On the Correlation of Graph Edit Distance and L_1 Distance in the Attribute Statistics Embedding Space

Jaume Gibert[1], Ernest Valveny[1], Horst Bunke[2] and Alicia Fornés[1]

[1] Computer Vision Center, Universitat Autònoma de Barcelona
Edifici O Campus UAB, 08193 Bellaterra, Spain
{jgibert,ernest}@cvc.uab.es
[2] Institute for Computer Science and Applied Mathematics, University of Bern,
Neubrückstrasse 10, CH-3012 Bern, Switzerland
bunke@iam.unibe.ch

Abstract. Graph embeddings in vector spaces aim at assigning a pattern vector to every graph so that the problems of graph classification and clustering can be solved by using data processing algorithms originally developed for statistical feature vectors. An important requirement graph features should fulfil is that they reproduce as much as possible the properties among objects in the graph domain. In particular, it is usually desired that distances between pairs of graphs in the graph domain closely resemble those between their corresponding vectorial representations. In this work, we analyse relations between the edit distance in the graph domain and the L_1 distance of the attribute statistics based embedding, for which good classification performance has been reported on various datasets. We show that there is actually a high correlation between the two kinds of distances provided that the corresponding parameter values that account for balancing the weight between node and edge based features are properly selected.

1 Introduction

The comparison of relational structures has been widely studied over the past years [3]. Graph edit distance constitutes a major paradigm due to its ability to handle arbitrary graph structures [2,10]. It is defined as the minimum amount of distortion that is needed to transform one graph into another. This distance measure is very intuitive in nature since the edit path it looks for is based on substituting, deleting and inserting nodes and edges such that the source and the target graph become isomorphic.

Graph edit distance, however, has a high computational complexity. Modern ways for graph matching try to avoid this high complexity. Extracting graph features and building up pattern vectors for the analysis of graphs —known as graph embedding— is a common way to reduce the computational complexity and make efficient learning algorithms available for the domain of graphs. A desired property of any generic graph embedding scheme is that it should be

able to approximate the original distribution of patterns in the graph domain. In other words, distances between objects in the graph domain should be similar to their corresponding distances in the embedding space. For instance, for the dissimilarity space embedding proposed in [9] it has been shown that the graph edit distance between two graphs is an upper bound of the Euclidean distance between the corresponding vectorial maps. Similarly, in [6], the Ihara coefficients have been experimentally shown to be a set of features with distances that correlate linearly with the edit distance.

In this paper, we investigate how the edit distance is related to a discrete version of the embedding methodology proposed in [5]. The features under this embedding methodology account for the number of nodes with a certain label that appear in a graph, and the number of edges with a given label that exist between two nodes with certain labels. In other words, this kind of embedding is based on occurrence and co-occurrence statistics of labels in the underlying graph. Absolute differences between node-based features indicate how many nodes with a certain label exist in one graph that are not present in the other graph. This is, in fact, exactly the same situation that occurs when performing the edit distance computation between graphs with discrete attributes under a cost function that disregards substitution of nodes with different labels and forces node deletions and insertions instead. This observation is one of the main motivations of our work.

In particular, we express both ways of computing graph distances —the edit distance and the L_1 distance for the embedding methodology— in terms of a weighting parameter balancing the impact of nodes and edges in the resulting distance values. We investigate how distances are correlated as a function of these two parameters, and also how corresponding distance-based classifiers behave.

The rest of the article is organized as follows. Graph edit distance is reviewed in the next section, and the edit cost function used for the case of discretely attributed graphs is specified. Section 3 describes the embedding methodology based on statistics of labelling information. Correlation experiments of both ways of comparing graphs and a discussion of the results are presented in Section 4. Finally, Section 5 draws conclusions from this work.

2 Graph Edit Distance

A graph $g = (V, E, \mu, \nu)$ is a 4-tuple where V is the set of nodes, $E \subseteq V \times V$ the set of edges, and $\mu : V \to L_V$ and $\nu : E \to L_E$ are the labelling functions of nodes and edges, respectively. In this work, we use undirected graphs where the labels come from finite discrete domains.

As already stated above, the main idea of graph edit distance is to define a dissimilarity measure between graphs by the minimum amount of distortion that is needed to transform one graph into the other [2,10]. Distortions are defined in terms of edit operations between two graphs, such as node and edge deletion, insertion and substitution. A sequence of edit operations transforming the source graph into the target graph is called an edit path. Edit costs define whether a

Table 1. Edit cost function.

	Deletion / Insertion	Substitution
Nodes	$c(u \to \epsilon) = c(\epsilon \to v) = 1 - \rho$	$c(u \to v) = \begin{cases} 0, & \text{if } \mu(u) = \mu(v) \\ 2 \cdot (1-\rho), & \text{otherwise} \end{cases}$
Edges	$c(e_1 \to \epsilon) = c(\epsilon \to e_2) = \rho$	$c(e_1 \to e_2) = \begin{cases} 0, & \text{if } \nu(e_1) = \nu(e_2) \\ 2 \cdot \rho, & \text{otherwise.} \end{cases}$

given operation constitutes a large deformation between the two involved graphs or not. Between similar graphs there should exist an inexpensive edit path, while dissimilar graphs are characterized by an edit path with high cost. The edit distance between two graphs is thus defined as the cost of the edit path with the minimum cost among all possible edit paths between two graphs.

The exact computation of the edit distance is a computationally hard task and many approximations have been proposed in the literature. In this work, we use the suboptimal approach of [8] where an approximate solution of graph edit distance is provided by means of solving the assignment problem of nodes of one graph to nodes of the other. A cost matrix regarding the substitution of the local structure of every node of the source graph by the local structure of every other node in the target graph is built. Then the optimal assignment is extracted by the Munkres' algorithm, and an edit path can be inferred from this assignment.

As a prerequisite, we need to assign costs to every edit operation between graph elements, i.e., nodes and edges. In particular, in this work we focus on the same cost function used in [1], where substitutions of nodes and edges with different labels are heavily penalized, forcing the (sub)optimal path to, first, delete the source node (or edge) and then insert the target node (or edge). Formally, deleting or inserting a given node (or edge) has a constant cost c, while substituting it has at least twice that cost if the corresponding labels are different. Without loss of generality, we set $c = 1$. Furthermore, we assume null cost of substituting two nodes (or edges) with the same label. In order to weight the node operations against those on the edges we introduce a parameter $\rho \in [0, 1]$ and multiply the node costs by $1 - \rho$ and the edge costs by ρ. The resulting cost function is summarized in Table 1.

3 Attribute Statistics based Embedding

Consider a set of graphs $\mathcal{G} = \{g_1, \ldots, g_N\}$, with $g_i = (V_i, E_i, \mu_i, \nu_i)$ being the ith graph in the set with labelling alphabet L_{V_i} for the nodes and L_{E_i} for the edges. We assume that all graphs in $\mathcal{G}$ have the same labelling alphabets, this is $L_{V_i} = L_{V_j}$ and $L_{E_i} = L_{E_j}$ for all $i, j \in \{1, \ldots, N\}$. We do not assume, however, that each node and edge label necessarily occurs in each graph. Let $L_V = \{\alpha_1, \ldots, \alpha_p\}$ and $L_E = \{\omega_1, \ldots, \omega_q\}$ be the discrete common labelling alphabets.

For each graph $g = (V, E, \mu, \nu) \in \mathcal{G}$, we define p unary features measuring the number of times each label in L_V appears in the graph, this is

$$U_i = \#(\alpha_i, g) = |\{v \in V \,|\, \alpha_i = \mu(v)\}|, \quad \forall i \in \{1, \ldots, p\}. \tag{1}$$

We also define $\frac{1}{2} \cdot q \cdot p \cdot (p+1)$ binary features counting the frequency of an edge with a specific label (ω_k) between two nodes with two given labels (α_i and α_j). Formally,

$$\begin{aligned} B_{ij}^k &= \#\left([\alpha_i \leftrightarrow \alpha_j]_{\omega_k}, g\right) \\ &= |\{e = (u, v) \in E \,|\, \alpha_i = \mu(u) \wedge \alpha_j = \mu(v) \wedge \omega_k = \nu(e)\}| \end{aligned} \tag{2}$$

where $k \in \{1, \ldots, q\}$ and $1 \leq i \leq j \leq p$. These features describe the local structure of every graph in terms of how frequently a simple substructure —an edge with a given label between two given node labels— occurs in a given graph.

These two sets of features can be combined in order to give a more global structural representation of the graphs by bringing together various pieces of local information. Formally, we define the embedding of graphs in the following way.

Definition 1 (Graph Embedding). *Given a graph $g \in \mathcal{G}$, let $\varphi_n(g)$ and $\varphi_e(g)$ be the vectors*

$$\varphi_n(g) = \left(\{U_i\}_{1 \leq i \leq p}\right) \tag{3}$$

$$\varphi_e(g) = \left(\{B_{ij}^k\}_{1 \leq i \leq j \leq p}^{1 \leq k \leq q}\right) \tag{4}$$

where U_i and B_{ij}^k are defined in Eqs. (1) and (2), respectively. The embedding of graph g is defined as the concatenation of these two vectors,

$$\varphi(g) = [\,\varphi_n(g)\ \varphi_e(g)\,]. \tag{5}$$

The above definition has been proved successful in our previous work [5]. In the current paper, we go one step further and assign a different weight to the node related vector $\varphi_n(g)$ and the edge related vector $\varphi_e(g)$. This leads to a generalized distance between the map of two graphs, where the information included in the nodes can be weighted differently from the information included in the edges. Given two graphs g_1 and g_2, we define the vectorial distance between them by

$$D(g_1, g_2) = (1 - \alpha) \cdot d_{L_1}(\varphi_n(g_1), \varphi_n(g_2)) + \alpha \cdot d_{L_1}(\varphi_e(g_1), \varphi_e(g_2)), \tag{6}$$

where $\alpha \in [0, 1]$ and $d_{L_1}(\cdot, \cdot)$ is the L_1 distance $d_{L_1}(x, y) = \sum_{i=1}^{n} |x_i - y_i|$. Clearly, the case $\alpha = 0.5$ is identical to the scenario in [5]. Now parameter α of Eq. (6) can be related to parameter ρ of the edit distance introduced in Section 2. As a matter of fact, Eq. (6) emulates the edit operations defined by the cost function of the edit distance. As described above, the cost function maintains all nodes and edges with identical labels, but deletes and subsequently inserts all

nodes and edges with different labels. Concerning the features we have defined, these operations translate into checking the absolute differences between vector coordinates. Note that the distance of Eq. (6) can alternatively be obtained if we would first weight both components of vector in Eq. (5) with $1-\alpha$ and α, respectively, and then compute the L_1 distance between the weighted vectors.

The parameter α measures the strength we give to the components of $\varphi_n(g)$ relative to $\varphi_e(g)$. In this way, there is a clear resemblance with ρ which weights the cost of operations on the nodes relative to the cost of operations on the edges. In Section 4.2, we experimentally check for the correlation of these parameters. From the definitions given above, it follows that the pair $(\rho, \alpha) = (0, 0)$ will result in a correlation coefficient equal to 1.

4 Experiments

4.1 Databases

We work with four datasets of discretely attributed graphs. These datasets are divided into two categories: object image datasets and molecule datasets. The object images are subsets of the ALOI and ODBK collections [4,11]. Images are segmented and a region adjacency graph is built, where nodes are labelled with a color name of the color naming theory and edges are labelled according to whether the common border of two adjacent regions is *short*, *medium* or *long*.

The molecule datasets are the AIDS and MUTAG collections from the IAM repository [7]. Nodes correspond to atoms labelled with the corresponding chemical element and edges represent chemical bonds with the corresponding covalent number.

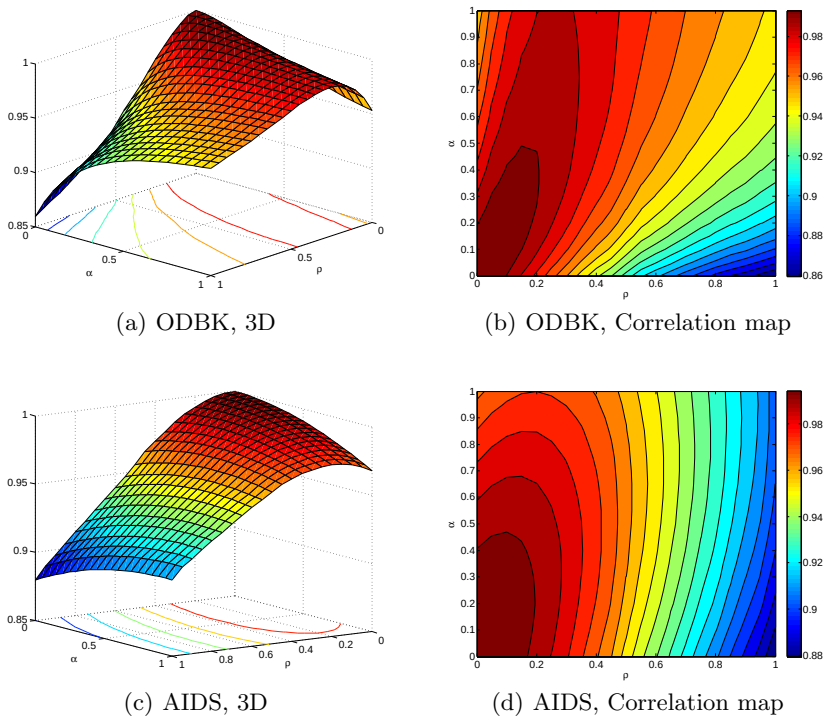
All four dataset are divided into a training, a validation and a test sets. In the following, we will use the training and the validation sets for computing pairwise distances. The test sets are not used.

4.2 Distance Correlation

Given a pair of values (ρ, α), we compute the sets of all pairwise graph distances X_ρ and Y_α between all graphs in the training set and all graphs in the validation set, using parameter value ρ for the edit distance and parameter value α for the embedding distance. For these two sets of distance values, X_ρ and Y_α, we compute the correlation coefficient by

$$C_{(\rho,\alpha)} = \frac{cov(X_\rho, Y_\alpha)}{\sigma_{X_\rho}\sigma_{Y_\alpha}}, \tag{7}$$

where $cov(X_\rho, Y_\alpha)$ is the covariance between distributions X_ρ and Y_α, and σ_{X_ρ} and σ_{Y_α} are the corresponding standard deviations. We compute such a coefficient for all pairs $(\rho, \alpha) \in [0, 1]^2$ and plot the corresponding 3D functions and correlation maps. Results can be seen in Fig. 1 (because of limited space we omit



(a) ODBK, 3D (b) ODBK, Correlation map

(c) AIDS, 3D (d) AIDS, Correlation map

Fig. 1. Correlation values as a function of the weighting parameters.

the ALOI and MUTAG cases but their behavior is very similar to that of ODBK and AIDS, and thus all discussions are valid for them as well).

First of all, we note how values close to $(\rho, \alpha) = (0, 0)$ have, both in the object and molecule datasets, a high correlation coefficient. This confirms that the embedding features under the L_1 metric replicate the edit distances when node information is considered as more relevant than edge information. If this is the case in the underlying application, we suggest to use the attribute statistics based graph embedding rather than working with graph edit distance because, first, the relative graph distribution is well maintained and, second, the computation efficiency is much higher.

In Fig. 1, we can also observe the biased effect of the correlation values with respect to the ideal case, where a diagonal behavior should be observed. The explanation for the biased relation is the fact that the edge-based embedding features still keep quite some information of the node labels. In particular, the co-occurrence of a certain pair of node labels at the end of an edge tells us that these particular node labels do appear in the graph. Therefore, it is clear that considering edge-based features only, the embedding representation still keeps information about the node attributes. As a consequence of this phenomenon, the correlation of the embedding distances for $\alpha = 1$ is maximized by values

$\rho \simeq 0.2$, suggesting that 80% of the node information in the graph domain is still included in the embedding representation when only edge-based features are considered.

In Fig. 1, when both ρ and α tend to 1, low correlation values result. This might be explained by the fact that the edit distance computation looks for an edit path that completely disregards the information of the nodes. Thus, since edge-based embedding features still keep some of this information, the behavior of distances in both domains becomes different.

Also worth noting is the shape of the correlation regions, which is more ellipse-like in the molecule datasets than in the object datasets. This observation has an interpretation in terms of how important the actual structural configuration of graphs is in each dataset. In the molecule datasets edge information is more salient than in the object datasets. The more weight we put on the edge-based features ($\alpha \to 1$) the faster the correlation values for $\rho \simeq 0.2$ descend, which means that edge-based features are less correlated with the node information in the graph domain and thus we should put more attention on the edges.

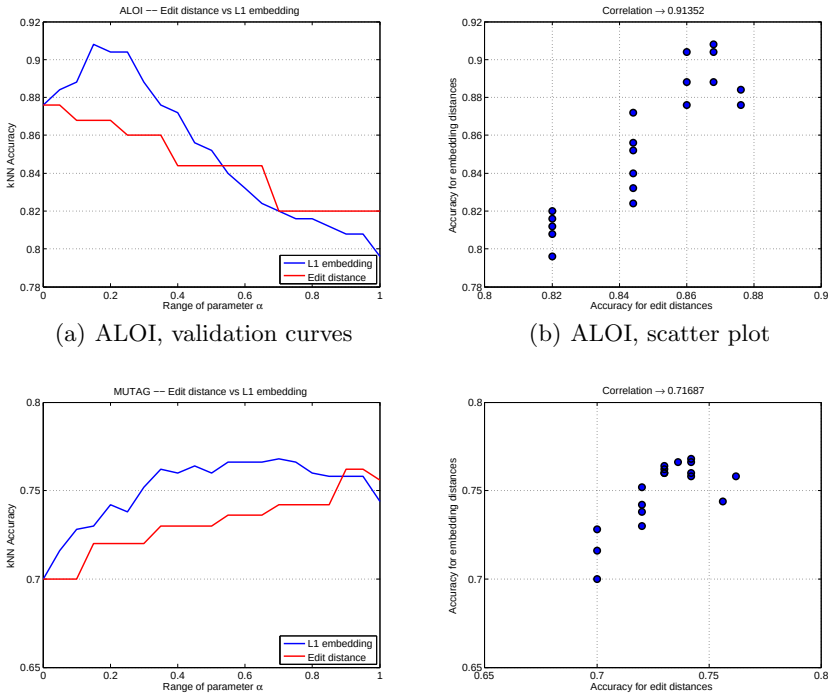
4.3 Classifier Correlation

Another way to check how well the edit distances are reproduced in the embedding space is to see how a distance based classifier performs. In this paper, we use a kNN classifier with both ways of computing the distances between graphs and look for the difference in performance. In particular, we compare the performance of the classifier based on the distances in the vector space for all values of α with the performance of the classifier using the edit distances in the graph domain for those values of ρ that maximize the correlation for every value of α. We indicate by

$$\overline{\rho}_\alpha = \underset{\rho \in [0,1]}{\operatorname{argmax}} C_{(\rho,\alpha)} \tag{8}$$

the ρ value that maximizes the correlation coefficient for a given α value. In Fig. 2 we show the corresponding classification curves on the validation sets. The x-axis shows the range of parameter α and the y-axis the classification rates. In particular, the results' curves for the embedding distances are stretched in such a way that the value $\alpha = 1$ coincides with the ρ values maximizing its correlation, and the corresponding intermediate values of α are maximally correlated with the respective ρ values in the curve. This is, for each value of α, the corresponding result of the edit distance curve is that of $\overline{\rho}_\alpha$. We also show the corresponding scatter plots of the accuracies of both classifiers, for all pairs $(\alpha, \overline{\rho}_\alpha)$ and give the correlation coefficient for these scatter plots on top of Figs. 2(b) and 2(d).

With regards to the correlation of the classifiers, we observe a great degree of similarity of both curves, supporting the hypothesis of a high correlation. We notice that the classifiers' correlation is higher for the object datasets than for the molecule ones. This result is explained by the same reason we have been discussing before. The fact that the embedding features correlate with edit distance whenever the node information of graphs is actually relevant makes the



(a) ALOI, validation curves (b) ALOI, scatter plot

(c) MUTAG, validation curves (d) MUTAG, scatter plot

Fig. 2. Classifiers performance: L_1 embedding as a function of α, edit distance as a function of $\overline{\rho}_\alpha$, and scatter plots of the accuracies of all pairs $(\alpha, \overline{\rho}_\alpha)$.

classifiers perform in similar ways. On the other hand, the molecules need some more attention on the edge structure and therefore the edit distance and the embedding based distance differ more.

With respect to the performance of the classifier on the molecules dataset, we observe how the embedding curve obtains its highest result for an intermediate value of the parameter, thus confirming that here edges have higher importance than in the objects case, where the highest result is obtained by a value of the parameter closer to 0. Another point supporting this idea is the fact that for the object datasets the case $\alpha = 0$ gives a better result than that of $\alpha = 1$, and for the molecule datasets this is the other way around.

As a final comment, we note how in most of the cases the performance of the embedding classifier for a given α outperforms that of the edit distance classifier for $\overline{\rho}_\alpha$. This suggests that whenever both ways of computing distances are regarding the same type of information, the embedding distances are more capable to distinguish among graph categories. Because of this observation and because of its much higher efficiency, the use of the embedding methodology for graph comparison is recommended.

5 Conclusions

In this work we have established a relation between graph edit distance and the L_1 vectorial distance in the attribute statistics embedding space. It has been shown that under a special class of cost functions, where node and edge label insertions and deletions are favored over substitutions, there is a close relation between the graph edit distance and the L_1 distance of the corresponding vectors obtained through graph embedding. Our formal analysis has been confirmed in a series of experiments. We have experimentally shown that there exists a high correlation between both types of graph distances and between the corresponding classifiers, provided that corresponding parameter values are chosen for both distances. The analysis provided in this paper may help in developing a better understanding of label statistics based embedding [5], which has been demonstrated to perform very well in practice but has been lacking, until now, a more rigorous formal investigation of its properties.

The current paper is limited to graphs with discrete labels. However, in future, a similar study for the case of continuous attributed graphs is planned. In addition, it would be interesting to exploit the embedding features to derive necessary conditions for subgraph isomorphism in terms of component-wise relations between the corresponding vectorial representations of graphs.

References

1. H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.
2. H. Bunke and G. Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1:245–253, 1983.
3. D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.
4. J.M. Geusebroek, G.J. Burghouts, and A.W.M. Smeulders. The Amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.
5. J. Gibert, E. Valveny, and H. Bunke. Graph embedding in vector spaces by node attribute statistics. *Pattern Recognition*, 45(9):3072–3083, 2012.
6. P. Ren, R. Wilson, and E. Hancock. Graph characterization via Ihara coefficients. *IEEE Transactions on Neural Networks*, 22(2):233–245, 2011.
7. 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, *Proceedings of the S+SSPR*, volume 5342 of *LNCS*, pages 287–297. Springer, 2008.
8. K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27:950–959, 2009.
9. K. Riesen and H. Bunke. *Graph Classification and Clustering Based on Vector Space Embedding*. World Scientific, 2010.
10. A. Sanfeliu and K.S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 13(3):353–363, 1983.
11. M.J. Tarr. The object databank. *http://www.cnbc.cmu.edu/tarrlab/stimuli/objects/index.html*.