edit distance of two graphs are typically based on combinatorial search procedures. A major drawback of those procedures is their computational complexity, which is exponential in the number of nodes. To render graph edit distance computation less computationally demanding, we employ the often used approximation algorithm BP [15].

3 Matching-Graphs

3.1 Creating Matching-Graphs

Our novel approach is based on matching-graphs originally proposed in [12]. The general idea of the matching-graphs is to extract information on the matching of pairs of graphs in a new data structure that formalizes and encodes the matching parts of the two graphs.

Formally, we assume k sets of training graphs $G_{\omega_1}, \ldots, G_{\omega_k}$ stemming from k different classes $\omega_1, \ldots, \omega_k$. For all pairs of graphs g_i, g_j stemming from the same class ω_l, the graph edit distance is computed. Hence, a (suboptimal) edit

path $\lambda(g_i, g_j)$ is obtained for each pair of graphs $g_i, g_j \in G_{\omega_l} \times G_{\omega_l}$. For each edit path $\lambda(g_i, g_j)$, two matching-graphs $m_{g_i \times g_j}$ and $m_{g_j \times g_i}$ are eventually built (for the source and the target graph g_i and g_j, respectively). To this end, all nodes of g_i and g_j that are actually substituted in edit path $\lambda(g_i, g_j)$ are added to $m_{g_i \times g_j}$ and $m_{g_j \times g_i}$, respectively. Vice versa, all nodes that are deleted in g_i or inserted in g_j are neither considered in the two matching-graphs.

Note that this procedure can result in matching-graphs with isolated nodes, which are eventually removed. If a node is not included in the matching-graph (since it was either deleted or inserted in the underlying edit path), the incident edges of this node are not included in the resulting matching-graph. Edges that connect two substituted nodes, however, are included in the matching-graphs. That is, if two nodes $u_1, u_2 \in V_i$ of a source graph g_i are substituted with nodes $v_1, v_2 \in V_j$ in a target graph g_j and there is an edge $(u_1, u_2) \in E_i$ available, (u_1, u_2) is actually included in the matching-graph $m_{g_i \times g_j}$ (whether or not edge (v_1, v_2) is available in E_j).

As shown in [12], the complete process leads to graph structures that can be interpreted as denoised core structures of their respective class. The present paper is built upon these matching-graphs by pursuing the goal of gradually improving the matching-graphs of the first iteration.

3.2 Iterative Building of Matching-Graphs

Using the described procedure for creating matching-graphs out of two input graphs, we now propose an algorithm that iteratively creates sets of matching-graphs out of existing sets of matching-graphs. The proposed procedure is formalized in Algorithm 1.

Algorithm 1: Algorithm for iterative matching-graph creation.

input : sets of graphs from k different classes $\mathcal{G} = \{G_{\omega_1}, \ldots, G_{\omega_k}\}$, number of matching-graphs c
output: sets of matching-graphs for each of the k different classes $\mathcal{M} = \{M_{\omega_1}, \ldots, M_{\omega_k}\}$

Initialize $\mathcal{M}$ as the empty set: $\mathcal{M} = \{\}$
foreach *set of graphs* $G \in \mathcal{G}$ **do**
  Initialize M as the empty set: $M = \{\}$
  foreach *pair of graphs* $g_i, g_j \in G \times G$ *with* $j > i$ **do**
    $M = M \cup \{m_{g_j \times g_i}, m_{g_i \times g_j}\}$
  end
  reduce M to the c matching-graphs with highest quality q
  do
    foreach *pair of graphs* $m_i, m_j \in M \times M$ *with* $j > i$ **do**
      $M = M \cup \{m_{m_j \times m_i}, m_{m_i \times m_j}\}$
    end
    reduce M to the c matching-graphs with highest quality q
  while *M has changed in the last iteration*
  $\mathcal{M} = \mathcal{M} \cup M$
end

The input of the algorithm is a set $\mathcal{G}$ that contains several sets of graphs $\{G_{\omega_1}, \ldots, G_{\omega_k}\}$ each representing members of a certain class ω_k. Additionally, the number of matching-graphs per class is fixed to a user-defined value c. The output is a set $\mathcal{M}$ which consists of k different sets $M_{\omega_1}, \ldots, M_{\omega_k}$ each containing c matching-graphs that represent one of the given classes.

First $\mathcal{M}$ is initialized to the empty set. The algorithm then actually starts on line 2 by iterating over each set of graphs $G \in \mathcal{G}$. For each of these sets the corresponding result M is initialized as the empty set.

As seen on line 4 to 6, before beginning the main iterative process, we first loop through all possible combinations of graphs g_i, $g_j \in G \times G$, where $j > i$. As mentioned in Sect. 3.1, from one edit path $\lambda(g_i, g_j)$ two matching-graphs are inferred, viz. $m_{g_i \times g_j}$ and $m_{g_j \times g_i}$, where g_i and g_j is the source and target graph, respectively (line 5). Hence, this process yields $n(n-1)$ matching-graphs, where n is the number of graphs in the current set G[3].

On line 7 we proceed to select the c graphs from M with the highest quality q by calculating the relative frequency of occurrence in their own class with respect to the occurrence in other classes. Formally, for a matching-graph $m \in M$ derived from graphs stemming from class ω_l, we verify for all graphs $g \in G_{\omega_l}$ whether or not m is a subgraph of g and store the number of positive matches in f_1. Likewise, we count all graphs $g' \in G_{\omega_i}$, where $\omega_i \neq \omega_l$, that contain m as subgraph and store this number in f_2.

Clearly, the higher f_1 and simultaneously the lower f_2 for a given matching-graph m, the better the quality of m. With $f_2 = max(1, f_2)$ (in order to avoid divisions by zero), we formalize the quality q of a matching-graph m by means of

$$q(m) = \frac{f_1}{f_2} \quad . \tag{2}$$

Given this initial set M of matching-graphs, the whole process is eventually repeated (lines 9 to 12). Yet, instead of creating the matching-graphs from the training set G, we produce matching-graphs from pairs of existing matching-graphs. This process is repeated as long as the c graphs in M have altered in the last iteration (line 13). Once the algorithm terminates, we obtain k sets of c matching-graphs for each class which are stored in $\mathcal{M}$.

4 Experimental Evaluation

4.1 Experimental Setup

The main question – from the experimental point of view – is whether or not our novel procedure is able to create more representative matching-graphs than

[3] Note that edit path $\lambda(g_i, g_j)$ is not necessarily the same as $\lambda(g_j, g_i)$ and thus, it could actually happen that the resulting matching-graphs stemming from these edit paths also differ. Yet, due to computational reasons we omit the computations of the edit paths and matching-graphs in both directions and assume two matching-graphs per graph pair.

the initial procedure (proposed in [12]). In order to answer this question, we count the occurrences of the matching-graphs found in a given test set (via subgraph isomorphism verification from graph-tool[4] that is based on the VF2 algorithm [16]). That is, we create the matching-graphs using the aforementioned algorithm, on the training sets and then count the actual occurences of the created graphs as subgraphs in the corresponding test sets.

The proposed approach is evaluated on two different data sets from the IAM graph repository both providing graphs from two classes [17][5]:

- AIDS (active vs. inactive)
- Mutagenicity (mutagen vs. nonmutagen)

The single parameter of our algorithm – namely, the number of matching-graphs being generated – is set to $c = 15$ in our experiments for the sake of convenience.

4.2 Test Results and Discussion

First, we aim at researching whether or not the quality of the matching-graphs actually improves from iteration to iteration. To this end, we plot the qualities (according to Eq. 2) of the top c matching-graphs from the first to the last iteration (see Fig. 1). It is clearly observable that the quality of the matching-graphs increases by each iteration. For instance, for the AIDS data set and class active the initial matching-graphs offer quality values between 20 and 38, while the qualities of the final matching-graphs are between 39 and 45, which means that the final matching-graphs occur about 39 to 45 times more often in their own class than in the other class.

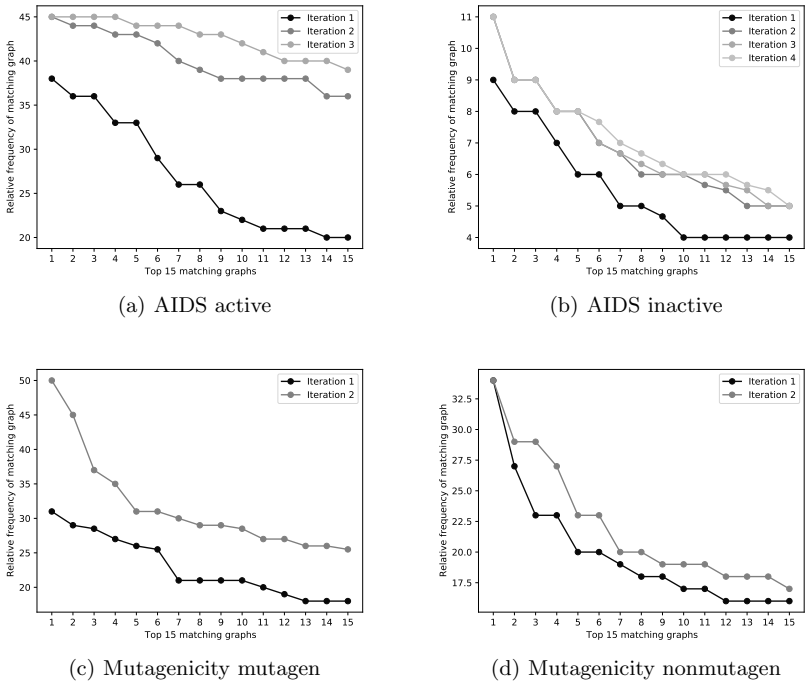
One could assume that this increase is mainly due to the fact that the matching-graphs become smaller from iteration to iteration (and are therefore found more often in the correct class). In fact, we observe only a marginal reduction of the average graph size (if any). This can be seen in Table 1 where we show the number of iterations per data set and class as well as the average number of nodes of the matching-graphs in the first and last iteration.

Table 1. Development of the average number of nodes from the top matching-graphs between the first and last iteration.

		# iterations	Avg. # nodes first iteration	Avg. # nodes last iteration
MUTA	nonmutagen	2	17.9	18.1
	mutagen	2	14.4	14.3
AIDS	inactive	4	6.6	5.5
	active	3	14.5	12.1

[4] https://graph-tool.skewed.de/static/doc/topology.htmlgraph_tool.topology.subgraph_isomorphism
[5] `www.iam.unibe.ch/fki/databases/iam-graph-database`

Next, we analyze the absolute frequencies of the resulting matching-graphs in the correct and false classes (see Fig. 2). It can be clearly observed that the resulting matching-graphs occur siginficantly more often in their correct classes than in the wrong class for the AIDS active, Mutagenicity mutagen and nonmutagen classes.



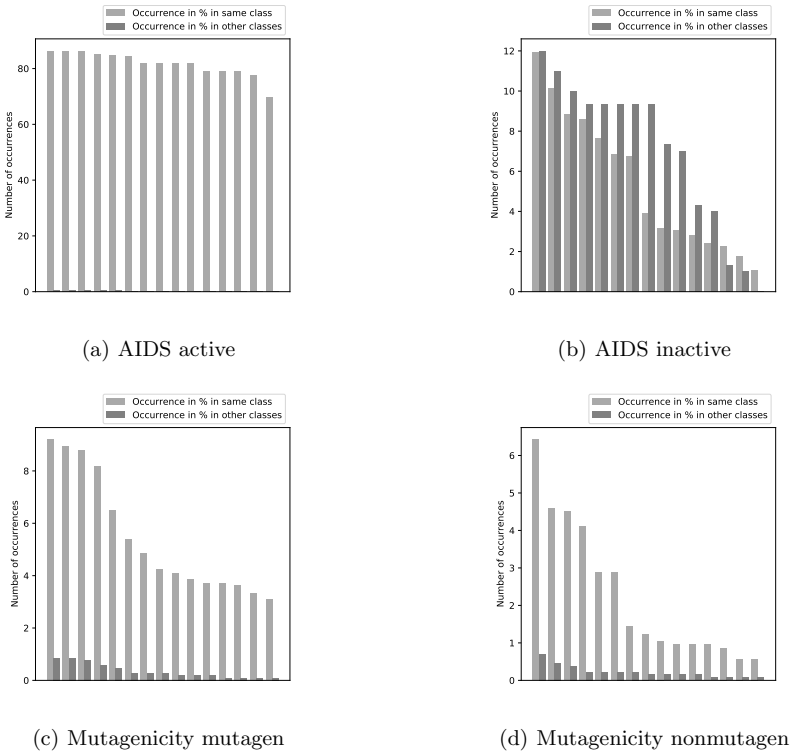
(a) AIDS active (b) AIDS inactive

(c) Mutagenicity mutagen (d) Mutagenicity nonmutagen

Fig. 1. Evolution of the relative frequencies of which a matching-graph occurs in the correct and incorrect class during the iterations.

In particular for AIDS active (Fig. 2 (a)) we can report exciting results, where most of the matching-graphs occur in about 80% of the test graphs of the correct class, and only about 1% in the other class. However, for the AIDS inactive (Fig. 2 (b)), the resulting matching-graphs do not seem to be representative.

Finally, we conduct a qualitative evaluation. In Fig. 3 we visualize the three matching-graphs with the best quality (according to Eq. 2) of each class for both data sets. Interestingly, as seen in Fig. 3 (a), the matching-graphs for the AIDS active class consist of carbon atoms only (in very specific combinations, that seems to be exclusive for this class). The matching-graphs of the inactive class on the other hand consist of chains of various atoms, that seem to be less common overall and not very specific to the inactive class (as seen in the quantitative analysis in Fig. 2).



(a) AIDS active

(b) AIDS inactive

(c) Mutagenicity mutagen

(d) Mutagenicity nonmutagen

Fig. 2. Percentage of frequency of the final c matching-graphs in the test set. Bars in light gray show the frequency in the correct class while the darker bars show the frequency in the incorrect class.

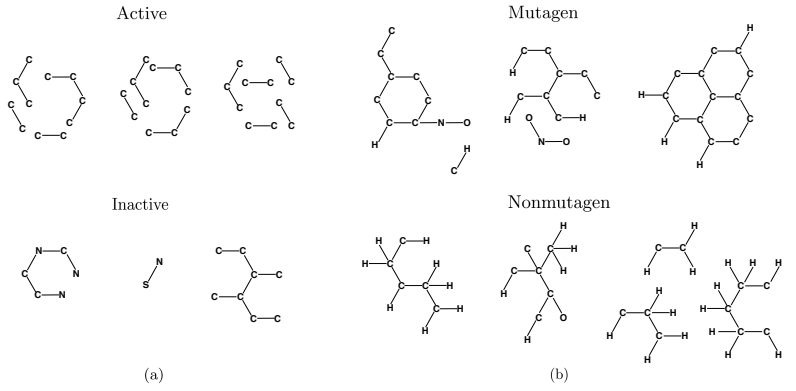


Fig. 3. The three matching-graphs with the best quality for the AIDS data set (a) and the Mutagenicity data set (b).

In Fig. 3 (b) we show the top matching-graphs for the Mutagenicity data set. One of the major differences is, that for the mutagen class the matching-graphs found often contain carbon rings or partial carbon rings, whereas in the nonmutagen class we find much more hydrogen atoms. Also very interesting to see is that the second sample of the Mutagen class in Fig. 3 (b) contains a NO_2 compound, which is well known to be mutagenic [18]. Overall the NO_2 compound occurs in 5 out of the 15 matching-graphs. This is especially interesting as the matching-graphs are automatically created on the basis of the edit path between training and matching-graphs without any further knowledge.

5 Conclusions and Future Work

We propose to build matching-graphs on the basis of the edit path between two graphs. The resulting matching-graphs basically include the nodes that are substituted via graph edit distance. In the present paper we advance the creation of matching-graphs by means of an iterative algorithm. That is, starting with an initial set of matching-graphs, novel sets of matching-graphs are iteratively created by means of further matchings of matching-graphs.

In an experimental evaluation on two graph data sets, we empirically confirm that our novel approach is able to produce matching-graphs that accurately represent significant and frequent substructures of a given class. Moreover, by means of a qualitative evaluation we confirm that our novel procedure offers high potential for detecting novel and relevant substructures in sets of graphs. To the best of our knowledge this is the first time that a graph matching algorithm is employed for this specific task.

There are several promising paths to be pursued in future work. First we feel that the classification accuracy of the framework presented in [12] can further be increased by using the novel improved matching-graphs. Moreover, one could evaluate the procedure on more data sets (especially on graphs with continuous labels). Furthermore, it might be interesting to compare our novel method with well known graph mining algorithms. Last but not least, one could integrate the improved matching-graphs in a classification scheme (e.g. in a distance based classifier or in a subgraph-kernel).

References

1. Maergner, P., Pondenkandath, V., Alberti, M., Liwicki, M., Riesen, K., Ingold, R., Fischer, A. (2018, August). Offline signature verification by combining graph edit distance and triplet networks. In Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR) (pp. 470-480). Springer, Cham.
2. Jing, Y., Wang, J., Wang, W., Wang, L., Tan, T. (2020). Relational graph neural network for situation recognition. Pattern Recognition, 108, 107544.
3. Khan, S., Islam, N., Jan, Z., Din, I. U., Rodrigues, J. J. C. (2019). A novel deep learning based framework for the detection and classification of breast cancer using transfer learning. Pattern Recognition Letters, 125, 1-6.

4. Rieck, B., Bock, C., Borgwardt, K. (2019, May). A persistent weisfeiler-lehman procedure for graph classification. In International Conference on Machine Learning (pp. 5448-5458). PMLR.
5. Yang, G., Cao, J., Chen, Z., Guo, J., Li, J. (2020). Graph-based neural networks for explainable image privacy inference. Pattern Recognition, 105, 107360.
6. Curado, M., Escolano, F., Lozano, M. A., Hancock, E. R. (2020). Early Detection of Alzheimer's Disease: Detecting Asymmetries with a Return Random Walk Link Predictor. Entropy, 22(4), 465.
7. Conte, D., Foggia, P., Sansone, C., Vento, M. (2004). Thirty years of graph matching in pattern recognition. International journal of pattern recognition and artificial intelligence, 18(03), 265-298.
8. Foggia, P., Percannella, G., Vento, M. (2014). Graph matching and learning in pattern recognition in the last 10 years. International Journal of Pattern Recognition and Artificial Intelligence, 28(01), 1450001.
9. Bunke, H., Allermann, G. (1983). Inexact graph matching for structural pattern recognition. Pattern Recognition Letters, 1(4), 245-253.
10. Sanfeliu, A., Fu, K. S. (1983). A distance measure between attributed relational graphs for pattern recognition. IEEE transactions on systems, man, and cybernetics, (3), 353-362.
11. Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., Borgwardt, K. M. (2010). Graph kernels. Journal of Machine Learning Research, 11, 1201-1242.
12. Fuchs, M., Riesen, K. Matching of Matching-Graphs - A Novel Approach for Graph Classification. In Proceedings of the 25th International Conference on Pattern Recognition, ICPR 2020, Milano, Italy, January 10-15, 2021
13. Mitchell, M. (1996). An introduction to genetic algorithms. Cambridge.
14. Jiang, C., Coenen, F., Zito, M. (2013). A survey of frequent subgraph mining algorithms. Knowledge Engineering Review, 28(1), 75-105.
15. K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(4):950–959, 2009.
16. Cordella, L. P., Foggia, P., Sansone, C., Vento, M. (2004). A (sub) graph isomorphism algorithm for matching large graphs. IEEE transactions on pattern analysis and machine intelligence, 26(10), 1367-1372.
17. K. Riesen and H. Bunke. IAM graph database repository for graph based pattern recognition and machine learning. In N. da Vitoria Lobo et al., editor, *Structural, Syntactic, and Statistical Pattern Recognition*, LNCS 5342, pages 287–297, 2008.
18. Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., Zhang, X. (2020). Parameterized explainer for graph neural network. arXiv preprint arXiv:2011.04573.