SN Computer Science (2021) 2:438
https://doi.org/10.1007/s42979-021-00792-5

ORIGINAL RESEARCH

Redefining the Graph Edit Distance

Francesc Serratosa[1]

Received: 17 December 2020 / Accepted: 22 July 2021 / Published online: 31 August 2021

Abstract
Graph edit distance has been used since 1983 to compare objects in machine learning when these objects are represented by attributed graphs instead of vectors. In these cases, the graph edit distance is usually applied to deduce a distance between attributed graphs. This distance is defined as the minimum amount of edit operations (deletion, insertion and substitution of nodes and edges) needed to transform a graph into another. Since now, it has been stated that the distance properties have to be applied [(1) non-negativity (2) symmetry (3) identity and (4) triangle inequality] to the involved edit operations in the process of computing the graph edit distance to make the graph edit distance a metric. In this paper, we show that there is no need to impose the triangle inequality in each edit operation. This is an important finding since in pattern recognition applications, the classification ratio usually maximizes in the edit operation combinations (deletion, insertion and substitution of nodes and edges) that the triangle inequality is not fulfilled.

Keywords Distance definition · Graph edit distance · Attributed graphs · Learning

Introduction

Several kinds of problems have been used to model attributed graphs for more than 4 decades [1–3]. This is because they have the flexible ability to define an element through their structural and semantic information. Some reviews have been published through these years [4–7]. Figure 1 shows several examples of representing an element through an attributed graph. Applications range from optical character recognition to chemical compound toxicity analysis. Other examples are image processing, text analysis or geolocalization. If elements in pattern recognition are modelled through attributed graphs, a distance between them is needed since, in pattern recognition, comparing elements is an essential task. One of the most widely used distances between attributed graphs is the graph edit distance [8–11].

Graph edit distance depends on some parameters, which have to be tuned to maximize an objective function, which is related on the task to be carried out. In the earliest presentations of the graph edit distance in 1983, [1, 2], this function was considered a metric subject to certain properties on these parameters. A more contemporary book published in 2015 [12] also suggests the same properties. That book analyses some pattern recognition applications and several algorithms that compute the graph edit distance in an optimal and sub-optimal manner.

As an example of how important is to define the graph edit distance as a metric, we could mention the graph databases structured by metric-trees [13]. In this case, if the function to compare attributed graphs is not a metric, some attributed graphs could not be retrieved from the database although they had properly been introduced. Thus, the system performs as these graphs have not never been introduced in the database.

Another example are the pattern recognition methods that compact a set of attributed graphs that have been considered to belong to one class into only one graph representative [14–16]. The aim of this type of representations is twofold. First, they want to reduce the runtime of deducing the class of an element to be classified. This is because in these methods, only one comparison is needed between this unclassified graph and each class in the set. In contrast, in other methods, such as the k-nearest neighbours [17], the number of comparisons needed is the number of graphs in the set. Second, they want to increase the representational power of the set. This is because, by compacting the general features of the set, the local discrepancies between the elements in

✉ Francesc Serratosa
francesc.serratosa@urv.cat
http://deim.urv.cat/~francesc.serratosa/

[1] Universitat Rovira i Virgili, Tarragona, Catalonia, Spain

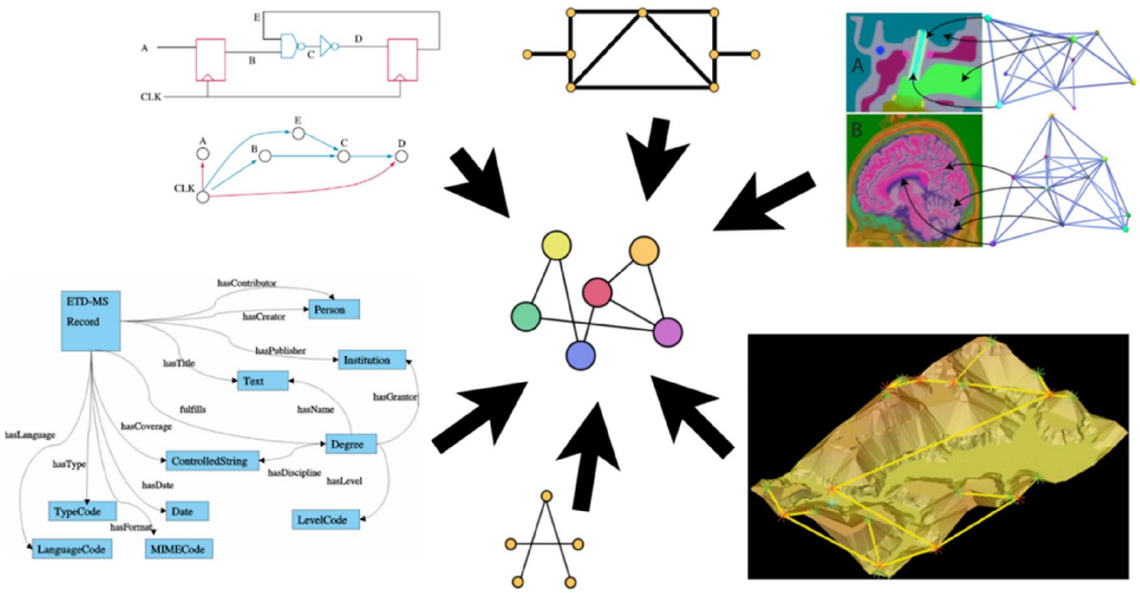

Fig. 1 Representing objects in pattern recognition applications by graphs with attributes on their nodes and edges

the set tend to be softened. In this case, if the function to compare attributed graphs is not a metric, the representative graph is not properly defined (the construction of the representative is carried out by several comparisons between the graphs in the set) and thus, there is a quality reduction of the pattern recognition task.

For this reason, in the previously commented examples, the graph edit distance parameters are set such as the graph edit distance becomes a metric. If we could somehow relax some of the distance properties while maintaining the graph edit distance a metric, we could extend the parameters' domain and thus, set them at a value where the query precision becomes higher (in the first example) and the representative power of the set also becomes higher (in the second example). Note we are aware that the dissimilarity between patterns in other pattern recognition applications can be defined as non-metric. Finally, the configuration of the graph edit distance parameters considering some matching algorithms is analysed in [18]. That paper demonstrates that only some graph matching algorithms return a metric although some distance properties are not fulfilled. Thus, it concludes that the selection of the graph matching is decisive not only for the run time point of view but also to assure the deduced graph edit distance between the graphs in the application is a metric.

To summarize, in this paper, we show that some properties usually considered in the graph edit distance [1, 2, 12] can be relaxed while the graph edit distance being a metric. This simplification is central in several pattern recognition applications. This is because all these applications have an objective function to be maximized (or minimized), for instance, the maximization of the recognition ratio or classification accuracy, between others. Thus, it turns out that several machine learning methods have been presented [19–23] to automatically deduce the graph edit distance parameters and, frequently, the objective function of these methods is maximized where these properties do not hold. Due to the findings of this paper, we currently know that the automatically learned graph edit distance parameters makes the graph edit distance to be a metric. The experimental section shows the classification ratio given some graph edit distance parameters and also shows that the maximum classification is achieved at the parameters that we currently can assure the graph edit distance is a metric.

Note that several algorithms have been presented to compute the graph edit distance. This is because its computation has been demonstrated to be NP complete [24]. Basically, there are algorithms that return the exact distance in exponential computational cost [25] and other ones that return an approximation of this value in polynomial computational cost [26–28]. The fact of using different edit costs do not influence over the computational cost but only on the algorithm per se. For instance, the computational cost of [25] is exponential, the computational cost of [26–28] is cubic and the computational cost of [29] is linear with respect to the number of nodes as it can be deduced in [30, 31]. For this reason, in this paper, the experimental section does not show a comparison between these methods but only shows that the edit cost that return the highest classification rations do not fulfil the triangle inequality. For a specific comparison of graph matching methods, we refer to [32].

Graph Edit Distance

A **distance** is a function that holds the following properties [33]:

1) $\text{dist}(x, y) \geq 0$. Non-negativity.
2) $\text{dist}(x, y) = \text{dist}(y, x)$. Symmetry.
3) $\text{dist}(x, y) = 0 \Leftrightarrow x = y$. Identity of indiscernible elements :
4) $\text{dist}(x, y) \leq \text{dist}(x, z) + \text{dist}(z, y)$. Triangle inequality, (1)

where x, y and z are elements in a specific domain. The distance domain is $[0, \infty)$.

An attributed graph $G = (V, E)$ is composed of a set of n nodes $V = \{v_1, \ldots, v_n\}$ and a set of m edges $E = \{e_{1,2}, \ldots, e_{t,k}\}$. Moreover, there is a function on the nodes, γ_v, and on the edges, γ_e, which assign an attribute on these nodes and edges. The **graph edit distance** between two attributed graphs, *GED*, G and G' is defined as,

$$\text{GED}(G, G') = \min_{\forall (e_1, e_2, \ldots e_k) \in E(G, G')} \left\{ \underset{(e_1, e_2, \ldots e_k)}{\text{CED}}(G, G') \right\} \tag{2}$$

where $E(G, G')$ denotes the set of all edit paths $(e_1, e_2, \ldots e_k)$ that transform G into G' and the cost of the edit path is CED,

$$\begin{aligned} \underset{(e_1, e_2, \ldots e_k)}{\text{CED}}(G, G') = & \sum_{\forall e_t \in \text{vs}} C_{\text{vs}}(e_t) + \sum_{\forall e_t \in \text{es}} C_{\text{es}}(e_t) + \sum_{\forall e_t \in \text{vd}} C_{\text{vd}}(e_t) \\ & + \sum_{\forall e_t \in ed} C_{\text{ed}}(e_t) + \sum_{\forall e_t \in vi} C_{\text{vi}}(e_t) + \sum_{\forall e_t \in ei} C_{\text{ei}}(e_t), \end{aligned} \tag{3}$$

The set of node substitutions, deletions and insertions is represented by vs, vd and vi. Moreover, the costs of these edit operations are represented by C_{vs}, C_{vd} and C_{vi}. Edit operations and costs on edges have similar definitions: es, ed, ei, C_{es} C_{ed} and C_{ei}.

Figure 2 shows a toy example of a graph edit distance between two simple graphs, which are composed of only two nodes and an edge between them. In this case, the transformation between both graphs is carried out by five edit operations: ed, vd, vi, ei and vs. Note that several edit paths transform one graph into the other one that generate different costs (Eq. 3). The edit path that is considered in the graph edit distance is the one that achieves the minimum cost through the previously defined $C_{\text{vs}}, C_{\text{vd}}, C_{\text{vi}}, C_{\text{es}}$ C_{ed} and C_{ei} (Eq. 2).

The edit operations e_t depend on the attributes on the nodes or on the arcs. For this reason, we define, for the rest of the paper, the following relations:

If $e_t \in vs$, then $e_t = [v_a, v_i']$. $v_a \in G$ and $v_i' \in G'$.
If $e_t \in es$, then $e_t = [e_{ab}, e_{ij}']$. $e_{ab} \in G$ and $e_{ij}' \in G'$.

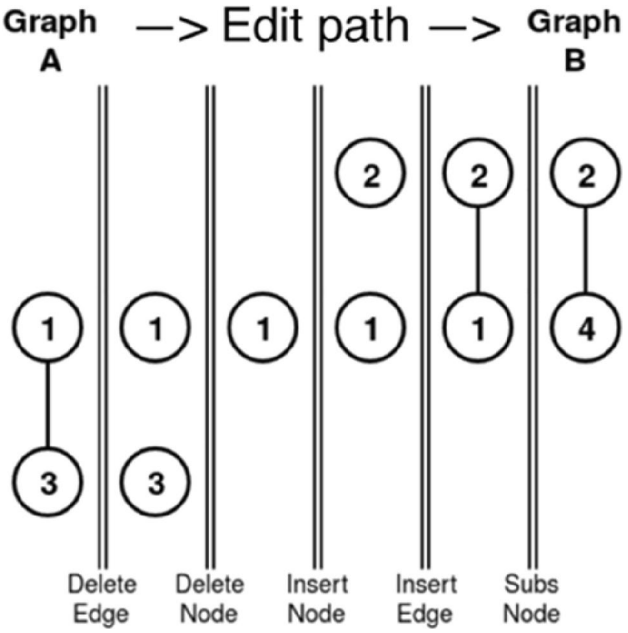

Fig. 2 An example of an edit path composed of five edit operations between two graphs. Graph A is composed of nodes 1 and 3 and an edge. Graph B is composed of nodes 2 and 4 and an edge

If $e_t \in vd$, then $e_t = [v_a]$. $v_a \in G$.
If $e_t \in ed$, then $e_t = [e_{ab}]$.$e_{ab} \in G$.
If $e_t \in vi$, then $e_t = [v_i']$. $v_i' \in G'$.
If $e_t \in ei$, then $e_t = [e_{ij}']$.$e_{ij}' \in G'$.

The distance properties (Eq. 1) are taken into consideration in the first definition of the graph edit distance [1] through the edit operations (Lemma 5 in [1]) and they are rewritten (Sect. 2.1.1 in [12]). Authors propose that if the edit operations fulfil the metric properties (Eq. 1) in their domain, then, it is clear that the graph edit distance is a metric. This is due to the graph edit distance is a linear function of the edit operations. In [1, 12], the distance properties are written in the following three points. Note the nomenclature has been slightly changed with regard to [1]. Besides, we write $C_{\cdot}(\cdot)$ instead of $C_{\cdot}([\cdot])$.

(1)

A. $C_{\text{vs}}(v_a, v_i') > 0$ if $\gamma_v(v_a) \neq \gamma_v'(v_i')$.
$C_{\text{vs}}(v_a, v_i') = 0$ otherwise.
B. $C_{\text{es}}(e_{ab}, e_{ij}') > 0$ if $\gamma_e(e_{ab}) \neq \gamma_v'(e_{ij}')$.
$C_{\text{es}}(e_{ab}, e_{ij}') = 0$ otherwise.
C. $C_{\text{vd}}(v_a, v_i') > 0$.
D. $C_{\text{ed}}(e_{ab}, e_{ij}') > 0$.
E. $C_{\text{vi}}(v_a, v_i') > 0$.
F. $C_{\text{ei}}(e_{ab}, e_{ij}') > 0$.

(2)

A. $C_{\mathrm{vs}}(v_a, v_i') = C_{\mathrm{vs}}(v_i', v_a)$.
B. $C_{\mathrm{es}}(e_{ab}, e_{ij}') = C_{\mathrm{es}}(e_{ij}', e_{ab})$.
C. $C_{\mathrm{vd}}(v_a) = C_{\mathrm{vi}}(v_i')$.
D. $C_{\mathrm{ed}}(e_{ab}) = C_{\mathrm{ei}}(e_{ij}')$.

(3)

A. $C_{\mathrm{vs}}(v_a, v_i') \leq C_{\mathrm{vs}}(v_a, v_j'') + C_{\mathrm{vs}}(v_j'', v_i')$.
B. $C_{\mathrm{es}}(e_{ab}, e_{ij}') \leq C_{\mathrm{es}}(e_{ab}, e_{kl}'') + C_{\mathrm{es}}(e_{kl}'', e_{ij}')$.
C. $C_{\mathrm{vs}}(v_a, v_i') \leq C_{\mathrm{vd}}(v_a) + C_{\mathrm{vi}}(v_i')$.
D. $C_{\mathrm{es}}(e_{ab}, e_{ij}') \leq C_{\mathrm{ed}}(e_{ab}) + C_{\mathrm{ei}}(e_{ij}')$.

Theorem 1, defined below, demonstrates that the previously defined point 3.C and point 3.D (the ones that relate the triangle inequality to the deletion, insertion and substitution costs) do not need to be imposed to properly define the graph edit distance. We also comment how this finding influence over the string and tree edit distance.

Theorem 1

The GED is a metric if points 1.A,…,1.F, 2.A,…,2.D, 3.A and 3.B are fulfilled (Points 3.C and 3.D are not need to be fulfilled).

Proof It is trivial to realise that the non-negativity: $\mathrm{GED}(G, G') \geq 0$; the identity of indiscernible elements: $\mathrm{GED}(G, G') = 0 \Leftrightarrow G = G'$; and the symmetry properties: $\mathrm{GED}(G, G') = GED(G', G)$ are fulfilled in the graph edit distance in the case that points 1. and 2. are fulfilled.

Thus, we have to prove the triangle inequality, which means that $GED(G, G') \leq \mathrm{GED}(G, G'') + \mathrm{GED}(G'', G')$ holds. We assume $e_{G,G'} \in E(G, G')$, $e_{G,G''} \in E(G, G'')$ and $e_{G'',G'} \in E(G'', G')$ are three edit path that make the minimum cost CED in $E(G, G')$, $E(G, G'')$ and $E(G'', G')$, respectively. In other words, the ones used to compute $\mathrm{GED}(G, G')$, $\mathrm{GED}(G, G'')$ and $\mathrm{GED}(G'', G')$. We define an edit path $\hat{e}_{G,G'}$ that transforms G into G' as a concatenation of $e_{G,G''}$ and $e_{G'',G'}$. By construction, it holds that $\mathrm{GED}(G, G') = \mathrm{CED}(G, G'') + \mathrm{CED}(G'', G')$ and by definition of $\mathrm{CED}(G, G')$, it holds that $\mathrm{GED}(G, G') \leq \mathrm{CED}(G, G')$. Therefore, $\mathrm{GED}(G, G') \leq \mathrm{CED}(G, G') = \mathrm{GED}(G, G'') + \mathrm{GED}(G'', G')$. □

Strings and trees can be modelled as graphs. In the case of strings, each element in the string is represented as a node in the graph and the order of the string elements is represented through edges in the graph that connect the string elements that are placed side by side. In the case of trees, nodes in the trees are represented as nodes in the graphs and the father—sibling relation is represented as an edge in the graph from the father node to the sibling node. For this reason, Theorem 1 can be applied to strings and trees and thus, we have demonstrated that the tree edit distance and the string edit distance do not need to fulfil the triangle inequality between their edit operations to be considered a metric.

Practical Experiment

The aim of the paper is to show that the triangle inequality applied to the edit operations is not needed to be the graph edit distance a metric. Thus, it could be presented without practical evaluation. Nevertheless, we included this evaluation to show that it is usual to have the optimal configuration of the parameters at the point that this triangle inequality is not fulfilled, and therefore, it is worth this demonstration. Thus, in this section, we show that the highest classification ratios appear in the values of the edit operations, such that the triangle inequality between deletion and insertion operations and substitution operation is not fulfilled. Yet, these values were rarely applied since the graph edit distance was not considered a metric. We observe that the algorithms, such as [34], designed to learn the edit operations, do not intrinsically take into consideration fulfilling the triangle inequality properties but they do, for the other distance properties. Thanks to Theorem 1, we know that the graph edit distance continues to be a metric and therefore it is worth setting the edit operations in the values such that the recognition ratio is maximized (even if the triangle inequality on the edit operations is not fulfilled).

The following five public graph databases have been used: Coil-Rag, Grec, Letter-low, Letter-med and Letter-high. These databases are explained in [32, 35], and their most important features and details are summarized in Table 1. For instance, the number of graphs and classes, the average number of nodes and edges per graph and the number of attributes on nodes.

Table 2 shows the minimum, maximum, mean and standard deviation of the node substitution costs of these databases. Given that the insertion and deletion edit operations are constants, we have to assure that these constants are set such that $C_{\mathrm{vd}} + C_{\mathrm{vi}} \geq \max(C_{\mathrm{vs}})$, to fulfil the triangle inequality on the edit operations. Hence, the minimum value of C_{vd} and C_{vi} would have to be $C_{\mathrm{vd}} = C_{\mathrm{vi}} = 0.5 \max(C_{\mathrm{vs}})$. The last column shows $0.5 \max(C_{\mathrm{vs}})$ per each database.

Given these databases, we have computed the classification ratio as follows. We used the 1-nearest neighbour classification algorithm [17] for the classification purposes. Moreover, we have used the fast bipartite graph matching to deduce the graph edit distance between graphs [26, 27].

Table 1 Main features of the testing and learning sets of the databases

Database	Number of graphs	Number of classes	Average number of nodes	Average number of edges	Number of attributes
Coil-Rag	1000	100	3.01	6	64
Grec	528	22	11.4	23.8	2
Letter Low	750	15	4.6	6.2	2
Letter Med	750	15	4.6	6.4	2
Letter High	750	15	4.6	9	2

Table 2 Different statistics of the node substitution costs C_{vs}

	Min	Max	Mean	Std. Dev	½ Max
Coil-Rag	0	1.3	0.8	0.36	0.69
Grec	0	823	246.5	128	411
Letter-low	0	4.09	1.6	0.82	2.04
Letter-med	0	4.09	1.6	0.82	2.04
Letter-high	0	4.95	1.6	0.83	2.46

The edit operations are set as follows: C_{vs} is the Euclidean distance between node attributes.C_{es} is not defined (these databases do not have attributes on the edges). Finally, we impose the insertion and deletion operations on nodes and edges to have the same value and to be the constant K_v: $K_v = C_{vd} = C_{ed} = C_{vi} = C_{ei}$. We first deduced the class of each graph in the test set. Then, the classification ratio is computed as the number of times the graphs in the test set are properly classified, divided by the number of graphs in the test set. To obtain the class of a graph in the test set, it is needed to compute the graph edit distance between this graph and all the graphs in the learning set. We conclude the class of the graph in the test set is the class of the graph in the learning set that has returns the minimum distance.

Figure 3 shows the classification ratio in the vertical axis and the constant K_v in the horizontal axis. Each value in the plot represents the classification ratio when the graph edit distance has been computed considering K_v as the deletion and insertion costs on nodes and edges. The range of values of K_v has been set from the minimum value of the substitution costs (first column in Table 2) to the maximum value of the substitution costs (second column in Table 2). Moreover, the arrow shows the range of values such that the triangle inequality between edit costs are fulfilled. It ranges from the last column of Table 2 to infinitive. Computing each plot took approximately two hours (MacMini, I5, Matlab). We realize that the classification ratio is maximized at the value of the insertion and deletion costs that are much smaller than half of the maximum value of the substitution cost (the last column of Table 2), which is the point that the triangle inequality begins not to be fulfilled. Therefore, we conclude that it is worth knowing that in the whole range of K_v the graph edit distance is a metric although the triangle inequality on the edit operations does not hold on the lower part of the range of K_v.

Discussion and Conclusion

Considering the practical experiment, we could deduce that imposing the insertion and deletion costs (represented as K_v) to be the mean of half of the substitution cost is a good option since it is at the point where the classification ratio is maximized. Nevertheless, in the past, this setting was not used due to the graph edit distance was not considered to be a metric. This paper relaxes the properties needed to be the graph edit distance a metric and for this reason, we can confirm that at this point, the graph edit distance keeps being a metric. Thus, it is worth using these costs.

Three algorithms have been presented to automatically deduce the edit operations [19–21]. Given the topic of this paper, we observe that their automatically found edit operations (the values that maximize the classification ratio) do not fulfil the triangle inequality. Nevertheless, we currently confirm that the graph edit distance is a metric at the insertion and deletion costs optimized by those learning algorithms. Clearly, this paper does not state that in all the applications, the classification ratio is maximized in the point that the triangle inequality between edit operations is not fulfilled. We only state that in these cases, we can consider using the learned parameters since we currently know that the graph edit distance is a metric. Finally, note that the learning methods impose, by construction of the algorithms, to fulfil the non-negativity, the symmetry and the identity of the edit operations.

As a future work, we want to analyse how influences the relaxation of other distance properties, specifically, the symmetry property. Note that in the classification scenarios, we want to know the distance between a graph of the test set and a graph of the learning set. Since, the symmetric distance, that is, the distance between a graph of the learning set and a graph of the test set is never computed, it could be worth not to fulfil the symmetry property at the edit operation level. This means that the insertion costs do not have to be the same as the deletion costs on nodes or edges. To do so, we are computing some experiments similar to the ones presented in Fig. 3 but taking some values for the insertion cost that are different from the deletion cost, that is, $K_{vd} = C_{vd} \neq K_{ve} = C_{ed}$.

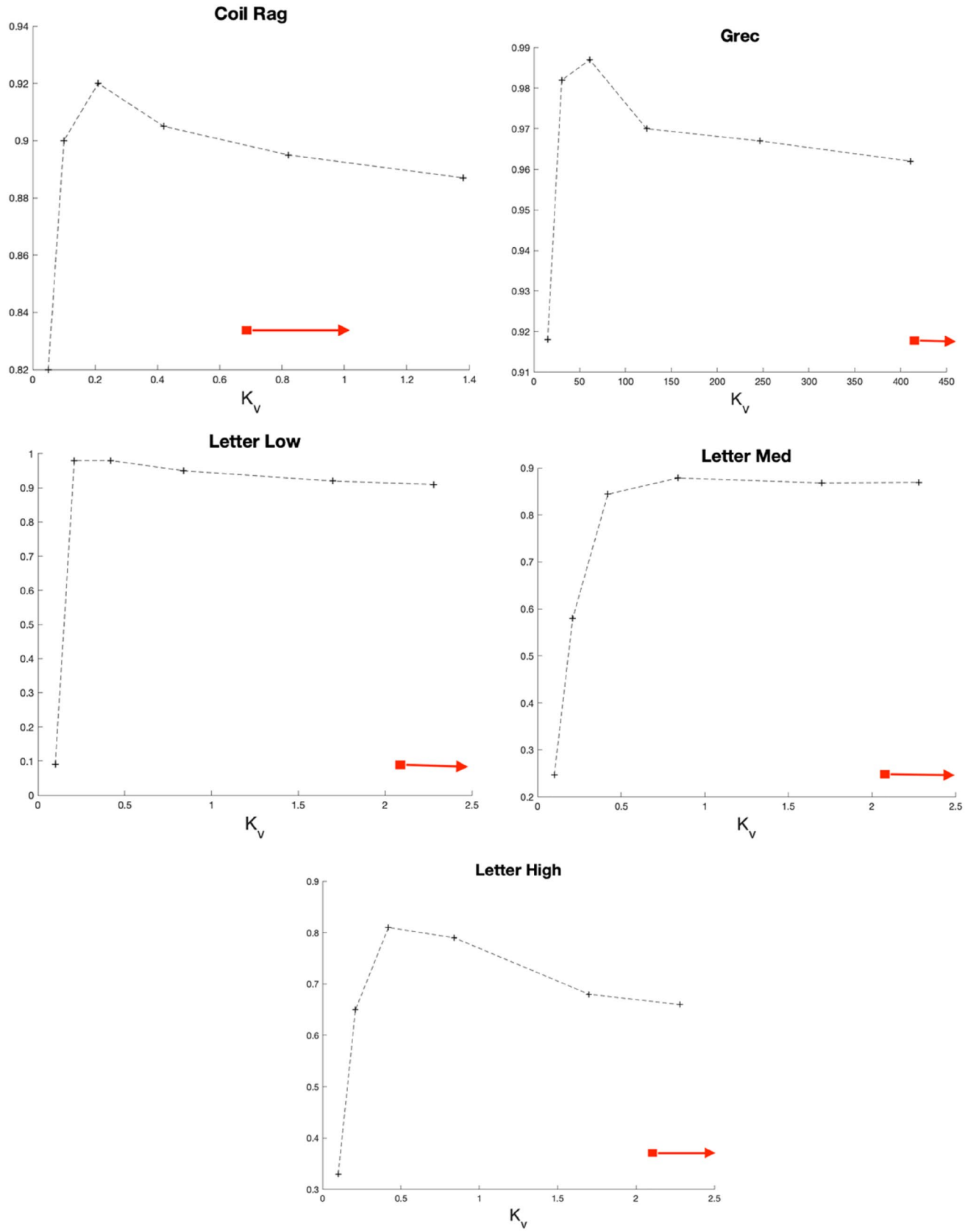

Fig. 3 Classification ratio of Coil-Rag, Grec, Letter-low, Letter-med and letter high given some values of the insertion and deletion costs K_v

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Declarations

Conflict of Interest On behalf of all authors, the author states that there is no conflict of interest.

References

1. Bunke H, Allermann G. Inexact graph matching for structural pattern recognition. Pattern Recogn Lett. 1983;1(4):245–53.
2. Sanfeliu A, Fu KS. A Distance measure between attributed relational graphs for pattern recognition. IEEE Trans Syst Man Cybern. 1983;13(3):353–62.
3. Sanfeliu A, Alquézar R, Andrade J, Climent J, Serratosa F, Vergés J. Graph-based representations and techniques for image processing and image analysis. Pattern Recogn. 2002;35(3):639–50.
4. Conte D, Foggia P, Sansone C, Vento M. Thirty years of graph matching in pattern recognition. Int J Pattern Recognit Artif Intell. 2004;18(3):265–98.
5. Vento M. A long trip in the charming world of graphs for pattern recognition. Pattern Recognit. 2015;48:291–301.
6. Livi L, Rizzi A. The graph matching problem. Pattern Anal Appl. 2013;16(3):253–83.
7. Foggia P, Percannella G, Vento M. Graph matching and learning in Pattern Recognition in the last 10 years. Int J Pattern Recognit Artif Intell. 2014;28(1):1450001 (**40 pages**).
8. Solé A, Serratosa F, Sanfeliu A. On the graph edit distance cost: properties and applications. Int J Pattern Recognit Artif Intell. 2012;26(5):1260004 (**21 pages**).
9. Serratosa F, Cortés X. Graph Edit Distance: moving from global to local structure to solve the graph-matching problem. Pattern Recogn Lett. 2015;65:204–10.
10. Gao X, Xiao B, Tao D, Li X. A survey of graph edit distance. Pattern Anal Appl. 2010;13(1):113–29.
11. Serratosa F. A general model to define the substitution, insertion and deletion graph edit costs based on an embedded space. Pattern Recogn Lett. 2020;138:115–22.
12. Riesen K. Structural pattern recognition with graph edit distance. Approximation algorithms and applications. Springer; 2015.
13. Serratosa F, Cortés X, Solé-Ribalta A. Component retrieval based on a database of graphs for hand-written electronic-scheme digitalisation. Expert Syst Appl. 2013;40:2493–502.
14. Sanfeliu A, Serratosa F, Alquézar R. Second-order random graphs for modelling sets of attributed graphs and their application to object learning and recognition. Int J Pattern Recognit Artif Intell. 2004;18(3):375–96.
15. Serratosa F, Alquézar R, Sanfeliu A. Function-Described Graphs for modelling objects represented by attributed graphs. Pattern Recogn. 2003;36(3):781–98.
16. Serratosa F, Alquézar R, Sanfeliu A. Synthesis of function-described graphs and clustering of attributed graphs. Int J Pattern Recognit Artif Intell. 2002;16(6):621–55.
17. Cover TM, Hart PE. Nearest neighbours pattern classification. IEEE Trans Inf Theory. 1967;13(1):21–7.
18. Serratosa F. Graph edit distance: restrictions to be a metric. Pattern Recogn. 2019;90:250–6.
19. Cortés X, Serratosa F. Learning graph matching substitution weights based on the ground truth node correspondence. Int J Pattern Recognit Artif Intell. 2016;30(2):1650005 (**22 pages**).
20. Cortés X, Serratosa F. Learning graph-matching edit-costs based on the optimality of the oracle's node correspondences. Pattern Recogn Lett. 2015;56:22–9.
21. Algabli S, Serratosa F. Embedding the node-to-node mappings to learn the Graph edit distance parameters. Pattern Recogn Lett. 2018;112:353–60.
22. Santacruz P, Serratosa F. Learning the graph edit costs based on a learning model applied to sub-optimal graph matching. Neural Process Lett. 2020;51:881–904.
23. Conte D, Serratosa F. Interactive online learning for graph matching using active strategies. Knowl Based Syst. 2020;105:106275.
24. Garey M, Johnson D. Computers and intractability: a guide to the theory of NP-completeness. Siam Rev. 1979;24:90.
25. Ferrer M, Serratosa F, Riesen K. Improving Bipartite graph matching by assessing the assignment confidence. Pattern Recogn Lett. 2015;65:29–36.
26. Serratosa F. Fast computation of bipartite graph matching. Pattern Recogn Lett. 2014;45:244–50.
27. Serratosa F. Speeding up Fast bipartite Graph Matching trough a new cost matrix. Int J Pattern Recognit Artif Intell. 2015;29(2):1550010 (**17 pages**).
28. Serratosa F. Computation of graph edit distance: reasoning about optimality and speed-up. Image Vis Comput. 2015;40:38–48.
29. Santacruz P, Serratosa F. Error-tolerant graph matching in linear computational cost using an initial small partial matching. Pattern Recognit Lett. 2020;134:10–9.
30. Hart P, Nilsson N, Raphael B. A formal basis for the heuristic determination of minimum cost paths. Trans Syst Sci Cybern. 1968;4(2):100–7.
31. Kuhn HW. The Hungarian method for the assignment problem. Nav Res Log Q. 1955;2:83–97.
32. Moreno-Garcia C, Cortés X, Serratosa F. A graph repository for learning error-tolerant graph matching. Syn Struct Pattern Recognit SSPR2016 LNCS. 2016;10029:519–29.
33. Arkhangel'skii AV, Pontryagin LS. General Topology I: basic concepts and constructions dimension theory, encyclopaedia of mathematical sciences. Springer; 1990.
34. Neuhaus M, Bunke H. Self-organizing maps for learning the edit costs in graph matching. IEEE Trans Syst Man Cyber Part B (Cybernetics). 2005;35(3):503–14.
35. Riesen K, Bunke H. IAM graph database repository for graph based pattern recognition and machine learning. In: Structural Syntactic and Statistical Pattern Recognition. Lecture Notes in Computer Science book series (LNCS, volume 5342); 2008, p. 287–97.