Computational Geometry: Theory and Applications 118 (2024) 102056

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

journal homepage: www.elsevier.com/locate/comgeo

Distance measures for geometric graphs

Sushovan Majhi [a,*], Carola Wenk [b,1]

[a] School of Information, University of California, Berkeley, CA, 94720-4600, USA
[b] Department of Computer Science, Tulane University, New Orleans, USA

ARTICLE INFO

Article history:
Received 25 January 2023
Received in revised form 5 July 2023
Accepted 2 October 2023
Available online 6 October 2023

Keywords:
Graph comparison
Geometric graphs
Graph edit distance
NP-hard

ABSTRACT

A geometric graph is a combinatorial graph, endowed with a geometry that is inherited from its embedding in a Euclidean space. Formulation of a meaningful measure of (dis-)similarity in both the combinatorial and geometric structures of two such geometric graphs is a challenging problem in pattern recognition. We study two notions of distance measures for geometric graphs, called the geometric edit distance (GED) and geometric graph distance (GGD). While the former is based on the idea of editing one graph to transform it into the other graph, the latter is inspired by inexact matching of the graphs. For decades, both notions have been lending themselves well as measures of similarity between attributed graphs. If used without any modification, however, they fail to provide a meaningful distance measure for geometric graphs—even cease to be a metric. We have curated their associated cost functions for the context of geometric graphs. Alongside studying the metric properties of GED and GGD, we investigate how the two notions compare. We further our understanding of the computational aspects of GGD by showing that the distance is $\mathcal{NP}$-hard to compute, even if the graphs are planar and arbitrary cost coefficients are allowed.

As a computationally tractable alternative, we propose in this paper the Graph Mover's Distance (GMD), which has been formulated as an instance of the earth mover's distance. The computation of the GMD between two geometric graphs with at most n vertices takes only $O(n^3)$-time. The GMD demonstrates extremely promising empirical evidence at recognizing letter drawings.

1. Introduction

Graphs have been a widely accepted object for providing structural representation of patterns involving relational properties. The framework of representing complex and repetitive patterns using graphical structures can facilitate their description, manipulation, and recognition. While hierarchical patterns are commonly reduced to a string [1] or a tree representation [2], non-hierarchical patterns generally require a graph representation. One of the most important aspects of such representation is that the problem of pattern recognition becomes the problem of quantifying (dis-)similarity between a query graph and a model or prototype graph. The problem of defining a relevant distance measure for a class of graphs has been looked into for almost five decades now and has a myriad of applications including chemical structure matching [3], fingerprint matching [4], face identification [5], and symbol recognition [6]. All these applications demand a reliable

* Corresponding author.
E-mail addresses: smajhi@berkeley.edu (S. Majhi), cwenk@tulane.edu (C. Wenk).
[1] Partially supported by NSF grant CCF 2107434.

https://doi.org/10.1016/j.comgeo.2023.102056
0925-7721/

and efficient means of comparing two graphs. A meaningful graph distance measure is expected to yield a small distance implying similarity, and a large distance revealing disparity.

Depending on the class of graphs of interest and the area of application, several methods have been proposed. If the use case requires a perfect matching of two graphs, then the problem of graph isomorphism can be considered [7]; whereas, subgraph isomorphism can be applied for a perfect matching of parts of two graphs. These techniques are not, however, lenient with (sometimes minor) local and structural deformations of the two graphs. To address this issue, several alternative distance measures have been studied. We particularly investigate *edit distance* [8,9] and *inexact matching distance* [10]. The former makes use of elementary edit transformations (such as deletion, insertion, relabeling of vertices and edges), while the latter is based on partially matching two graphs through an inexact matching relation (Definition 11). And, the distance is defined as the minimum cost of transforming or matching one graph to the other. Although these distance measures have been battle-proven for attributed graphs (i.e., combinatorial graphs with finite label sets), the formulations seem inadequate in providing meaningful similarity measures for geometric graphs.

A geometric graph belongs to a special class of attributed graphs having an embedding into a Euclidean space $\mathbb{R}^d$, where the vertex and edge labels are inferred from the Euclidean locations of the vertices and Euclidean lengths of the edges, respectively. In the last decade, there has been a gain in practical applications involving comparison of geometric graphs. Examples include road-network or map comparison [11], detection of chemical structures using their spatial bonding geometry, etc. In addition, large datasets like [12] are being curated by pattern recognition and machine learning communities.

Despite a rich literature on the matching of attributed graphs and a fair count of algorithms benchmarked by both the database community and the pattern recognition community, most of the frameworks become untenable for matching geometric graphs. They remain oblivious to the spatial geometry such graphs are endowed with, consequently giving rise to very *artificial* measures of similarity for geometric graphs. This is not surprising at all—geometric graphs are a special class of labeled graphs after all! For a geometric graph, the significant differences include:

(i) Edge relabeling is not an independent edit operation, but vertex labels dictate the incident edge labels.
(ii) Vertex relabeling amounts to its translation to a different location in the ambient space, and additionally incurs the cost of relabeling of all its adjacent edges.

1.1. Our contribution

We study two distance measures, the *geometric edit distance* (GED) and *geometric graph distance* (GGD), in order to provide a meaningful measure of similarity between two geometric graphs. For attributed graphs the corresponding distance measures are equivalent as shown in [13, Proposition 1]. In contrast, we show in Section 2.3 they are not equivalent for geometric graphs. In addition to bounding each distance measure by a constant factor of the other in Proposition 18, we provide polynomial-time computable bounds on them.

We mention here the contribution of [14] for introducing GGD as well as discussing different definitions of edit distance in the context of geometric graphs. The authors also prove certain complexity results for GGD, which we improve upon in this paper. One of the major contributions of our study is to further our understanding of the computational complexity of GGD. In [14], the authors show that computing GGD is $\mathcal{NP}$-hard for non-planar graphs, when arbitrary cost coefficients C_V, C_E (as defined in Definition 13) are allowed. For planar graphs, $\mathcal{NP}$-hardness is proved under a very strict condition that $C_V << C_E$. We show in Proposition 21 that computing the GGD is $\mathcal{NP}$-hard, even if the graphs are planar and arbitrary C_V, C_E are allowed. The paper is organized in the following way. In Section 2.1 and Section 2.2, we formally define the two distances GGD and GED, respectively, and explore some of their important properties. We then compare the two distances in Section 2.3. Section 3 is devoted to our findings on the computational complexity of the GGD.

We define and study the *graph mover's distance* (GMD) in Section 4. The GMD has been shown to render a *pseudo*-metric on the class of (ordered) geometric graphs. Finally, we apply the GMD to classify letter drawings in Section 5. Our experiment involves matching each of 2250 test drawings, modeled as geometric graphs, to 15 prototype letters from the English alphabet. For the drawings (with low distortion), the correct letter has been found among the top 3 matches at a rate of 98.93%, which is extremely promising.

2. Two distances for geometric graphs

A geometric graph is a combinatorial graph that is also embedded in a Euclidean space. We begin with the formal definition.

Definition 1 *(Geometric graph).* A (finite) combinatorial graph $G = (V^G, E^G)$ is called a *geometric graph* of $\mathbb{R}^d$ if the vertex set $V^G \subset \mathbb{R}^d$ and the Euclidean straight-line segments $\left\{\overline{ab} \mid (a, b) \in E^G\right\}$ intersect (possibly) at their endpoints.

We denote the set of all geometric graphs of $\mathbb{R}^d$ by $\mathcal{G}(\mathbb{R}^d)$, and the subset of geometric graphs without any isolated vertex by $\mathcal{G}_0(\mathbb{R}^d)$. Two geometric graphs $G = (V^G, E^G)$ and $H = (V^H, E^H)$ are said to be *equal*, written $G = H$, if and only if $V^G = V^H$ and $E^G = E^H$. We make no distinction between a geometric graph $G = (V^G, E^G)$ and its *geometric realization* as

Table 1
Allowed edit operations on a geometric graph and associated costs.

Operation	Cost
delete (isolated) vertex u	0
insert vertex $u \in \mathbb{R}^d$	0
add edge e between existing vertices	$C_E\|e\|$
delete edge e	$C_E\|e\|$
translate a vertex at $u \in \mathbb{R}^d$ to vertex at $v \in \mathbb{R}^d$	$C_V\|u-v\| + \sum_{(s,u)\in E} C_E\big\|\|\overline{su}\| - \|\overline{sv}\|\big\|$

a subset of $\mathbb{R}^d$; an edge $(u, v) \in E^G$ can be identified as the line-segment $\overline{uv}$ in $\mathbb{R}^d$, and its length by the Euclidean length $|\overline{uv}|$. We denote by $\mathrm{Vol}(G)$ the sum of the edge lengths of G.

2.1. Geometric edit distance (GED)

Given two geometric graphs $G, H \in \mathcal{G}(\mathbb{R}^d)$, we transform G into H by applying a sequence of edit operations. The allowed edit operations and their costs are

(i) inserting (and deleting) a vertex costs nothing,
(ii) inserting (and deleting) an edge costs C_E times its length, and
(iii) translating a vertex costs C_V times the displacement of the vertex plus C_E times the total change in the length of *all* its incident edges.

The operations and their costs are summarized in Table 1. Throughout the paper, we assume that the cost coefficients C_V and C_E are positive constants. In order to denote a deleted vertex and a deleted edge, we introduce the *dummy vertex* ϵ_V and the *dummy edge* ϵ_E, respectively. While computing edit costs, we follow the convention that $|\epsilon_E| = 0$, $|a - \epsilon_V| = 0$ for any $a \in \mathbb{R}^d$, and $(u, v) = \epsilon_E$ if either $u = \epsilon_V$ or $v = \epsilon_V$. For each operation o listed in Table 1, note that its inverse, denoted o^{-1}, is also an edit operation with the same cost.

Definition 2 *(Edit path).* Given two geometric graphs $G, H \in \mathcal{G}(\mathbb{R}^d)$, an edit path P from G to H is a (finite) sequence of edit operations $\{o_i\}_{i=1}^k$ that satisfies the following:

(a) $(o_k \circ \ldots \circ o_2 \circ o_1)(G) = H$, i.e., $P(G) = H$, and
(b) o_{i+1} is a legal edit operation on $(o_i \circ \ldots \circ o_2 \circ o_1)(G)$ for any $1 \le i \le k-1$.

Note that we do not require for an intermediate edit operation to yield a geometric graph. The set of all edit paths between $G, H \in \mathcal{G}(\mathbb{R}^d)$ is denoted by $\mathcal{P}(G, H)$. For an edit path $P = \{o_i\}_{i=1}^k$, the edit path $\{o_i^{-1}\}_{i=1}^k$ from H to G is called its *inverse path*, and is denoted by P^{-1}. For any vertex $u \in V^G$ (resp. edge $e \in E^G$), we denote by $P(u)$ (resp. $P(e)$) the end result after its evolution under P. If P deletes the vertex u (resp. edge e), we write $P(v) = \epsilon_V$ (resp. $P(e) = \epsilon_E$). We now define the cost, $\mathrm{Cost}(P)$, of an edit path P to be the total cost of the individual edits.

Definition 3 *(Cost of edit paths).* The cost of an edit path $P \in \mathcal{P}(G, H)$, denoted $\mathrm{Cost}(P)$, is the sum of the cost of the individual edits, i.e.,

$$\mathrm{Cost}(P) \stackrel{\text{def}}{=} \sum_{o_i \in P} \mathrm{Cost}(o_i).$$

It is not difficult to note that $\mathrm{Cost}(P) = \mathrm{Cost}(P^{-1})$. Then, $\mathrm{GED}(G, H)$ is defined as cost of the *least* expensive edit path.

Definition 4 *(Geometric edit distance).* For geometric graphs $G, H \in \mathcal{G}(\mathbb{R}^d)$, their geometric edit distance, denoted $\mathrm{GED}(G, H)$, is defined to be the infimum cost of the edit paths, i.e.,

$$\mathrm{GED}(G, H) \stackrel{\text{def}}{=} \inf_{P \in \mathcal{P}(G,H)} \mathrm{Cost}(P).$$

In Proposition 10, we prove that GED is, in fact, a metric on the space of geometric graphs without any isolated vertex. As also observed in [14], the following example demonstrates that the distance may not be attained by an edit path, unless an infinite number of edits are allowed: Consider $G, H \in \mathcal{G}(\mathbb{R}^2)$, where G has only one edge (u_1, u_2) and H has only one edge (v_1, v_2) as shown in Fig. 1. For any fixed $k \ge 1$, consider the edit path $P_k = \{o_i\}_{i=1}^{2k}$, where o_i translates the left vertex of G up by a distance $1/k$ and then o_{i+1} moves the right vertex by the same distance for any odd i. So, for any i

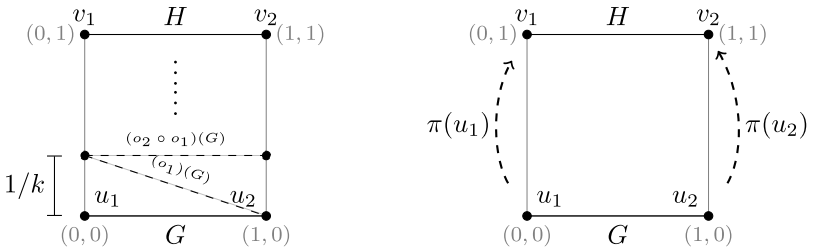

Fig. 1. Left: the edit path P_k alternatively moves the left and right vertices of G by distance $1/k$. Consequently, $\mathrm{GED}(G,H) = 2C_V$. Right: The inexact matching π between G and H has been shown to attain the same distance for $\mathrm{GGD}(G,H)$.

$$\mathrm{Cost}(o_i) = C_V\frac{1}{k} + C_E\left[\sqrt{(1/k)^2 + 1^2} - 1\right] = C_V\frac{1}{k} + C_E\frac{\frac{1}{k^2}}{\sqrt{\frac{1}{k^2}+1}+1}, \text{ and therefore}$$

$$\mathrm{GED}(G,H) \le \mathrm{Cost}(P_k) = \sum_{i=1}^{2k}\mathrm{Cost}(o_i) = 2C_V + C_E\frac{\frac{2}{k}}{\sqrt{\frac{1}{k^2}+1}+1} \xrightarrow{k\to\infty} 2C_V.$$

Now, if we assume that $C_E > C_V$, then any edit path with an edge deletion costs more than $2C_V$ from the inequality (2) on the next page. Therefore, $\mathrm{GED}(G,H) = 2C_V$. However, there is no edit path that attains this cost.

In Definition 3, the cost of an edit path P is defined as the aggregated cost from the individual edits involved in P. Another perspective of the cost of P is the total amount paid by P for the evolution of each vertex and edge of G and H. We make this notion more precise by tracking the evolution of vertices and edges through their orbit.

Definition 5 *(Orbit of a vertex).* Let $P \in \mathcal{P}(G,H)$ be an edit path and u a vertex of G. The orbit of u under $P = \{o_i\}_{i=1}^k$ is the sequence of vertices $\{u_i\}_{i=0}^k$, where $u_0 = u$ and $u_i = (o_i \circ o_{i-1} \circ \ldots \circ o_1)(u)$ for $i \ge 1$. And, the cost of the orbit, denoted $\mathrm{Cost}_P(u)$, is defined by

$$\mathrm{Cost}_P(u) \stackrel{\text{def}}{=} C_V \sum_{i=1}^{k} |u_i - u_{i-1}|.$$

The ith summand above is positive only if o_i is a translation of the vertex. Using the triangle inequality, we can immediately note the following fact.

Lemma 6 *(Cost of vertex orbit). For a vertex $u \in V^G$ and $P \in \mathcal{P}(G,H)$, we have*

$$\mathrm{Cost}_P(u) \ge C_V|u - P(u)|.$$

We similarly define the orbit of an edge and its cost.

Definition 7 *(Orbit of an edge).* Let $P \in \mathcal{P}(G,H)$ be an edit path and e an edge of G. The orbit of e under $P = \{o_i\}_{i=1}^k$ is the sequence of edges $\{e_i\}_{i=0}^k$, where $e_0 = e$ and $e_i = (o_i \circ o_{i-1} \circ \ldots \circ o_1)(e)$ for $i \ge 1$. And, the cost of the orbit, denoted $\mathrm{Cost}_P(e)$, is defined by

$$\mathrm{Cost}_P(e) \stackrel{\text{def}}{=} C_E \sum_{i=1}^{k} \big||e_i| - |e_{i-1}|\big|.$$

We note that deletion of the edge or translation of an incident vertex are the only edit operations in P that can potentially contribute to a positive summand in the cost function above. Again, the triangle inequality implies the following lemma.

Lemma 8 *(Cost of edge orbit). For an edge $e \in E^G$ and $P \in \mathcal{P}(G,H)$, we have*

$$\mathrm{Cost}_P(e) \ge C_E\big||e| - |P(e)|\big|.$$

In particular, $\mathrm{Cost}_P(e) \ge C_E|e|$ *if P eventually deletes e, i.e., $P(e) = \epsilon_E$.*

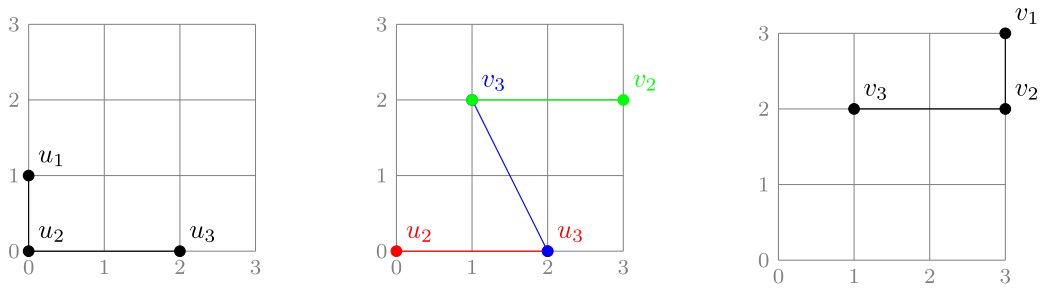

Fig. 2. Two graphs $G, H \in \mathcal{G}(\mathbb{R}^2)$ have been shown on the left and right, respectively. In the middle, the evolution of G under an edit path $P = \{o_1, o_2, o_3, o_4, o_5, o_6\}$ is demonstrated. The edit o_1 deletes the edge (u_1, u_2), o_2 deletes the vertex u_1, then o_3 translates u_2 to v_3, after that o_4 translates u_3 to v_2, o_5 inserts the vertex v_1, and finally o_6 inserts the edge (v_1, v_2). The orbit of the vertex u_2 is $\{u_2, u_2, v_3, v_3, v_3\}$, whereas the orbit of (u_2, u_3) is $\{(u_2, u_3), (u_2, u_3), (v_3, u_3), (v_3, v_2), (v_3, v_2)\}$.

For examples of vertex and edge orbits see Fig. 2. In order to describe $\mathrm{Cost}(P)$ in terms of the costs of individual orbits, we note that $\mathrm{Cost}(P)$ accounts for the costs of the orbits of:

(a) vertices $u \in V^G$ that end up as a vertex of H, i.e., $P(u) \neq \epsilon_V$
(b) vertices $u \in V^G$ with $P(u) = \epsilon_V$
(c) vertices $v \in V^H$ that have been inserted, i.e., $P^{-1}(v) = \epsilon_V$
(d) edges $e \in E^G$ that end up as an edge of H, i.e., $P(e) \neq \epsilon_E$
(e) edges $e \in E^G$ with $P(e) = \epsilon_E$
(f) edges $f \in E^H$ that have been inserted, i.e., $P^{-1}(f) = \epsilon_E$
(g) vertices and edges that have been inserted at some point and have also been deleted eventually.

Moreover, we observe that two vertex (resp. edge) orbits $\{x_i\}$ and $\{y_i\}$ intersect at the i_0th position only if $x_i = y_i = \epsilon_V$ (resp. $x_i = y_i = \epsilon_E$) for all $i \geq i_0$. As a consequence, the positive summands in the costs of two orbits are necessarily distinct. Accumulating the costs for all orbits of type (a)–(f), we can, therefore, write

$$\begin{aligned}\mathrm{Cost}(P) \geq & \underbrace{\sum_{\substack{u \in V^G \\ P(u) \neq \epsilon_V}} \mathrm{Cost}_P(u)}_{\text{vertex translations}} + \underbrace{\sum_{\substack{u \in V^G \\ P(u) = \epsilon_V}} \mathrm{Cost}_P(u)}_{\text{vertex deletions}} + \underbrace{\sum_{\substack{v \in V^H \\ P^{-1}(v) = \epsilon_V}} \mathrm{Cost}_{P^{-1}}(v)}_{\text{vertex insertions}} \\ & + \underbrace{\sum_{\substack{e \in E^G \\ P(e) \neq \epsilon_E}} \mathrm{Cost}_P(e)}_{\text{edge translations}} + \underbrace{\sum_{\substack{e \in E^G \\ P(e) = \epsilon_E}} \mathrm{Cost}_P(e)}_{\text{edge deletions}} + \underbrace{\sum_{\substack{f \in E^H \\ P^{-1}(f) = \epsilon_E}} \mathrm{Cost}_{P^{-1}}(f)}_{\text{edge insertions}}.\end{aligned} \tag{1}$$

Equation (1) together with Lemma 6 and Lemma 8 readily imply the following useful result.

Lemma 9. *For any edit path $P \in \mathcal{P}(G, H)$, it holds that*

$$\mathrm{Cost}(P) \geq \sum_{\substack{u \in V^G \\ P(u) \neq \epsilon_V}} C_V |u - P(u)| + \sum_{\substack{e \in E^G \\ P(e) \neq \epsilon_E}} C_E \big| |e| - |P(e)| \big| + \sum_{\substack{e \in E^G \\ P(e) = \epsilon_E}} C_E |e| + \sum_{\substack{f \in E^H \\ P^{-1}(f) = \epsilon_E}} C_E |f|. \tag{2}$$

Proposition 10 (GED *is a metric). The* GED *defines a metric on $\mathcal{G}_0(\mathbb{R}^d)$, the space of geometric graphs without any isolated vertex.*

Proof. Non-negativity. Since the cost of edit paths is non-negative, Definition 4 implies that $\mathrm{GED}(G, H)$ is non-negative for any $G, H \in \mathcal{G}_0(\mathbb{R}^d)$.

Separability. If $\mathrm{GED}(G, H) = 0$, we claim that $G = H$, i.e., $V^G = V^H$ and $E^G = E^H$. In order to show that $V^G = V^H$, it suffices to show that the Hausdorff distance $r := d_H(V^G, V^H)$ between the vertex sets is zero. Fix

$$2\xi = \begin{cases} C_E \min\{l^G, l^H\}, & \text{if } r = 0 \\ \min\{C_V r, C_E l^G, C_E l^H\}, & \text{if } r \neq 0 \end{cases}$$

where l^G and l^H denote the smallest edge lengths of G and H, respectively. Since $\xi > 0$, the definition of GED implies that there is an edit path $P \in \mathcal{P}(G, H)$ with $\text{Cost}(P) \leq \xi$. Consequently, each of the four summands in (2) is no larger than ξ. We immediately see that there is no edge $e \in E^G$ such that $P(e) = \epsilon_E$. Otherwise, the third summand in (2) would be at least

$$C_E|e| \geq C_E l^G \geq 2\xi > \xi,$$

leading to a contradiction. The last inequality above is due to the observation that $\xi > 0$. Similarly using the fourth summand in (2), we conclude there is no edge $f \in E^H$ such that $P^{-1}(f) = \epsilon_E$. In other words, P does not delete any edge of G or H, i.e., $|E^G| = |E^H|$. As a result, we can further say that no vertex of G can be removed and no vertex of H can be inserted, since the input graphs do not have any isolated vertices. Since $H = P(G)$, the graphs G and H must be isomorphic. Lastly, we show that $V^G = V^H$, i.e., $r = 0$. If not, i.e., $r \neq 0$ and $u_0 \in V^G$ such that all the vertices of H are at least r distance away from it, then

$$C_V|u_0 - P(u_0)| \geq C_V r \geq 2\xi > \xi.$$

This is a contradiction, because the first term in (2) exceeds ξ. So, $r = 0$. Therefore, $G = H$.

Symmetry. Each elementary edit operation can be reversed at exactly the same cost. Given an edit path $P \in \mathcal{P}(G, H)$, we can reverse the operations to get an edit path $P^{-1} \in \mathcal{P}(H, G)$ with $\text{Cost}(P) = \text{Cost}(P^{-1})$. By Definition 4, for an arbitrary $\xi > 0$ there exists $P \in \mathcal{P}(G, H)$ such that $\text{Cost}(P) \leq \text{GED}(G, H) + \xi$. On the other hand,

$$\text{GED}(H, G) \leq \text{Cost}(P^{-1}) = \text{Cost}(P) \leq \text{GED}(G, H) + \xi.$$

Since ξ is arbitrary, this implies $\text{GED}(H, G) \leq \text{GED}(G, H)$. By a similar argument, one can also show $\text{GED}(H, G) \geq \text{GED}(G, H)$. Together, they imply $\text{GED}(H, G) = \text{GED}(G, H)$.

Triangle inequality. Fix an arbitrary $\xi > 0$ and $G, H, I \in \mathcal{G}_0(\mathbb{R}^d)$. By Definition 4, there must exist edit paths $P_1 \in \mathcal{P}(G, H)$ and $P_2 \in \mathcal{P}(H, I)$ such that $\text{Cost}(P_1) \leq \text{GED}(G, H) + \xi/2$ and $\text{Cost}(P_2) \leq \text{GED}(H, I) + \xi/2$. If we define P to be the concatenation of the edit operations from P_1 and P_2 in the same order, then $P \in \mathcal{P}(G, I)$. Moreover, $\text{Cost}(P) = \text{Cost}(P_1) + \text{Cost}(P_2)$. Now,

$$\begin{aligned}
\text{GED}(G, I) &\leq \text{Cost}(P), \text{ from the Definition of GED} \\
&= \text{Cost}(P_1) + \text{Cost}(P_2) \\
&\leq \left[\text{GED}(G, H) + \frac{\xi}{2}\right] + \left[\text{GED}(H, I) + \frac{\xi}{2}\right] \\
&= \text{GED}(G, H) + \text{GED}(H, I) + \xi.
\end{aligned}$$

Since the choice of ξ is arbitrary, we get $\text{GED}(G, I) \leq \text{GED}(G, H) + \text{GED}(H, I)$. □

2.2. Geometric graph distance (GGD)

The definition of GED is very intuitive but not at all suited for computational purposes. Firstly, there could be infinitely many locations a vertex is allowed to be translated to. Secondly, there are infinitely many edit paths between two graphs—even if the vertices are located on a finite grid. Due to the infinite search space, it is not clearly understood how to compute the GED. As a feasible alternative we study the GGD. The definition is inspired by the concept of inexact matching first proposed in [10] for attributed graphs, and later introduced for geometric graphs in [14]. We follow the notation of [10] in order to define it. We first define an *(inexact) matching*.

Definition 11 *(Inexact matching).* Let $G, H \in \mathcal{G}(\mathbb{R}^d)$ be two geometric graphs. A relation $\pi \subseteq (V^G \cup \{\epsilon_V\}) \times (V^H \cup \{\epsilon_V\})$ is called an (inexact) matching if for any $u \in V^G$ (resp. $v \in V^H$) there is exactly one $v \in V^H \cup \{\epsilon_V\}$ (resp. $u \in V^G \cup \{\epsilon_V\}$) such that $(u, v) \in \pi$.

The set of all matchings between graphs G, H is denoted by $\Pi(G, H)$. Intuitively speaking, a matching π is a relation that covers the vertex sets V^G, V^H exactly once. As a result, when restricted to V^G (resp. V^H), a matching π can be expressed as a map $\pi : V^G \to V^H \cup \{\epsilon_V\}$ (resp. $\pi^{-1} : V^H \to V^G \cup \{\epsilon_V\}$). In other words, when $(u, v) \in \pi$ and $u \neq \epsilon_V$ (resp. $v \neq \epsilon_V$), it is justified to write $\pi(u) = v$ (resp. $\pi^{-1}(v) = u$). It is evident from the definition that the induced map

$$\pi : \{u \in V^G \mid \pi(u) \neq \epsilon_V\} \to \{v \in V^H \mid \pi^{-1}(v) \neq \epsilon_V\}$$

is a bijection. Additionally for edges $e = (u_1, u_2) \in E^G$ and $f = (v_1, v_2) \in E^H$, we introduce the short-hand $\pi(e) := (\pi(u_1), \pi(u_2))$ and $\pi^{-1}(f) := (\pi^{-1}(v_1), \pi^{-1}(v_2))$.

Another perspective of π is discerned when viewed as a matching between portions of G and H, (possibly) after applying some edits on the two graphs. For example, $\pi(u) = \epsilon_V$ (resp. $\pi^{-1}(v) = \epsilon_V$) encodes deletion of the vertex u from G (resp. v from H), whereas $\pi(e) = \epsilon_E$ (resp. $\pi^{-1}(f) = \epsilon_E$) encodes deletion of the edge e from G (resp. f from H). Once the above deletion operations have been performed on the graphs, the resulting subgraphs of G and H become isomorphic, which are finally matched by translating the remaining vertices u to $\pi(u)$. Now, the cost of the matching π is defined as the total cost for all of these operations:

Definition 12 *(Cost of a matching).* Let $G, H \in \mathcal{G}(\mathbb{R}^d)$ be geometric graphs and $\pi \in \Pi(G, H)$ an inexact matching. The cost of π, denoted $\text{Cost}(\pi)$, is defined as

$$\text{Cost}(\pi) = \underbrace{\sum_{\substack{u \in V^G \\ \pi(u) \neq \epsilon_V}} C_V |u - \pi(u)|}_{\text{vertex translations}} + \underbrace{\sum_{\substack{e \in E^G \\ \pi(e) \neq \epsilon_E}} C_E \big||e| - |\pi(e)|\big|}_{\text{edge translations}} + \underbrace{\sum_{\substack{e \in E^G \\ \pi(e) = \epsilon_E}} C_E |e|}_{\text{edge deletions}} + \underbrace{\sum_{\substack{f \in E^H \\ \pi^{-1}(f) = \epsilon_E}} C_E |f|}_{\text{edge insertions}}. \tag{3}$$

Definition 13 *(GGD).* For geometric graphs $G, H \in \mathcal{G}(\mathbb{R}^d)$, their geometric graph distance, denoted $\text{GGD}(G, H)$, is defined as the cost of a least expensive inexact matching, i.e.,

$$\text{GGD}(G, H) \stackrel{\text{def}}{=} \min_{\pi \in \Pi(G,H)} \text{Cost}(\pi).$$

The minimum cost matching between two graphs along with their GGD has been illustrated in Fig. 1. The above definition readily yields the following result.

Lemma 14. *Let $G, H \in \mathcal{G}(\mathbb{R}^d)$ be geometric graphs. For any $\pi \in \Pi(G, H)$, we have*

$$\text{Cost}(\pi) \geq \sum_{\substack{u \in V^G \\ \pi(u) \neq \epsilon_V}} C_V |u - \pi(u)| + C_E |\text{Vol}(G) - \text{Vol}(H)| + 2 \min \left\{ \sum_{\substack{e \in E^G \\ \pi(e) = \epsilon_E}} C_E |e|, \sum_{\substack{f \in E^H \\ \pi^{-1}(f) = \epsilon_E}} C_E |f| \right\}.$$

Proof. Without any loss of generality, we assume that

$$\sum_{\substack{e \in E^G \\ \pi(e) = \epsilon_E}} C_E |e| \leq \sum_{\substack{f \in E^H \\ \pi^{-1}(f) = \epsilon_E}} C_E |f|. \tag{4}$$

From (3), we have

$$\begin{aligned}
\text{Cost}(\pi) &= \sum_{\substack{u \in V^G \\ \pi(u) \neq \epsilon_V}} C_V |u - \pi(u)| + \sum_{\substack{e \in E^G \\ \pi(e) \neq \epsilon_E}} C_E \big||e| - |\pi(e)|\big| + \sum_{\substack{e \in E^G \\ \pi(e) = \epsilon_E}} C_E |e| + \sum_{\substack{f \in E^H \\ \pi^{-1}(f) = \epsilon_E}} C_E |f| \\
&= \sum_{\substack{u \in V^G \\ \pi(u) \neq \epsilon_V}} C_V |u - \pi(u)| + \sum_{\substack{e \in E^G \\ \pi(e) \neq \epsilon_E}} C_E \big||\pi(e) - |e|\big| + \sum_{\substack{f \in E^H \\ \pi^{-1}(f) = \epsilon_E}} C_E |f| - \sum_{\substack{e \in E^G \\ \pi(e) = \epsilon_E}} C_E |e| + 2 \sum_{\substack{e \in E^G \\ \pi(e) = \epsilon_E}} C_E |e| \\
&= \sum_{\substack{u \in V^G \\ \pi(u) \neq \epsilon_V}} C_V |u - \pi(u)| + \sum_{\substack{e \in E^G \\ \pi(e) \neq \epsilon_E}} C_E \big||\pi(e) - |e|\big| + \left| \sum_{\substack{f \in E^H \\ \pi^{-1}(f) = \epsilon_E}} C_E |f| - \sum_{\substack{e \in E^G \\ \pi(e) = \epsilon_E}} C_E |e| \right| \\
&\quad + 2 \sum_{\substack{e \in E^G \\ \pi(e) = \epsilon_E}} C_E |e|, \text{ from (4)} \\
&\geq \sum_{\substack{u \in V^G \\ \pi(u) \neq \epsilon_V}} C_V |u - \pi(u)| + \left| \sum_{\substack{e \in E^G \\ \pi(e) \neq \epsilon_E}} C_E (|\pi(e)| - |e|) \right| + \left| \sum_{\substack{f \in E^H \\ \pi^{-1}(f) = \epsilon_E}} C_E |f| - \sum_{\substack{e \in E^G \\ \pi(e) = \epsilon_E}} C_E |e| \right| \\
&\quad + 2 \sum_{\substack{e \in E^G \\ \pi(e) = \epsilon_E}} C_E |e|, \text{ by the triangle inequality}
\end{aligned}$$

$$
\begin{aligned}
&\geq \sum_{\substack{u\in V^G\\ \pi(u)\neq\epsilon_V}} C_V|u-\pi(u)| + \Bigg|\sum_{\substack{e\in E^G\\ \pi(e)\neq\epsilon_E}} C_E(|\pi(e)|-|e|) + \sum_{\substack{f\in E^H\\ \pi^{-1}(f)=\epsilon_E}} C_E|f| - \sum_{\substack{e\in E^G\\ \pi(e)=\epsilon_E}} C_E|e|\Bigg| \\
&\quad + 2\sum_{\substack{e\in E^G\\ \pi(e)=\epsilon_E}} C_E|e|, \text{ by the triangle inequality} \\
&= \sum_{\substack{u\in V^G\\ \pi(u)\neq\epsilon_V}} C_V|u-\pi(u)| + C_E\Bigg|\sum_{\substack{e\in E^G\\ \pi(e)\neq\epsilon_E}} |\pi(e)| - \sum_{\substack{e\in E^G\\ \pi(e)\neq\epsilon_E}} |e| + \sum_{\substack{f\in E^H\\ \pi^{-1}(f)=\epsilon_E}} |f| - \sum_{\substack{e\in E^G\\ \pi(e)=\epsilon_E}} |e|\Bigg| + 2\sum_{\substack{e\in E^G\\ \pi(e)=\epsilon_E}} C_E|e| \\
&= \sum_{\substack{u\in V^G\\ \pi(u)\neq\epsilon_V}} C_V|u-\pi(u)| + C_E\Bigg|\Bigg(\sum_{\substack{e\in E^G\\ \pi(e)\neq\epsilon_E}} |\pi(e)| + \sum_{\substack{f\in E^H\\ \pi^{-1}(f)=\epsilon_E}} |f|\Bigg) - \Bigg(\sum_{\substack{e\in E^G\\ \pi(e)=\epsilon_E}} |e| + \sum_{\substack{e\in E^G\\ \pi(e)\neq\epsilon_E}} |e|\Bigg)\Bigg| + 2\sum_{\substack{e\in E^G\\ \pi(e)=\epsilon_E}} C_E|e| \\
&= \sum_{\substack{u\in V^G\\ \pi(u)\neq\epsilon_V}} C_V|u-\pi(u)| + C_E\Bigg|\sum_{f\in E^H} |f| - \sum_{e\in E^G} |e|\Bigg| + 2\sum_{\substack{e\in E^G\\ \pi(e)=\epsilon_E}} C_E|e| \\
&= \sum_{\substack{u\in V^G\\ \pi(u)\neq\epsilon_V}} C_V|u-\pi(u)| + C_E\big|\mathrm{Vol}(H)-\mathrm{Vol}(G)\big| + 2\sum_{\substack{e\in E^G\\ \pi(e)=\epsilon_E}} C_E|e|.
\end{aligned}
$$

This proves the result. □

The follow proposition provides a lower and upper bound for the GGD that are computable in polynomial-time.

Proposition 15 *(Bounding the* GGD*). For geometric graphs $G, H \in \mathcal{G}(\mathbb{R}^d)$, we have*

$$C_E|\mathrm{Vol}(G)-\mathrm{Vol}(H)| \leq \mathrm{GGD}(G,H) \leq C_E|\mathrm{Vol}(G)+\mathrm{Vol}(H)|.$$

Proof. For any arbitrary matching $\pi \in \Pi(G,H)$, from Lemma 14 we get

$$C_E|\mathrm{Vol}(G)-\mathrm{Vol}(H)| \leq \mathrm{Cost}(P).$$

Since π is arbitrary, we conclude $C_E|\mathrm{Vol}(G)-\mathrm{Vol}(H)| \leq \mathrm{GGD}(G,H)$.

For the second inequality, we choose the trivial matching $\pi_0 \in \Pi(G,H)$, where $\pi_0(u) = \pi_0^{-1}(v) = \epsilon_V$ for all $u \in V^G$ and $v \in V^H$. So,

$$\mathrm{GGD}(G,H) \leq \mathrm{Cost}(\pi) = C_E[\mathrm{Vol}(G)+\mathrm{Vol}(H)]. \quad \square$$

As also shown in [14], the GGD is also a metric. We present a proof here, using our notation, for the sake of completion.

Proposition 16 *(*GGD *is a metric). The* GGD *defines a metric on $\mathcal{G}_0(\mathbb{R}^d)$, the space of geometric graphs without any isolated vertex.*

Proof. Non-negativity. Since the cost of any matching in $\Pi(G,H)$ is non-negative, Definition 13 implies that $\mathrm{GGD}(G,H)$ is non-negative for any $G, H \in \mathcal{G}_0(\mathbb{R}^d)$.

Separability. If $\mathrm{GGD}(G,H)=0$, then there is $\pi \in \Pi(G,H)$ with $\mathrm{Cost}(\pi)=0$. So, all the four summands in (3) are identically zero. In particular, the third and fourth summands imply that no edge has been deleted from G or H by π, i.e., $|E^G| = |E^H|$. Since the graphs do not have any isolated vertex, this implies that $\pi(u) \neq \epsilon_V, \pi(v) \neq \epsilon_V$ for all $u \in V^G$ and $v \in V^H$. As a result, $|V^G| = |V^H|$. Moreover, the first summand of (3) implies that $\pi(u) = u$ for all $u \in V^G$. Therefore, $G = H$.

Symmetry. We conclude that $\mathrm{GGD}(G,H) = \mathrm{GGD}(H,G)$ due to the fact that any matching in $\Pi(G,H)$ induces a matching in $\Pi(H,G)$ with exactly the same cost and vice versa.

Triangle inequality. For the triangle inequality, let us assume that $\mathrm{Cost}(\pi_1) = \mathrm{GGD}(G,H)$ and $\mathrm{Cost}(\pi_2) = \mathrm{GGD}(H,I)$ for some $\pi_1 \in \Pi(G,H)$ and $\pi_2 \in \Pi(H,I)$. For any $u \in V^G$ and $v \in V^I$, define $\pi \in \Pi(G,I)$ such that:

$$\pi(u) = \begin{cases} \pi_2 \circ \pi_1(u), & \text{if } \pi_1(u) \neq \epsilon_V \\ \epsilon_V, & \text{otherwise} \end{cases}$$

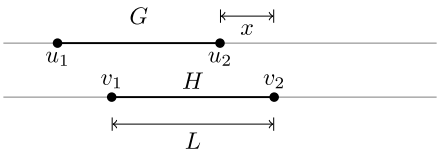

Fig. 3. The graphs G (top) and H (bottom) are embedded in the real line, where $u_2 - u_1 = v_2 - v_1 = L$ and $v_2 - u_2 = v_1 - u_1 = x$.

and

$$\pi^{-1}(v) = \begin{cases} \pi_1^{-1} \circ \pi_2^{-1}(v), & \text{if } \pi_2^{-1}(u) \neq \epsilon_V \\ \epsilon_V, & \text{otherwise} \end{cases}$$

Using the triangle inequality, it can be easily seen from (3) that $\text{Cost}(\pi) \leq \text{Cost}(\pi_1) + \text{Cost}(\pi_2)$. So,

$$\begin{aligned} \text{GGD}(G, I) &\leq \text{Cost}(\pi), \text{ from the Definition of GED} \\ &\leq \text{Cost}(\pi_1) + \text{Cost}(\pi_2) \\ &= \text{GGD}(G, H) + \text{GGD}(H, I). \end{aligned}$$

Therefore, we get $\text{GGD}(G, I) \leq \text{GGD}(G, H) + \text{GGD}(H, I)$ as desired. □

2.3. Comparing GED and GGD

As we now have the two notions of distances under our belts, the question of how they compare arises naturally. We have already pointed out that the analogous notions for attributed graphs yield equivalent distances. To our surprise, they are not generally equal for geometric graphs, as the following proposition demonstrates.

Proposition 17. *Given any $D > 0$, there exist graphs $G, H \in \mathcal{G}(\mathbb{R})$ such that*

$$\text{GGD}(G, H) = D \text{ and } \text{GED}(G, H) = \left(1 + \frac{C_E}{C_V}\right) D.$$

In particular, $\text{GGD}(G, H) < \text{GED}(G, H)$.

Proof. We take two graphs $G, H \in \mathcal{G}(\mathbb{R})$ as shown in Fig. 3. In each graph, the two vertices are separated by a distance L, whereas the second graph is a copy of the first but shifted by x. We also choose

$$x = \frac{D}{2C_V} \text{ and } L = \left(1 + \frac{2C_V}{C_E}\right) x.$$

To see that $\text{GGD}(G, H) = D$, we consider the matching $\pi(u_i) = v_i$ for $i = 1, 2$. The cost of the matching is

$$\text{Cost}(\pi) = C_V \sum_{i=1}^{2} |u_i - v_i| = C_V \sum_{i=1}^{2} x = 2C_V x = D.$$

It is worth noting here that a matching π' that is not bijective on the vertex sets has cost

$$\text{Cost}(\pi') \geq C_E L > C_E \times \frac{2C_V}{C_E} x = D = \text{Cost}(\pi).$$

Since $L > x$, the cost of π is also (strictly) smaller that $2C_V L$, which is the cost of the other possible bijective matching. So, we have $\text{GGD}(G, H) = D$.

In order to compute $\text{GED}(G, H)$, we consider the edit path P_0 that moves the vertex u_1 to v_1, then moves u_2 to v_2. The cost of P_0 is

$$2C_V x + 2C_E x = 2C_V x \left(1 + \frac{C_E}{C_V}\right) = \left(1 + \frac{C_E}{C_V}\right) D.$$

We now claim that the cost of *any* edit path P is at least $(1 + C_E/C_V)D$. Consider the following two cases:
Case I. If $P(u_1, u_2) = \epsilon_E$, then from (2), we have

$$\text{Cost}(P) \geq 2C_E L = 2(C_E + 2C_V)x = 2(C_E + 2C_V)\frac{D}{2C_V} = (2 + C_E/C_V)D > (1 + C_E/C_V)D.$$

Case II. For this case, we assume that $P(u_1, u_2) \neq \epsilon_E$. So, P contains only vertex translations. Let $O = \{o_i\}_{i=1}^{k}$ be the subsequence of P containing only those translations that do not flip the order of the endpoints of the incident edge. Due to the position of G and H, it is evident that O is non-empty. Moreover, the vertices must travel at least x distance each under O. When an endpoint u is moved to a location $w \in \mathbb{R}$ by such an o_i, the associated cost of translating the edge becomes $C_E|w - u|$. Therefore, the cost

$$\text{Cost}(P) \geq \text{Cost}(O) \geq 2C_V x + 2C_E 2x = 2(C_E + C_V)\frac{D}{2C_V} = (1 + C_E/C_V)D.$$

Considering the above the cases, we conclude that $\text{GED}(G, H) = (1 + C_E/C_V)D$. □

More generally, we prove that following result to compare the two distances.

Proposition 18. *For any two geometric graphs $G, H \in \mathcal{G}(\mathbb{R}^d)$, we have*

$$\text{GGD}(G, H) \leq \text{GED}(G, H) \leq \left(1 + \Delta\frac{C_E}{C_V}\right)\text{GGD}(G, H),$$

where Δ denotes the maximum degree of the graphs G, H.

Proof. Take an arbitrary edit path $P \in \mathcal{P}(G, H)$. Let us define a matching $\pi_P \in \Pi(G, H)$ such that

$$\pi_P \stackrel{\text{def}}{=} \{(u, P(u)) \mid u \in V^G\} \cup \{(P^{-1}(v), v) \mid v \in V^H\}.$$

This definition of π_P implies that $P(u) = \pi_P(u)$ for all $u \in V^G$, $P(e) = \pi_P(e)$ for all $e \in E^G$, and $P^{-1}(f) = \pi_P^{-1}(f)$ for all $f \in E^H$. From (3) and Lemma 9 it follows that $\text{Cost}(\pi_P) \leq \text{Cost}(P)$. The definition of $\text{GGD}(G, H)$ then implies that

$$\text{GGD}(G, H) \leq \text{Cost}(\pi_P) \leq \text{Cost}(P).$$

Since P is chosen arbitrarily, the definition of $\text{GED}(G, H)$ then implies the first inequality.

For the second inequality, we take an arbitrary $\pi \in \Pi(G, H)$. From π, we define an edit path P_π to be the sequence $(D_E, D_V, T_V, I_V, I_E)$ of edit operations, where

(i) D_E is a sequence of deletions of edges $e \in E^G$ with $\pi(e) = \epsilon_E$
(ii) D_V is a sequence of deletions of vertices $u \in V^G$ with $\pi(u) = \epsilon_V$,
(iii) T_V is a sequence of translations of vertices $u \in V^G$ with $\pi(u) \neq \epsilon_V$ to $\pi(u)$,
(iv) I_V is a sequence of insertions of vertices $v \in V^H$ with $\pi^{-1}(v) = \epsilon_V$, and
(v) I_E is a sequence of insertions of edges $f \in E^H$ with $\pi^{-1}(f) = \epsilon_E$.

Each of the above sequences (i)–(v) is unique up to the ordering of the operations. Also in P_π, the edges are deleted in D_E before deleting their endpoints in D_V, and the edges are inserted in I_E only after inserting their endpoints in I_V. Consequently, P_π defines a legal edit path from G to H, i.e., $P_\pi \in \mathcal{P}(G, H)$. We claim that

$$\text{Cost}(P_\pi) \leq \left(1 + \Delta\frac{C_E}{C_V}\right)\text{Cost}(\pi).$$

To prove the claim, we note that P_π does not insert any vertex or edge that will be later deleted. As a result, the item (g) above (1) has a zero cost. So, (1) is, in fact, an equality:

$$\begin{aligned}\text{Cost}(P_\pi) = &\sum_{\substack{u \in V^G \\ \pi(u) \neq \epsilon_V}} \text{Cost}_{P_\pi}(u) + \sum_{\substack{u \in V^G \\ \pi(u) = \epsilon_V}} \text{Cost}_{P_\pi}(u) + \sum_{\substack{v \in V^H \\ \pi^{-1}(u) = \epsilon_V}} \text{Cost}_{P_\pi^{-1}}(v) \\ &+ \sum_{\substack{e \in E^G \\ \pi(e) \neq \epsilon_E}} \text{Cost}_{P_\pi}(e) + \sum_{\substack{e \in E^G \\ \pi(e) = \epsilon_E}} \text{Cost}_{P_\pi}(e) + \sum_{\substack{f \in E^H \\ \pi^{-1}(f) = \epsilon_E}} \text{Cost}_{P_\pi}(f)\end{aligned}$$

Moreover, a deleted (resp. inserted) vertex has never been translated, yielding a zero cost for its orbit. So, the second and the third summands are identically zero. We can then write

$$\text{Cost}(P_\pi) = \sum_{\substack{u \in V^G \\ \pi(u) \neq \epsilon_V}} \text{Cost}_{P_\pi}(u) + \sum_{\substack{e \in E^G \\ \pi(e) \neq \epsilon_E}} \text{Cost}_{P_\pi}(e) + \sum_{\substack{e \in E^G \\ \pi(e) = \epsilon_E}} \text{Cost}_{P_\pi}(e) + \sum_{\substack{f \in E^H \\ \pi^{-1}(f) = \epsilon_E}} \text{Cost}_{P_\pi}(f)$$

$$= \sum_{\substack{u\in V^G \\ \pi(u)\neq\epsilon_V}} C_V|u-\pi(u)| + \sum_{\substack{e\in E^G \\ \pi(e)\neq\epsilon_E}} \mathrm{Cost}_{P_\pi}(e) + \sum_{\substack{e\in E^G \\ \pi(e)=\epsilon_E}} C_E|e| + \sum_{\substack{f\in E^H \\ \pi^{-1}(f)=\epsilon_E}} C_E|f|$$

$$= \left[\sum_{\substack{u\in V^G \\ \pi(u)\neq\epsilon_V}} C_V|u-\pi(u)| + \sum_{\substack{e\in E^G \\ \pi(e)=\epsilon_E}} C_E|e| + \sum_{\substack{f\in E^H \\ \pi^{-1}(f)=\epsilon_E}} C_E|f|\right] + \sum_{\substack{e\in E^G \\ \pi(e)\neq\epsilon_E}} \mathrm{Cost}_{P_\pi}(e)$$

$$\leq \mathrm{Cost}(\pi) + \sum_{\substack{e\in E^G \\ \pi(e)\neq\epsilon_E}} \mathrm{Cost}_{P_\pi}(e)$$

In order to get upper bound on the last term, we observe for any edge $e=(u_1,u_2)\in E^G$ with $\pi(e)\neq\epsilon_E$ that its orbit under T_V is $\{(u_1,u_2),(u_1,\pi(u_2)),(\pi(u_1),\pi(u_2))\}$. The cost of the orbit of each e then is

$$C_E\left(\big||u_1-\pi(u_2)|-|u_1-u_2|\big|+\big||\pi(u_1)-\pi(u_2)|-|u_1-\pi(u_2)|\big|\right) \leq C_E(|u_2-\pi(u_2)|+|u_1-\pi(u_1)|).$$

So,

$$\begin{aligned}
\mathrm{Cost}(P_\pi) &\leq \mathrm{Cost}(\pi) + \sum_{\substack{e\in E^G \\ \pi(e)\neq\epsilon_E}} \mathrm{Cost}_{P_\pi}(e) \\
&\leq \mathrm{Cost}(\pi) + \sum_{\substack{e=(u_1,u_2)\in E^G \\ \pi(e)\neq\epsilon_E}} C_E(|u_2-\pi(u_2)|+|u_1-\pi(u_1)|) \\
&\leq \mathrm{Cost}(\pi) + \Delta \sum_{\substack{u\in E^V \\ \pi(u)\neq\epsilon_V}} C_E|u-\pi(u)| \\
&= \mathrm{Cost}(\pi) + \Delta\frac{C_E}{C_V} \sum_{\substack{u\in E^V \\ \pi(u)\neq\epsilon_V}} C_V|u-\pi(u)| \\
&\leq \mathrm{Cost}(\pi) + \Delta\frac{C_E}{C_V}\mathrm{Cost}(\pi) = \left(1+\Delta\frac{C_E}{C_V}\right)\mathrm{Cost}(\pi).
\end{aligned}$$

By the definition GED, it is implied that $\mathrm{GED}(G,H) \leq \left(1+\Delta\frac{C_E}{C_V}\right)\mathrm{Cost}(\pi)$. Since π is chosen arbitrarily, we then conclude from the definition of GGD that $\mathrm{GED}(G,H) \leq \left(1+\Delta\frac{C_E}{C_V}\right)\mathrm{GGD}(G,H)$. □

We remark that the configuration in Fig. 1 and Proposition 17 show that the bounds presented in Proposition 18 are, in fact, tight.

3. Computational complexity

In this section, we discuss the computational aspects of the GGD. The computation is algorithmically feasible, since the there are only a finite number of matchings between two graphs. However, it has been already shown in [14] that the distance is generally hard to compute. We define the decision problem as follows.

Definition 19 *(PROBLEM* GGD*).* Given geometric graphs $G,H\in\mathcal{G}(\mathbb{R}^d)$ and $\tau\geq 0$, is there a matching $\pi\in\Pi(G,H)$ such that $\mathrm{Cost}(\pi)\leq\tau$?

We remark here that C_V, C_E, and τ can be chosen as arbitrary positive real numbers. In [14], the authors show that PROBLEM GGD is $\mathcal{NP}$-hard for non-planar graphs. For planar graphs, however, its $\mathcal{NP}$-hardness is proved under the very strict condition that $C_V << C_E$. In both cases, the problem instances seem *non-practical*. In Proposition 21, we prove a stronger result that the problem is $\mathcal{NP}$-hard, even if the graphs are planar and arbitrary C_V, C_E are allowed. Our reduction is from the well-known 3-PARTITION problem.

Definition 20 *(Problem* 3*-PARTITION).* Given positive integers $N>1$, B and a multiset of positive integers $S=\{a_1,a_2,\ldots,a_{3N}\}$ so that $\frac{B}{4}<a_i<\frac{B}{2}$ and $\sum_{i=1}^{3N}a_i=NB$, does there exist a partition of S into N multisets $S_1,S_2,\ldots,S_N$ such that $|S_i|=3$ and $\sum_{a\in S_i}a=B$ for all $1\leq i\leq N$?

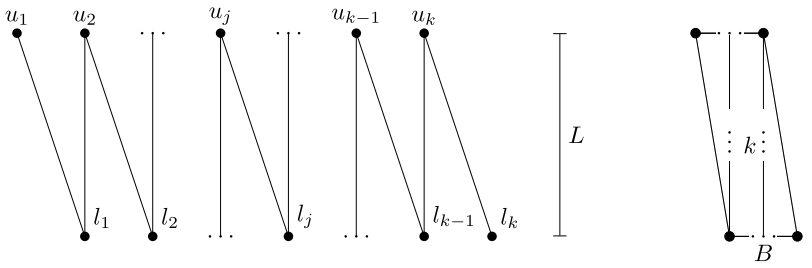

Fig. 4. Left: A typical blob B of size k is shown. Right: The shorthand for such a blob is depicted.

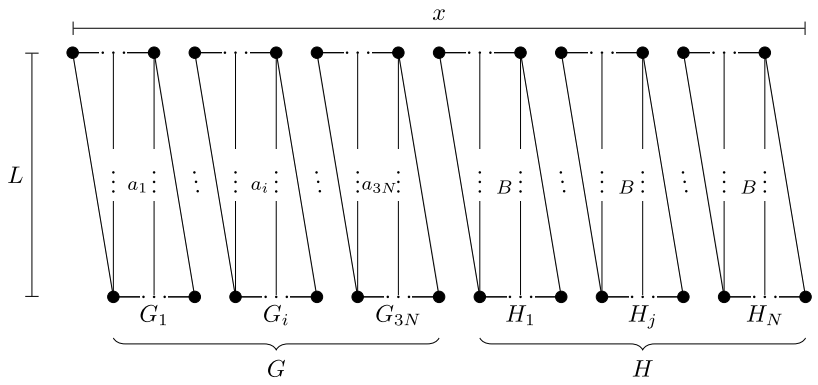

Fig. 5. Encoding an instance of 3-PARTITION into planar graphs G, H.

The problem is known to be strongly $\mathcal{NP}$-complete [15]. We reduce an instance $\mathcal{I} := (N, B, S)$ of 3-PARTITION to an instance of PROBLEM GGD.

Proposition 21 *(Hardness of PROBLEM* GGD*). The PROBLEM* GGD *is $\mathcal{NP}$-hard to decide. This result holds even if*

(i) *the input graphs are embedded in $\mathbb{R}^2$, and*
(ii) *the cost coefficients C_E, C_V are arbitrary.*

Proof. Given an instance $\mathcal{I} := (N, B, S)$ of 3-PARTITION, we construct two planar graphs G, H such that the existence of a 3-PARTITION of S implies $\mathrm{GGD}(G, H) \leq \tau$, otherwise $\mathrm{GGD}(G, H) > \tau$.

We now describe the construction of G and H. Each of them will have a certain number of connected components, which we call *blobs*. A blob of size k is a connected block of k vertices $\{u_1, u_2, \ldots, u_k\}$ in the upper row and k vertices $\{l_1, l_2, \ldots, l_k\}$ in the lower row. The two rows are separated by distance L, and the consecutive vertices in each row are equidistant. The choice of L will be made explicit later on. Except for u_1, each vertex u_j in the upper row is connected to l_{j-1} and l_j in the bottom row, making the blob path-connected. The configuration of such a typical blob and its shorthand are depicted in Fig. 4.

We define G as the graph with $3N$ many blobs $G_1, G_2, \ldots, G_{3N}$ of size $a_1, a_2, \ldots, a_{3N}$, respectively, placed side-by-side so that they do not overlap. Now, H is defined as the graph with N many blobs $H_1, H_2, \ldots, H_N$ of size B each placed side-by-side so that they do not overlap. Now, G and H are placed side-by-side in a bounding-box of width x and height L, where

$$x = \frac{\tau}{2C_V(N+1)NB}, \text{ and } L = \frac{\tau}{2C_E(N+1)}.$$

We remark that appropriately small inter-vertex and inter-blob distances can always be chosen to fit them in the bounding-box, keeping the length of all the vertical (resp. slanted) edges the same. See Fig. 5 for the configuration of the graphs.

Let us first assume that $\mathcal{I}$ is a YES instance, and that $\{S_1, S_2, \ldots, S_N\}$ is a partition of S. A (bijective) matching $\pi \in \Pi(G, H)$ can be defined in the following way. For any $i \in \{1, 2, \ldots, N\}$, if $S_i = \{a_{i_1}, a_{i_2}, a_{i_3}\}$ then the upper and lower vertices of the blobs G_{i_1}, G_{i_2}, and G_{i_3} of G are mapped, consecutively, to the corresponding upper and lower vertices of the ith blob H_i of H. We argue that $\mathrm{Cost}(\pi) \leq \tau$. In light of (3), the cost is the total contribution from the following two types:

(a) There are $(2NB - 3N)$ many edges in G, whereas there are $(2NB - N)$ many in H. So, there are exactly $2N$ many vertical edges e in H such that $\pi^{-1}(e) = \epsilon_V$. The resulting cost is at most $C_E \cdot 2N \cdot L$.

(b) Since no vertex in the upper row is mapped to a vertex in the lower row and vice versa, we have

$$|u - \pi(u)| \leq x \text{ for all } u \in G.$$

There are $2(\sum_i^{3N} a_i) = 2NB$ many vertices in G, so the total cost for vertex translation is at most $C_V \cdot x \cdot 2NB$.

As a result, the total cost is

$$\text{Cost}(\pi) \leq 2C_E NL + 2C_V NBx = 2C_E N\frac{\tau}{2C_E(N+1)} + 2C_V NB\frac{\tau}{2C_V(N+1)NB} = \tau.$$

Hence, $\text{GGD}(G, H) \leq \tau$.

For the other direction, we assume that $\text{GGD}(G, H) \leq \tau$, i.e., there is a matching $\pi \in \Pi(G, H)$ such that $\text{Cost}(\pi) \leq \tau$. We observe that $\pi(V^G) \neq \{\epsilon_V\}$. Otherwise, from (3) the cost of π would be

$$\text{Cost}(\pi) \geq C_E\text{Vol}(G) + C_E\text{Vol}(H) \geq C_E(4NB - 4N)L = 4C_E N(B-1)\frac{\tau}{2C_E(N+1)} = \frac{2N(B-1)\tau}{N+1} > \tau.$$

The above volume estimates use the fact that there are $(2NB - 3N)$ edges in G and $(2NB - N)$ edges in H, and the length of each edge is at least L. Also, the last inequality above is strict because $2N > N+1$ for any $N > 1$. Since this is a contradiction, there must be some $u_0 \in V^G$ with $\pi(u_0) \neq \epsilon_V$.

Moreover, we claim that $\pi : V^G \to V^H$ must be a bijection. Let us assume the contrary, i.e., there is $u_1 \in V^G$ such that $\pi(u_1) = \epsilon_V$. Since there is at least one edge (of length at least L) incident to u_1, we then have from Lemma 14,

$$\begin{aligned}
\text{Cost}(\pi) &\geq C_V|u_0 - \pi(u_0)| + C_E[\text{Vol}(H) - \text{Vol}(G)] + 2C_E L \\
&\geq C_V|u_0 - \pi(u_0)| + C_E \cdot 2N \cdot L + 2C_E L \\
&= C_V|u_0 - \pi(u_0)| + 2C_E(N+1)L \\
&= C_V|u_0 - \pi(u_0)| + 2C_E(N+1)\frac{\tau}{2C_E(N+1)} \\
&= C_V|u_0 - \pi(u_0)| + \tau.
\end{aligned}$$

Since the graphs are non-overlapping, $|u_0 - \pi(u_0)| > 0$. Hence, $\text{Cost}(\pi) > \tau$. This is a contradiction, so π must be a bijection. Finally, we show that π defines a partition of S by arguing that a blob G_r of G cannot split into two blobs H_s and H_t of H when mapped by π. If it did, there would an edge e_0 of G_r with $\pi(e_0) = \epsilon_E$, since the blobs H_s and H_t are not connected. This would lead to a contradiction using the exact same argument just presented. Therefore, π defines a partition of the blobs of G, so a partition of S. This completes the proof. □

4. Graph mover's distance (GMD)

We define the *Graph Mover's Distance* for two ordered geometric graphs. A geometric graph is called *ordered* if its vertices are ordered or indexed. In that case, we denote the vertex set as a (finite) sequence $V^G = \{u_i\}_{i=1}^m$. Let us denote by $\mathcal{G}^O(\mathbb{R}^d)$ the set of all ordered geometric graphs of $\mathbb{R}^d$. The formulation of the GMD uses the framework known as the earth mover's distance (EMD).

4.1. Earth mover's distance (EMD)

The EMD is a well-studied distance measure between weighted point sets, with many successful applications in a variety of domains; for example, see [16–19]. The idea of the EMD was first conceived by Monge [20] in 1781, in the context of transportation theory. The name "earth mover's distance" was coined only recently, and is well-justified due to the following analogy. The first weighted point set can be thought of as piles of earth (dirt) lying on the point sites, with the weight of a site indicating the amount of earth; whereas, the other point set as pits of volumes given by the corresponding weights. Given that the total amount of earth in the piles equals the total volume of the pits, the EMD computes the least (cumulative) cost needed to fill all the pits with earth. Here, a unit of cost corresponds to moving a unit of earth by a unit of "ground distance" between the pile and the pit.

The EMD can be cast as a transportation problem on a bipartite graph, which has several efficient implementations, e.g., the network simplex algorithm [21,22]. Let the weighted point sets $P = \{(p_i, w_{p_i})\}_{i=1}^m$ and $Q = \{(q_j, w_{q_j})\}_{j=1}^n$ be a set of suppliers and a set of consumers, respectively. The weight w_{p_i} denotes the total supply of the supplier p_i, and w_{q_j} the total demand of the consumer q_j. The matrix $[d_{i,j}]$ is the matrix of ground distances, where $d_{i,j}$ denotes the cost of transporting a unit of supply from p_i to q_j. We also assume the *feasibility condition* that the total supply equals the total demand:

$$\sum_{i=1}^{m} w_{p_i} = \sum_{j=1}^{n} w_{q_j}. \tag{5}$$

A *flow* of supply is given by a matrix $[f_{i,j}]$ with $f_{i,j}$ denoting the units of supply transported from p_i to q_j. We want to find a flow that minimizes the overall cost

$$\sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j}d_{i,j}$$

subject to:

$$f_{i,j} \geq 0 \text{ for any } i = 1,\ldots,m \text{ and } j = 1,\ldots,n \tag{6}$$

$$\sum_{j=1}^{n} f_{i,j} = w_i \text{ for any } i = 1,\ldots,m \tag{7}$$

$$\sum_{i=1}^{m} f_{i,j} = w_j \text{ for any } j = 1,\ldots,n. \tag{8}$$

Constraint (6) ensures a flow of units from P to Q, and not vice versa; constraint (7) dictates that a supplier must send all its supply—not more or less; constraint (8) guarantees that the demand of every consumer is exactly fulfilled.

The *earth mover's distance* (EMD) is then defined by the cost of the optimal flow. A solution always exists, provided condition (5) is satisfied. The weights and the ground distances can be chosen to be any non-negative numbers. However, we choose them appropriately in order to solve our graph matching problem.

4.2. Defining the GMD

Let $G, H \in \mathcal{G}^O(\mathbb{R}^d)$ be two ordered geometric graphs of $\mathbb{R}^d$ with $V^G = \{u_i\}_{i=1}^m$ and $V^H = \{v_j\}_{j=1}^n$. For each $i = 1,\ldots,m$, let E_i^G denote the (row) m–vector containing the lengths of (ordered) edges incident to the vertex u_i of G. More precisely, the

$$k\text{th element of } E_i^G = \begin{cases} |e_{i,k}^G|, \text{ if } e_{i,k}^G := (u_i, u_k) \in E^G \\ 0, \text{ otherwise.} \end{cases}$$

Similarly, for each $j = 1,\ldots,n$, we define E_j^H to be the (row) n–vector with the

$$k\text{th element of } E_j^H = \begin{cases} |e_{j,k}^H|, \text{ if } e_{j,k}^H := (v_j, v_k) \in E^H \\ 0, \text{ otherwise.} \end{cases}$$

In order to formulate the desired instance of the EMD, we take the point sets to be $P = \{u_i\}_{i=1}^{m+1}$ and $Q = \{v_j\}_{j=1}^{n+1}$. Here, u_{m+1} and v_{n+1} have been taken to be a dummy supplier and dummy consumer, respectively, to incorporate vertex deletion into our GMD framework. The weights on the sites are defined as follows:

$$w_{u_i} = 1 \text{ for } i = 1,\ldots,m \text{ and } w_{u_{m+1}} = n\,.$$

And,

$$w_{v_j} = 1 \text{ for } j = 1,\ldots,n \text{ and } w_{v_{m+1}} = m\,.$$

We note that the feasibility condition (5) is satisfied: $m + n$ is the total weight for both P and Q. An instance of the transportation problem is depicted in Fig. 6.

Finally, the ground distance from u_i to v_j is defined by:

$$d_{i,j} = \begin{cases} C_V|u_i - v_j| + & C_E\|E_i^G D_{m\times p} - E_j^H D_{n\times p}\|_1, \\ & \text{if } 1 \leq i \leq m, 1 \leq j \leq n \\ C_E\|E_j^H\|_1, & \text{if } i = m+1 \text{ and } 1 \leq j \leq n \\ C_E\|E_i^G\|_1, & \text{if } 1 \leq i \leq m \text{ and } j = n+1 \\ 0, & \text{otherwise.} \end{cases}$$

Here, $p = \min\{m, n\}$, the 1–norm of a row vector is denoted by $\|\cdot\|_1$, and D denotes a diagonal matrix with all the diagonal entries being 1.

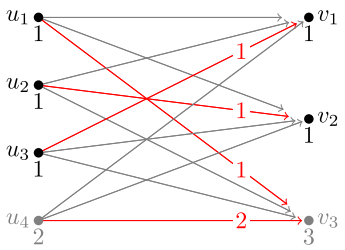

Fig. 6. The bipartite network used by the GMD is shown for two ordered graphs G, H with vertex sets $V^G = \{u_1, u_2, u_3\}$ and $V^H = \{v_1, v_2\}$, respectively. The dummy nodes u_4 for G and v_3 for H, respectively, have been shown in gray. Below each node, the corresponding weights are shown. A particular flow has been depicted here. The gray edges do not transport anything. A red edge has a non-zero flow with the transported units shown on them. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

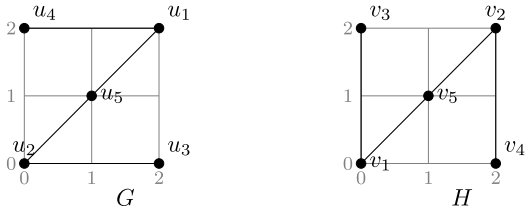

Fig. 7. For the geometric graph $G, H \in \mathcal{G}^O(\mathbb{R}^2)$, the GMD is zero. The optimal flow is given by the matching $\pi(u_1) = v_2$, $\pi(u_2) = v_1$, $\pi(u_3) = v_4$, $\pi(u_4) = v_3$, and $\pi(u_5) = v_5$.

4.3. Metric properties

We can see that the GMD induces a pseudo-metric on the space of ordered geometric graphs $\mathcal{G}^O(\mathbb{R}^d)$. Non-negativity, symmetry, and triangle inequality follow from those of the cost matrix $[d_{i,j}]$ defined in the GMD.

In addition, we note that $G = H$ (as ordered graphs) implies that $d_{i,j} = 0$ whenever $i = j$. The trivial flow, where each u_i sends its full supply to v_i, has a zero cost. So, $\mathrm{GMD}(G, H) = 0$. The GMD does not, however, satisfy the separability condition on $\mathcal{G}^O(\mathbb{R}^d)$.

For the graphs G, H as shown in Fig. 7, we have $\mathrm{GMD}(G, H) = 0$. We note that G and H have the following adjacency length matrices $[E_i^G]_i$ and $[E_j^H]_j$, respectively:

$$\begin{bmatrix} 0 & 0 & 0 & 2 & \sqrt{2} \\ 0 & 0 & 2 & 0 & \sqrt{2} \\ 0 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ \sqrt{2} & \sqrt{2} & 0 & 0 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & 0 & 2 & 0 & \sqrt{2} \\ 0 & 0 & 0 & 2 & \sqrt{2} \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ \sqrt{2} & \sqrt{2} & 0 & 0 & 0 \end{bmatrix}.$$

It can be easily checked that the flow that transports a unit of supply from $u_1 \mapsto v_2$, $u_2 \mapsto v_1$, $u_3 \mapsto v_4$, $u_4 \mapsto v_3$, $u_5 \mapsto v_5$, and five units from $u_6 \mapsto v_6$ has total cost zero. So, $\mathrm{GMD}(G, H) = 0$. However, the graphs G and H are not the same geometric graph.

One can easily find even simpler configurations for two distinct geometric graphs with a zero GMD—if the graphs are allowed to have multiple connected components.

4.4. Computing the GMD

As pointed out earlier, the GMD can be computed as an instance of transportation problem—using, for example, the network simplex algorithm. If the graphs have at most n vertices, computing the ground cost matrix $[d_{i,j}]$ takes $O(n^3)$-time. Since the bipartite network has $O(n)$ vertices and $O(n^2)$ edges, the simplex algorithm runs with a time complexity of $O(n^3)$, with a pretty good constant. Overall, the time complexity of the GMD is $O(n^3)$.

5. Experimental results

We have implemented the GMD in Python, using network simplex algorithm from the `networkx` package. We ran a pattern retrieval experiment on letter drawings from the IAM Graph Database [12]. The repository provides an extensive collection of graphs, both geometric and labeled.

In particular, we performed our experiment on the `LETTER` database from the repository. The graphs in the database represent distorted letter drawings. The database considers only 15 uppercase letters from the English alphabet: `A`, `E`, `F`, `H`,

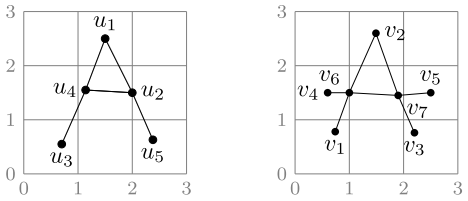

Fig. 8. The prototype geometric graph of the letter A is shown on the left. On the right, a (MED) distorted letter A is shown.

Table 2
Empirical result on the LETTER dataset.

	correct letter in first k models (%)		
Distortion	$k = 1$	$k = 3$	$k = 5$
LOW	96.66%	98.93%	99.37%
MED	66.66%	85.37%	91.15%
HIGH	73.73%	90.48%	95.51%

I, K, L, M, N, T, V, W, X, Y, and Z. For each letter, a prototype line drawing has been manually constructed. On the prototypes, distortions are applied with three different levels of strengths: LOW, MED, and HIGH, in order to produce 2250 letter graphs for each level. Each test letter drawing is a graph with straight-line edges; each node is labeled with its two-dimensional coordinates. Since some of the graphs in the dataset were not embedded, we had to compute the intersections of the intersecting edges and label them as nodes. The preprocessing guaranteed that all the considered graphs were geometric; a prototype and a distorted graph are shown in Fig. 8.

We devised a classifier for these letter drawings using the GMD. For this application, we chose $C_V = 4.5$ and $C_E = 1$ heuristically for best results. For a test letter, we computed its GMD from the 15 prototypes, then sorted the prototypes in an increasing order of their distance to the test graph. We then check if the letter generating the test graph is among the first k prototypes. For each level of distortion and various values of k, we present the rate at which the correct letter has been found in the first k models. A summary of the empirical results has been shown in Table 2. Although the graph edit distance (GED) based k-NN classifier still outperforms the GMD by a very small margin, our results have been extremely satisfactory. One possible reason why the GMD might fail to correctly classify some of the graphs is that it lacks the separability property as a metric.

6. Discussions and future work

We have studied two notions for a similarity measure between geometric graphs. In addition to the metric properties of GED and GGD, we also establish tight bounds in order to compare them. Although the distance measures induce equivalent metrics on the space of geometric graphs, it is not clear which one is better performant in practical applications. We have also shown the hardness of computing the GGD even for planar graphs. This naturally provokes the question of the hardness of its polynomial-time approximation. We conjecture that for any $\alpha > 1$, an α-approximation is also $\mathcal{NP}$-hard, i.e., PROBLEM GGD is generally $\mathcal{APX}$-hard. One can also investigate an alternative version of the GED that is algorithmically feasible to compute. This can probably be achieved by putting the graphs on a (Euclidean) grid and avoiding redundant edit operations in an edit path. It also remains unclear how to adjust the definitions of the proposed distances to incorporate merging and splitting of vertices and edges.

We have successfully introduced an efficiently computable and meaningful similarity measure for geometric graphs. However, the GMD lacks some of the desirable properties, like separability. It will be interesting to study the exact class of geometric graphs for which the GMD is, in fact, a metric.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Carola Wenk reports financial support was provided by National Science Foundation. Co-author Carola Wenk is on the editorial board of the current journal.

Data availability

No data was used for the research described in the article.

Acknowledgements

The authors thank Erfan Hosseini, Erin Chambers, and Elizabeth Munch for fruitful discussions.

References

[1] K.-S. Fu, P. Swain, On syntactic pattern recognition, in: J.T. Tou (Ed.), Computer and Information Sciences – 1969, in: SEN Report Series Software Engineering, vol. 2, Elsevier, 1971, pp. 155–182.
[2] K.-S. Fu, B. Bhargava, Tree systems for syntactic pattern recognition, IEEE Trans. Comput. C-22 (12) (1973) 1087–1099.
[3] P. Willett, Similarity searching in databases of three-dimensional chemical structures, in: H.-H. Bock, W. Lenski, M.M. Richter (Eds.), Information Systems and Data Analysis, Springer, 1994, pp. 280–293.
[4] J.W. Raymond, P. Willett, Effectiveness of graph-based and fingerprint-based similarity measures for virtual screening of 2D chemical structure databases, J. Comput.-Aided Mol. Des. 16 (1) (2002) 59–71.
[5] J. Liu, Y.T. Lee, Graph-based method for face identification from a single 2d line drawing, IEEE Trans. Pattern Anal. Mach. Intell. 23 (10) (2001) 1106–1119.
[6] J. Llados, E. Marti, J. Villanueva, Symbol recognition by error-tolerant subgraph matching between region adjacency graphs, IEEE Trans. Pattern Anal. Mach. Intell. 23 (10) (2001) 1137–1143.
[7] D.G. Corneil, C.C. Gotlieb, An efficient algorithm for graph isomorphism, J. ACM 17 (1) (1970) 51–64.
[8] A. Sanfeliu, K.-S. Fu, A distance measure between attributed relational graphs for pattern recognition, IEEE Trans. Syst. Man Cybern. SMC-13 (May 1983) 353–362.
[9] D. Justice, A. Hero, A binary linear programming formulation of the graph edit distance, IEEE Trans. Pattern Anal. Mach. Intell. 28 (Aug. 2006) 1200–1214.
[10] H. Bunke, G. Allermann, Inexact graph matching for structural pattern recognition, Pattern Recognit. Lett. 1 (May 1983) 245–253.
[11] M. Ahmed, S. Karagiorgou, D. Pfoser, C. Wenk, Map Construction Algorithms, first ed., Springer International Publishing, 2015.
[12] K. Riesen, H. Bunke, IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning, Structural, Syntactic, and Statistical Pattern Recognition, vol. 5342, Springer, 2008, pp. 287–297.
[13] S. Bougleux, L. Brun, V. Carletti, P. Foggia, B. Gaüzère, M. Vento, Graph edit distance as a quadratic assignment problem, Pattern Recognit. Lett. 87 (2017) 38–46.
[14] O. Cheong, J. Gudmundsson, H.-S. Kim, D. Schymura, F. Stehn, Measuring the similarity of geometric graphs, in: J. Vahrenhold (Ed.), Experimental Algorithms, vol. 5526, Springer, 2009, pp. 101–112.
[15] M.R. Garey, D.S. Johnson, Computers and Intractability; a Guide to the Theory of NP-Completeness, W. H. Freeman & Co., USA, 1990.
[16] C.J. Hargreaves, M.S. Dyer, M.W. Gaultois, V.A. Kurlin, M.J. Rosseinsky, The Earth mover's distance as a metric for the space of inorganic compositions, Chem. Mater. 32 (Dec. 2020) 10610–10620.
[17] M. Kusner, Y. Sun, N. Kolkin, K. Weinberger, From word embeddings to document distances, in: Proceedings of the 32nd International Conference on Machine Learning, PMLR, June 2015, pp. 957–966, ISSN 1938-7228.
[18] Z. Ren, J. Yuan, Z. Zhang, Robust hand gesture recognition based on finger-Earth mover's distance with a commodity depth camera, in: Proceedings of the 19th ACM International Conference on Multimedia, MM '11, New York, NY, USA, Association for Computing Machinery, Nov. 2011, pp. 1093–1096.
[19] Y. Rubner, C. Tomasi, L.J. Guibas, The Earth mover's distance as a metric for image retrieval, Int. J. Comput. Vis. 40 (Nov. 2000) 99–121.
[20] G. Monge, Mémoire sur la théorie des déblais et des remblais, Impr. R. (1781).
[21] R. Ahuja, T. Magnanti, J. Orlin, Network Flows: Theory, Algorithms, and Applications. Always Learning, Pearson, 2013.
[22] O. Pele, M. Werman, A linear time histogram metric for improved SIFT matching, in: D. Forsyth, P. Torr, A. Zisserman (Eds.), Computer Vision – ECCV 2008, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2008, pp. 495–508.