Chapter 5

Geometric Graph Matching

5.1 Introduction

Geometric graph matching is the process of evaluating the similarity between geometric graphs, where each vertex has an associated coordinate point in a plane. Exact matching computes a strict correspondence between two geometric graphs, whereas in error-tolerant matching an approximate similarity between two geometric graphs is computed. In this chapter, we present an approach to error-tolerant graph matching using geometric graphs. We introduce the vertex distance or dissimilarity and edge distance between two geometric graphs and use it to compute graph distance. Finally, we use graph distance to perform exact and error-tolerant geometric graph matching.

A graph is said to be a geometric graph when each of the vertices is assigned a coordinate point. The geometric graph matching is the process of computing the similarity between two geometric graphs. Geometric graph matching is a particular type of graph matching in which every input graph is modeled as a geometric graph. Depending on the nature of the matching, the graph matching process is broadly classified into exact and error-tolerant graph matching. Exact graph matching requires a strict correspondence between nodes and edges of the two geometric graphs. It is like a geometric graph isomorphism, which in addition to preserving the structure of the two geometric graphs, also preserves the coordinates of the corresponding vertices.

Exact geometric graph matching although theoretically appealing, may not be useful in many real-world applications, as due to the presence of noise or distortion during the processing, the input graph data may be altered. In such situations, we use error-tolerant geometric graph matching which is also known as inexact graph matching due to its flexibility to accommodate errors during the process of matching [1].

As each vertex of a geometric graph has a unique coordinate representation, we can use this information along with other geometric transformation properties to perform geometric graph matching. In [32] authors have proposed an algorithm to compute geometric graph isomorphism in polynomial time. In [112] authors have shown geometric graph matching using edit distance method to be *NP*-hard. Approximate solution for geometric graph matching using vertex edit distance is described in [113]. In [114] authors have described geometric graph matching by applying Monte Carlo tree search.

The graph matching problem seems to be inherently linked to geometry and topology of graphs. In this chapter, we extend the work given in [115] and describe a simple yet promising framework for geometric graph similarity and matching. We describe a graph matching algorithm using geometric graphs, which is both error-tolerant as well as computable in polynomial time [87]. We define the vertex distance between two geometric graphs (not between two vertices) using the position of their vertices and show it to be metric over the set of all graphs with vertices only. We define edge distance between two graphs based on the angular orientation and length of the edges. We also define a modified version of edge distance, which uses the position of edges as an additional attribute, and demonstrate it to be a metric. Then we combine the notion of vertex distance and edge distance to define the graph distance between two geometric graphs and show some of its properties. Finally, we use this graph distance to perform error-tolerant graph matching on letter and AIDS dataset, in which all the nodes have a coordinate position in the two-dimensional plane [87].

This chapter is organized as follows. Section 5.2, provides preliminaries and motivation. Section 5.3, presents the novel geometric graph similarity framework. Section 5.4, introduce geometric graph matching based on the proposed geometric graph similarity framework. It describes both exact as well as error-tolerant graph matching. Section 5.5, contains an experimental evaluation of the proposed geometric graph matching scheme. Finally, section 5.6, provides a summary.

5.2 Preliminaries and Motivation

In this section, we review the basic definitions and notations used in this chapter. We also explain the motivation for the proposed work.

A *geometric graph* G is defined as $G = (V, E, l, c)$, where V is the set of nodes, E is the set of edges, l is a labeling function $l : \{V \cup E\} \rightarrow \Sigma$ which assigns a label from Σ to each vertex and edge, c is a function $c : V \rightarrow \mathbb{R}^2$ which assigns a coordinate point to each vertex of G. If $\Sigma = \emptyset$ then G is called the unlabeled geometric graph.

Let X be a nonempty set. A mapping $d : X \times X \rightarrow \mathbb{R}$ is defined to be a *metric* on X if d satisfy the following conditions: (i) $d(u,v) \geq 0$ for all $u, v \in X$ and $d(u,v) = 0$ if and only if $u = v$
(ii) $d(u,v) = d(v,u)$ for all $u, v \in X$ (Symmetry)
(iii) $d(u,w) \leq d(u,v) + d(v,w)$ for all $u, v, w \in X$ (Triangle inequality). The pair (X,d) is called a *metric space*.

In this chapter, we consider the problem of finding a similarity score between two geometric graphs. Given two geometric graphs G_1,G_2 in graph domain $\mathscr{G}$ of the two-dimensional space, we would like to find a distance $d : \mathscr{G} \times \mathscr{G} \rightarrow \mathbb{R}$ which computes the similarity between G_1 and G_2. For general graphs, it is also known as graph comparison problem. In this chapter, we consider graph comparison problem synonymous to graph matching problem.

Edit-distance-based approach to graph matching, as well as geometric graph matching, have no efficient exact solution. For example, graph edit distance [73] and geometric graph matching based on edit distances [112] are *NP*-hard and therefore no efficient algorithms are available. Since our goal is not to reconstruct G_2 from G_1, therefore we can avoid the intractability of edit distance based methods for the problem of geometric graph similarity. Instead, we use the Linear Sum Assignment Problem (LSAP) formulation-based approach to geometric graph distance using the various attributes of vertices and edges of the geometric graphs.

Geometric graph isomorphism, as opposed to graph isomorphism, can be solved efficiently in polynomial time [32] so we can take advantage of this to find the best geometric

representation of a geometric graph G_2 with respect to an input geometric graph G_1 which make them similar. We define geometric graph similarity which intuitively captures the minimum error required to make the two geometric graphs similar.

5.3 Geometric Graph Similarity

In this section, we introduce vertex distance, edge distance between the geometric graphs G_1 and G_2. We use these distance measures to compute the dissimilarity between two geometric graphs.

Initially, we consider two geometric graphs G_1 and G_2 with vertices only without connections. To find similarity between these graphs, one approach is to assign each vertex of G_1 to one of the vertices of G_2, so that the total distance is minimum. Suppose vertex coordinate points of G_1 be $\{(a_1,b_1),(a_2,b_2),\ldots,(a_n,b_n)\}$ and coordinate points of G_2 be $\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$, as shown in Figure 5.1. If $l_i = \{\sqrt{(a_i-x_j)^2+(b_i-y_j)^2} | 1 \leq j \leq n\}, \forall 1 \leq i \leq n$. Then vertex distance between G_1 and G_2 is $\sum_{i=1}^{n} l_i = \sum_{i=1}^{n} \min_{1\leq j\leq n}\{\sqrt{(a_i-x_j)^2+(b_i-y_j)^2}\}$. One limitation of this expression is that more than one vertices of G_1 can be assigned to the same vertex of G_2.

For a more realistic optimization problem formulation, let $C = (c_{ij})$ be a $n \times n$ cost matrix, where c_{ij} is the Euclidean distance from i-th vertex of G_1 to j-th vertex of G_2. We have, $c_{ij} = \sqrt{(a_i-x_j)^2+(b_i-y_j)^2}$. Now, we can minimize the objective function $\sum_{i=1}^{n} c_{i\varphi(i)}$ to get the minimum distance between G_1 and G_2 as $\min_{\varphi\in S_n} \sum_{i=1}^{n} c_{i\varphi(i)}$. Where S_n is the set of all assignments φ from V_1 of G_1 to V_2 of G_2. This is the standard LSAP formulation of vertex assignments from V_1 to V_2. The LSAP can be solved using Hungarian algorithm in $O(n^3)$ time [116].

In the following definition of vertex distance, we assume that the number of vertices in two graphs are equal. For arbitrary graph in which the number of vertices is not same, we can insert additional vertices in the smaller graph, with coordinates equal to mean of all its existing coordinate values, to make the vertex set of both graphs equal.

Definition 5.3.1. Let $G_1 = (V_1,E_1,c_1)$ and $G_2 = (V_2,E_2,c_2)$ be two geometric graphs with $|V_1| = |V_2| = n$. Let coordinate points of V_1 be $\{(a_1,b_1),(a_2,b_2),\ldots,(a_n,b_n)\}$ and

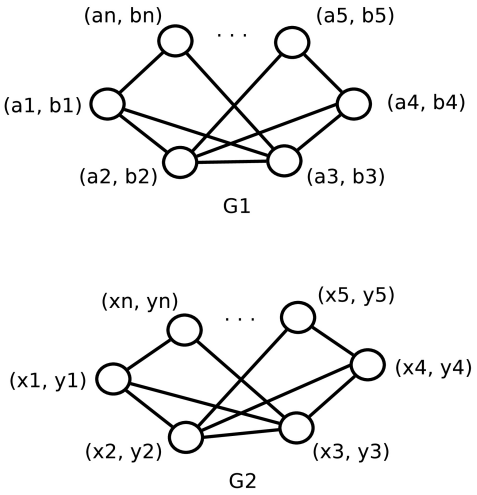

FIGURE 5.1: Geometric Graphs G_1 and G_2

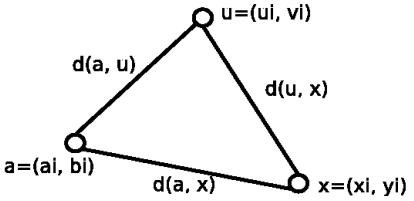

FIGURE 5.2: Triangle inequality: $d(a,x) \leq d(a,u)+d(u,x)$

coordinate points of V_2 be $\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$ then the vertex distance between the two geometric graphs G_1 and G_2 is defined as

$$VD(G_1,G_2) = \min_{\varphi \in S_n} \sum_{i=1}^{n} \sqrt{(a_i - x_{\varphi(i)})^2 + (b_i - y_{\varphi(i)})^2} \tag{5.1}$$

Where S_n and φ are as defined above.

Here, *VD* represents an LSAP formulation for the assignment of the vertex set V_1 of G_1 to the vertex set V_2 of G_2. The geometric graphs having a similar position of vertices will have less *VD*, whereas the graphs having a very different position of vertices have large *VD*. In the above definition, we can also represent vertex set V_1 and V_2 using their offsets from the mean position of all vertex coordinates of G_1 and G_2, respectively.

Lemma 5.3.1. *$VD(G_1,G_2)$ is a metric on the set of all geometric graphs $\mathscr{G}$ having edge set $E = \emptyset$.*

Proof. Property (i) of metric: Here by definition $VD(G_1,G_2) \geq 0$, If $G_1 = G_2$ then clearly $VD(G_1,G_2) = \min_{\varphi \in S_n} \sum_{i=1}^{n} [(a_i - x_{\varphi(i)})^2 + (b_i - y_{\varphi(i)})^2]^{1/2} = 0$, since in each component of summation we have $a_i = x_{\varphi(i)}$ and $b_i = y_{\varphi(i)}$.
Conversely, if $VD(G_1,G_2) = 0$ then $\min_{\varphi \in S_n} \sum_{i=1}^{n} [(a_i - x_{\varphi(i)})^2 + (b_i - y_{\varphi(i)})^2]^{1/2} = 0$ implies that for some combination of $i, \varphi(i)$ each term of summation
$\sum_{i=1}^{n} [(a_i - x_{\varphi(i)})^2 + (b_i - y_{\varphi(i)})^2]^{1/2}$ is 0,
$\Rightarrow (a_i - x_{\varphi(i)})^2 = 0$ and $(b_i - y_{\varphi(i)})^2 = 0$,
$\Rightarrow a_i = x_{\varphi(i)}$ and $b_i = y_{\varphi(i)}$. Hence $G_1 = G_2$
Property (ii):
$VD(G_1,G_2) = \min_{\varphi \in S_n} \sum_{i=1}^{n} [(a_i - x_{\varphi(i)})^2 + (b_i - y_{\varphi(i)})^2]^{1/2}$
$= \min_{\varphi \in S_n} \sum_{i=1}^{n} [(x_{\varphi(i)} - a_i)^2 + (y_{\varphi(i)} - b_i)^2]^{1/2} = VD(G_2,G_1)$
Reversibility of φ can also be demonstrated. Let $V_1 = u_1, u_2, \cdots, u_n$ and $V_1 = v_1, v_2, \cdots, v_n$. Let ϕ from $V_1 \rightarrow V_2$ using LSAP assign $u_1 \rightarrow v_1, u_2 \rightarrow v_2, \cdots, u_n \rightarrow v_n$. Now suppose ϕ^{-1} from $V_2 \rightarrow V_1$ assigns $v_1 \rightarrow x_1, v_2 \rightarrow x_2, \cdots, v_n \rightarrow x_n$. There are three cases to consider.
Case 1: Cost of ϕ from $V_1 \rightarrow V_2$ = cost of ϕ^{-1} from $V_2 \rightarrow V_1$ then $VD(G_1,G_2) = VD(G_2,G_1)$.
Case 2: Cost of ϕ from $V_1 \rightarrow V_2 >$ cost of ϕ^{-1} from $V_2 \rightarrow V_1$ Then $x_1 \neq u_1, x_2 \neq u_2, \cdots, x_n \neq u_n$. It is a contradiction since cost using ϕ was supposed to be minimum using LSAP. Here the cost is a distance. So ϕ could have selected x_1 instead of u_1, x_2 instead of u_2 and so on, to return a lower value.
Case 3: Cost of ϕ from $V_1 \rightarrow V_2 <$ cost of ϕ^{-1} from $V_2 \rightarrow V_1$. Arguments similar to Case 2 applies here.
Property (iii):
Let $G_3 = (V_3, E_3, c_3)$ be a geometric graph with $|V_3| = n$, $E_3 = \emptyset$ and coordinate points of V_3 be $\{(u_1,v_1),(u_2,v_2),\ldots,(u_n,v_n)\}$. Figure 5.2 illustrate geometrically the triangle inequality among the points $(a_i,b_i),(x_i,y_i)$ and (u_i,v_i) of the graphs G_1, G_2 and G_3 respectively. We can observe that
$VD(G_1,G_3) + VD(G_3,G_2)$
$= \min_{\varphi \in S_n} \sum_{i=1}^{n} [(a_i - u_{\varphi(i)})^2 + (b_i - v_{\varphi(i)})^2]^{1/2}$
$+ \min_{\varphi' \in S_n} \sum_{i=1}^{n} [(u_i - x_{\varphi'(i)})^2 + (v_i - y_{\varphi'(i)})^2]^{1/2}$
let the minimum function over all $\varphi(i)$ and $\varphi'(i)$ returns the following
$= \sum_{i=1}^{n} [(a_i - u_{\varphi(i)})^2 + (b_i - v_{\varphi(i)})^2]^{1/2} + \sum_{i=1}^{n} [(u_i - x_{\varphi'(i)})^2 + (v_i - y_{\varphi'(i)})^2]^{1/2}$
In the first term of the above expression, for $i = 1$, let $\varphi(1) = k$. In the second term suppose $u_{\varphi(1)} = u_k$ in G_3 is assigned to $x_{\varphi'(k)}$ for some $x \in G_2$. So, for $i = 1$ we have the

above expression
$[(a_1-u_{\varphi(1)})^2+(b_1-v_{\varphi(1)})^2]^{1/2}+[(u_k-x_{\varphi'(k)})^2+(v_k-y_{\varphi'(k)})^2]^{1/2}$
$=[(a_1-u_k)^2+(b_1-v_k)^2]^{1/2}+[(u_k-x_{\varphi'(k)})^2+(v_k-y_{\varphi'(k)})^2]^{1/2}$
let $p_1=a_1-u_k, p_2=b_1-v_k$,
$q_1=u_k-x_{\varphi'(k)}, q_2=v_k-y_{\varphi'(k)}$, then we have
$[(a_1-u_k)^2+(b_1-v_k)^2]^{1/2}+[(u_k-x_{\varphi'(k)})^2+(v_k-y_{\varphi'(k)})^2]^{1/2}$
$=[(p_1)^2+(p_2)^2]^{1/2}+[(q_1)^2+(q_2)^2]^{1/2}$
using Minkowski inequality
$\sum_{i=1}^{n}[(x_i+y_i)^2]^{1/2}\leq(\sum_{i=1}^{n}x_i^2)^{1/2}+(\sum_{i=1}^{n}y_i^2)^{1/2}$
and putting the values of p_1,q_1,p_2,q_2 we get
$[(p_1)^2+(p_2)^2]^{1/2}+[(q_1)^2+(q_2)^2]^{1/2}$
$\geq=[(p_1+q_1)^2+(p_2+q_2)^2]^{1/2}$
$=[(a_1-u_k+u_k-x_{\varphi'(k)})^2+(b_1-v_k+v_k-y_{\varphi'(k)})^2]^{1/2}$
$=[(a_1-x_{\varphi'(k)})^2+(b_1-y_{\varphi'(k)})^2]^{1/2}$
taking sum over all i-th term and a $\varphi''\in S_n$
$\sum_{i=1}^{n}[(a_i-x_{\varphi''(i)})^2+(b_i-y_{\varphi''(i)})^2]^{1/2}$
and taking minimum of summation over all i and a $\varphi''(i)$ we get
$\min_{\varphi''\in S_n}\sum_{i=1}^{n}[(a_i-x_{\varphi''(i)})^2+(b_i-y_{\varphi''(i)})^2]^{1/2}=VD(G_1,G_2)$
hence, $VD(G_1,G_3)+VD(G_3,G_2)\geq VD(G_1,G_2)$
Note that if either $G_3=G_1$ or $G_3=G_2$ then
$VD(G_1,G_3)+VD(G_3,G_2)=VD(G_1,G_2)$
and when $G_3\neq G_1$ and $G_3\neq G_2$ then
$VD(G_1,G_3)+VD(G_3,G_2)>VD(G_1,G_2)$.

□

VD considers only the position of vertices of geometric graphs in the plane. Now, we use the features of edges of geometric graphs to define the edge distance. The main features of an edge of a geometric graph include its orientation and length along with the coordinates of its endpoints.

Let $\theta_{\{(a,b),(c,d)\}}$ denote the angle subtended between the extended line joining the coordinate points $(a,b),(c,d)$ and positive x-axis. In the following definition of edge distance, we assume that the number of edges in the two graphs is equal. For graphs with

unequal edge set, we can append additional empty edges to the smaller graph, to make the total number of edges in both graphs same.

Definition 5.3.2. Let $G_1 = (V_1, E_1, c_1)$ and $G_2 = (V_2, E_2, c_2)$ be two geometric graphs with $|E_1| = |E_2| = m$. Here c_1 and c_2 are the set of coordinate points of vertex set V_1 and V_2 respectively. Then the edge distance between the two geometric graphs G_1 and G_2 is defined as

$$ED(G_1, G_2) = \min_{\varphi \in S_m} \sum_{i=1}^{m} c_{i\varphi(i)} \tag{5.2}$$

such that,

$$c_{ij} = \sqrt{((\Theta_i - \Theta_j)\frac{\pi}{180^\circ})^2} + \sqrt{(d_i - D_j)^2}$$

Where, $\Theta_i = \theta_{\{(a_i,b_i),(a_p,b_p)|((a_i,b_i),(a_p,b_p))\in E_1\}}$,
$\Theta_j = \theta_{\{(x_j,y_j),(x_q,y_q)|((x_j,y_j),(x_q,y_q))\in E_2\}}$,
$d_i = \{\sqrt{(a_i - a_p)^2 + (b_i - b_p)^2} | ((a_i, b_i), (a_p, b_p)) \in E_1\}$,
$D_j = \{\sqrt{(x_j - x_q)^2 + (y_j - y_q)^2} | ((x_j, y_j), (x_q, y_q)) \in E_2\}$
and S_m is the set of all assignments φ from E_1 of G_1 to E_2 of G_2.

The above definition of ED finds the best assignment of edges from E_1 to E_2. The cost function $C = (c_{ij})$ is itself made up of two components. The first term $\sqrt{((\Theta_i - \Theta_j)\frac{\pi}{180^\circ})^2}$ calculates difference in orientation of E_i in E_1 and E'_j in E_2. We denote this term of ED by E^A_{ij}, where A stands for an angular difference. It is translation and scaling invariant, as we can observe in Figure 5.3 that G_1 and G_4 have identical value of this term. The second term $\sqrt{(d_i - D_j)^2}$ of ED denotes the difference of length between E_i and E'_j. We call this component as the length term and represent it by E^L_{ij}, where L is used for the length of edges. E^L_{ij} is translation and rotation invariant, as we can see in Figure 5.3 that all four graphs G_1 to G_4 have the equivalent value of the second term of ED. Following properties of ED are easy to establish.

Lemma 5.3.2. *$ED(G_1, G_2)$ satisfy the following properties:*
(i) $ED(G_1, G_2) \geq 0$, if $G_1 = G_2$ then $ED(G_1, G_2) = 0$
(ii) $ED(G_1, G_2) = ED(G_2, G_1)$
(iii) $ED(G_1, G_2) \leq ED(G_1, G_3) + ED(G_3, G_2)$

Proof. (i) $ED(G_1, G_2) \geq 0$, since both terms in the definition of ED is always non-negative. Also when $G_1 = G_2$, we have $\Theta_i = \Theta_j$ and $d_i = D_j$ for each term of

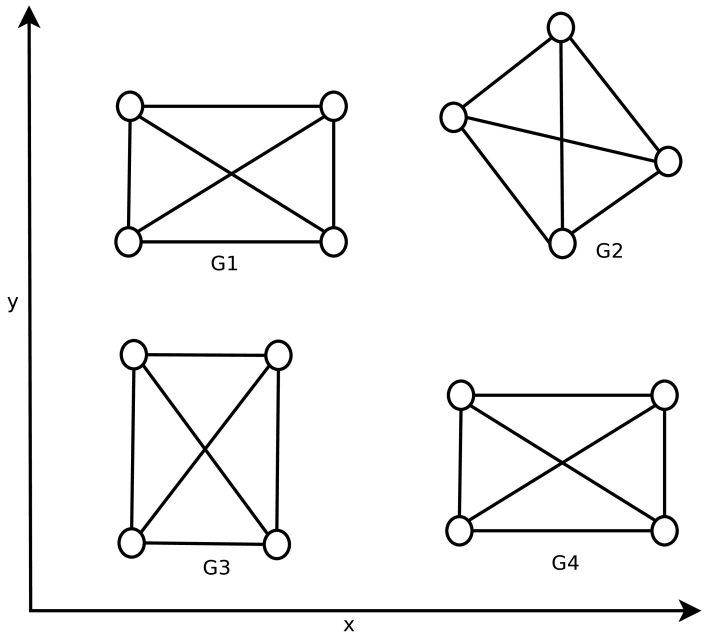

FIGURE 5.3: Edge Distance from G_1 to G_4 is 0

summation. Therefore $ED(G_1,G_2) = 0$. Note that converse may not be true i.e., $ED(G_1,G_2) = 0$ not necessarily imply $G_1 = G_2$. We can observe in Figure 5.3 that graphs G_2 and G_3 are rotation followed by translation of G_1, while G_4 is a translation of $G1$. Here $ED(G_1,G_4) = 0$. Thus ED between two translated version of a graph turns out to be 0.

(ii) $ED(G_1,G_2) = \min_{\varphi\in S_m}\sum_{i=1}^{m}([((\Theta_i - \Theta_{\varphi(i)})\frac{\pi}{180^\circ})^2]^{1/2} + [(d_i - D_{\varphi(i)})^2]^{1/2}) = \min_{\varphi\in S_m}\sum_{i=1}^{m}([((\Theta_{\varphi(i)} - \Theta_i)\frac{\pi}{180^\circ})^2]^{1/2} + [(D_{\varphi(i)} - d_i)^2]^{1/2}) = ED(G_2,G_1)$

(iii) Let $G_3 = (V_3,E_3,c_3)$ be a geometric graph with $|V_3| = n$ and coordinate points of V_3 be $\{(u_1,v_1),(u_2,v_2),\ldots,(u_n,v_n)\}$. Suppose $\Theta_k = \theta_{\{(u_k,v_k),(u_r,v_r)|(u_k,v_k),(u_r,v_r)\in E_3\}}$ and $D_k = \{\sqrt{(u_k-u_r)^2+(v_k-v_r)^2}|(u_k,v_k),(u_r,v_r)\in E_3\}$, then $ED(G_1,G_3) + ED(G_3,G_2)$

$= \min_{\varphi\in S_m}\sum_{i=1}^{m}([((\Theta_i-\Theta_k)\frac{\pi}{180^\circ})^2]^{1/2} + [(d_i-D_k)^2]^{1/2})$

$\quad + \min_{\varphi\in S_m}\sum_{k=1}^{m}([((\Theta_k-\Theta_j)\frac{\pi}{180^\circ})^2]^{1/2} + [(D_k-D_j)^2]^{1/2})$

$= \min_{\varphi\in S_m}\sum_{i=1}^{m}(|(\Theta_i-\Theta_k)\frac{\pi}{180^\circ}| + |d_i-D_k|)$

$\quad + \min_{\varphi\in S_m}\sum_{k=1}^{m}(|(\Theta_k-\Theta_j)\frac{\pi}{180^\circ}| + |D_k-D_j|)$

Let the minimum function over all i and $\varphi(i)$ returns the following

$\sum_{i=1}^{m}(|(\Theta_i-\Theta_k)\frac{\pi}{180^\circ}| + |d_i-D_k|) + \sum_{k=1}^{m}(|(\Theta_k-\Theta_j)\frac{\pi}{180^\circ}| + |D_k-D_j|)$

now, using triangle inequality $|x| + |y| \geq |x+y|$

and inequality of summation $\sum_{x\in X}|f(x)| \geq |\sum_{x\in X} f(x)|$

$\sum_{i=1}^{m}(|(\Theta_i-\Theta_k)\frac{\pi}{180^\circ}| + |d_i-D_k|) + \sum_{k=1}^{m}(|(\Theta_k-\Theta_j)\frac{\pi}{180^\circ}| + |D_k-D_j|)$

$= \sum_{i=1}^{m}|(\Theta_i-\Theta_k)\frac{\pi}{180^\circ}| + \sum_{i=1}^{m}|d_i-D_k| + \sum_{k=1}^{m}|(\Theta_k-\Theta_j)\frac{\pi}{180^\circ}| + \sum_{k=1}^{m}|D_k-D_j|$

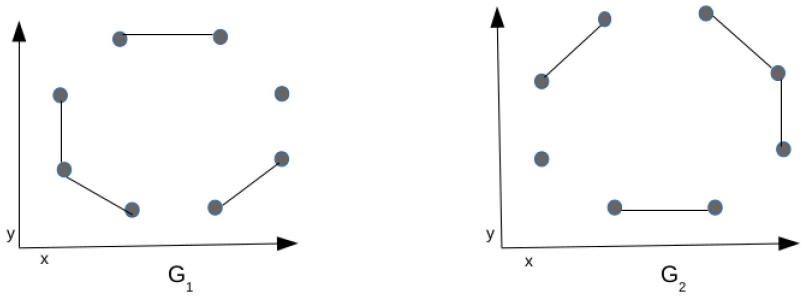

FIGURE 5.4: Geometric graphs with VD and ED both 0

$\geq \sum_{i,k=1}^{m}(|(\Theta_i-\Theta_k+\Theta_k-\Theta_j)\frac{\pi}{180^\circ}|+|d_i-D_k+D_k-D_j|)$
$=\sum_{i=1}^{m}(|(\Theta_i-\Theta_j)\frac{\pi}{180^\circ}|+|d_i-D_j|)$
and therefore $ED(G_1,G_3)+ED(G_3,G_2)\geq ED(G_1,G_2)$.

□

We observe that VD express the relative position of the vertices between two geometric graphs, whereas ED captures the orientation and length of edges between two geometric graphs. Nevertheless, ED does not encapsulate the relative position of edges of graphs in the two-dimensional plane. For example in Figure 5.4, two geometric graphs G_1 and G_2 are shown such that $VD(G_1,G_2)=ED(G_1,G_2)=0$ even though geometrically $G_1\neq G_2$, because ED does not capture the relative position of edges. To accommodate the relative position of edges, we introduce a third term in the definition of ED called position component denoted by E_{ij}^P.

Let i and j denote the i-th and j-th edge of G_1 and G_2 respectively, such that i-th edge of G_1 is $\{(a_i,b_i),(a_p,b_p)\}$ and j-th edge of G_2 is $\{(x_j,y_j),(x_q,y_q)\}$. We define the position term of ED by

$$E_{ij}^P=(\sqrt{(a_i-x_j)^2+(b_i-y_j)}^2+\sqrt{(a_p-x_q)^2+(b_p-y_q)}^2)/2$$

The term E_{ij}^P is similar to VD, which associate the position of one vertex of G_1 to another vertex of G_2. E_{ij}^P compares position of one edge of G_1 to another edge of G_2. Now, we define Edge Distance Modified or Edge Distance Metric (EDM), which use an objective function to combine all three features of edges.

Definition 5.3.3. Let $G_1 = (V_1, E_1, c_1)$ and $G_2 = (V_2, E_2, c_2)$ be two geometric graphs with $|E_1| = |E_2| = m$. Then the edge distance metric between the two geometric graphs G_1 and G_2 is defined as

$$EDM(G_1, G_2) = \min_{\varphi \in S_m} \sum_{i=1}^{m} c_{i\varphi(i)} \tag{5.3}$$

such that,

$$c_{ij} = E_{ij}^A + E_{ij}^L + E_{ij}^P$$

Where, $E_{ij}^A = \sqrt{((\Theta_i - \Theta_j)\frac{\pi}{180^\circ})^2}$, $E_{ij}^L = \sqrt{(d_i - D_j)^2}$,
$E_{ij}^P = (\sqrt{(a_i - x_j)^2 + (b_i - y_j)}^2 + \sqrt{(a_p - x_q)^2 + (b_p - y_q)}^2)/2$
and S_m is the set of all assignments φ from E_1 of G_1 to E_2 of G_2.

Lemma 5.3.3. *$EDM(G_1, G_2)$ is metric over the set of all geometric graphs $\mathcal{G}$ without isolated vertices.*

Proof. Property (i) of metric: $EDM(G_1, G_2) \geq 0$, since E_{ij}^A, E_{ij}^L, and E_{ij}^P are all non-negative. Now, when $G_1 = G_2$, we have $c_{ij} = E_{ij}^A + E_{ij}^L + E_{ij}^P = 0$, hence $EDM(G_1, G_2) = 0$. Conversely, suppose $EDM(G_1, G_2) = 0$. Then for some $\varphi \in S_m$, we get $\sum_{i=1}^{m} c_{i\varphi(i)} = 0$. So, $E_{ij}^A = E_{ij}^L = E_{ij}^P = 0$. We notice that $E_{ij}^A = 0$ implies $\Theta_i = \Theta_j^{'}$ and since E_{ij}^A is translation and scaling invariant therefore $E_{ij}^A = 0$ indicates that rotation is preserved between G_1 and G_2. Also E_{ij}^L is translation and rotation invariant so $E_{ij}^L = 0$ denotes that scaling is preserved. Similarly, $E_{ij}^P = 0$ implies $a_i = x_j, b_i = y_j, a_p = x_q, b_p = y_q$ and therefore translation is also preserved between G_1 and G_2. Hence $EDM(G_1, G_2) = 0$ implies that $G_1 = G_2$ as long as G_1 and G_2 does not have 0-degree vertices. For example, if the position of the isolated vertex of graph G_1 in Figure 5.4 is changed to construct another graph G_1' then also we get $EDM(G_1, G_1') = 0$.
Property (ii): In the definition of EDM, we have $c_{ij} = E_{ij}^A + E_{ij}^L + E_{ij}^P = E_{ji}^A + E_{ji}^L + E_{ji}^P = c_{ji}$. Hence $EDM(G_1, G_2) = EDM(G_1, G_2)$.
Property (iii): Let $G_3 = (V_3, E_3, c_3)$ be a geometric graph as defined in Lemma 2 having $|E_3| = m$, then
$EDM(G_1, G_3) + EDM(G_3, G_2)$
$= \min_{\varphi \in S_m} \sum_{i=1}^{m} ([((\Theta_i - \Theta_k)\frac{\pi}{180^\circ})^2]^{1/2} + [(d_i - D_k)^2]^{1/2} + (\sqrt{(a_i - u_k)^2 + (b_i - v_k)}^2 + \sqrt{(a_p - u_r)^2 + (b_p - v_r)}^2)/2)$
$+ \min_{\varphi \in S_m} \sum_{k=1}^{m} ([((\Theta_k - \Theta_j)\frac{\pi}{180^\circ})^2]^{1/2} + [(D_k - D_j)^2]^{1/2} + (\sqrt{(u_k - x_j)^2 + (v_k - y_j)}^2 + \sqrt{(u_r - x_q)^2 + (v_r - y_q)}^2)/2)$

$= \min_{\varphi \in S_m} \sum_{i=1}^{m} (E_{ik}^A + E_{ik}^L + E_{ik}^P) + \min_{\varphi \in S_m} (\sum_{k=1}^{m} E_{kj}^A + E_{kj}^L + E_{kj}^P)$ using Lemma 2 and the fact that each E_{ij}^P is itself a Euclidean distance function, we have
$\min_{\varphi \in S_m} \sum_{i,k=1}^{m} [(E_{ik}^A + E_{kj}^A) + (E_{ik}^L + E_{kj}^L) + (E_{ik}^P + E_{kj}^P)]$
$\geq \sum_{i=1}^{m} (E_{ij}^A + E_{ij}^L + E_{ij}^P) = EDM(G_1, G_2)$ □

For the purpose of showing the triangle inequality property of VD, ED or EDM; we can use the properties of their cost matrix. We can observe that $EDM(G_1, G_2)$ is the minimum cost of assignment of edges from G_1 to G_2 using its cost function. Similarly $EDM(G_1, G_3)$ and $EDM(G_3, G_2)$ are the minimum cost of selecting edges from G_1 to G_3 and G_3 to G_2 to respectively. So, $EDM(G_1, G_3) + EDM(G_3, G_2)$ is a valid assignment from G_1 to G_2. Since, $EDM(G_1, G_2)$ pertains to assignment of edges from G_1 to G_2 with minimum cost, we have $EDM(G_1, G_2) \leq EDM(G_1, G_3) + EDM(G_3, G_2)$.

Graph distance between two geometric graphs is defined as a linear combination of both VD and ED for the same pair of graphs.

Definition 5.3.4. Let $G_1 = (V_1, E_1, c_1)$ and $G_2 = (V_2, E_2, c_2)$ be two geometric graphs with $|V_1| = |V_2| = n$ and $|E_1| = |E_2| = m$. Then the graph distance between the two geometric graphs G_1 and G_2 is defined as

$$GD(G_1, G_2) = VD(G_1, G_2) + ED(G_1, G_2). \tag{5.4}$$

GD combines both VD and ED. We call the first component in the definition of GD as VD-term and the second component as ED-term.

Theorem 5.3.4. *$GD(G_1, G_2)$ satisfy the following properties:*
(i) $GD(G_1, G_2) \geq 0$, if $G_1 = G_2$ then $GD(G_1, G_2) = 0$
(ii) $GD(G_1, G_2) = GD(G_2, G_1)$
(iii) $GD(G_1, G_2) \leq GD(G_1, G_3) + GD(G_3, G_2)$

The above follows from Lemma 5.3.1 and Lemma 5.3.2. Since both VD-term and ED-term separately satisfy the above properties.

Definition 5.3.5. Let $G_1 = (V_1, E_1, c_1)$ and $G_2 = (V_2, E_2, c_2)$ be two geometric graphs with $|V_1| = |V_2| = n$ and $|E_1| = |E_2| = m$. Then the Graph Distance Modified or Graph Distance

Metric (*GDM*) between the two geometric graphs G_1 and G_2 is defined as

$$GDM(G_1, G_2) = VD(G_1, G_2) + EDM(G_1, G_2). \tag{5.5}$$

In the above definition of graph distance modified we use EDM instead of ED for the computation of edge distance between two graphs.

Theorem 5.3.5. *$GDM(G_1, G_2)$ is a metric over set of all geometric graphs $\mathscr{G}$.*

Proof. From Lemma 5.3.1 and Lemma 5.3.3, we observe that both VD-term and EDM-term of GDM satisfy properties (ii), (iii) of metric. Now, we consider property (i):
Here certainly, $GDM(G_1, G_2) \geq 0$ since $VD(G_1, G_2) \geq 0$ and $EDM(G_1, G_2) \geq 0$.
If $G_1 = G_2$ then $VD(G_1, G_2) = 0$ and $EDM(G_1, G_2) = 0$ and therefore $GDM(G_1, G_2) = 0$.
Now if $GDM(G_1, G_2) = 0$ then both VD-term and EDM-term are 0. When VD-term is 0, i.e., $VD(G_1, G_2) = 0$, then from Lemma 1, $G_1 = G_2$ without considering the links, in other words positions of all matched vertices between G_1 and G_2 coincide. When EDM-term is 0, then from Lemma 3, $E^A_{ij} = E^L_{ij} = E^P_{ij} = 0$ and therefore, angular orientation, length and positions of each matched edges are exactly the same. By combining the above two facts we get $G_1 = G_2$, whenever $GDM(G_1, G_2) = 0$. □

5.4 Geometric Graph Matching

5.4.1 Exact Geometric Graph Matching

In this section, we describe an algorithm to check whether two geometric graphs are isomorphic. Before checking the isomorphism between two geometric graphs, their reference coordinate must be identical. Therefore, first, we carry out graph alignment of two input graphs so that their reference coordinates become identical. Steps to perform graph alignment of G_2 with respect to G_1 is described in Algorithm 6. The input to this algorithm is two geometric graphs G_1 and G_2 and output is the geometric transform of G_2 having the maximum number of edges having similar orientation to G_1. Line 1 of this algorithm selects an edge e of E_1 to be a reference axis. The *for* loop in line 2 of the

algorithm choose an edge f of E_2 to perform geometric-transform on G_2. The function geometric-transform process the coordinates of G_2 using a transformation matrix to make edge f to have same orientation to reference axis e with the left coordinate of edge f coincident to left coordinate of e. This particular geometric configuration of G_2 will be translation and rotation invariant to $G1$. To make this geometric configuration scaling invariant also, we make the edges e and f of unit length and uniformly scale the other edges of G_1 and G_2. Line 4 of the algorithm computes ED between G_1 and G_2' and in line 6, the algorithm updates d_{min} to find the minimum ED over all the geometric configuration of G_2. Line 10 of the algorithm returns the graph G corresponding to minimum ED value.

Algorithm 6 :Graph-Alignment(G_1, G_2)

Input: Two undirected geometric graphs G_1, G_2, where $G_1 = (V_1, E_1, c_1), G_2 = (V_2, E_2, c_2)$ with $|V_1| = |V_2| = n, |E_1| = |E_2| = |m|$

Output: Geometric transform of G_2 with maximum edges aligned to G_1

```
1:  Select an edge e ∈ E1 as a reference axis
2:  for (each edge f ∈ E2) do
3:      G2' ← geometric-transform(G2)
4:      d ← ED(G1, G2')
5:      if (d < dmin) then
6:          dmin ← d
7:          G ← G2'
8:      end if
9:  end for
10: return (G)
```

Test to perform geometric graph isomorphism between two graphs is presented in Algorithm 7. The input to this algorithm is two geometric graphs G_1 and G_2. Output to this algorithm is isomorphic graphs when G_1 is isomorphic to G_2; t-tolerant isomorphic graphs when G_1 is t-tolerant isomorphic to G_2; and graph distance GD when G_1 is neither isomorphic to G_2 nor t-tolerant isomorphic to G_2. Line 1 of Geometric-Graph-Isomorphism algorithm calls Algorithm 6 to perform graph alignment of G_2 with respect to G_1. Vertex assignments from G_1 to G_2 using the definition of VD is given in line 2. We can use the Hungarian algorithm for optimal assignment of vertices. Based on the assignment VD is calculated in line 3. Angle and length component of ED is initialized in lines 4–5. Line 7 computes ED based on edge assignments from E_1 to E_2. Final GD value is calculated in line 8. The *if* loop in line 9 checks whether GD is 0, and line 10 also check uniqueness of edge assignments which ensures that each assigned edge from G_1 to G_2 connects the same set of coordinate points there by implying a valid geometric graph isomorphism. The algorithm

then returns in line 11, indicating that G_1 is isomorphic to G_2. The *for* loop in the lines 13–15 of algorithm checks if all the assigned coordinates from G_1 to G_2 are within distance t, and again it checks uniqueness of edge assignments, it then returns in line 17 showing that G_1 is t-tolerant isomorphic to G_2. When GD is neither 0 nor the assigned coordinates form G_1 to G_2 are within specified distance t, the algorithm outputs the GD value in line 19.

Algorithm 7 :Geometric-Graph-Isomorphism(G_1, G_2)

Input: Two undirected geometric graphs G_1, G_2, where $G_1 = (V_1, E_1, c_1), G_2 = (V_2, E_2, c_2)$ with $|V_1| = |V_2| = n, |E_1| = |E_2| = |m|$

Output: Isomorphic graphs or t-tolerant isomorphic graphs or graph distance between G_1 and G_2

1: $G_2 \leftarrow$ Graph-Alignment(G_1, G_2)
2: Compute vertex assignment from V_1 to V_2
3: $VD \leftarrow \sum_{i=1}^{n} \sqrt{(a_i - x_{\varphi(i)})^2 + (b_i - y_{\varphi(i)})^2}$
4: $E^A_{ij} \leftarrow \sqrt{((\Theta_{ij} - \Theta'_{ij})\frac{\pi}{180^\circ})^2}$
5: $E^L_{ij} \leftarrow \sqrt{(d_{ij} - D_{ij})^2}$
6: Compute edge assignment from E_1 to E_2
7: $ED \leftarrow \sum_{i=1}^{n} (E^A_{i\varphi(i)} + E^L_{i\varphi(i)})$
8: $GD \leftarrow VD + ED$
9: **if** $(GD == 0)$
10: Check uniqueness of edge assignments **then**
11: **return** G_1 is isomorphic to G_2
12: **else if**
13: **for** each $i = 1$ to n **do**
14: $(a_i - x_i) < t$ **and** $(b_i - y_i) < t$
15: **end for**
16: Check uniqueness of edge assignments **then**
17: **return** G_1 is t-tolerant isomorphic to G_2
18: **else**
19: **return** (GD)
20: **end if**

Proposition 5.4.1. *Graph-Alignment algorithm executes in* $O(m.n^3)$ *time.*

The geometric-transform operation in line 3 of Algorithm 7 takes $O(n^2)$ time, once a transformation matrix is constructed. The EDM operation in line 4 is performed in $O(n^3)$ time, hence total time taken by this algorithm is $O(m.n^3)$.

Proposition 5.4.2. *Geometric-Graph-Isomorphism algorithm executes in* $O(n^3 + m^3)$ *time exclusive of Algorithm 6 in line 1.*

The assignment of vertices in line 2 can be performed in $O(n^3)$ using Munkres or Hungarian algorithm. Similarly, the assignment of edges in line 6 can be achieved in $O(n^3)$. Remaining steps can be computed in $O(n^2)$ time. Therefore overall execution time becomes $O(n^3+m^3)$ time.

5.4.2 Error-Tolerant Geometric Graph Matching

The computation of graph distance between two geometric graphs G_1 and G_2 is described in Algorithm 8. The input to the algorithm is two undirected geometric graphs G_1 and G_2 and four weighting parameters w_1, w_2, w_3 and w_4, which are application dependent. G_1 has n_1 vertices and m_1 edges, whereas G_2 has n_2 vertices and m_2 edges. For simplicity, we have assumed that $n_1 \geq n_2$. The output of the algorithm is the geometric graph distance between G_1 and G_2.

When the number of vertices in $|V_1| = n_1$ is greater than that of $|V_2| = n_2$ then line 1 of the Geometric-Graph-Distance algorithm computes the mean values of x and y coordinate of graph G_2 as $x = (\sum_{i=1}^{n_2} x_i)/n_2$, $y = (\sum_{i=1}^{n_2} y_i)/n_2$ and $n_1 - n_2$ vertices with these coordinate positions are appended to V_2 in line 4 of the *for* loop in lines 3–5 to make $|V_1| = |V_2| = n$. Similarly, *if* loop in lines 7–11 check the number of edges in G_1 and G_2 and it makes them equal by appending empty edges with length and angle value both 0 to E_2 or E_1 depending on whether $m_1 > m_2$ or $m_1 < m_2$ respectively.

One optional step of this algorithm in line 12 is the preprocessing phase for graph alignment to make their reference coordinates identical. Geometric-Graph-Distance algorithm computes vertex assignment from V_1 to V_2 based on the cost function given in the definition of VD in line 13. We can use Munkres or Hungarian algorithm for optimal assignment of vertices. Based on the assignment VD is calculated in line 14. Similarly, line 15 computes edge assignment from E_1 to E_2 using the cost function given in the definition of EDM. Line 19 computes EDM by multiplying weighting factor w_2 to orientation term E_{ij}^A, w_3 to length term E_{ij}^L and w_4 to position term E_{ij}^P. Finally, line 20 computes GDM by multiplying w_1 to VD and adding it to EDM.

Steps to perform graph alignment of G_2 with respect to G_1 is described in Graph-Alignment-Modified algorithm. This algorithm is same as Graph-Alignment

Algorithm 8 :Geometric-Graph-Distance(G_1, G_2)

Input: Two undirected geometric graphs G_1, G_2, where $G_1 = (V_1, E_1, c_1), |V_1| = n_1, |E_1| = m_1, G_2 = (V_2, E_2, c_2), |V_2| = n_2, |E_2| = |m_2|$ and weighting factors w_i for $i = 1$ to 4; assume $n_1 \geq n_2$

Output: Geometric graph distance between G_1 and G_2

1: $(x, y) \leftarrow ((\sum_{i=1}^{n_2} x_i)/n_2, (\sum_{i=1}^{n_2} y_i)/n_2)$
2: **if** $(n_1 > n_2)$ **then**
3: **for** $i = 0$ to $(n_1 - n_2)$ **do**
4: $V_2 \leftarrow V_2 \cup \{(x, y)\}$
5: **end for**
6: **end if**
7: **if** $(m_1 > m_2)$ **then**
8: Append $(m_1 - m_2)$ empty edges to E_2
9: **else**
10: Append $(m_2 - m_1)$ empty edges to E_1
11: **end if**
12: $G_2 \leftarrow$ Graph-Alignment-Modified(G_1, G_2) ▷ preprocessing step
13: Compute vertex assignment φ from V_1 to V_2
14: $VD \leftarrow \sum_{i=1}^{n} \sqrt{(a_i - x_{\varphi(i)})^2 + (b_i - y_{\varphi(i)})^2}$
15: Compute edge assignment φ from E_1 to E_2
16: $E_{ij}^A \leftarrow \sqrt{((\Theta_i - \Theta_j)\frac{\pi}{180^\circ})^2}$; $E_{ij}^L \leftarrow \sqrt{(d_i - D_j)^2}$
17: $E_{ij}^P \leftarrow \sqrt{(a_i - x_j)^2 + (b_i - y_j)}^2 + \sqrt{(a_p - x_q)^2 + (b_p - y_q)}^2$
18: $E_{ij}^P \leftarrow (E_{ij}^P)/2$
19: $EDM \leftarrow \sum_{i=1}^{n} (w_2 \cdot E_{i\varphi(i)}^A + w_3 \cdot E_{i\varphi(i)}^L + w_4 \cdot E_{i\varphi(i)}^P)$
20: $GDM \leftarrow w_1 \cdot VD + EDM$
21: **return** (GDM)

algorithm except that it calls $EDM(G_1, G_2')$ instead of $ED(G_1, G_2')$ in line 4 to find the minimum value of d.

Proposition 5.4.3. *Geometric-Graph-Distance algorithm executes in* $O(n^3 + m^3)$ *time.*

The assignment of vertices in line 13 can be performed in $O(max\{n_1, n_2\}^3)$ using Munkres or Hungarian algorithm. Similarly, the assignment of edges in line 15 can be achieved in $O(max\{m_1, m_2\}^3)$. Remaining steps can be computed in $O(n^2)$ time. Let $n = max(n_1, n_2)$ and $m = max(m_1, m_2)$. Therefore overall execution time becomes $O(n^3 + m^3)$ time.

Proposition 5.4.4. *Graph-Alignment-Modified algorithm executes in* $O(m_2.n^3)$ *time.*

Algorithm 9 :Graph-Alignment-Modified(G_1, G_2)

Input: Two undirected geometric graphs G_1, G_2, where $G_1 = (V_1, E_1, c_1), |V_1| = n_1, |E_1| = m_1, G_2 = (V_2, E_2, c_2), |V_2| = n_2, |E_2| = |m_2|$

Output: Geometric transform of G_2 having maximum edges and vertices aligned to G_1

1: Select an edge $e \in E_1$ as a reference axis
2: **for** (each edge $f \in E_2$) **do**
3: $G_2' \leftarrow$ geometric-transform(G_2)
4: $d \leftarrow EDM(G_1, G_2')$
5: **if** $(d < d_{min})$ **then**
6: $d_{min} \leftarrow d$
7: $G \leftarrow G_2'$
8: **end if**
9: **end for**
10: **return** (G)

The *for* loop in line 2 of Algorithm 9 executes m_2 times, where $|E_2| = |m_2|$. The geometric-transform operation in line 3 takes $O(n_2^2)$ time; once a transformation matrix is constructed, where $|V_2| = n_2$. The EDM operation in line 4 is performed in $O(n^3)$ time, where $n = max(n_1, n_2)$. Hence total time taken by this algorithm is $O(m_2.n^3)$.

5.5 Experimental Evaluation

5.5.1 Exact Geometric Graph Matching

To assess the effectiveness Geometric-Graph-Isomorphism algorithm, we apply it for exact graph matching. We use letter dataset of IAM graph database repository for the demonstration of exact graph matching [109]. Each node in letter graphs is labeled with an (x, y) coordinate to represent its location in the two-dimensional plane.

Depending on three *if* loop conditions in the Geometric-Graph-Isomorphism algorithm, there are three cases to be considered. In the first case, we take several geometric graphs, apply composite geometric transformation using the translation, rotation, scaling and their combination to these sample graphs and verify whether the concerned pair is geometrically isomorphic using Geometric-Graph-Isomorphism algorithm. In the second case, we apply random distortion within a specified range to the vertices of the sample graphs and check

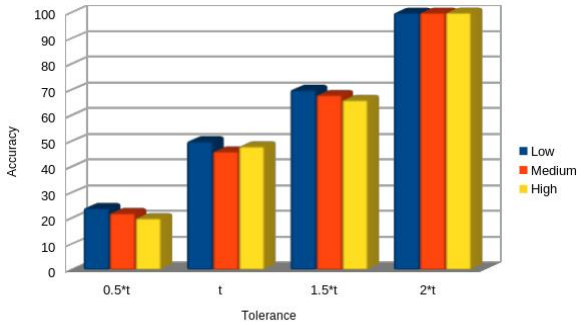

FIGURE 5.5: t-tolerant geometric graph isomorphism

whether they are t-tolerant isomorphic with respect to each other. Finally, in the third case, we apply the proposed algorithm to low, medium and high distortions of letter dataset to compute the graph matching, and compare their relative graph distances.

In the first set of experiments, we take letter graphs from each of low, medium and high class and apply an arbitrary transformation to the coordinate of vertices. We then apply the Geometric-Graph-Isomorphism algorithm to each pair of the original and transformed graph to verify the isomorphism. Since the Graph-Alignment algorithm makes the reference coordinate of two graphs identical, it undoes the transformations applied on the processed letter graph to output the two graphs to be isomorphic in each test.

In the second set of experiments, we select letter graphs from each of low, medium, and high class, and apply random distortions from 0 to t distance on each coordinate of vertices of the graph. We check t-tolerant isomorphism for each pair of graphs for four different values of t which are $t/2, t, 3t/2, 2t$. Comparison of accuracy of t-tolerant isomorphism for different values of t, when Geometric-Graph-Isomorphism algorithm is applied to letter graphs of low, medium and high distortion classes is shown in Figure 5.5. Here we observe that for the value of $2t$ every pair of graphs is correctly identified to be isomorphic.

In the third set of experiments, we apply Geometric-Graph-Isomorphism algorithm on each letter dataset to perform exact graph matching. For this experiment, we select graphs of equal vertices and edges. Based on the *GD* value returned by the algorithm, we compute the classification accuracy of letter dataset using the nearest neighborhood classifier. Classification accuracy of each of the fifteen letters of letter graphs of low distortion class is shown in Figure 5.6.

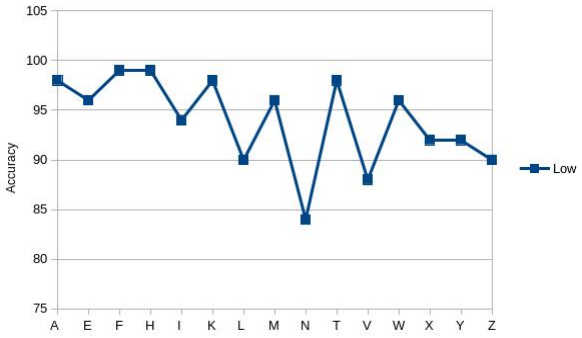

FIGURE 5.6: Accuracy Low

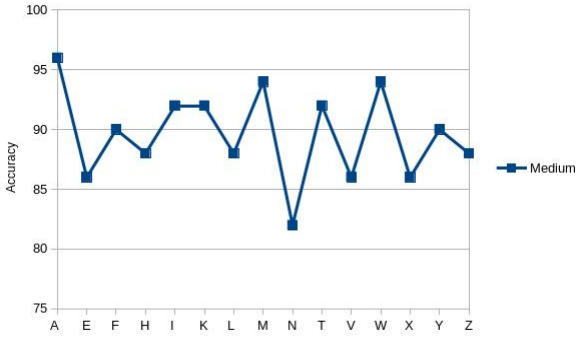

FIGURE 5.7: Accuracy Medium

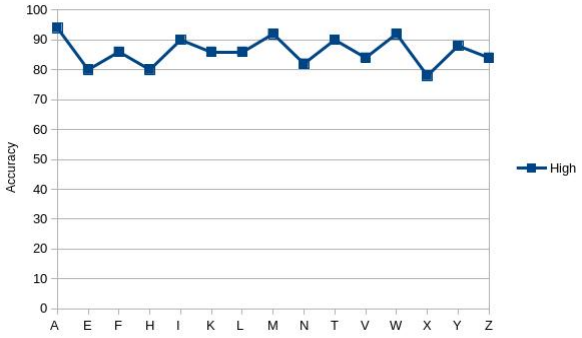

FIGURE 5.8: Accuracy High

Similarly, classification accuracy of each letters for medium and high distortion class of letter dataset is shown in Figure 5.7 and Figure 5.8 respectively.

The comparison of classification accuracy of each of the low, medium, and high class of letter dataset is shown in Figure 5.9. Here we observe that the accuracy of letter dataset of the low class is higher than that of medium class, which in turn is higher than that of letter dataset of high distortion.

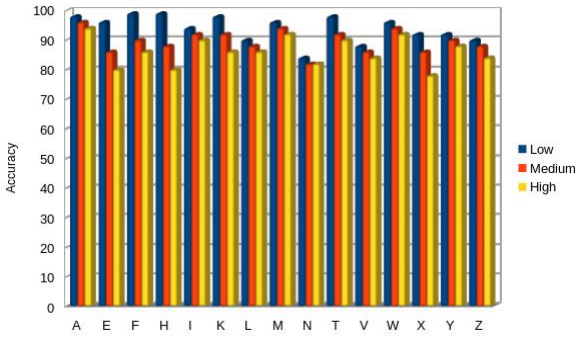

FIGURE 5.9: Accuracy

TABLE 5.1: Comparison of average computation time in *ms*

Algorithm	T_{Letter}	Algorithm	T_{AIDS}
Geometric	5.7	Geometric	1.39
Beam	4.5	Beam	6.48
GED	8.1	Bipartite	12.11

5.5.2 Error-Tolerant Geometric Graph Matching

To evaluate the usefulness of the geometric graph similarity framework, we apply it for inexact graph matching task. In this section, we use Geometric-Graph-Distance algorithm for geometric graph matching. We use the letter dataset and AIDS dataset of IAM graph database repository [109] for comparison of execution time and accuracy of geometric graph matching with other important graph matching algorithms. Each node in letter graphs is labeled with an (x,y) coordinate to represent its location in the two-dimensional plane. Every node of AIDS graphs also has its associated coordinate position.

5.5.2.1 Execution Time Comparison

Results in this section are computed using a system having 9.8 GB of memory and running the processor at 3.40 GHz. For comparison purpose, we use A^* based graph edit distance, beamsearch [75] with beam width $s = 10$, and bipartite graph matching [76]. Comparison of the average running time of graph matching methods in milliseconds for geometric graph matching, beamsearch having beam width $w = 10$, and GED is shown in Table 5.1.

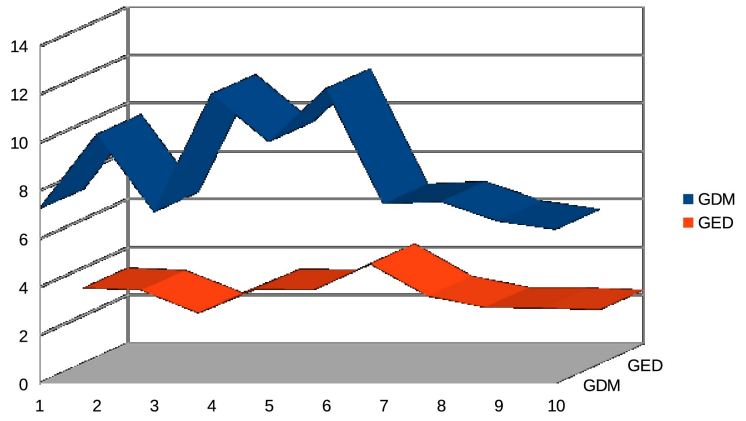

FIGURE 5.10: GDM vs GED for high distortion letter graphs

We may not use A^* based graph edit distance on AIDS dataset, as it usually becomes infeasible beyond 10 to 20 nodes. Comparison of the average execution time of graph matching in milliseconds for geometric graph matching, beamsearch and bipartite graph matching is shown in Table 5.1. On an average geometric graph matching takes less execution time than either beamsearch optimization or bipartite graph matching heuristic.

5.5.2.2 Accuracy Comparison

To compare the *GDM* computed by Algorithm 8 to the GED for the same pair of graphs, we compute these values for the first few letter graphs from each of high, medium and low distortion letter graphs as shown in Figure 5.10, Figure 5.11, and Figure 5.12 respectively. We note that *GDM* and GED values are more similar in low distortion letter graphs differing only by a shift as compared to high distortion letter graphs. It implies that in lower distortion, the two distance measures seems to correlate more than that with higher distortion levels. Also, the accuracy for low distortion letter dataset is higher than the accuracy for high distortion dataset. It shows that the proposed model is more accurate for unmodified graphs.

For accuracy evaluation of geometric graph matching, we consider the problem of classification of letter graphs and AIDS graphs using nearest neighbor classifier. For accuracy evaluation of letter dataset, we use high distortion letter graphs consisting of 50 training, validation and test graphs for each of the 15 classes of letters. We have trained the proposed model using the training set, optimized the weight parameters using the

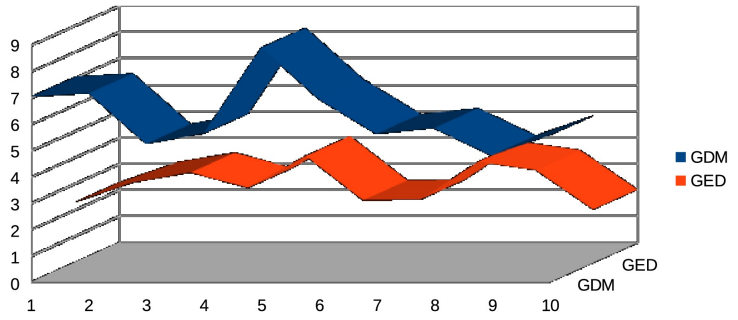

FIGURE 5.11: GDM vs GED for medium distortion letter graphs

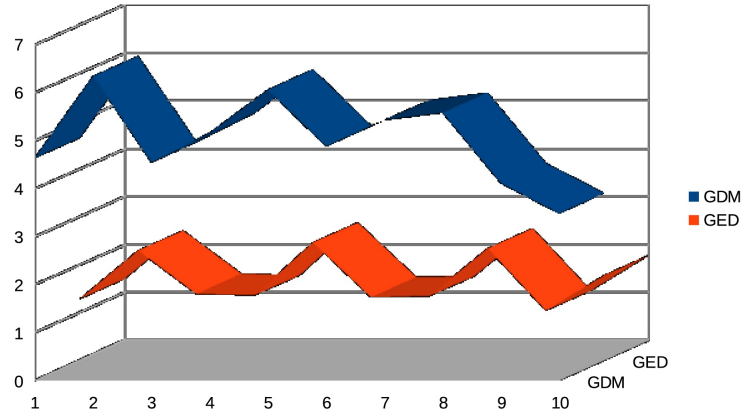

FIGURE 5.12: GDM vs GED for low distortion letter graphs

validation set and computed the classification accuracy using the test set. We use steepest-ascent search in the parameter space to find the weight parameters that are as well performing on the validation set as possible. The best weight parameter obtained are $w_1 = 0.35$, $w_2 = 0.23$, $w_3 = 0.11$, and $w_4 = 0.31$.

Table 5.2 shows the accuracy ratio for each of the 15 letters of high distortion letter graphs using Algorithm 8. For comparison we have shown accuracy using VD, ED and EDM separately along with GD and GDM computed using Algorithm 8. We observe that accuracy using GDM is always higher than that of VD and EDM individually; similarly, the accuracy using GD is more than that of VD and ED separately. We also note that the average accuracy of GDM is higher than that of GD for the computation of graph distance in Algorithm 8.

TABLE 5.2: Accuracy on letter dataset using *VD*, *ED*, *EDM*, *GD* and *GDM*

Class	*VD*	*ED*	*EDM*	*GD*	*GDM*
A	84	80	84	98	98
E	36	78	80	80	82
F	72	64	70	86	88
H	38	72	74	82	84
I	96	76	78	96	94
K	42	74	82	92	94
L	84	52	60	88	90
M	38	88	88	96	96
N	34	64	68	84	84
T	84	62	74	90	92
V	74	50	58	88	88
W	38	82	82	94	96
X	30	52	66	70	72
Y	46	56	64	84	88
Z	40	68	72	88	90

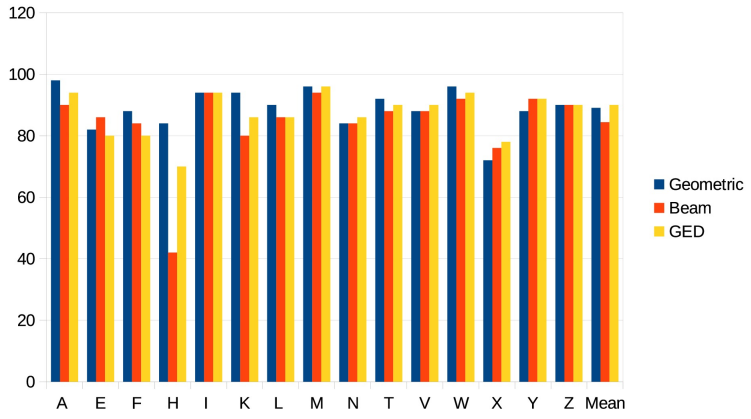

FIGURE 5.13: Comparison of accuracy on letter dataset

To compare the accuracy of geometric graph matching with other important graph matching techniques, we apply these algorithms to letter dataset with high distortion. The average classification accuracy achieved using geometric *GDM*, beamsearch and GED are 89.06, 84.40 and 90 respectively. Comparison of the accuracy of letter dataset using geometric *GDM*, beamsearch and, GED for every individual letter is shown in Figure 5.13.

AIDS training dataset consists of 250 graphs, out of which 50 graphs are from active class

TABLE 5.3: Accuracy on AIDS dataset

Algorithm used	Active	Inactive
Geometric	99.33	98.50
Beam	98.33	99.91
Bipartite	98.66	99.91

and the remaining 200 from inactive class. Test dataset includes 1500 graphs, having 300 graphs from active and rest 1200 graph are from inactive class. The classification accuracy of AIDS dataset using geometric GM, beamsearch and bipartite GM is shown in Table 5.3. We can observe that recognition rate of confirmed active molecules using geometric GM is more than both of beamsearch and bipartite GM, whereas classification rate of inactive molecules is less than that of beamsearch and bipartite GM.

The proposed graph distance measure can be used to compare the structural similarity between different graphs. It can be particularly useful in real-world applications, where the graph data is large and can be modified by noise. Depending on application requirement, we can select weighting factors such that $\sum_{i=1}^{n} w_i = 1$. When the position of vertices is more dominant, we can select w_1 to be higher, if angular structures are more important then w_2 can be prominent. Otherwise, if edge differences are more essential, we can select w_3 to be higher. We can either use equal weight factors or train them to optimize the classification ratio. In this section, we have trained the weight parameters for the computation of *GD* and *GDM* in the Geometric-Graph-Distance algorithm.

5.6 Summary

In this chapter, we described an intuitive approach to compute similarity between two geometric graphs and used it to perform exact and error-tolerant graph matching. In a geometric graph, every vertex has an associated coordinate, which specify its distinct position in the plane. We use this fact to define the *VD* between the two graphs. We defined *EDM* between the two graphs using the angular, edge length, and position components between corresponding edges of two graphs. Finally, we combined *VD* and *EDM* to form *GDM* and used it to compute error-tolerant graph matching between two graphs. The

experimental results show that this graph matching approach can be promising to graph dataset in which every node has a coordinate position in a two-dimensional plane.