Vertex Similarity - A Basic Framework for Matching Geometric Graphs

Ayser Armiti and Michael Gertz

Institute of Computer Science, Heidelberg University, Germany
{ayser.armiti,gertz}@informatik.uni-heidelberg.de

Abstract. Solutions to the graph matching problem play an important role in many application domains, such as chemistry, proteomics, or image processing. Especially in these domains, graphs have geometric properties that describe the positions of the vertices in some 2- or 3-dimensional space. Several exact and approximate approaches have been proposed to address the problem of matching graphs, which is known to be NP-hard in general. For this, most approaches depend on the concept of vertex similarity to iteratively increase the matching quality.
In this paper, we study the vertex similarity problem for geometric graphs. We formally define such a problem and prove that its complexity is NP-hard. For geometric graphs in 2D, we propose an approximate solution with polynomial runtime. For this, we utilize techniques underlying attributed cyclic string matching and customized edit operations that consider spatial properties and labeling information. In our evaluations, we show that our approach outperforms existing vertex similarity approaches in terms of classification accuracy and matching quality.

1 Introduction

Searching for and exploring similar objects is an important task in many application domains, such as in social networks, biology, or pattern recognition. For such domains and many more, graphs are used as a powerful data structure for the representation of objects and object relationships. By this, searching for similar objects turns to be the tasks of finding similar (sub)graphs, which is estimated by a *graph matching* algorithm [9]. Approaches to graph matching search for correspondences between the vertices of two graphs such that the matched vertices have similar labels and connectivity.

The problem of *inexact graph matching*, which is matching graphs in the presence of noise and outliers, has been shown to be NP-hard [25]. As a consequence, most of the approaches to solving the matching problem focus on finding

 In: T. Seidl, M. Hassani, C. Beecks (Eds.): Proceedings of the LWA 2014 Workshops: KDML, IR, FGWM, Aachen, Germany, 8-10 September 2014, published at http://ceur-ws.org

approximate solutions, e.g., [11, 12]. These approaches follow an iterative *continuous optimization* procedure. The objective function used by such optimization procedures depends on the concept of *vertex similarity* [7, 26].

For many graph matching algorithms, vertex similarity means the similarity of the labels assigned to the vertices and edges [25]. But in application domains such as chemistry, proteomics, or image processing, vertex similarity is much more complex and subjective. For such disciplines, it is critical to study the *spatial properties* of vertices, in addition to their labels and connectivity [3].

In this paper, we study the vertex similarity problem for geometric graphs. We build on our previous work [5, 6] and prove that such a problem is NP-hard in general. For geometric graphs in 2D space (as common in many application domains), we propose a novel approach to estimate the similarity between different vertices. Our solution is based on the concept that two vertices are similar when their direct neighboring vertices are similar. For this, we propose to extract a feature for each vertex based on the properties of its neighborhood. We use string edit distance to compute the similarity of two vertices using their features. To realize this, we propose customized edit operations that utilize spatial properties and labeling information. Comprehensive empirical studies using real geometric graph datasets from different application domains are used to demonstrate the accuracy of our proposed approaches compared to related work.

The remainder of the paper is organized as follows. In Section 2, we survey related work. Section 3 discusses the problem settings. In Section 4, we propose our novel approaches to solve vertex similarity for geometric graphs in 2D space. In Section 5, we present experimental results of our proposed approach. Finally, Section 6 summarizes the paper.

2 Related Work

Several graph matching algorithms use the Euclidean distance to estimate the similarity between real-valued labels that are assigned to the vertices and edges [19, 25]. For geometric graphs, the coordinates of the vertices cannot be simply treated as real-valued attributes since they are measured with respect to the particular reference axis frame for each graph. This makes the Euclidean distance incapable of estimating the spatial distance between vertices of two graphs under a geometric transformation. In addition to this, pure structural graph matching approaches cannot be applied to geometric graphs because they do not consider the spatial properties of a graph. Several other approaches have been proposed to solve the vertex similarity problem for non-geometric graphs [27, 28]. However, for geometric graphs, little can be found in literature. We believe that this is a consequence of the complexity of the problem in the case of geometric graphs, which will be discussed in later sections. In the following, we discuss current approaches to estimate the similarity between vertices of geometric graphs and divide them into two classes: global and local approaches.

The global approaches extract a feature for each vertex using the overall graph structure. Algorithms that utilize graph spectra are classified as global

approaches. The main idea is to extract a feature for each vertex based on the values of the Eigenvectors. Such features are then used by the Hungarian algorithm for graph matching [22]. To use the same concept for geometric graphs, the spectra of the *weighted* adjacency or the weighted Laplacian matrices are used. The weight of an entry represents the length of an edge, which is computed using the Euclidean distance between the coordinates of its incident vertices. Then, eigen-decomposition is used to generate a spectral feature for each vertex, which is represented by the values of the Eigenvectors with respect to that vertex. Since graphs with different numbers of vertices create different numbers of Eigenvectors, the spectral features for the vertices are truncated by keeping the values with respect to the most dominant Eigenvectors [26], i.e., the Eigenvectors that correspond to the largest Eigenvalues. Based on this, the distance between two vertices equals the Euclidean distance between their spectral features. A major drawback of the spectral approach is that it cannot handle labeling information. Also, such an approach is sensitive to differences in the number of vertices, the structure of the graph, and the lengths of the edges.

Another global vertex similarity approach is based on the *landmark distance* concept [8]. First, a set of vertices from each graph is selected as landmarks. Then, every vertex from the graph is represented by a feature vector containing the distances to the landmarks. The distance is measured as the length of the shortest path between the vertex and a landmark. Then, the distance between two vertices is computed using the Manhattan distance between their landmark-based features. The basis of such an approach is the selection of landmarks for each graph. Cheong *et el.* [8] propose to use four landmarks as the extreme vertices in the boundaries of the graph, i.e., peripheral vertices. However, such an approach is incapable of matching graphs that differ in the number of vertices.

To overcome the problems of the global approaches, local features are extracted from the neighborhood of each vertex. One of the earliest approaches to estimate the similarity of different vertices is the *histogram-based* approach [10, 13, 21]. A histogram is created from the spatial properties of the neighborhood of each vertex. It stores the pair-wise relationships between the edges that are incident to that vertex, which consists of the ratio of the lengths of the edges in addition to the angle between them. As a result, the local-feature is a 2D histogram of edge lengths and angle values. Based on this, the distance between two vertices is estimated by the distance between their geometric histograms, which is computed by the χ^2 or the Bhattacharyya distances. Unfortunately, histogram approaches face problems in binning and normalization, especially when dealing with real-valued attributes, i.e., the edge length and the angle value.

Notice that the above approaches extract features that are invariant to geometric transformations. Another approach to solve vertex similarity is to use *geometric hashing* based on the coordinates of the vertices [24]. The basis of this approach is to create several local frames for the neighborhood of each vertex, which are defined again by that vertex and its direct neighbors. Then, the coordinates of the vertices in the neighborhood of a vertex are measured with respect to each local frame. After that, hashing is used to speed up the search for the

local frame that best estimates the distance between two vertices. Geometric hashing is efficient in the case of matching vertices that have a homogeneous transformation, i.e., rigid transformation. But, in the case of inexact matching, such an approach fails to estimate the similarity of the vertices.

3 General Problem Setting

In our framework, we consider (non-)planar, labeled, undirected geometric graphs that do not contain self-loops or multi-edges.

Definition 1. ***(Geometric Graph)*** *A labeled undirected geometric graph* $G = (V, E, l, c)$ *consists of a finite set of vertices* V*, a finite set of edges* $E \subseteq V \times V$*, a labeling function* $l : \{V \cup E\} \rightarrow \Sigma$*, assigning a label to every vertex and every edge from a label alphabet* Σ*, and a function* $c : V \rightarrow \mathbb{R}^d$*, assigning a coordinate in* $\mathbb{R}^d$ *to every vertex.*

Without loss of generality and throughout the rest of this paper, we represent a geometric graph G as $G = (V, E)$. The size $|G|$ of a graph is the number of vertices in G. The degree of a vertex v, denoted $\deg(v)$, is the number of vertices that are directly connected to v. The set of direct neighboring vertices of a vertex v is denoted by $N(v)$.

In our framework, we follow a local-based vertex similarity approach, which has been proved to give good results for general non-geometric graphs [19, 25]. It is based on the concept that two vertices are similar when their neighbors are similar. For our framework, we call the neighborhood of a vertex its *signature*.

Definition 2. ***(Vertex Signature)*** *Given a vertex* v_i *in a graph* $G = (V, E)$*, the vertex signature* $S(v_i)$ *is a subgraph* $G^{'} = (V^{'}, E^{'})$ *of* G *such that* $V^{'} = \{v_i \cup \{v_j | (v_i, v_j) \in E\}\}$*. For each vertex* $v_j \in V^{'}$*,* $v_j \neq v_i$*, there exists an edge* $(v_i, v_j) \in E^{'}$*.* v_i *is called the* ***root vertex*** *of* $S(v_i)$*.*

After defining the meaning of locality, the similarity between two vertices is estimated by computing the similarity of their vertex signatures. A function that quantifies the similarity between two vertex signatures must satisfy geometric transformations, i.e., two vertex signatures are considered spatially identical if there is a geometric transformation (rotation, translation, and scaling) that makes the coordinates of one vertex signature identical to the coordinates of the other [16]. In addition to this, and for many scientific applications, two similar objects are often represented by two non-identical graphs. In pattern recognition applications, acquisition methods often introduce noise in the number of vertices and their locations. Also, the structure and connectivity of vertices often vary between graphs representing similar objects. As a result, two vertex signatures representing similar vertices have differences in the number of neighbors, labeling information, the lengths of the edges, and the distances between the neighboring vertices. This leads to the concept of *inexact vertex similarity*, which will be detailed in the following section.

3.1 Vertex Edit Distance

To compute the inexact similarity between two vertex signatures, we adopt the *edit distance* concept that is been used in matching strings and graphs [20, 23]. It is defined as the minimum amount of changes that is needed to make a string or a graph identical to another. We call the edit distance of two vertex signatures the *vertex edit distance (VED)*. For two vertex signatures $S(v)$ and $S(u)$, the key idea of the VED is to delete some vertices and edges from $S(v)$, re-label some other vertices and edges, change the coordinates of some vertices, and insert some vertices and edges into $S(u)$ such that the two vertex signatures become identical. For this, we adopt three *edit operations*: substitution (re-label), insertion, and deletion. A sequence of edit operations that transfer one vertex signature to be identical to another is called an *edit path*. Obviously, there are many possible edit paths from one vertex signature to another. As a result, the VED is defined as the distance with the minimum cost of all of them:

Definition 3. ***(Vertex Edit Distance)*** *Let $\phi(S(v), S(u))$ be the set of all geometric transformations between the coordinates of $S(v)$ and $S(u)$, $\Upsilon_{\phi_i}(S(v), S(u))$ be the set of all edit paths between $S(v)$ and $S(v)$ after applying the geometric transformation ϕ_i, then the vertex edit distance is defined as:*

$$d(v, u) = \min_{\phi_i \in \phi(S(v),S(u)), p_j \in \Upsilon_{\phi_i}(S(v),S(u))} cost(p_j) \tag{1}$$

where $cost(p_j)$ is the total cost of all edit operations of the path p_j

The cost of an edit path depends on the cost of its edit operations, which we define as the following. The cost of a substitution between two vertices is defined by the Euclidean distance between their coordinates, the distance between their labels, and the substitution costs of their edges. The substitution cost between two edges is defined as the distance between their labels in addition to the distance between their lengths. The cost of vertex insertion or deletion equals to a constant α.

Lemma 1. *The problem of vertex edit distance for geometric graphs in the $\mathbb{R}^d$ space is NP-hard such that $d \geq 2$.*

In the following, and without loss of generality, we give a sketch proof for the above lemma in the case of unlabeled geometric graphs.

Proof Sketch: For two unlabeled geometric graphs, we reduce the problem of inexact point set matching to the problem of vertex edit distance. Let $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, q_2, \ldots, q_m\}$ be two point sets in $\mathbb{R}^d$, $d \geq 2$. The two point sets are reduced to two vertex signatures in polynomial time as follows. All points from the point set P become vertices directly connected to a dummy root vertex v_p. The coordinate of v_p is computed as the center of the point set P. In the same way, the points from Q create a vertex signature with a dummy root vertex v_q. The optimal match between P and Q is the optimal mapping of the neighbors of v_p and the neighbors v_q. Non-matched points (vertices) represent

the insertion and deletion operations. A substitution operation is indicated by a correspondence from one point to another.

The problem of inexact point set matching in the $\mathbb{R}^d$ space is proved to be NP-hard [4] where $d \geq 2$. As result, the problem of computing the optimal solution of vertex edit distance for geometric graphs in $\mathbb{R}^d$ space is also NP-hard. □

4 Vertex Similarity for 2D Geometric Graphs

In this section, we propose an approximate solution to the VED problem for geometric graphs in 2D space, which can be computed in $O(mn \log n)$, such that n and m are the numbers of edges for two vertex signatures. For this, we require that the edges of a vertex signature are sorted in counter-clockwise order around the root vertex. As a result, the feature that is extracted from each vertex signature is a cyclic string that is defined as follows:

Definition 4. ***(Spatial Feature)*** *Given a geometric graph $G = (V, E, l, c)$, the spatial feature F_v for the vertex signature $S(v)$ is a* ***cyclic string*** $F_v = [f_1, f_2, \ldots, f_n]$, $n = \deg(v)$, *such that each token f_i is defined as:*

$$f_i := (|e_i|, \angle e_i e_{i-1}, l(e_i), l(v_i))$$

where $|e_i|$ denotes the length of the edge e_i, $\angle e_i e_{i-1}$ denotes the angle between the edges e_i and e_{i-1} in counter-clockwise order, $l(e_i)$ is the label of edge e_i, and $l(v_i)$ is the label of the neighboring vertex incident to edge e_i.

The feature is created by selecting an edge from the vertex signature and going over the rest of the edges in a counter-clockwise order. For a vertex signature of n edges, there will be n different ways to represent its feature. However, all of them are considered equivalent with a cyclic shift from one to another.

Once the spatial features are represented as cyclic strings, the VED between two vertex signatures is estimated by the *cyclic string edit distance* (CS) [17], which is a natural extension to the string edit distance. A naive approach to solve it runs in $O(nm^2)$, where n and m are numbers of edges for two vertex signatures. This is done by applying the algorithm by Wagner and Fisher [23] to the first spatial feature and all cyclic shifts of the second one. Maes in [17] proposed a faster solution to the cyclic string edit distance that runs in time $O(nm \log m)$. So, the cyclic string edit distance gives an approximate solution to the VED problem with a runtime complexity of $O(nm \log m)$.

To utilize the CS approach, we define three edge edit operations: substitution, insertion, and deletion. We propose edit operations that combine spatial attributes and labeling information. In the following we discuss two sets of edit operations. The first one computes the edit operations based on the absolute values of the edge length and the angel value. The second one uses a polar distance based on the lengths of two edges and the angle between them.

Edit operations using the Manhattan distance

Given two vertex signatures $S(v)$ and $S(u)$ such that $n = |S(v)|$ and $m = |S(u)|$, let edge $e_i \in S(v)$, edge $e_j \in S(u)$, $f_i = (|e_i|, \angle e_i e_{i-1}, l(e_i), l(v_i))$, $f_j = (|e_j|, \angle e_j e_{j-1}, l(e_j), l(u_j))$, then, the substitution $\gamma(f_i \rightarrow f_j)$ is defined as:

$$\gamma(f_i \rightarrow f_j) := d_L(f_i, f_j) + d_S(f_i, f_j) \tag{2}$$

In the case of labeled graphs, the function $d_L(f_i, f_j)$ computes the distance between the label of edge e_i and the label of e_j in addition to the distance between the labels of the vertices that are incident to them, i.e., v_i and u_j. The function $d_S(f_i, f_j)$ calculates the spatial distance based on the angle and the edge length. For an edge e, let θ_e and l_e denote the angle and edge length, as defined earlier in Definition 4. The function d_S is formally defined as follows:

$$d_S(f_i, f_j) := \frac{|\theta_{e_i} - \theta_{e_j}|}{2\pi} + \left| \frac{l_{e_i}}{\sum_{k=1}^{n} l_{e_k}} - \frac{l_{e_j}}{\sum_{k=1}^{m} l_{e_k}} \right| \tag{3}$$

The angles and edge lengths at a vertex signature are normalized, as can be seen by the denominators used in the above equation. An angle is normalized by 2π since the sum of angles at a local signature sums up to this value. Also, an edge length is normalized by the sum of edge lengths at a local signature. For example, for a local signature $S(v)$, the edge length normalization factor is $\frac{l_{e_i}}{\sum_{k=1}^{n} l_{e_k}}$, where n is the number of edges connected to v.

In the following, we define the insertion and deletion operations. Let λ represent the null (non-existent) edge, then the insertion $\gamma(\lambda \rightarrow f_i)$ and deletion $\gamma(f_i \rightarrow \lambda)$ with respect to f_i are defined as follows:

$$\gamma(\lambda \rightarrow f_i) = \gamma(f_i \rightarrow \lambda) := c(f_i) + \left(\frac{\theta_{e_i}}{2\pi} + \frac{l_{e_i}}{\sum_{k=1}^{n} l_{e_k}} \right) \tag{4}$$

The cost of edge insertion or deletion is computed based on the angle value, edge length, and labeling information. For labeled graphs, the function c defines the cost of inserting or deleting the label assigned to that edge in addition to the label assigned to its incident vertex.

For unlabeled graphs, the cost of an edit operation lies in the range [0,2]. This is because each of the angle value and edge length is normalized to the range [0,1]. For labeled graphs, the range increases depending on the range of the function d_L for the substitution operation and c for the insertion and deletion.

Edit operations using polar coordinate

The second set of edit operations shares many similarities with the previously defined edit operations. However, the spatial distance between two vertex signatures is computed based on the polar distance between the neighboring vertices

of two vertex signatures. Given two vertex signature $S(v)$ and $S(u)$, let edge $e_i \in S(v)$ and edge $e_j \in S(u)$. The substitution cost $\gamma(f_i \to f_j)$ is defined as:

$$\gamma(f_i \to f_j) = d_L(f_i, f_j) + d_S(f_i, f_j) \tag{5}$$

In the case of labeled graphs, the function $d_L(f_i, f_j)$ computes the distance between the label of edge e_i and the label of e_j. It also computes the distance between the label of the neighboring vertex connected to e_i to the label of the one connected to e_j. The function $d_S(f_i, f_j)$ calculates the spatial distance based on the angles and the lengths of the edges. For an edge e, let θ_e denote the angle between e and the previous edge in a counter-clockwise order, and let l_e denote the edge length. The function d_S is defined as:

$$d_S(f_i, f_j) = \sqrt{l_{e_i}^2 + l_{e_j}^2 - 2\ l_{e_i} l_{e_j} cos(|\theta_{e_i} - \theta_{e_j}|)} \tag{6}$$

The substitution cost is defined as the distance needed for the neighboring vertex of edge e_i to align with the neighboring vertex of e_j. This can be seen as the polar distance between them such as the polar axis for each vertex is the edge that precedes it in the counter-clockwise order. Analogously, we define the insertion and deletion operations. Let λ represent the null (non-existent) edge. Then, the insertion $\gamma(\lambda \to f_i)$ and deletion $\gamma(f_i \to \lambda)$ with respect to f_i are defined as:

$$\gamma(\lambda \to f_i) = \gamma(f_i \to \lambda) = c(f_i) + l_{e_i} \tag{7}$$

The cost of edge insertion or deletion is computed based on the edge length (l_{e_i}). For labeled graphs, the function c defines the cost of inserting or deleting the label assigned to the edge e_i in addition to the label assigned to the neighboring vertex connected to e_i.

5 Evaluation

In this section, our proposed approach to the vertex similarity problem and its usage for graph matching is empirically evaluated. We use three different data sets: 1) Chinese characters [1], 2) the COIL-100 image data set [18], and 3) the CMU house and hotel image data sets [2]. Besides coming from different application domains, our data sets vary in many aspects such as the size of the data set, the number of classes (in case geometric graphs have been assigned to classes), as well as the number of vertices and edges.

We compare our algorithms **(CSv1)**, which uses the first set of edit operations, and **(CSv2)**, which uses the second set of edit operations, with three other approaches: a graph spectral approach **(SP)** [22, 26], a geometric histogram of the pair-wise relations between the edges in the neighborhood of a vertex **(GH)** [14, 21], and a unary vertex distance function based on only the coordinates of the vertices **(CO)** such as neither global nor local structural information is used. We test such an approach to evaluate the effect of using the coordinates of the vertices on their similarities.

To evaluate the different approaches, we embed them in a unified graph matching algorithm. It consists of two steps. First, a vertex-to-vertex distance matrix is created using any of the previous approaches. Second, the Hungarian algorithm [15] is used to select the best match between the two graphs. Since all the approaches use the Hungarian algorithm for graph matching, the differences in the matching results are affected only by the approach that is used to estimate the similarity between two vertices.

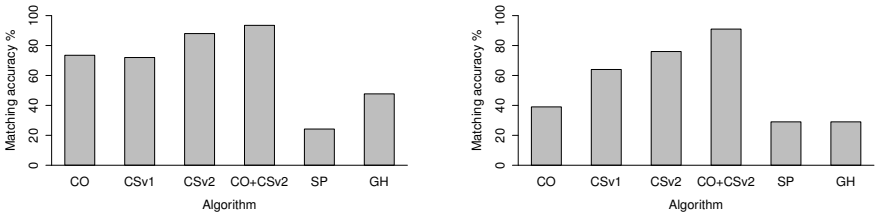
To evaluate the performance of a vertex similarity approach, we use two criteria. The first one is the effect of vertex similarity on a graph similarity metric. This is evaluated by embedding the graph matching algorithm in a classification task. The higher the classification accuracy the better the vertex similarity approach. To create a graph distance metric, we follow a graph edit distance approach. This means that the distance between two graphs consists of the cost of the match between them, i.e., the substitution cost, in addition to the cost of inserting the unmatched vertices. The second criterion is the selectivity power, which means that a vertex similarity approach reflects the similarity notion of an application domain. This is measured by the quality of the match computed by the graph matching algorithm.

5.1 Graph Similarity and Classification

In this section, we evaluate the relation between different vertex similarity approaches and graph similarity in general. To measure this, we test the different approaches in a graph classification task. In our experiments, we used the first nearest neighbor classifier (1-NN) based on the similarities of the graphs. For this experiment, we use the COIL-100 data set [18], which consists of images of 100 different objects taken at different degrees. A geometric graph is then extracted from each image. From 3900 graphs, we select 2900 for training, 29 graphs for each object. For testing, we select 1000 graphs, 10 graphs for each object. We also use the Chinese data set [1], which contains a total of 9384 characters that belong to 6 different fonts, i.e., 1564 characters from each font. A test data set of 1564 graphs is extracted from the Dotum Korean font. The remaining five fonts build a training data set of 7820 graphs. Ideally, for a query character, its most similar character from the train data set should have the same Unicode.

Figure 1(a) shows the classification accuracies for the COIL-100 data set for the different approaches. The lowest classification accuracy is for the **SP** approach. This is because spectral approaches are sensitive to the changes in the number of vertices in addition to their spatial properties. We conclude that a local-based vertex similarity approach is better than a global-based one. The best classification accuracy is for the **CSv2+CS** approach, followed by the **CSv2** approach. Notice that the use of invariant spatial features by our approaches (**CSv1** and **CSv2**) gives better results than using the coordinates of the vertices (**CO**). However, combining both of them gives the best result.

The classification accuracy for the Chinese data set is shown in Figure 1(b). Also, for this data set, the best results is for the **CO+CSv2** approach. On the



a) COIL-100 data set b) Chinese data set

Fig. 1: Classification accuracy for different vertex similarity approaches.

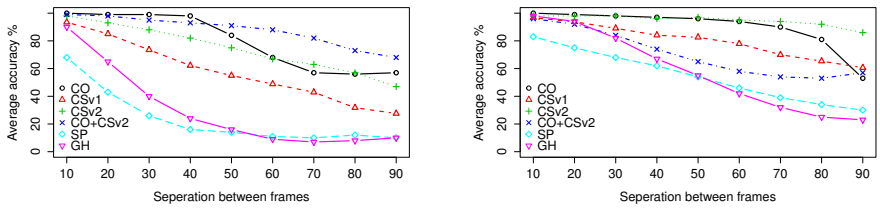
other hand, the **CSv1** and **CSv2** approaches are much better than **CO** alone. The lowest classification accuracy is for **SP** and **GH**.

From these two data sets we conclude that using invariant spatial features is better than using only the coordinates of the vertices. However, still the coordinates of the vertices can be used to give good graph matching results for many applications. Also, using the second set of edit operations, i.e., **CSv2**, gives better results that the first set. i.e., **CSv1**. This is justified since **CSv1** gives the same weight for the differences in the angle value and the edge length.

5.2 Graph Matching

In this section, we evaluate the quality of the match computed by the graph matching algorithm. Higher matching quality indicates higher selectivity power for a vertex similarity approach. We use the matching accuracy to estimate the quality of a match. It is defined as the number of correct matches, computed by a matching algorithm, over the actual total number of correct matches. For this test, we use the CMU hotel and house data sets. They contain images for a toy house and hotel, subjected to rotation in 3D. For each data set, we match all images spaced at 10, 20, 30, 40, 50, 60, 70, 80, and 90 in the rotation sequence, and compute the average matching accuracy.

From Figures 2(a) and 2(b), one can see that for all the approaches, the matching accuracy decreases when the distance in the rotation sequence between the images increases. This is a consequence of the increase in the structural differences between the geometric graphs. The lowest matching accuracy is for the **SP** approach, which is sensitive to the changes in the structure of the graphs. The best matching accuracy is for the **CSv2** and **CO**. However, **CO+CSv2** is not always better than using each of the single approaches alone. Also, the matching accuracy of **CSv2** is better than the one of **CSv1**. This means that using the polar distance gives better results than using just the absolute values of the length of the edge and the angle value.



a) Hotel data set b) House data set

Fig. 2: Matching quality for the CMU hotel/house data sets.

6 Conclusions

In this paper, we discussed the problem of vertex similarity for geometric graphs. Our focus is on local-based vertex similarity approaches, which use the properties of the neighborhoods of the vertices to estimate their similarities. One of the main results that we introduced is the sketch proof that the problem of vertex similarity for geometric graphs is NP-hard in general. On the other side, we proposed an algorithm to approximate the similarity between vertices for geometric graphs in 2D space. Our solution utilizes the property that the direct neighbors of a vertex has a total order, which is a consequence of the embedding of the neighboring vertices in 2D space. To find the similarity between two vertices, first, a spatial feature is extracted, which is a cyclic string of the lengths of the edges in addition to the angles between them. After that, the cyclic string edit distance is used to estimate the similarity of different vertices. For this, we proposed edit operations that utilize spatial properties and labeling information. We demonstrated the accuracy of our approach using different real-world data sets from image processing and character recognition. We also showed that our approach compares favorably to existing vertex similarity techniques.

References

1. CJK Fonts: Chinese, Japanese, and Korean Fonts. `http://bookr-mod.googlecode.com/files/cjk-fonts-1.zip`. Accessed: 01/12/2011.
2. CMU house and hotel data sets. `http://vasc.ri.cmu.edu/idb/html/motion`. Accessed: 16/02/2012.
3. C. Aggarwal and H. Wang. *Managing and mining graph data*. Springer, 2010.
4. T. Akutsu, K. Kanaya, A. Ohyama, and A. Fujiyama. Point matching under non-uniform distortions. *Discrete Applied Mathematics*, 127(1):5–21, 2003.
5. A. Armiti and M. Gertz. Efficient Geometric Graph Matching Using Vertex Embedding. In *SIGSPATIAL*, pages 234–243, 2013.
6. A. Armiti and M. Gertz. Geometric Graph Matching and Similarity: A Probabilistic Approach. In *SSDBM*, pages 27:1–27:12, 2014.
7. T. Caetano, J. McAuley, L. Cheng, Q. Le, and A. Smola. Learning graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(6):1048–1058, 2009.

8. O. Cheong, J. Gudmundsson, H. Kim, D. Schymura, and F. Stehn. Measuring the Similarity of Geometric Graphs. *Experimental Algorithms*, pages 101–112, 2009.
9. D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty Years Of Graph Matching In Pattern Recognition. *IJPRAI*, 18(3):265–298, 2004.
10. X. Gao, B. Xiao, D. Tao, and X. Li. Image categorization: Graph edit distance+edge direction histogram. *Pattern Recognition*, 41:3179–3191, 2008.
11. X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Anal. Appl.*, 13(1):113–129, 2010.
12. S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(4):377–388, 1996.
13. B. Huet and E. Hancock. Inexact graph retrieval. In *IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 40–44, 1999.
14. B. Huet and E. R. Hancock. Relational Object Recognition from Large Structural Libraries. *Pattern Recognition*, 35:1895–1915, 2002.
15. H. Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.
16. M. Kuramochi and G. Karypis. Discovering Frequent Geometric Subgraphs. In *ICDM*, pages 258–265, 2002.
17. M. Maes. On a cyclic string-to-string correction problem. *Information Processing Letters*, 35(2):73–78, 1990.
18. S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-100). Technical report, Feb 1996.
19. K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(7):950–959, 2009.
20. A. Sanfeliu and K.-S. Fu. A Distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. Syst., Man, Cybern., Syst.*, 13(3):353–362, 1983.
21. N. Thacker, P. Riocreux, and R. Yates. Assessing the completeness properties of pairwise geometric histograms. *Image and Vision Computing*, 13(5):423–429, 1995.
22. S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(5):695–703, 1988.
23. R. Wagner and M. Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, 1974.
24. X. Wang, D. Shasha, B. Shapiro, I. Rigoutsos, and K. Zhang. Finding Patterns in Three-Dimensional Graphs: Algorithms and Applications to Scientific Data Mining. *IEEE Trans. on Knowl. and Data Eng.*, 14(4):731–749, July 2002.
25. Z. Zeng, A. K. H. Tung, J. Wang, J. Feng, and L. Zhou. Comparing Stars: On Approximating Graph Edit Distance. *PVLDB*, 2(1):25–36, 2009.
26. Y. Zhu, L. Qin, J. X. Yu, Y. Ke, and X. Lin. High efficiency and quality: large graphs matching. In *CIKM*, pages 1755–1764, 2011.
27. G. Wang, B. Wang, X. Yang, and G. Yu. Efficiently Indexing Large Sparse Graphs for Similarity Search. *IEEE Trans. Knowl. Data Eng.*, 24(3):440–451, 2012.
28. C. Xiao, X. Lin, X. Zhao, and W. Wang. Efficient Graph Similarity Joins with Edit Distance Constraints. In *ICDE*, pages 834–845, 2012.