A Graph Database Repository and Performance Evaluation Metrics for Graph Edit Distance

Zeina Abu-Aisheh, Romain Raveaux, and Jean-Yves Ramel

Laboratoire d'Informatique (LI), Université François Rabelais,
37200, Tours, France
{zeina.abu-aisheh,romain.raveaux,jean-yves.ramel}@univ-tours.fr
http://www.li.univ-tours.fr/

Abstract. Graph edit distance (GED) is an error tolerant graph matching paradigm whose methods are often evaluated in a classification context and less deeply assessed in terms of the accuracy of the found solution. To evaluate the accuracy of GED methods, low level information is required not only at the classification level but also at the matching level. Most of the publicly available repositories with associated ground truths are dedicated to evaluating graph classification or exact graph matching methods and so the matching correspondences as well as the distance between each pair of graphs are not directly evaluated. This paper consists of two parts. First, we provide a graph database repository annotated with low level information like graph edit distances and their matching correspondences. Second, we propose a set of performance evaluation metrics to assess the performance of GED methods.

Keywords: Graph Edit Distance, Performance Evaluation Metrics, Matching Correspondence.

1 Introduction

Attributed relational graphs are powerful representation structures that have been widely used to represent structural description of objects in pattern recognition (PR), computer vision and other related fields. When objects are represented by graphs, the problem of object comparison is turned into a graph matching (GM) one where an evaluation of structural and attributed similarity of two graphs has to be found [1]. The similarity evaluation, based on GM, consists in finding correspondences between vertices and edges of two graphs that satisfy some constraints ensuring that similar substructures in one graph are mapped to similar substructures in the other [2].

Among *error-tolerant* GM problems, GED is of great interests. GED is a graph matching paradigm whose concept was first reported in [3, 4]. Its basic idea is to find the best set of edit sequences that can transform graph g_1 into graph g_2 by means of edit operations on graph g_1. The allowed operations are inserting, deleting and/or substituting vertices and their corresponding edges.

In the literature, many exact and approximate approaches have been proposed to solve GED [5–8]. As a first step to evaluate such approaches, one needs to find repositories dedicated to evaluating GM in general or GED in particular.

Lots of graphs repositories have been made publicly available for the community [9, 10]. However, to the best of our knowledge, most of these repositories have been put forward for classification and clustering experiments. Moreover, only high level information has been given to the community such as the class labels of the objects represented by graphs. When evaluating classification, the matching quality is evaluated indirectly through a recognition rate which highly depends on the classifier and does not allow a clear analysis of the matching algorithms. On the other hand, low level information has not been provided. For instance, the matching between vertices and edges of each pair of graphs.

We, authors, believe that providing low level information is of great interest for understanding the behavior of GED methods in terms of accuracy and speed as a function of graph size and attribute types. Hence, instead of proposing yet a completely new graph database repository, we propose adding low level information for well-known and publicly used databases. For that purpose, the GREC, Mutagenicity, Protein, and CMU databases were selected [9, 11]. Added information consists of the best found distance for each pair of graphs as well as their vertex to vertex and edge to edge matching. This information helps at assessing the feasibility of exact and approximate methods.

Our repository aims at making a first step towards a GM repository that is able to assess the accuracy of error-tolerant GM methods. All the graph databases and their added information are publicly available[1]. Moreover, we propose novel performance evaluation metrics that aim at comparing a set of GED approaches based on several significant criteria. For instance, deviation and solution optimality. All the provided criteria are assessed under time and memory constraints.

The remainder of this paper is organized as follows: In Section 2, a focus on the related works is given. In Section 3, the graph set repository is described. In Section 4, the different performance evaluation metrics are put forward to evaluate GED approaches. Section 5 demonstrates a use case of these performance evaluation metrics. Finally, Section 6 is devoted to conclusions and perspectives.

2 Related Works

Table 1 synthesizes the graph databases presented in the literature. One may notice that exact GM has been evaluated at the matching level. However, error-tolerant GM has been evaluated at the classification level rather than the matching one. [11] is devoted to error-tolerant GM with ground truth information. However, graphs have the same number of vertices and thus the scalability measure cannot be assessed. Indeed, there is a lack of performance comparison measures for error-tolerant GM methods, whether exact or approximate ones. Moreover, none of the graph repositories was dedicated to assessing the performance

[1] http://www.rfai.li.univ-tours.fr/PagesPerso/zabuaisheh/GED-benchmark.html

of GED methods. In this paper, we focus on GED methods since GED is a flexible paradigm that has been used in different applications in PR. To understand better the behavior of each GED method, we provide new performance metrics.

Ref	Problem Type	Graph Type	Database Type	Measure Type	Purpose
[12]	Exact GM	Non-attributed	Synthetic	Accuracy and scalability	Matching
[11]	Error-tolerant GM	Attributed	Real-world	Memory consumption, accuracy and matching quality	Matching
[9]	Error-tolerant GM	Attributed	Real-world	Accuracy and running time	Classification
[13, 14]	Exact GM	Attributed	Synthetic	Accuracy and scalability	Matching
[15]	Exact GM	(Non)attributed	Real-world	Scalability	Matching

Table 1. Synthesis of graph databases.

3 Graph Set Repository

This section is devoted to the description and the justification of the selected graph databases as well as the details of the information added to each database.

3.1 Databases

Several databases are integrated in our repository. This section is devoted to the advantages of each of these databases as well as graphs selection.

We aim at evaluating GED approaches when increasing the number of vertices. To this end, we decomposed each database into disjoint subsets, each of which contains graphs that have the same number of vertices. Moreover, we aim at studying the behavior of each algorithm on different types of attributes.

For each database, two non-negative meta parameters associated to GED are included (τ_{node} and τ_{edge}) where τ_{node} denotes a vertex deletion or insertion whereas τ_{edge} denotes an edge deletion or insertion. A third meta parameter α is integrated to control whether the edit operation cost on the vertices or on the edges is more important. From each graph matching pair, we derive two notions: a distance between each pair of graphs and vertex-vertex and edge-edge matching or so-called edit sequence.

GREC Database: This database consists of a subset of the symbol database underlying the GREC 2005 competition [16].

Database Interest: GREC is composed of undirected graphs of rather small size (*i.e.,* up to 24). In addition, continuous attributes on vertices and edges play an important role in the matching process. Such graphs are representative of pattern recognition problems where graphs are involved in a classification stage.

Cost Function: Additionally to (x, y) coordinates, the vertices of graphs from the GREC database are labeled with a type (ending point, corner, intersection, circle). For edges, two types (line, arc) are employed. The cost functions of vertex and edge substitutions, deletions and insertions are defined as follows:

- $c(u \rightarrow \epsilon) = c(\epsilon \rightarrow v) = \alpha.\tau_{node}$
- $c(u \rightarrow v) = \begin{cases} \alpha.|\mu(u) - \mu(v)|, & \text{if labels are similar} \\ \alpha.2.\tau_{node}, & \text{otherwise} \end{cases}$
- $c(p \rightarrow q) = 2.(1-\alpha).Dirac(\mu(p), \mu(q))$
- $c(p \rightarrow \epsilon) = c(\epsilon \rightarrow q) = (1-\alpha).\tau_{edge}$

where $u \in V_1$, $v \in V_2$, $p \in E_1$ and $q \in E2$. V_* and E_* refer to the sets of vertices and edges of graph g_*, respectively. μ is a function which returns the attribute(s) of each vertex/edge. $(* \rightarrow \epsilon)$ and $(\epsilon \rightarrow *)$ denote vertex/edge deletion and insertion, respectively. In our experiments, we have set τ_{node}, τ_{edge} and α to 90, 15 and 0.5 respectively. The meta parameters' values of the GREC, Mutagenicity and Protein databases are taken from the PhD thesis of Riesen [17].

Database Decomposition: We filtered and decomposed the database into the following subsets: (GREC-5, GREC-10, GREC-15 and GREC-20), each subset has 10 graphs of size 5, 10, 15 and 20, respectively in addition to GREC-mix that has graphs of different sizes taken from the aforementioned subsets. Graphs were chosen from the train set and from different classes to capture, at best, graph variability. 100 pairwise comparisons per subset are conducted.

Mutagenicity Database: The Mutagenicity database was originally prepared by the authors of [18]. For simplicity, we denote this database as MUTA.

Database Interest: MUTA is representative of GM problems where graphs have only symbolic attributed. MUTA gathers large graphs up to 70 vertices.

Cost Function: The cost functions of operations on vertices and edges are defined as follows:

- $c(u \rightarrow \epsilon) = c(\epsilon \rightarrow v) = 2.\alpha.\tau_{node}$
- $c(u \rightarrow v) = \begin{cases} 0, & \text{if they have the same symbols} \\ \alpha.2.\tau_{node}, & \text{otherwise} \end{cases}$
- $c(p \rightarrow \epsilon) = c(\epsilon \rightarrow q) = (1-\alpha).\tau_{edge}$ $\quad$ $-$ $c(p \rightarrow q) = 0$

where (τ_{node}, τ_{edge},α) values of Mutagenicity are set to (11,1.1,0.25).

Database Decomposition: (MUTA-10, MUTA-20, MUTA-30 ... MUTA-70) were selected and shrunk to 10 graphs per subset. In some subsets of MUTA, the number of train graphs was less than 10. Hence, to complete the subsets, we took some graphs from the test or the validation subsets. We also added another subset, denoted by MUTA-mix, that contains 10 graphs of various number of vertices. As in GREC, 100 comparisons are carried out in each subset.

Protein Database: The protein database was first reported in [18].

Database Interest: This database contains numeric attributes on each vertex as well as a string sequence that is used to represent the amino acid sequence.

Cost Function: The cost functions of matching operations are defined as follows:

- $c(u \to \epsilon) = c(\epsilon \to v) = \alpha.\tau_{node}$
- $c(u \to v) = \begin{cases} \text{SED}(\mu(u), \mu(v)), & \text{if labels are similar} \\ \alpha.\tau_{node}, & \text{otherwise} \end{cases}$
- $c(p \to \epsilon) = c(\epsilon \to q) = (1-\alpha).\tau_{edge}$
- $c(p \to q) = \begin{cases} 0, & \text{if labels are similar} \\ \alpha.\tau_{edge}, & \text{otherwise} \end{cases}$

where (τ_{node},τ_{edge},α) are set to (11,1,0.75) and SED is string edit distance [19]. Given two amino acid sequences (s_1 and s_2), the corresponding cost of matching symbols in s_1 and s_2 is defined as follows:
c(k $\to$ m) = c(k $\to \epsilon$) = c($\epsilon \to$ m) = 1, c(k $\to$ m) = 0, s.t. $k, m \in s_1, s_2$.

Database Decomposition: 10 graphs were selected from Protein-20, Protein-30 and Protein-40. In addition, 10 graphs were picked up from the aforementioned Protein subsets and were put in the mixed database referred to as Protein-mix. 100 pairwise matchings per subset are conducted and integrated in the repository.

CMU Database: The CMU model house sequence is made up of a series of images of a toy house that has been captured from different viewpoints. 111 images in total are publicly available. 660 comparisons are carried out.

Database Interest: Unlike the aforementioned databases, the CMU database has a model house sequence that consists of a series of images. Each of which has been manually annotated by a human being providing its key points (corner features). Moreover, the ground truth is attached with each pair of graphs.

Cost Function: We empirically set (τ_{node},α) to (∞,0.5). The cost functions of matching vertices and edges are similar to [20]. We formalize them as follows:

- $c(u \to \epsilon) = c(\epsilon \to v) = \alpha.\tau_{node}$
- $c(u \to v) = 0$
- $c(p \to \epsilon) = c(\epsilon \to q) = (1-\alpha).\mu(q)$
- $c(p \to q) = (1-\alpha).|\mu(p) - \mu(q)|$

Graphs Construction: A manual identification of corner features, or points, was done to represent vertices on each of the rotated images. Then, the Delaunay triangulation was applied on the corner-features in order to identify edges and finally transform the images into graphs. Vertices are labeled with (x,y) coordinates while edges are labeled with the distance between vertices. Each graph has 30 vertices and matched with graphs at 10, 20, 30, 40, 50, 60, 70, 80, and 90 in the rotation sequence. Since vertex to vertex matchings are given, the accuracy of the final matching sequence of any method p can be computed by verifying whether or not the matched vertices are correct.

3.2 Added Low Level Information

For each graphs pair (g_i, g_j), the initial content of the databases in addition to the following low level information are provided: the optimal or sub-optimal solution provided by the most accurate GED method for $d(g_i, g_j)$, the name of the most accurate GED method, the solution status (*i.e.,* optimal or sub-optimal) and the edit path sequence. Table 2 illustrates an example of two graphs taken from GREC-5 and their added information in our repository. All these information are available as csv files in our website.

G1Name	G2Name	Method	Distance	Optimal	Matching
image3_23	image3_25	$BS-100$	135.178	false	*Vertex*:0 $\rightarrow$ 0=37.476/ *Vertex*:1 $\rightarrow$ 1=6.519/ *Vertex*:2 $\rightarrow$ 2=32.070/ *Vertex*:4 $\rightarrow$ 4=34.409/ *Vertex*:3 $\rightarrow$ 3=24.703/ *Edge*:2 $\leftrightarrow$ 3 $\rightarrow$ 2 $\leftrightarrow$ 3 =0.0/ *Edge*:0 $\leftrightarrow$ 4 $\rightarrow$ 0 $\leftrightarrow$ 4=0.0

Table 2. Low level information (taken from the file GREC5-lowlevelinfo.csv).

4 Performance Evaluation Metrics

In this section, we define the new metrics used for evaluating any GED method. We propose to analyze the behavior of the different methods under a time constraint (C_T) as well as a memory constraint (C_M):

Deviation: We compute the error committed by each method p over the reference distances. For each pair of graphs matched by method p, we provide the following deviation measure:

$$dev(g_i, g_j)^p = \frac{|d(g_i, g_j)^p - R_{g_i,g_j}|}{R_{g_i,g_j}},\ \forall (i,j) \in [\![1, m]\!]^2, \forall p \in \mathcal{P} \tag{1}$$

where m is the number of graphs. $d(g_i, g_j)^p$ is the distance obtained when matching g_i and g_j using method p while R_{g_i,g_j} corresponds to the ground truth provided in our csv files. R_{g_i,g_j} represents the best distance among all methods p for matching graphs g_i and g_j. As all comparisons are evaluated under a small or a big C_T, R_{g_i,g_j} does not necessarily have to be the optimal distance. In other words, R_{g_i,g_j} represents the best solution found under a small or a big C_T. This solution could be optimal when C_T is reasonable to solve the given matching problem. For each method p, the mean $dev(g_i, g_j)^p$ is computed.

Matching Dissimilarity: Let EP refer to vertex to vertex mappings between g_1 into g_2. We aim at finding how dissimilar are two EPs (*i.e.,* EP_p and EP_q) that correspond to matching g_i and g_j using methods p and q. Thus, the idea here is to see how far an EP from the ground truth EP integrated in our repository. To this objective, we count the differences between the two solutions, we exclude edge correspondences in order to only concentrate on vertex correspondences.

Mathematically saying, the distance between EP_p and EP_q is defined as:

$$\mathrm{d}(EP_p, EP_q) = \sum_{i=1}^{n}\sum_{j=1}^{m}\delta(EP_p(i), EP_q(j)) \tag{2}$$

where $n = |EP_p|$, $m = |EP_q|$ and $\delta(EP_p(i), EP_q(j))$ is the well-known Kronecker Delta function:

$$\delta(EP_p(i), EP_q(j)) = \begin{cases} 0, & \text{if } EP_p(i) = EP_q(j) \\ 1, & \text{if } EP_p(i) \neq EP_q(j) \end{cases} \tag{3}$$

Number of Explored Nodes: We propose to measure the number of explored nodes in the tree search for each comparison $d(g_i, g_j)$. This number represents the number of moves in the search tree needed to obtain the best found solution. The mean number of explored nodes in the tree search is calculated as follows:

$$\overline{\#explored\_nodes}_p = \frac{1}{m \times m}\sum_{i=1}^{m}\sum_{j=1}^{m} expnd(d(g_i, g_j)^p) \tag{4}$$

where $m \times m$ is the total number of comparisons per subset, m is the number of graphs to be matched and $expnd(d(g_i, g_j)^p)$ is the number of explored nodes obtained when matching graphs g_i and g_j of subset s using method p. Note that p has to be a tree-search based algorithm (*e.g., A∗* [5] and BeamSearch [21]).

Number of Best Found Solutions: For each subset, we count the number of times the best solution is found by method p. This number indicates that method p was able to find the best solution, not necessarily the optimal one. The solution is supposed to be the best one when compared to all the involved methods.

Number of Unfeasible Solutions: For each method p, we measure the number of times method m was not able to find an *EP*. This case can happen when C_T or C_M is violated before finding a complete solution. (*i.e.,* when there is only incomplete matching correspondences). Lower bound methods [8] always give unfeasible solutions as they only output distances without matching sequences.

Number of Time-Out and Out-Of-Memory Cases: For each subset, we count the number of times method p violates C_T and C_M respectively.

Running Time: We measure the overall time in milliseconds (ms), for each GED computation, including all the inherits costs computations. The mean running time is calculated per subset s and for each method p.

Time-Deviation Scores: To sum up advantages and drawbacks of each method p, a projection of p on a two-dimensional space ($\mathbb{R}^2$) is achieved by using *speed-score* and *deviation-score* features defined in equations 7 and 8 where speed and

deviation are two concurrent criteria to be minimized. First, for each database, the mean deviation and the mean time is derived as follows:

$$\overline{dev^p} = \frac{1}{m \times m} \sum_{i=1}^{m} \sum_{j=1}^{m} dev(g_i, g_j)^p \tag{5}$$

$$\overline{time^p} = \frac{1}{m \times m} \sum_{i=1}^{m} \sum_{j=1}^{m} time(g_i, g_j)^p \; and \; (i,j) \in [\![1, m]\!]^2 \tag{6}$$

where $dev(g_i, g_j)$ is the deviation of each $d(g_i, g_j)$ and $time(g_i, g_j)$ is the running time of each $d(g_i, g_j)$. To obtain comparable results between databases, mean deviations and times are normalized between 0 and 1 as follows:

$$\overline{deviation\_score^p} = \frac{1}{\#subsets} \sum_{i=1}^{\#subsets} \frac{\overline{dev_i^p}}{max\_dev_i} \tag{7}$$

$$\overline{speed\_score^p} = \frac{1}{\#subsets} \sum_{i=1}^{\#subsets} \frac{\overline{time_i^p}}{max\_time_i} \tag{8}$$

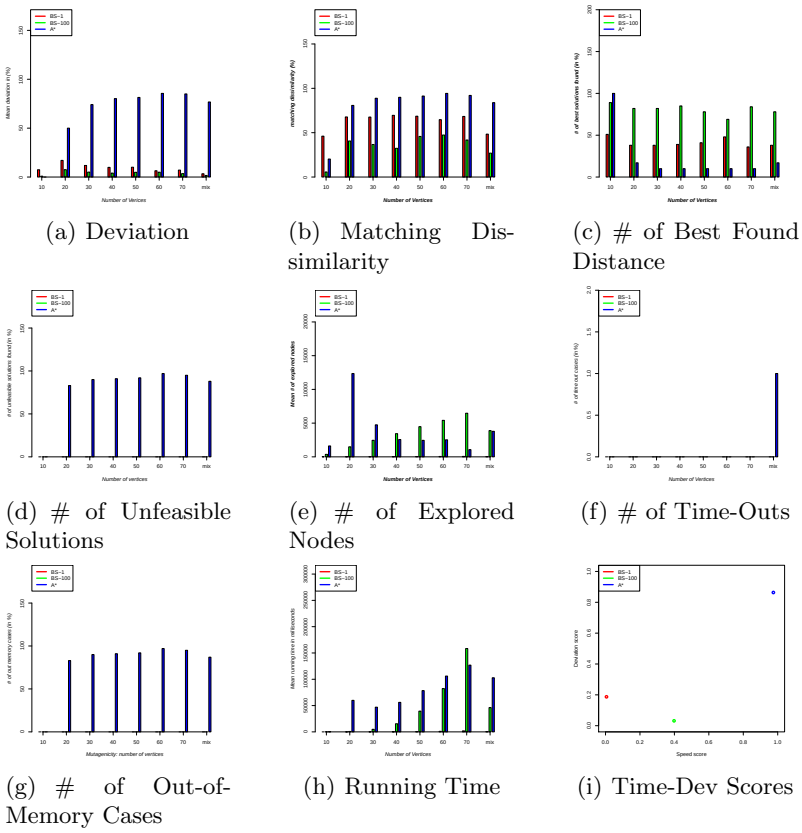
where $max\_dev_i = \max(\overline{dev_i^p})$ and $max\_time_i = \max(\overline{time_i^p})$ $\forall p \in \mathcal{P}$

5 Use Case

In this section, we demonstrate a use case of the aforementioned metrics. Two GED methods were evaluated. On the exact method side, the A^* algorithm applied to GED problem is a foundation work [5]. It is the most well-known exact method that is often used to evaluate the accuracy of approximate methods. On the approximate method side, the truncated version of A^*, *i.e.,* Beam Search (*BS-x*), was chosen. This method is one of the most accurate heuristics of the literature. In this use case, x in *BS-x* has been set to 1 and 100, respectively.

In this paper, we only provide results on the aforementioned methods. The results on the other databases and other methods are demonstrated in our website. Also, due to the large number of matchings considered and the exponential complexity of the tested methods, C_T has been set to 300 seconds which is large enough to let the methods search deeply into the solution space and to ensure that many search tree nodes will be explored. C_M has been set to 1Gb.

Figure 1(a) represents the deviation from the best distances integrated in our repository. Figure 1(b) points out the matching dissimilarity when comparing $d(g_i, g_j)^p$ with the best matching in our repository. Both figures reveal that *BS-100* had the least deviation. Similarly, Figure 1(c) shows that *BS-100* had the highest number of best found solutions. In average, *BS-100* beat the other methods with 38 more best solutions. $A*$ suffered from high memory consumption, see Figure 1(g) and thus it outputted unfeasible solutions, as demonstrated in Figure 1(d). In average, on MUTA-40, MUTA-50, MUTA-60 and MUTA-70, *BS-100* explored around 4043 search nodes more than the other methods. Figure 1(h) shows that, in average, *BS-1* was the fastest algorithm. However, according to Figure 1(i), *BS-100* gave the best trade-off between deviation and speed.



(a) Deviation (b) Matching Dissimilarity (c) # of Best Found Distance

(d) # of Unfeasible Solutions (e) # of Explored Nodes (f) # of Time-Outs

(g) # of Out-of-Memory Cases (h) Running Time (i) Time-Dev Scores

Fig. 1. Results on MUTA and under a reasonable time constraint ($C_T = 300$ seconds).

6 Conclusions

The contribution of this paper is two-fold: First, additional low level annotation has been made publicly available for some representative graph databases. Each graph comparison is coupled with the best solution found by a GED method. Four databases with different characteristics are integrated (GREC, MUTA, Protein and CMU). We have proposed to evaluate the scalability of GED methods on GREC, MUTA and Protein as they have graphs of different number of vertices. Thus, these databases were decomposed into subsets, each of which has graphs whose number of vertices are equal. On the other hand, CMU has geometric graphs of 30 vertices, each of which has been subjected to rotations. Second, this paper has presented performance evaluation metrics that assess GED methods under time and memory constraints. The aim of this paper is to make GED methods better comparable against each other. For that reason, we highly encourage the community not only to use the information provided in the

repository, but also to integrate their algorithms' answers when obtaining more accurate results.

In future work, we will further expand this repository by integrating other publicly available databases.

References

1. M. Vento, A long trip in the charming world of graphs for pattern recognition, Pattern Recognition 48 (2) (2015) 291–301.
2. S. Sorlin, C. Solnon, J. michel Jolion, A generic graph distance measure based on multivalent matchings (2007).
3. A. Sanfeliu, K. Fu, A distance measure between attributed relational graphs for pattern recognition, IEEE Transactions on S, M, and C 13 (1983) 353–362.
4. H. Bunke, Inexact graph matching for structural pattern recognition, Pattern Recognition Letters 1 (4) (1983) 245–253.
5. S. Fankhauser, et al., Speeding up graph edit distance computation with a bipartite heuristic, no. 6658, 2011, pp. 102–111.
6. D. Justice, A. Hero, A binary linear programming formulation of the graph edit distance, IEEE Transactions on PA and MI 28 (8) (2006) 1200–1214.
7. B. H. Riesen, K., Approximate graph edit distance computation by means of bipartite graph matching., Image and Vision Computing. 28 (2009) 950–959.
8. A. Fischer, et al., A fast matching algorithm for graph-based handwriting recognition, GbRPR (2013) 194–203.
9. K. Riesen, H. Bunke, Iam graph database repository for graph based pattern recognition and machine learning (2008) 287–297.
10. K. B. Xu, Benchmarks with hidden optimum solutions for graph problems. URL `http://www.nlsde.buaa.edu.cn/kexu/benchmarks/graph-benchmarks.htm`
11. Cmu house and hotel datasets, http://vasc.ri.cmu.edu/idb/html/motion.
12. A large database of graphs and its use for benchmarking graph isomorphism algorithms 24 (8) (2003) 1067 – 1079.
13. D. Conte, et al., Challenging complexity of maximum common subgraph detection algorithms 11 (1) (2007) 99–143.
14. P. Foggia, et al., A performance comparison of five algorithms for graph isomorphism, in: IAPR TC-15, 2001, pp. 188–199.
15. V. Carletti, P. Foggia, M. Vento, Performance comparison of five exact graph matching algorithms on biological databases., Vol. 8158, 2013, pp. 409–417.
16. Grec competition, http://symbcontestgrec05.loria.fr/index.php.
17. K. Riesen, H. Bunke, Graph Classification and Clustering Based on Vector Space Embedding, 2010.
18. J. Kazius, et al., Derivation and validation of toxicophores for mutagenicity prediction, Journal of Medicinal Chemistry 48(1) (2005) 312–20.
19. Wagner, et al., The string-to-string correction problem 21 (1) (1974) 168–173.
20. F. Zhou, F. De la Torre, Factorized graph matching, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.
21. M. Neuhaus, et al., Fast suboptimal algorithms for the computation of graph edit distance, Structural and Syntactic Pattern Recognition (2006) 163–172.