Computational Geometry: Theory and Applications 95 (2021) 101743

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Distance measures for embedded graphs ☆

Hugo A. Akitaya [a,1], Maike Buchin [b], Bernhard Kilgus [b,*,2], Stef Sijben [b], Carola Wenk [c,3]

[a] Department of Computer Science, Tufts University, Medford, MA, USA
[b] Department of Mathematics, Ruhr University Bochum, Bochum, Germany
[c] Department of Computer Science, Tulane University, New Orleans, USA

A R T I C L E I N F O

Article history:
Received 9 March 2020
Received in revised form 30 November 2020
Accepted 20 December 2020
Available online 5 January 2021

Keywords:
Fréchet distance
Graph comparison
Embedded graphs

A B S T R A C T

We introduce new distance measures for comparing straight-line embedded graphs based on the Fréchet distance and the weak Fréchet distance. These graph distances are defined using continuous mappings and thus take the combinatorial structure as well as the geometric embeddings of the graphs into account. We present a general algorithmic approach for computing these graph distances. Although we show that deciding the distances is NP-hard for general embedded graphs, we prove that our approach yields polynomial time algorithms if the graphs are trees, and for the distance based on the weak Fréchet distance if the graphs are planar embedded and if the embedding meets a certain geometric restriction. Moreover, we prove that deciding the distances based on the Fréchet distance remains NP-hard for planar embedded graphs and show how our general algorithmic approach yields an exponential time algorithm and a polynomial time approximation algorithm for this case.

1. Introduction

There are many applications that work with graphs that are embedded in Euclidean space. One task that arises in such applications is comparing two embedded graphs. For instance, the two graphs to be compared could be two different representations of a geographic network (e.g., roads or rivers). Oftentimes these networks are not isomorphic, nor is one interested in subgraph isomorphism, but one would like to have a mapping of one graph to the other, and ideally such a mapping would be continuous. For instance, this occurs when we have a ground truth of a road network and a simplification or reconstruction of the same network and we would like to measure the error of the latter. In this case, a mapping would identify the parts of the ground truth that are reconstructed/simplified and would allow one to study the local error.

We present new graph distance measures that are well-suited for comparing such graphs. Our distance measures are natural generalizations of the Fréchet distance [9] to graphs and require a continuous mapping, but they don't require graphs to be homeomorphic. One graph is mapped continuously to a portion of the other, in such a way that edges are

☆ This paper is a full version of the work presented at the 30th International Symposium on Algorithms and Computation (ISAAC 2019) [7].

* Corresponding author.
E-mail addresses: hugo.alves_akitaya@tufts.edu (H.A. Akitaya), maike.buchin@rub.de (M. Buchin), bernhard.kilgus@rub.de (B. Kilgus), stef.sijben@rub.de (S. Sijben), cwenk@tulane.edu (C. Wenk).

[1] Supported by National Science Foundation grants CCF-1422311 and CCF-1423615, and the Science Without Borders scholarship program.
[2] Supported by the Deutsche Forschungsgemeinschaft (DFG), project BU 2419/3-1.
[3] Supported by National Science Foundation grant CCF-1618469.

https://doi.org/10.1016/j.comgeo.2020.101743
0925-7721/

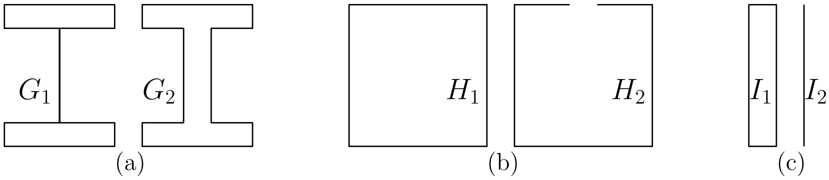


Fig. 1. Examples where our graph distances, the traversal distance, and the geometric edit distance differ. For clarity the graphs are shown side-by-side, but in the embedding they lie on top of each other. (a): Graphs G_1 and G_2 have large graph distance (because G_1 needs to mapped to one side of G_2), large edit distance (because a long edge needs to be added), but small traversal distance. (b): Graphs H_1 and H_2 have large graph distance (because all of H_1 must be mapped to only one side of H_2), but small traversal distance and small edit distance. (c): Graphs I_1 and I_2 have small graph distance and small traversal distance, but a large edit distance (because a long edge needs to be added).

mapped to paths in the other graph. The graph distance is then defined as the maximum of the strong (or weak) Fréchet distances between the edges and the paths they are mapped to. This results in a directed or asymmetric notion of distance, and we define the corresponding undirected distances as the maximum of both directed distances. The directed distances naturally arise when seeking to measure subgraph similarity, which requires mapping one graph to a subgraph of the other.

For comparing two not necessarily isomorphic graphs only few measures were known previously. One such measure is the traversal distance suggested by Alt et al. [8] and another is the geometric edit distance suggested by Cheong et al. [15]. The traversal distance converts graphs into curves by traversing the graphs continuously and comparing the resulting curves using the Fréchet distance. It is also a directed distance that compares the traversal of one graph with the traversal of a part of the other graph. However, an explicit mapping between the two graphs is not established, and part of the connectivity information of the graphs is lost due to the conversion to curves. The geometric edit distance minimizes the cost of edit operations from one graph to another, where cost is measured by Euclidean lengths and distances. But again, connectivity is not well maintained. Fig. 1 shows some examples of graphs where our graph distances, the traversal distance, and the geometric edit distance differ. In particular, only our graph distances capture the difference in connectivity between graphs G_1 and G_2, as well as between H_1 and H_2.

Our graph distances map one graph onto a subgraph of the other and they measure the Fréchet distance between the mapped parts (see Section 2.1 for a formal definition). Hence connectivity information is preserved and an explicit mapping between the two (sub-)graphs is established. One possible application of these new graph distances is the comparison of geographic networks, for instance evaluating the quality of map reconstructions and map simplification. In Section 5, we show some experimental results on graphs of map reconstructions that illustrate that our approach considers both geometry and connectivity.

Related work. A few approaches have been proposed in the literature for comparing geometric embedded graphs. Subgraph-isomorphism considers only the combinatorial structure of the graphs and not its geometric embedding. It is NP-hard to compute in general, although it can be computed in linear time if both graphs are planar and the pattern graph has constant size [18]. If we consider the graphs as metric spaces, the Gromov-Hausdorff distance (GH) between two graphs is the minimum Hausdorff distance between isometric embeddings of the graphs into a common metric space. While it is unknown how to compute GH for general graphs, recently Agarwal et al. [1] gave a polynomial time approximation algorithm for the GH between a pair of metric trees. We are however interested in measuring the similarity between two specific embeddings of the graphs. Armiti et al. [10] suggest a probabilistic approach for comparing graphs that are not required to be isomorphic, using spatial properties of the vertices and their neighbors. However, they require vertices to be matched to vertices, which can result in a large graph distance when an edge in one graph is subdivided in the other graph. Furthermore, the spatial properties used are invariant to translation and rotation, whereas we consider a fixed embedding. Cheong et al. [15] proposed the geometric edit distance for comparing embedded graphs, however it is NP-hard to compute. Alt et al. [8] defined the traversal distance, which is most similar to our graph distance measures, but it does not preserve connectivity. See Section 2.2 for a detailed comparison with the traversal distance.

For assessing the quality of map construction algorithms, several approaches have been proposed. One approach is to compare all paths [2] or random samples of shortest paths [19]. However, these measures ignore the local structure of the graphs. In order to capture more topological information, Biagioni and Eriksson developed a sampling-based distance [11] and Ahmed et al. introduced the local persistent homology distance [3]. The latter distance measure focuses on comparing the topology and does not encode geometric distances between the graphs. The sampling-based distance is not a formally defined distance measure, and it crucially depends on parameters (in particular *matched_distance*, to decide if points are sufficiently close to be matched); in practice it is unclear how these parameters should be chosen. However, it captures the number of matched edges, which is useful when comparing reconstructed road networks. In contrast to these measures, our graph distances capture more topology than the path-based distance [2], and capture differences in geometry better than the local persistent homology distance [3]. Also our graph distances are well-defined distance measures that do not require specific parameters to be set, unlike [11].

Contributions. We present new graph distance measures that compare graphs based on their geometric embeddings while respecting their combinatorial structure. To the best of our knowledge, our graph distances are the first to establish a continuous mapping between the embedded graphs. In Section 2 we define several variants of our graph distances (weak, strong, directed, undirected) and study their properties. In Section 3 we develop an algorithmic approach for computing the graph distances. On the one hand, we prove that for general embedded graphs, deciding these distances is NP-hard. On the other hand, we also show that our algorithmic approach gives polynomial time algorithms in several cases, e.g., when one graph is a tree. The most interesting case is when both graphs are plane. Here, we show that our algorithmic approach yields a quadratic time algorithm for the weak Fréchet distance for embeddings with a certain geometric property to be defined precisely in Lemma 7. In Section 4 we focus on plane graphs and the strong Fréchet distance. For this case, we show that the problem is NP-hard, even though it is polynomial time solvable for the weak Fréchet distance. Also, we show how to obtain an approximation, that depends on the angle between incident edges, in polynomial time and an exact result in exponential time.

2. Graph distance definition and properties

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two undirected graphs with vertices embedded as points in $\mathbb{R}^d$ (typically $\mathbb{R}^2$) that are connected by straight-line edges. We refer to such graphs as *(straight-line) embedded graphs*. Generally, we do not require the graphs to be planar. We denote a crossing free embedding of a planar graph shortly as a *plane graph*. Note that for plane graphs G_1 and G_2, crossings between edges of G_1 and edges of G_2 are still allowed.

2.1. Strong and weak graph distance

We define distance measures between embedded graphs that are based on mapping one graph to the other. We consider a particular type of graph mappings, as defined below:

Definition 1 *(Graph Mapping). We call a mapping $s\colon G_1 \to G_2$ a* graph mapping *if*

1. *it maps each vertex $v \in V_1$ to a point $s(v)$ on an edge of G_2, and*
2. *it maps each edge $\{u, v\} \in E_1$ to a simple path from $s(u)$ to $s(v)$ in G_2.*

When we say that we map to a point on an edge in G_2 we mean that we allow to map to a point in the embedding of the edge (or equivalently, inserting a new vertex at that point in the abstract graph). Note that a graph mapping results in a continuous map if we consider the graphs as suitable topological spaces. For planar embedded graphs, a graph mapping is a continuous mapping between the embeddings. This is however not the case for non-planar embeddings, as intersections in the embeddings are not considered to be vertices in the graphs.

To measure similarity between edges and mapped paths, our graph distances use the Fréchet distance or the weak Fréchet distance, which are popular distance measures for curves [9]. For two curves $f, g\colon [0,1] \to \mathbb{R}^d$ their *Fréchet distance* is defined as

$$\delta_F(f, g) = \inf_{\sigma\colon [0,1]\to[0,1]} \max_{t\in[0,1]} ||f(t) - g(\sigma(t))||,$$

where σ ranges over orientation preserving homeomorphisms. The *weak Fréchet distance* is

$$\delta_{wF}(f, g) = \inf_{\alpha,\beta\colon [0,1]\to[0,1]} \max_{t\in[0,1]} ||f(\alpha(t)) - g(\beta(t))|| ,$$

where α, β range over all continuous onto functions that keep the endpoints fixed.

Typically, the Fréchet distance is illustrated by a man walking his dog. Here, the Fréchet distance equals the shortest length of a leash that allows man and dog to walk on their curves from beginning to end. For the weak Fréchet distance man and dog may walk backwards on their curves, for the Fréchet distance they may not. The (weak) Fréchet distance between two polygonal curves of complexity n can be computed in $O(n^2 \log n)$ time [9]. Now, we are ready to define our graph distance measures.

Definition 2 *(Graph Distances). We define the* directed (strong) graph distance $\vec{\delta}_G$ *as*

$$\vec{\delta}_G(G_1, G_2) = \min_{s:G_1\to G_2} \max_{e\in E_1} \delta_F(e, s(e))$$

and the directed weak graph distance $\vec{\delta}_{wG}$ *as*

$$\vec{\delta}_{wG}(G_1, G_2) = \min_{s:G_1\to G_2} \max_{e\in E_1} \delta_{wF}(e, s(e)) ,$$

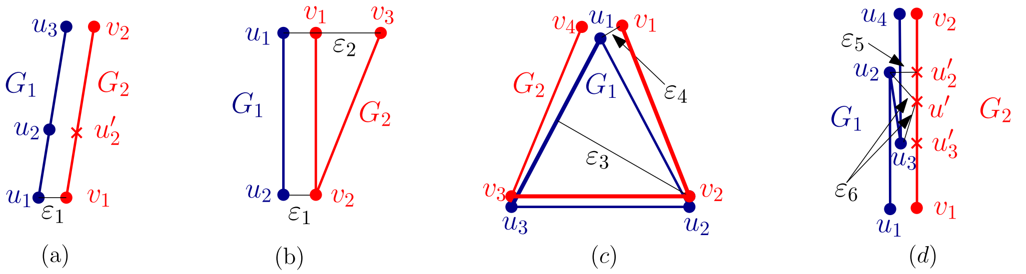


Fig. 2. Examples of graph mappings $s_1 : G_1 \to G_2$ and $s_2 : G_2 \to G_1$, and the resulting graph distances. Mapped vertices are drawn with crosses and are not graph vertices. (a) $\vec{\delta}_G(G_1, G_2) = \vec{\delta}_G(G_2, G_1) = \varepsilon_1$. $s_1(u_1) = v_1, s_1(u_2) = u'_2, s_1(u_3) = v_2$ and $s_2 = s_1^{-1}$. (b) $\vec{\delta}_G(G_1, G_2) = \varepsilon_1 < \varepsilon_2 = \vec{\delta}_G(G_2, G_1)$. The mapping $s_1(u_1) = v_1$ and $s_1(u_2) = v_2$ is not surjective, and $s_2(v_1) = s_2(v_3) = u_1$ and $s_2(v_2) = u_2$ is not injective. (c) $\vec{\delta}_G(G_1, G_2) = \varepsilon_3 > \varepsilon_4 = \vec{\delta}_G(G_2, G_1)$. $s_1(u_i) = v_i$ and $s_2(v_i) = u_i$ for $i = 1, 2, 3$; $s_2(v_4) = u_1$. (a)-(c) The weak graph distances equal the strong graph distances. (d) $\vec{\delta}_G(G_1, G_2) = \vec{\delta}_{wG}(G_1, G_2) = \vec{\delta}_{wG}(G_2, G_1) = \varepsilon_5 < \varepsilon_6 = \vec{\delta}_G(G_2, G_1)$. Here, the mappings that attain the strong graph distance are $s_1(u_1) = v_1, s_1(u_2) = u'_2, s_1(u_3) = u'_3, s_1(u_4) = v_2$ and $s_2(v_1) = u_1, s_2(v_2) = u_4$, where s_2 in the limit maps u' to all points on the edge from u_2 to u_3. The mappings attaining the weak graph distances are $s_1^w = s_1$ and $s_2^w = s_1^{-1}$.

where s ranges over graph mappings from G_1 to G_2, and e and its image $s(e)$ are interpreted as curves in the plane.[4] *The* undirected graph distances *are*

$$\delta_G(G_1, G_2) = \max(\vec{\delta}_G(G_1, G_2), \vec{\delta}_G(G_2, G_1)) \qquad \textit{and}$$
$$\delta_{wG}(G_1, G_2) = \max(\vec{\delta}_{wG}(G_1, G_2), \vec{\delta}_{wG}(G_2, G_1)).$$

According to Definition 1, a graph mapping s maps each edge e of G_1 to a simple path $s(e)$ in G_2. This is justified by the following observation: Mapping the edge e to a non-simple path $s'(e)$, where $s(e)$ and $s'(e)$ have the same endpoints and $s(e) \subset s'(e)$, does not decrease the (weak) graph distance because $\delta_{(w)F}(e, s(e)) \leq \delta_{(w)F}(e, s'(e))$. From this observation also follows that we cannot decrease $\vec{\delta}_G(G_1, G_2)$ by adding additional vertices to subdivide an edge e of G_1: While the concatenation of the resulting mapped paths in G_2 may not be simple, it can be replaced by the image of the entire edge e, which by the observation has to be simple.

The difference between the strong and weak graph distance goes back to the distance between the strong and weak Fréchet distance: the weak graph distance allows to map an edge of G_1 with small distance to a zig-zag path in G_2, the strong graph distance does not.

We state a first important property of the graph distances:

Lemma 1. *For embedded graphs, the strong graph distances and the weak graph distances fulfill the triangle inequality. The undirected distances are pseudo-metrics. For plane graphs they are metrics.*

Proof. Symmetry follows immediately for the undirected distances. The directed distances fulfill the triangle inequality because we can concatenate two maps and use the triangle inequality of $\mathbb{R}^d$: Let G_1, G_2 and G_3 be three embedded graphs. An edge e of G_1 is mapped to a simple path p in G_2. The segments of p are again mapped to a sequence of simple paths in G_3. Thus, when concatenating two maps, one possible mapping maps each edge e of G_1 to a sequence S of simple paths in G_3. Note, that S need not be simple. However, in that case we can instead map e to a shortest path $\hat{p}$ in S from beginning to end. As $\delta_{(w)F}(e, \hat{p}) \leq \delta_{(w)F}(e, S)$ for each edge of G_1, we have $\vec{\delta}_G(G_1, G_2) + \vec{\delta}_G(G_2, G_3) \geq \vec{\delta}_G(G_1, G_3)$ and $\vec{\delta}_{wG}(G_1, G_2) + \vec{\delta}_{wG}(G_2, G_3) \geq \vec{\delta}_G(G_1, G_3)$ by definition of the directed (weak) graph distance as the maximum Fréchet distance of an edge and its mapping. Analogously, the undirected distances fulfill the triangle inequality as well.

For plane graphs their (weak) graph distance is zero iff their embeddings are the same. Indeed, if a (weak) graph distance is zero, every edge needs to be mapped to itself, hence the embeddings are the same. If on the other hand, the embeddings are the same, a graph mapping may map every edge to itself in the embedding. Hence, for plane graphs the identity of discernible is fulfilled and the strong and weak graph distances are metrics. □

Note that for non-plane graphs the (weak) graph distance does not fulfill the identity of indiscernibles. For example, if G_1 consists of two crossing line segment edges, and G_2 has visually the same embedding but consists of four edges and includes the intersection point as a vertex, then both, $\vec{\delta}_G(G_1, G_2) = \vec{\delta}_{wG}(G_1, G_2) = 0$ and $\vec{\delta}_G(G_2, G_1) = \vec{\delta}_{wG}(G_2, G_1) = 0$, and therefore $\delta_G(G_1, G_2) = \delta_{wG}(G_1, G_2) = 0$. Also note that we do not require graph mappings to be injective or surjective. And an optimal graph mapping from G_1 to G_2 may be very different from an optimal graph mapping from G_2 to G_1. See Fig. 2 for examples of graphs and their graph distances.

[4] Here we slightly abuse notation, as we use e to refer to both the edge in (the abstract graph) G_1 as well as its embedding.

In the following, we show that the traversal distance between a graph G_1 and a graph G_2 is a lower bound for $\vec{\delta}_{wG}(G_1, G_2)$, which follows from the observation that the traversal distance captures the combinatorial structure of the graphs to a lesser extent than our graph distances. Furthermore, we apply the graph distances to measure the similarity between two polygonal paths to examine how these new definitions are generalizations of the (weak) Fréchet distance for curves to graphs.

2.2. Relation to traversal distance

A related distance measure for graphs was proposed by Alt et al. [8]. They define the *traversal distance* of two connected embedded graphs G_1, G_2 as

$$\delta_T(G_1, G_2) = \inf_{f,g} \max_{t\in[0,1]} ||f(t) - g(t)|| ,$$

where f ranges over all traversals of G_1 and g over all partial traversals of G_2. A *traversal* of G_1 is a continuous, surjective map $f: [0, 1] \to G_1$, and a *partial traversal* of G_2 is a continuous map $g: [0, 1] \to G_2$.

Thus, graphs G_1, G_2 have small traversal distance if there is a traversal of G_1 and a partial traversal of G_2 that stay close together. This could also be used for comparing a graph G_1 to a larger graph G_2. However, as we observe below, the traversals are not required to maintain the combinatorial structure of G_1 within G_2. First, we observe that our distance measures are stronger distances in the sense that

$$\delta_T(G_1, G_2) \le \vec{\delta}_{wG}(G_1, G_2) \le \vec{\delta}_G(G_1, G_2).$$

This follows because a graph mapping that realizes $\vec{\delta}_G(G_1, G_2) \le \varepsilon$ maps any traversal of G_1 to a partial traversal of G_2 with distance at most ε. For the weak graph distance, the traversal might need to be adjusted, so that it moves back along an already traversed path where the weak Fréchet matching requires it. Note that a traversal need not be injective.

However, the traversal distance captures the combinatorial structure of the graphs to a lesser extent than our measures. Fig. 2 (c) shows two graphs that have large graph distance (in particular the directed distance from G_1 to G_2 is large) but small traversal distance. If indeed G_1 is a map reconstruction and G_2 the ground truth we are comparing to, then the distance from G_1 to G_2 should be large.

2.3. Graph distance for paths

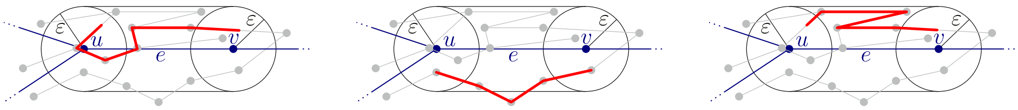
Consider the simple case that the graphs are paths embedded as polygonal curves. In this case, the (weak) graph distance is closely related to the (weak) Fréchet distance. If the graphs are straight-line embedded paths, that is each edge is embedded as a straight segment, the curve and graph distances are related, but not identical, as we show next. If instead the graphs are single edges embedded as polygonal curves, the graph and curve distances are in fact equal except for orientation of the curves.

A graph mapping between two paths maps vertices from one path to points on the other path, and it maps edges to the corresponding subpaths. In this case, we can characterize the graph distance in the *free space* [9], the geometric structure used for computing the Fréchet distance. Recall that for curves $f, g: [0, 1] \to \mathbb{R}^d$ the free space is defined as $F_\varepsilon(f, g) = \{(s, t) \mid d(f(s), g(t)) \le \varepsilon\}$, i.e., the subset of the product of parameter spaces such that the corresponding points in the image space have distance at most ε.

Observation 1. *Let P_1, P_2 be two polygonal curves parameterized over $[0, m]$ and $[0, n]$, respectively. A graph mapping realizing $\vec{\delta}_G(P_1, P_2) \le \varepsilon$ can be characterized as an x-monotone path in $[0, m] \times [0, n]$ from the left boundary to the right boundary that is y-monotone (either increasing or decreasing) in each column of the free space. A graph mapping realizing $\vec{\delta}_{wG}(P_1, P_2) \le \varepsilon$ is characterized by a path in $[0, m] \times [0, n]$ from the left boundary to the right boundary that is vertex-x-monotone, i.e., it is monotone in the traversal of the vertices on the x-axis.*

This observation implies relationships between graph distance and (weak) Fréchet distance that are summarized in Lemma 2. In this lemma we use the non-standard variant of (weak) Fréchet distance that does not require the homeomorphism to be orientation preserving, but allows one to choose an orientation.[5] Our graph distance does this naturally by choosing where to map. This variant of the Fréchet distance for curves can be computed by running the standard algorithm twice, i.e. searching for a path from bottom-left to top-right corner, as well as from top-left to bottom-right corner in the free space. Alternatively, we could enforce an orientation for the graph distance, e.g., using directions on the graphs.

[5] For the weak Fréchet distance one can drop the requirement that the reparameterizations α, β keep the endpoints fixed, also called *boundary restriction* [13]. Doing so, the distance possibly decreases.



(a) An ε-placement of e. (b) Not an ε-placement. (c) A weak ε-placement.

Fig. 3. (a) Illustration of ε-placements of an edge e. (b) Not an ε-placement because the path leaves the ε-tube around e. (c) The Fréchet distance is too large, but e can be mapped to the path if backtracking is allowed. Thus, it is a weak ε-placement.

Lemma 2. *Let P_1, P_2 be paths embedded as polygonal curves. Then*

$$\delta_{wF*}(P_1, P_2) \leq \delta_{wG}(P_1, P_2) \leq \delta_G(P_1, P_2) \leq \delta_F(P_1, P_2),$$

where δ_{wF*} denotes the weak Fréchet distance without boundary restriction.

Proof. The last inequality holds because a path in the free space realizing the Fréchet distance is a monotone path in x and y from the lower left to upper right corner, hence also realizes both undirected graph distances. For the first inequality, observe that two paths in the free space realizing the two directed weak graph distances can be combined to a path from the left to the right boundary realizing the weak Fréchet distance without boundary restriction. $\square$

Note that if we require mapping endpoints to themselves then the weak graph distance is also lower bounded by the weak Fréchet distance with boundary restriction. A 1D-example of two paths where δ_{wF} is strictly smaller than δ_{wG} when enforcing to map endpoints is the following: $P_1 = (0, 2, 0, 2)$ and $P_2 = (0, 1, 2)$. Here, $\delta_{wF}(P_1, P_2) = 0 < 1 = \delta_{wG}(P_1, P_2)$.

Also note that if we consider P_1, P_2 as single edges embedded as polygonal paths, the graph embedding is simply the same as a curve parameterization. Hence in this case both the weak and the strong distances for the curve and graph distance are equal.

Intuitively, and supported by the above lemma, our graph distance measures are at least as hard to compute as the Fréchet distance. It is known that the Fréchet distance of polygonal curves cannot be computed in less than subquadratic time unless the strong exponential time hypothesis fails [12]. Hence we also do not expect to compute our graph distance measures more efficiently than quadratic time.

3. Algorithms and hardness for embedded graphs

Recall that $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are two straight-line embedded graphs, and let $n_1 = |V_1|$, $m_1 = |E_1|$, $n_2 = |V_2|$ and $m_2 = |E_2|$. First, we consider the decision variants for the different graph distances defined in Definition 2. Given G_1 and G_2 and a value $\varepsilon > 0$, the decision problem for the graph distances is to determine whether $\vec{\delta}_G(G_1, G_2) \leq \varepsilon$ (resp., $\vec{\delta}_{wG}(G_1, G_2) \leq \varepsilon$). Equivalently, this amounts to determining whether there exists a graph mapping from G_1 to G_2 realizing $\vec{\delta}_G(G_1, G_2) \leq \varepsilon$ (resp., $\vec{\delta}_{wG}(G_1, G_2) \leq \varepsilon$). Note that the undirected distances can be decided by answering two directed distance decision problems. As we show in Section 3.3, the value of ε can be optimized by parametric search.

In Section 3.1 we describe a general algorithmic approach for solving the decision problems by computing *valid ε-placements* for vertices. We show that for general embedded graphs the decision problems for the strong and weak directed graph distances are NP-hard, see Section 3.2. However, we prove in Section 3.3 that our algorithmic approach yields polynomial-time algorithms for the strong graph distance if G_1 is a tree, and for the weak graph distance if G_1 is a tree or if both are plane graphs. In the latter scenario (G_1 and G_2 plane graphs), deciding if $\vec{\delta}_G(G_1, G_2) \leq \varepsilon$ remains NP-hard, see Section 4.1.

3.1. Algorithmic approach

Recall, that a (directional) graph mapping that realizes a given distance ε maps each vertex of G_1 to a point in (the embedding of) G_2 and each edge of G_1 to a simple path in (the embedding of) G_2 within this distance. In order to determine whether such a graph mapping exists, we define the notion of *ε-placements* of vertices and edges; see Figs. 3 and 4 (a).

Definition 3 *(ε-Placement). An ε-*placement *of a vertex v is a maximally connected part of G_2 restricted to the ε-ball $B_\varepsilon(v)$ around v. An ε-*placement *of an edge $e = \{u, v\} \in E_1$ is a path P in G_2 connecting placements of u and v such that $\delta_F(e, P) \leq \varepsilon$. In that case, we say that C_u and C_v are* reachable *from each other. An ε-*placement *of G_1 is a graph mapping $s\colon G_1 \to G_2$ such that s maps each edge e of G_1 to an ε-placement.*

A weak ε-placement *of an edge $e = \{u, v\}$ is a path P in G_2 connecting placements of u and v such that $\delta_{wF}(e, P) \leq \varepsilon$. A* weak ε-placement *of G_1 is a graph mapping $s\colon G_1 \to G_2$ such that s maps each edge e of G_1 to a weak ε-placement.*

Note that an ε-placement of a vertex v consists of edges and portions of edges of G_2, depending on whether $B_\varepsilon(v)$ contains both, one or zero endpoint(s) of the edge, see Fig. 4. Also note that each vertex has $O(m_2)$ ε-placements, since

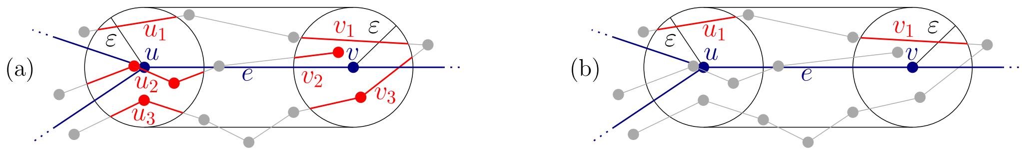


Fig. 4. Illustration of valid and invalid vertex placements. (a) Placements u_3 (resp. v_3) are invalid because they are not connected to a placement of v (resp. u) by an ε-placement of the edge e. Placement v_2 is only valid when considering e in isolation. However, it cannot be connected to any placement of a vertex other than u adjacent to v. Thus, it is also invalid. As a result of pruning v_2 (right), u_2 becomes invalid as well. Thus, only u_1 and v_1 remain as potentially valid placements of u and v (b).

an ε-placement is defined as a connected part of G_2 of maximal size inside $B_\varepsilon(v)$. Furthermore, we consider two graph mappings s_1 and s_2 from G_1 to G_2 to be equivalent in terms of the directed (weak) graph distance if for each vertex $v \in V_1$, $s_1(v)$ and $s_2(v)$ are points on the same ε-placement of v.

General Decision Algorithm. Our algorithm consists of the following four steps, which we describe in more detail below. We assume ε is fixed and use the term *placement* for an ε-placement.

Observe that each connected component of G_1 needs to be mapped to a connected component of G_2, and each connected component of G_1 can be mapped independently of the other components of G_1. Hence we can first determine the connected components of both graphs, and then consider mappings between connected components only. In the following we present an algorithm for determining if a mapping from G_1 to G_2, that realizes a given distance ε, exists, where both G_1 and G_2 are connected graphs.

Algorithm 1 General Decision Algorithm.

1: Compute vertex placements.
2: Compute reachability information for vertex placements.
3: Prune invalid placements.
4: Decide if there exists a placement for the whole graph G_1.

1. Compute vertex placements. We iterate over all vertices $v \in V_1$ and compute all their placements. Each vertex has $O(m_2)$ placements, so the total number of vertex-placements is $O(n_1 \cdot m_2)$, and they can be computed in $O(n_1 \cdot m_2)$ time using standard algorithms for computing connected components.

2. Compute reachability information of vertex placements. Next, we iterate over all edges $e = \{u, v\} \in E_1$ to determine all placements of its vertices that allow a placement of the edge. That is, we search for all pairs of vertex-placements C_u, C_v that are reachable from each other according to Definition 3.

For the weak graph distance, we need to find all pairs of placements of u and placements of v that can reach one another using paths contained in the ε-tube $T_\varepsilon(e)$ around e, i.e., the set of all points with distance $\leq \varepsilon$ to a point on e, see Fig. 3 (c). If we restrict G_2 to its intersection with the ε-tube, all placements in the same connected component are mutually reachable. Thus, each edge is processed in time linear in the size of G_2 using linear space per edge: For each connected component a pair of lists containing the placements of u and v in that component, respectively, is computed. So, all reachability information can be computed in $O(m_1 \cdot m_2)$ time and space. Note that the weak Fréchet distance between a straight line edge $e \in E_1$ and a simple path $s(e)$ in G_2 is the maximum of the Hausdorff distance between e and $s(e)$ and the distances of the endpoints of e and $s(e)$.

For the strong graph distance, existence of a path inside the ε-tube is not sufficient to describe the connectivity between placements. We must ensure that the Fréchet distance between e and P is at most ε, i.e., a continuous and monotone map s must exist from e to P such that $\|t - s(t)\| \leq \varepsilon$ for all $t \in e$. This can be decided in $O(|P|)$ time using the original dynamic programming algorithm for computing the Fréchet distance [9]. In order to determine whether such a path P exists, every placement of u stores a list of all placements of v that are reachable. The connectivity information can be computed by running a graph exploration, starting from each placement, which prunes a branch if the search leaves the ε-tube or backtracking on e is required to map it. This method runs a search for every placement of the start vertex and thus needs $O(m_2^2)$ time per edge of G_1. Since the connectivity is explicitly stored as pairs of placements that are mutually reachable, it also needs $O(m_2^2)$ space per edge. Hence, in total over all edges, $O(m_1 \cdot m_2^2)$ time and space are needed. Summing up, we have:

Lemma 3. *To run step 1 and step 2 of Algorithm 1, we need $O(m_1 \cdot m_2)$ time and space for the weak graph distance and $O(m_1 \cdot m_2^2)$ time and space for the strong graph distance.*

3. Prune invalid placements. Now, after having processed all vertices and edges, it still needs to be decided whether G_1 as a whole can be mapped to G_2. To this end, we delete *invalid* placements of vertices.

Definition 4 *(Valid Placement). An ε-placement C_v of a vertex v is* (weakly) valid *if for every neighbor u of v there exists an ε-placement C_u of u such that C_v and C_u are connected by a (weak) ε-placement of the edge $\{u, v\}$. Otherwise, C_v is* (weakly) invalid.

See Fig. 4 for an illustration of (in)valid placements. As shown in the Figure, deleting an invalid placement possibly sets former valid placements to be invalid. Thus, we need to process all placements recursively until all invalid placements are deleted and no new invalid placements occur. Note that the ordering of processing the placements does not affect the final result. To decide which placements of vertices u and v incident to an edge e are valid, we use the reachability information computed in Step 2.

Initially there are $O(n_1 \cdot m_2)$ vertex-placements, each of which may be deleted once. For the weak graph distance, connectivity is stored using connected components inside the ε-tube surrounding an edge $\{u, v\}$. On deleting a placement C_v of v, it is removed from the list containing placements of v. If a component no longer contains placements of v (i.e. its list becomes empty), then all placements of u in that component become invalid. A placement C_v is deleted at most once and upon deletion it must be removed from one list for every edge incident to v. Thus, the time for pruning C_v is $O(\deg(v))$. Since the sum of all degrees is $2m_1$, all invalid placements can be pruned in $O(m_1 \cdot m_2)$ time. For the strong graph distance, every placement has a list of placements to which it is connected. On deleting C_v, it must be removed from the lists of all placements C_u to which C_v is connected. Each vertex has $O(m_2)$ placements which have to be removed from a list for each neighbor of v. Thus, pruning a placement runs in $O(\deg(v) \cdot m_2)$ time and pruning all invalid placements in $O(m_1 \cdot m_2^2)$ time.

Lemma 4. *Pruning all invalid placements takes $O(m_1 \cdot m_2)$ time for the weak graph distance and $O(m_1 \cdot m_2^2)$ time for the strong graph distance.*

Note that after the pruning step all remaining vertex placements are (weakly) valid. However, the existence of a (weakly) valid placement for each vertex is not a sufficient criterion for $\vec{\delta}_G(G_1, G_2)$ ($\vec{\delta}_{wG}(G_1, G_2)$) in general, see Fig. 11.

4. Decide if there exists a placement for the whole graph G_1. After pruning all invalid placements, we want to decide if the remaining valid vertex-placements allow a placement of the whole graph G_1. The complexity of this step depends on the graph and the distance measure: for plane graphs we show that we can concatenate weakly valid placements of two adjacent faces (Lemma 7), whereas this is not possible for the directed strong graph distance in this setting (Theorem 5) or for general graphs for both distances (Theorem 2). Although deciding the directed (weak) graph distance is NP-hard for general graphs, there are two settings which may occur after running steps 1-3 of Algorithm 1, making step 4 of the algorithm trivial. Clearly $\vec{\delta}_G(G_1, G_2) > \varepsilon$ ($\vec{\delta}_{wG}(G_1, G_2) > \varepsilon$) if there is a vertex that has no (weakly) valid ε-placement. Furthermore, we have the following:

Lemma 5. *If, after running steps 1-3 of Algorithm 1, each internal vertex (degree at least two) has exactly one valid ε-placement (resp., weakly valid ε-placement) and each vertex of degree one has at least one valid ε-placement (resp., weakly valid ε-placement), then G_1 has an ε-placement (resp., weak ε-placement). Thus, $\vec{\delta}_G(G_1, G_2) \le \varepsilon$ (resp., $\vec{\delta}_{wG}(G_1, G_2) \le \varepsilon$).*

Lemma 3, Lemma 4 and Lemma 5 imply the following Theorem.

Theorem 1. *If there is a vertex that has no valid ε-placement or if each vertex has exactly one valid ε-placement after running steps 1-3 of Algorithm 1, the directed strong graph distance can be decided in $O(m_1 \cdot m_2^2)$ time and space. Analogously, if there is a vertex that has no weakly valid ε-placement or if each vertex has exactly one weakly valid ε-placement after running steps 1-3 of Algorithm 1, the directed weak graph distance can be decided in $O(m_1 \cdot m_2)$ time and space.*

Proof. Map each internal vertex v to a point $s(v)$ on its unique (weakly) valid placement C_v. Consider an edge $e = \{u, v\} \in E_1$. In the previous step, at least one (weak) placement P_e of e was discovered that connects points p_0 and p_k on C_u and C_v, respectively, since otherwise C_u and C_v would be invalid. If $p_k \ne s(v)$, P_e can be adapted by shortening it and/or concatenating a path on C_v (i.e. inside $B_\varepsilon(v)$) without causing its (weak) Fréchet distance to e to become $> \varepsilon$. Adapt P_e to a path P'_e that has endpoints $p'_0 = s(u)$ and $p'_k = s(v)$ and define $s(e) = P'_e$. Now, s is a graph mapping from G_1 to G_2 and each edge is mapped to a path with (weak) Fréchet distance at most ε, so $\vec{\delta}_G(G_1, G_2) \le \varepsilon$ (or $\vec{\delta}_{wG}(G_1, G_2) \le \varepsilon$). Recall, that each vertex w of degree one is either connected to an internal vertex i, or G_1 consists of only one edge $\{w, x\}$. The first case is already covered, since the unique valid (weak) placement C_i for i is reachable from any valid (weak) placement of w. The latter case follows because every vertex placement for w is valid, i.e., for each placement C_w for w there is a (weakly) reachable placement C_x for x and any combination of two reachable placements C_w and C_x yields a valid (weak) placement of G_1. □

Note that Theorem 1 yields an FPT-algorithm to decide $\vec{\delta}_G(G_1, G_2)$ parameterized in the product of the number of valid placements per vertex, as each combination of placements can be processed in $\mathcal{O}(m_1)$ time to determine if the combination induces a valid mapping of the whole graph G_1.

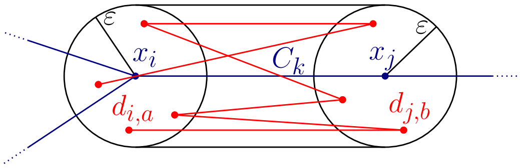


Fig. 5. Illustration of the reduction from BCSP. The blue edge represents a constraint on the variables x_i and x_j. The red edges represent eligible pairs of values for x_i and x_j. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

3.2. NP-hardness for the general case

Notwithstanding the special cases in Theorem 1, deciding the (weak) graph distance is not tractable for general graphs.

Theorem 2. *Deciding whether $\vec{\delta}_G(G_1, G_2) \le \varepsilon$ and deciding whether $\vec{\delta}_{wG}(G_1, G_2) \le \varepsilon$ for two graphs G_1 and G_2 embedded in $\mathbb{R}^2$ is NP-complete.*

Proof. First, we observe that deciding the (weak) graph distance is in NP: We can specify and verify a graph mapping realizing the distance in polynomial time. Note that although the original algorithm to decide the Fréchet distance computes Euclidean distances, squaring these distances avoids computing square roots.

We show NP-hardness with a reduction from the binary constraint satisfaction problem (CSP), which is defined as follows:

Problem 1. Binary Constraint Satisfaction Problem (CSP)
Instance: *A set of variables $X = \{x_1, \dots, x_n\}$, variable domains $D = \{D_1, \dots, D_n\}$ and a set of constraints $C = \{C_1, \dots, C_m\}$, where each constraint C has two variables x_i, x_j and a relation $R_C \subseteq D_i \times D_j$.*
Question: *Can each variable x_i be assigned a value $d_i \in D_i$ such that for each constraint C on variables x_i, x_j, their values (d_i, d_j) satisfy R_C?*

Consider an instance $\langle X, D, C\rangle$. Set $\varepsilon = 1$. Every variable x_i is represented by a vertex v_i in G_1 and for each constraint C_k on variables x_i, x_j, G_1 has an edge $\{v_i, v_j\}$ in G_1. We embed G_1 such that all adjacent ε-balls are separated by at least 2ε and no ε-tube of an edge overlaps an ε-ball that does not belong to one of the edge's endpoints. This can for example be realized by placing all vertices on a sufficiently large circle.

Every value $d_{i,a} \in D_i$ is represented by a vertex $u_{i,a}$ in G_2 that is inside the ε-ball of v_i. For each pair of values $d_{i,a} \in D_i$, $d_{j,b} \in D_j$ allowed by a constraint on x_i and x_j, G_2 has an edge $\{u_{i,a}, u_{j,b}\}$. This way, every vertex $u_{i,a}$ of G_2 defines exactly one ε-placement of the corresponding vertex v_i in G_1. Fig. 5 illustrates the construction.

A solution to the CSP consists of selecting a value $d_i \in D_i$ for each variable x_i, such that if there is a constraint on v_i and v_j, the pair $\{d_i, d_j\}$ satisfies the constraint. This is equivalent to selecting a placement d_i of each vertex v_i, such that if G_1 has an edge $\{v_i, v_j\}$, then G_2 has an edge connecting u_i, u_j representing d_i and d_j, respectively. The graph distance problem has the weaker requirement that there exists a path $P_{i,j}$ between u_i and u_j, such that $\delta_F(\{v_i, v_j\}, P_{i,j}) \le \varepsilon$. However, the construction is such that only paths consisting of a single edge are permitted for the strong distance, since ε-balls must be sufficiently separated and nonoverlapping with ε-tubes. So G_2 must have an edge $\{u_i, u_j\}$ if G_1 has an edge $\{v_i, v_j\}$. So, $\vec{\delta}_G(G_1, G_2) \le \varepsilon$ if and only if $\langle X, D, C\rangle$ is a satisfiable binary CSP.

For the weak graph distance, edges in G_1 can be mapped to paths consisting of multiple edges in G_2. In this case, there may be weak placements of G_1 that do not represent a solution to the constraint satisfaction instance. To remedy this, we insert a vertex in the middle of each edge of G_1. The vertex is placed such that its ε-ball is separated from the ε-balls of the original endpoints of the edge by at least 2ε, so each of the new edges is mapped to part of a single edge in G_2. In this construction $\vec{\delta}_{wG}(G_1, G_2) \le \varepsilon$ if and only if $\langle X, D, C\rangle$ is a satisfiable binary CSP. □

3.3. Efficient algorithms for plane graphs and trees

Here, we show that Algorithm 1 yields polynomial-time algorithms for deciding the strong graph distance if G_1 is a tree (Theorem 4), and the weak graph distance if G_1 is a tree or if both are plane graphs (Theorem 3). More precisely, we show that the existence of at least one (weakly) valid placement for each vertex is a sufficient condition for $\vec{\delta}_G(G_1, G_2) \le \varepsilon$ or $\vec{\delta}_{wG}(G_1, G_2) \le \varepsilon$.

Lemma 6. *If G_1 is a tree and every vertex of G_1 has at least one (weakly) valid ε-placement after running steps 1-3 of Algorithm 1, then G_1 has a (weak) ε-placement. Thus, $\vec{\delta}_G(G_1, G_2) \le \varepsilon$ (or $\vec{\delta}_{wG}(G_1, G_2) \le \varepsilon$).*

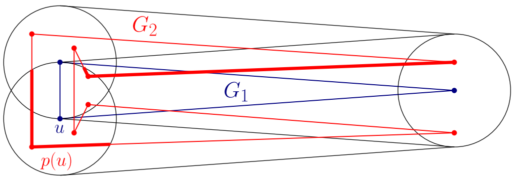


Fig. 6. G_1 consists of one thin cycle that can be mapped to the bold path in G_2. The placement $p(u)$ of the vertex v is only locally valid, but cannot be part of a valid mapping of the whole cycle.

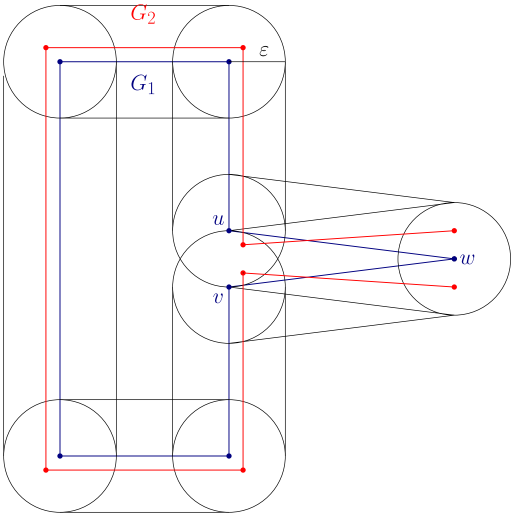


Fig. 7. All placements of G_1 are valid with respect to ε, but a mapping for the whole graph G_1 does not exist as we cannot start and end at the same placement of v.

Proof. We view G_1 as a rooted tree, selecting an arbitrary vertex as the root. We map all vertices of G_1 from the root outwards. First, map the root to an arbitrary (weakly) valid placement. When processing a vertex v: Map v to an arbitrary (weakly) valid placement that is reachable from the placement its parent p is mapped to. Recall, that a (weak) ε-placement of a vertex is *(weakly) valid* if there is an ε-placement for every incident edge. Since p was mapped to a (weakly) valid placement and there is an edge $\{p, v\}$ in G_1, there must be at least one such placement of v by definition of a (weakly) valid placement. Since all edges in G_1 are tree edges, this ensures that every edge is mapped correctly, that is to a path with (weak) Fréchet distance at most ε. $\square$

Plane graphs are more difficult to handle than trees and we need to distinguish between the strong and weak distance and between several geometric properties of the embeddings. In this section, we discuss the weak graph distance for plane graphs. The strong graph distance for plane graphs is discussed in Section 4.

First, let us consider a graph G_1 that consist only of a *thin cycle*, that is a cycle such that the shorter side of a minimum size bounding box around G_1 is at most 2ε. In this case, some of the vertex placements determined to be valid by step 3 of Algorithm 1 cannot be used to construct a valid mapping for the whole cycle G_1, see Fig. 6 for an illustration. Still, the property of a valid placement of each vertex ensures the existence of an (open) path in the planar embedding of G_2 or the existence of a simple cycle inside the ε-tube around G_1.

However, for constructing a valid placement of G_1, one needs to run a graph exploration that is exponential in the number of vertices of G_1. Thus, we can decide if $\vec{\delta}_{wG}(G_1, G_2) \leq \varepsilon$ in polynomial time if the embedding of G_1 consists of a thin cycle, but computing a specific valid mapping requires exponential time in general. Note that in the proof of the Lemma 7 we will construct a specific valid mapping for each cycle that can be merged iteratively. Hence, we conjecture that we cannot decide $\vec{\delta}_{wG}(G_1, G_2) \leq \varepsilon$ efficiently if G_1 contains at least one thin cycle.

Next, we observe that if G_1 consists of a *thick cycle*, that is a cycle were both sides of a minimum size bounding box have lengths greater than 2ε, the existence of a valid mapping for the whole graph does not follow from the existence of a valid placement for each vertex. In this case, we cannot guarantee that we start and end at the same placement when traversing the cycle, as shown in Fig. 7.

Thus, we conjecture that if the graph G_1 consists of or contains one thick cycle, we cannot decide $\vec{\delta}_{wG}(G_1, G_2)$ in polynomial time even if the embedding of G_2 is planar, because we need to run an exponential-time graph exploration to find a valid mapping of G_1 if such a mapping exists. The following conjecture summarizes the previous observations.

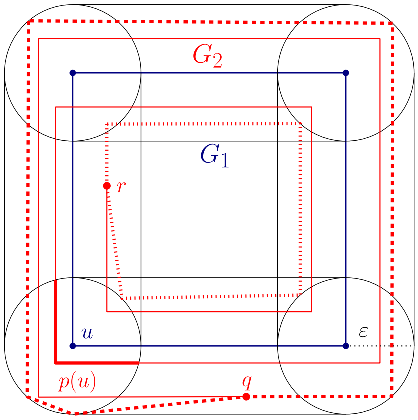


Fig. 8. The placement $p(u)$ is not part of a valid mapping for the whole cycle. However a path starting at $p(u)$ in either direction of the cycle must eventually self intersect (here at points q and r). In this example, this results in two simple cycles, an inner (dotted) and an outer (dashed) cycle that C can be mapped to. The outer cycle will be formally defined as *outermost cycle* in the proof of Lemma 7.

Conjecture 1. *Given two plane graphs G_1 and G_2, the problem of deciding whether $\vec{\delta}_{wG}(G_1, G_2) \le \varepsilon$ is NP-complete if we do not demand further geometric properties of the embedding of G_1.*

In the following, we consider only graphs that do not contain a thin cycle. We define a geometric property of G_1 that allows to conclude $\vec{\delta}_{wG}(G_1, G_2) \le \varepsilon$ if, after step 3 of Algorithm 1, each vertex has a valid placement.

Definition 5. *Let C be a simple cycle. We say that C is* spike-free, *if C is thick and if $T_\varepsilon(uw) \cap B_\varepsilon(v) = T_\varepsilon(vw) \cap B_\varepsilon(u) = \emptyset$ for all triples of vertices u, v, w on C such that u and v are adjacent to w and u and v are not adjacent. A plane graph G is* spike-free *if all cycles of G are* spike-free*.*

In the example given in Fig. 7, G_1 is not spike-free as the ε-ball around u intersects the ε-tube around the edge vw.

Note that, unlike for trees, even for a spike free cycle we cannot start at an arbitrary placement of an arbitrary vertex to map the whole cycle. An example can be found in Fig. 8. However, consider starting a traversal at an arbitrary placement p of an arbitrary vertex v on a cycle C. We traverse C simultaneously clockwise and counter-clockwise and construct a path P_1 within the ε-tube of C in clockwise direction and a path P_2 within the ε-tube of C in counter-clockwise direction extending the paths by connecting to eligible vertex placements (that must always exists as the placements are valid). Then, P_1 and P_2 must self intersect or merge inside the tube: Assume, that this is not the case. Then, due to the validity of all placements and the planarity of G_2, both paths must wind around C in a snail shell shape and must eventually leave the ε-tube or must end on invalid placements. The intersection(s) induce the existence of a (several) simple cycle(s) D where C can be mapped to. See again Fig. 8 for an illustration. In particular, if all vertices of C have at least one valid placement we can guarantee the existence of an *outermost placement* of C that is defined in the proof of the following Lemma.

Lemma 7. *Let G_2 be a plane graph and let G_1 be a spike-free graph. If every vertex of G_1 has at least one weakly valid ε-placement after running steps 1-3 of Algorithm 1, then G_1 has a weak ε-placement. Thus, $\vec{\delta}_{wG}(G_1, G_2) \le \varepsilon$.*

Proof. The difficulty in finding a consistent mapping arises from the cycles, hence we first map these. That is, we first consider only cycles in the graph, and ignore tree-like parts of the graph, which can later be easily mapped as shown in the proof of Lemma 6.

For this, let $\hat{G}_1$ be the subgraph of G_1 induced by the vertices of all simple cycles of G_1. Then, the *tree-substructures* of G_1 are the components of the closure of $G_1 \backslash \hat{G}_1$. See Fig. 9 for an illustration. Next, we consider all faces of $\hat{G}_1$ and show how to iteratively map them.

Consider a cycle C of $\hat{G}_1$ bounding a face F and let e_1 and e_2 be two edges of C incident to a vertex v. We define an *outermost placement* of v as a placement that maximizes its maximal distance to v, see Fig. 10 (a). Further, we define an *outermost path* in G_2 of an edge $e = \{u, v\}$ of G_1 as the path P with maximum distance to F connecting the outermost placements of u and v. That is, no subpath Q of P can be replaced by a path R such that $\delta_H(R, B) \le \delta_H(Q, B)$, where δ_H is the Hausdorff distance and B is the boundary of the tube $T_\varepsilon(e)$ on the right of $\vec{uv}$, assuming that u is the predecessor of v within a clockwise order of the vertices of F. Note that if an edge e is shorter than 2ε, and hence the ε-balls around the vertices overlap, then so possibly do the placements. In particular, in this case the outer placements may overlap, in which case the edge placements degenerate, see Fig. 10 (a). Finally, we define an *outer placement* O of C in G_2 as the concatenation of all outermost paths of edges of C. Note that possibly more than one vertex placements fulfils the definition of an outermost placement. See for instance Fig. 8 (vertex w) or Fig. 11 (leftmost and rightmost vertex). However,

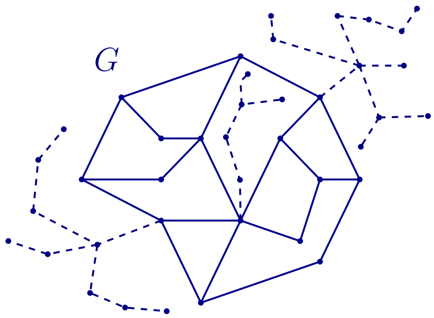
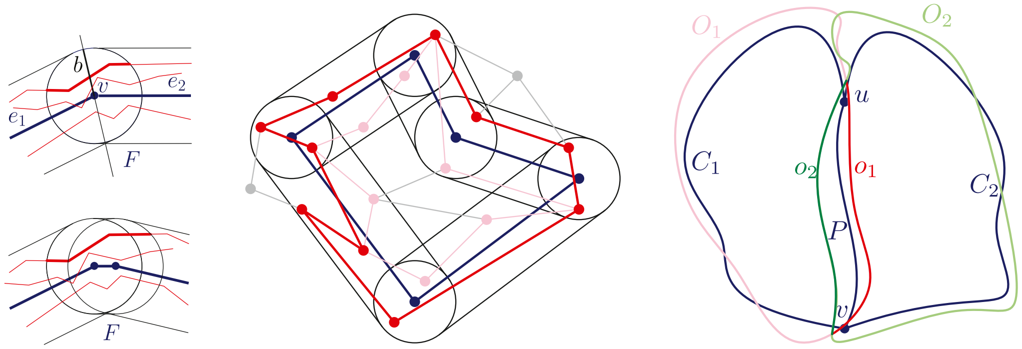


Fig. 9. The parts marked with dashed lines are the tree-substructures of the graph G.



(a) A vertex with its placement. (b) A cycle with its outer placement. (c) Merging two outer placements.

Fig. 10. Illustration of outer placements and how to merge them. In (c) the outer placements of cycles C_1 and C_2 can be merged by mapping the shared path P through o_1.

by planarity there always exists an (outermost) path inside the ε-tube of uv that connects two outermost placements of u and v if G_1 is spike free. Hence, for a cycle C in a spike-free graph there always exists an outermost placement.

Fig. 10 (b) shows an example, where the red outer placement bounds the outer face of G_2 restricted to red and pink vertices and edges. The outer placement of C is a weak ε-placement of C.

Now, consider two cycles C_1 and C_2 bounding adjacent faces of G_1, which share a single (possibly degenerate) path P between vertices u and v. Let O_1 and O_2 be the outer placements of C_1 and C_2, respectively. By definition of an outermost placement, O_1 and O_2 must intersect inside the intersection of the ε-tubes of C_1 and C_2. Let o_1 and o_2 of O_1 and O_2 be the parts between the intersections of O_1 and O_2 containing the respective images of P. Again, by definition of an outermost placement, it holds that o_1 is completely inside O_2 and o_2 is completely inside O_1.

This is illustrated in Fig. 10 (c). By planarity there must be a vertex at the intersections of O_1 and O_2. Thus, we can construct a mapping O'_2 of C_2 that consists of o_1 and $O_2 \setminus o_2$. This is a weak ε-placement of C_2 for which the image of the shared path P is identical to its image in O_1. Thus, we can merge O_1 and O'_2 to obtain a weak ε-placement of these two adjacent cycles. Note that the mapping of C_1 is not modified in this construction. Additionally, the image of the cycle bounding the outer face is its outer placement. The same argument can be applied iteratively when C_1 and C_2 share multiple paths.

If there are two cycles C_1 and C_2 which are connected by a path P such that one endpoint u of P lies on C_1, the other endpoint v of P lies on C_2 and all other vertices of P are no vertices of C_1 or C_2, we can still construct a common placement for C_1, C_2 and P: Let C_u, C_v be the outermost placements of u and v, respectively and let D_v be a valid placement of v which is connected by a path Q in G_2 to C_u such that $\delta_{wF}(Q, P) \leq \varepsilon$. Such a placement D_v must exist as C_u is a valid placement. If $D_v = C_v$ we have found a common valid placement for C_1, C_2 and P. If $D_v \neq C_v$, by definition of an outermost placement, the path Q must intersect the outermost placement O of C_2 inside the intersection of the tubes $T_\varepsilon(P)$ and $T_\varepsilon(C_2)$. As G_2 is plane, there is a vertex w at the intersection and the resulting path $R = Q_{C_u \to w} + O_{w \to C_v}$ with $\delta_{wF}(R, P) \leq \varepsilon$ connects C_u and C_v.

Now, we iteratively map the cycles bounding faces of G_1 until G_1 is completely mapped. Let $\langle F_1, F_2, \ldots, F_k \rangle$ be an ordering of the faces of G_1 such that each F_i, for $i \geq 2$ is on the outer face of the subgraph $\mathbb{G}_{i-1} := C_1 \cup C_2 \cup \ldots \cup C_{i-1}$ of G_1, where C_j is the cycle bounding face F_j. Thus, let F_1 be an arbitrary face of G_1 and subsequently choose faces adjacent to what has already been mapped. Hence when adding a cycle C_i, we have already mapped $\mathbb{G}_{i-1}$ such that the cycle bounding its outer face is mapped to its outer placement. Thus, we can treat $\mathbb{G}_{i-1}$ as a cycle, ignoring the part of it inside this cycle,

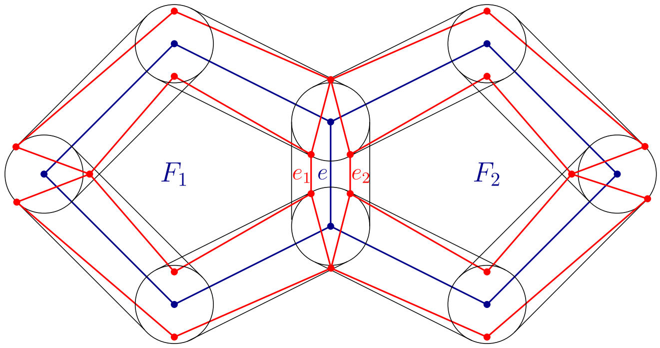


Fig. 11. An example of plane graphs G_1 (blue) and G_2 (red) where every vertex of G_1 has two valid placements, but there is no ε-placement of G_1: If the central edge e is mapped to a path through e_1, there is no way to map the cycle bounding face F_2 on the right, and if e is mapped to a path through e_2, the cycle bounding F_1 cannot be mapped.

and merge its mapping with C_i using the procedure described above. This leaves the mapping of $\mathbb{G}_{i-1}$ unchanged, hence this is still a weak ε-placement of $\mathbb{G}_{i-1}$. However, the mapping of C_i is now modified to be identical to that of $\mathbb{G}_{i-1}$ in the parts where they overlap. Thus, we can merge these mappings to obtain a weak ε-placement of $\mathbb{G}_i$. After mapping F_k we have completely mapped G_1. □

Lemma 3 and Lemma 4 together with Lemma 6 and Lemma 7 directly imply the following theorems. Note, that $m_1 = O(n_1)$ for plane graphs and trees, in particular.

Theorem 3 *(Decision Algorithm for Weak Graph Distance). Let $\varepsilon > 0$. If G_1 is a tree, or if G_1 is a spike-free graph and G_2 is a plane graph, then Algorithm 1 decides whether $\vec{\delta}_{wG}(G_1, G_2) \le \varepsilon$ in $O(n_1 \cdot m_2)$ time and space.*

Theorem 4 *(Decision Algorithm for Graph Distance). Let $\varepsilon > 0$. If G_1 is a tree, then Algorithm 1 decides whether $\vec{\delta}_G(G_1, G_2) \le \varepsilon$ in $O(n_1 \cdot m_2^2)$ time and space.*

Computing the Distance. To compute the graph distance, we proceed as for computing the Fréchet distance between two curves: We search over a set of critical values and employ the decision algorithm in each step. The following types of critical values can occur:

1. A new vertex-placement emerges: An edge in G_2 is at distance ε from a vertex in G_1.
2. Two vertex-placements merge: The vertex in G_2 where they connect is at distance ε from a vertex in G_1.
3. The (weak) Fréchet distance between a path and an edge is ε: these are described in [9]. There are exponentially many paths in G_2, but each value the Fréchet distance may attain is defined by either a vertex and an edge, or two vertices and an edge.

There are $O(n_1 \cdot m_2)$ critical values of the first two types, and $O(m_1 \cdot n_2^2)$ of type three. Parametric search can be used to find the distance as described in [9], using the decision algorithms from Theorems 3 and 4. This leads to a running time of $O(n_1 \cdot m_2 \cdot \log(n_1 + n_2))$ for computing the weak graph distance if G_1 is a tree or both are plane graphs. And the total running time for computing the graph distance if G_1 is a tree is $O(n_1 \cdot m_2^2 \cdot \log(n_1 + n_2))$.

4. Hardness results and algorithms for plane graphs

Lemma 7 does not hold for plane graphs and the directed strong graph distance because in general outer placements of cycles cannot be combined to a placement of G_1 as shown in the proof of Lemma 7. See Fig. 11 for a counterexample. Note that in this example G_1 is spike-free. In fact we show that deciding the directed strong graph distance for plane graphs is NP-hard.

4.1. NP-hardness for the strong distance for plane graphs

Theorem 5. *For plane graphs G_1 and G_2, deciding whether $\vec{\delta}_G(G_1, G_2) \le \varepsilon$ is NP-complete.*

Proof. Analogously to Theorem 2, the problem is in NP. We prove the NP-hardness by a reduction from Monotone-Planar-3-Sat. In this 3-Sat variant, the associated graph with edges between variables and clauses is planar and each clause contains only positive or only negative literals. The overall idea is to construct two graphs G_1 and G_2 based on a Monotone-Planar-3-Sat instance A, such that A is satisfiable if and only if $\vec{\delta}_G(G_1, G_2) \le \varepsilon$. That is, we construct subgraphs of G_1 and G_2, where some edges of G_2 are labeled TRUE or FALSE in a way, such that only certain combinations of TRUE and FALSE values can be realized by a placement of G_1 to G_2. To realize other combinations, backtracking, at least along one edge of G_1, is necessary. But this is not allowed for the strong graph distance. In the following, we describe the construction of the gadgets (subgraphs) for the variables and the clauses of a MONOTONE-PLANAR-3-SAT instance. Additionally, we need a gadget to split a variable if it is contained in several clauses and a gadget which connects the variable gadgets with the clause gadgets. Furthermore, we prove for each gadget which TRUE and FALSE combinations can be realized and which combinations are not possible. All constructed edges are straight line edges. The graph G_1 is shown in blue color and G_2 is shown in red color in the sketches used to illustrate the ideas of the proof. We denote $T_\varepsilon(e)$ as the ε-tube around the edge e. All vertices of the graph can be either mapped arbitrarily within a given ε-surrounding and with a given minimal distance from each other or must lie at the intersection of two lines. Thus, we can ensure that the construction uses rational coordinates only and can be computed in polynomial time.

When describing a reparameterization of an edge or a path (according to Definition 1) we say that we "walk along" an edge or a path. If backtracking is necessary to stay within ε-distance to another path or edge, then the reparameterization is not injective and thus one can conclude that the Fréchet distance between these paths or edges is greater than ε.

Furthermore, we call a path labeled TRUE (FALSE) shortly a TRUE (FALSE) signal.

In the following, we give a detailed description of the construction and the properties of the gadgets. The VARIABLE gadget: For each variable of A, we add a vertex v and two edges, e_1 and e_2, of G_1 incident to a vertex v, where the angle between e_1 and e_2 is between 90° and 120°. We add a vertex w_1 (w_2) of G_2 on the intersection of the outer boundaries of $T_\varepsilon(e_2)$ ($T_\varepsilon(e_1)$) and a line through e_1 (e_2). Furthermore, we add a vertex w_3 of G_2 at the intersection of the boundaries of $T_\varepsilon(e_1)$ and $T_\varepsilon(e_2)$. See the upper left sketch in Fig. 12 for an illustration. We connect w_1 and w_2 with w_3 and draw an edge from w_1 and w_2 inside the ε-tubes around e_1 and e_2, with labels TRUE. Analogously, we embed two edges from w_3 with the label FALSE. For the VARIABLE gadget a TRUE-TRUE combination is not possible: There are two placements p_1 and p_2 of the vertex v. Assume we choose p_1. Note that one can map e_1 to a path containing the edge of G_2 with the TRUE labeling inside $T_\varepsilon(e_1)$. Now, we want to map e_2 to a path P starting at some point of p_1, where P contains the edge of G_2 with the TRUE labeling inside $T_\varepsilon(e_2)$. In this case, one has to walk along e_2 up to q (the point on e_2 with distance ε to w_3) while walking simultaneously to w_3 on P. But then, when walking along P up to w_1, one must walk back along e_2 up to v as any point along the interior of e_1 has distance greater than ε to w_1. Thus, $\delta_F(e_2, P) > \varepsilon$. It is easy to verify that for any other combination of labels there is a placement p of v, such that e_1 and e_2 can be mapped to a path P_1 (P_2) starting at a point a of p with $\delta_F(e_2, P_1) \leq \varepsilon$ ($\delta_F(e_2, P_2) \leq \varepsilon$), because in these cases there is no zig-zag that forces backtracking."

A PERMUTE gadget is the same as the VARIABLE gadget, but with a different labeling, see Fig. 12. This ensures that a FALSE signal can never be converted to a TRUE signal in the PERMUTE gadget. However, a TRUE signal can, but need not to be converted to a FALSE signal in the PERMUTE gadget. Again, in both cases (converting TRUE to FALSE or not), there exists a mapping of the edges of G_1 that does not contain a zig-zag and thus the Fréchet distance between the edges and the mapping is at most ε. We will make use of this property to construct an eligible mapping based on an assignment of the variables that satisfies the given formula.

A SPLIT gadget is needed to connect a VARIABLE gadget to multiple CLAUSE gadgets that contain the variable and constructed similarly to the VARIABLE gadget. In comparison with to VARIABLE gadget we add a third edge e_3 of G_1 and edges of G_2 from w_2 and w_3 inside the ε-tube around e_3 to allow for splitting a path emerging from a VARIABLE gadget. Furthermore, we need to ensure that a FALSE signal can never be converted to a TRUE signal in the SPLIT gadget. Again, this is achieved by constructing zig-zags and an appropriate labeling such that backtracking is forced if the in-going edge is mapped to a path labeled FALSE and an out-going edge is mapped to path labeled TRUE. For an eligible labeling, see Fig. 12.

We construct the WIRE gadget used to connect all the other gadgets by drawing two edges e_1 and e_2 of G_1 incident to a vertex v (with arbitrary angle) and two vertices w_1 and w_2 of G_2 inside $B_\varepsilon(v)$ with non intersecting incident edges inside $T_\varepsilon(e_1)$ and $T_\varepsilon(e_2)$. Obviously, it is not possible to convert a TRUE signal to a FALSE signal or vice versa, here.

For the CLAUSE gadget, we first introduce a NAE-CLAUSE gadget. Here it is required that the three values in each clause are not all equal to each other. We start the construction of the NAE-CLAUSE gadget by drawing three edges e_1, e_2 and e_3 incident to a vertex v with a pairwise 120° angle. We draw three vertices w_1, w_2 and w_3 on the intersections of $T_\varepsilon(e_1)$, $T_\varepsilon(e_2)$ and $T_\varepsilon(e_3)$ with a maximum distance to v. Furthermore, we draw two edges of G_2 inside the tubes for each vertex and label them as shown in Fig. 12. Let q_1 be the point on e_1 with distance ε to w_1 and w_2. Here, it is not enough to simply connect w_1 and w_2 with one edge as shown in the bottom left sketch of Fig. 12. To force backtracking along e_1 for a combination of labels which we want to exclude, we have to ensure that a path from w_1 to w_2 leaves $B_\varepsilon(q_1)$ but stays, once entered, inside $B_\varepsilon(v)$. For the other pairs, (w_1, w_3) and (w_2, w_3) we do the same. A possible drawing of these paths maintaining the planarity of G_2 is shown in Fig. 12. The three placements of v are connected by the vertices w_1, w_2 and w_3 and it holds that there is no placement of v such that an all-FALSE or an all-TRUE labeling can be realized: Suppose we map v to $s(v)$ as shown in the Fig. 12. Then, edges e_2 and e_3 can be mapped to paths through edges labeled TRUE. But we cannot map e_1 to such a path P: When P reaches vertex w_1, any corresponding reparameterization of e_1 realizing $\delta_F(e_1, P) \leq \varepsilon$ must have reached q_1 as q_1 is the only point with distance at most ε to w_1 on e_2. As P leaves $B_\varepsilon(q_1)$ between w_1 and w_2 and any point on e_1 with distance at most ε to the part of P outside $B_\varepsilon(q_1)$ lies between v and q_1 it follows that $\delta_F(e_1, P) > \varepsilon$. For symmetric reasons it follows that any other all-equal labeling cannot be realized. However, there is a placement of v, such that all three edges e_1, e_2 and e_3 can be mapped to a path in G_2 with Fréchet distance at most ε, for each configuration where not all three signals have the same value.

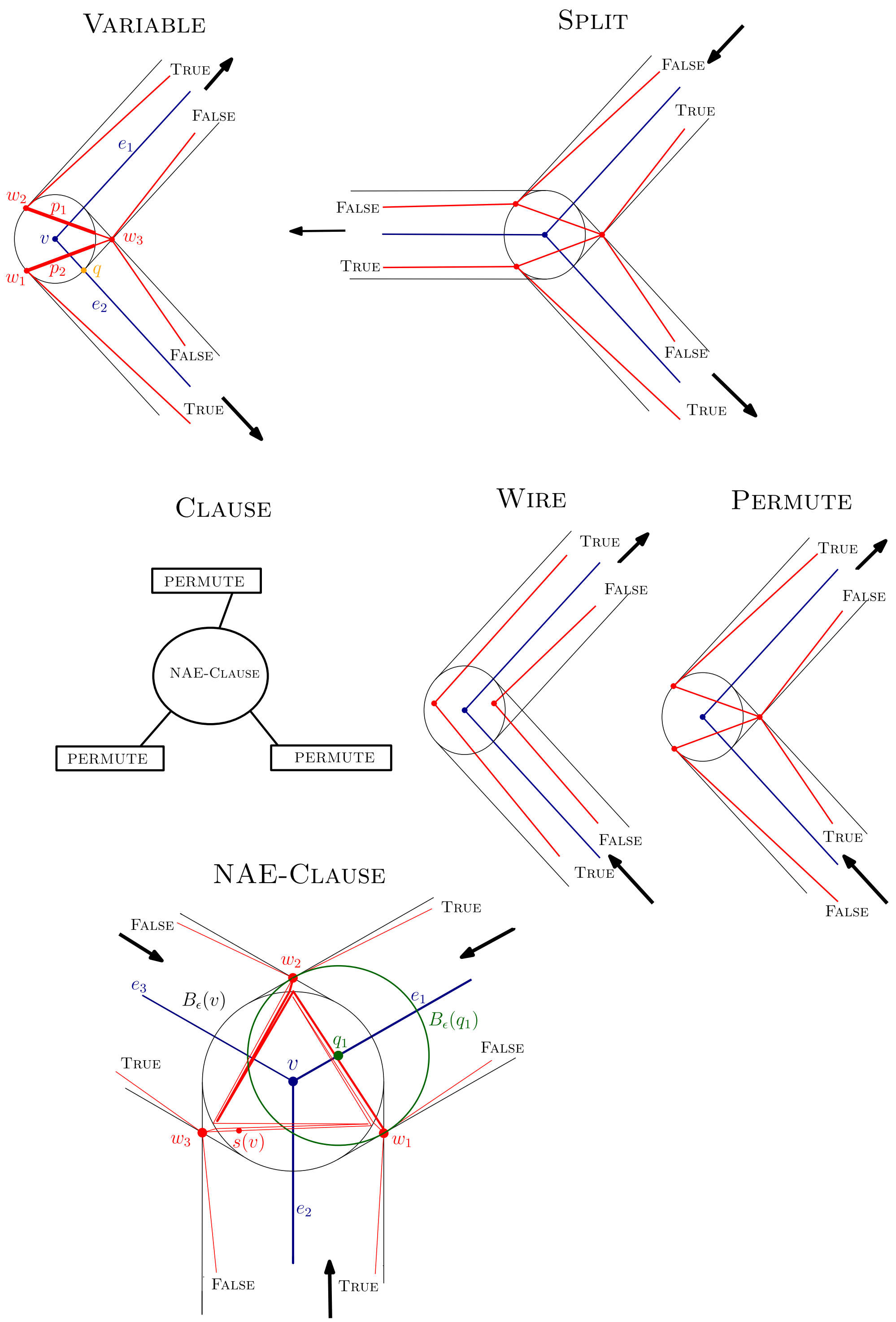


Fig. 12. Building blocks to build graph-similarity instance given a MONOTONE-PLANAR-3-SAT instance.

MONOTONE-PLANAR-NAE-3-SAT is in P, but we can use the NAE-CLAUSE gadget as core of our CLAUSE gadget referring to the NP-complete version MONOTONE-PLANAR-3-SAT: We obtain the CLAUSE gadget by connecting each NAE-CLAUSE gadget with three PERMUTE gadgets, as shown in Fig. 12.

Fig. 13 partially shows the constructed graphs for a given MONOTONE-PLANAR-3-SAT A consisting of the subgraphs (gadgets) described above.

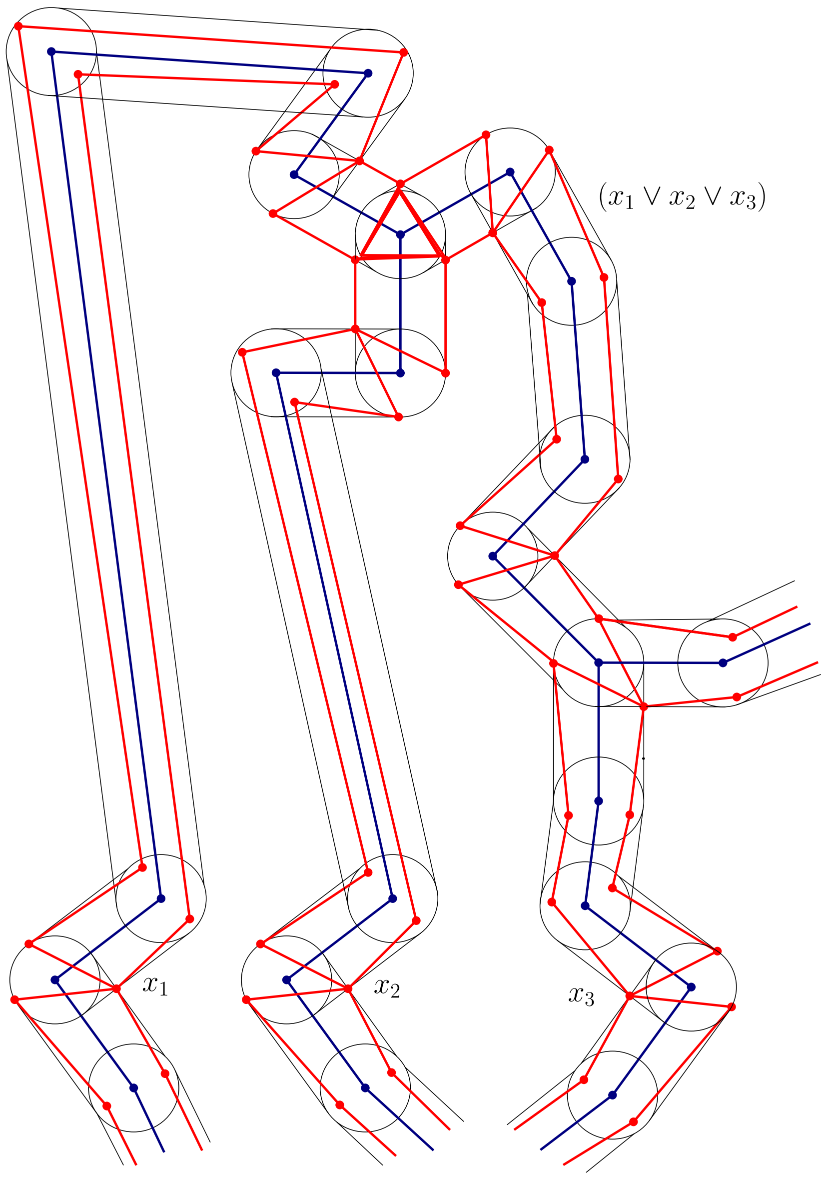


Fig. 13. For the Monotone-Planar-3-Sat instance A with variables $V = \{x_1, x_2, \ldots, x_5\}$ and clauses $C = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (\bar{x_1} \vee \bar{x_3} \vee \bar{x_5})\}$ the Figure shows the construction of the clause $(x_1 \vee x_2 \vee x_3)$.

Now, given a Monotone-Planar-3-SAT instance A, one can construct the graphs G_1 and G_2 with the gadgets described above. Note that all gadgets are plane subgraphs. By placing them next to each other with no overlap, we can ensure that G_1 and G_2 are plane graphs.

A valid placement of the whole graph G_1 induces a solution of A: In the corresponding gadget for each positive NAE-clauses, at least one of the outgoing edges of G_1 must be mapped to a path through an edge labeled True. By construction, this label cannot be converted to False in any of the gadgets and therefore the corresponding variable v gets the value True. In this case, v cannot set any of the negative clauses True because the other outgoing edge must be mapped to a path through the edge of G_2 labeled False and this signal can never be switched to True. The same holds for the case of negative NAE-clauses.

Conversely, given a solution S of the Monotone-Planar-3-SAT instance A, it is easy to construct a placement of G_1. In the variable gadget of variable x_1, we choose placement p_1 and map e_1 to a path through the edge of G_2, labeled True and map e_2 to a path through the edge labeled False if x_1 is positive in S. If x_1 is negative, we chose placement p_2 and map e_1 and e_2 accordingly. All edges of the other gadget now can be mapped to G_2 in a signal preserving manner (True stays True, False stays False). If there exists a clause C in A, such that all three variables of C are positive (negative) in S, we change one signal in the Permute gadget from True to False. Thus, we have found a placement for the whole graph G_1. □

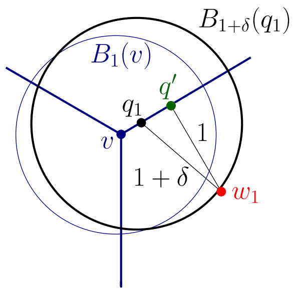


Fig. 14. Illustration of the proof of Theorem 6.

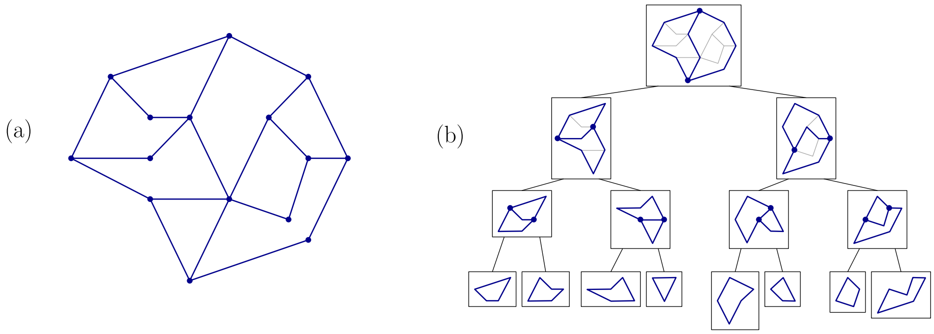


Fig. 15. A plane graph (a) is recursively decomposed into chordless cycles by splitting each cycle with a chord (b).

The following stronger result follows from the observation that the characteristics of the subgraphs we constructed in the proof of Theorem 5 still hold for a slightly larger ε value.

Theorem 6. *Approximating $\vec{\delta}_G(G_1, G_2)$ within a 1.10566 factor is an NP-complete problem.*

Proof. We give a detailed proof for the NAE-Clause gadget and note that a similar argument holds for the other gadgets. See Fig. 14 for an illustration of the arguments and the calculations below.

Let us fix $\varepsilon = 1$. As described in the proof of Theorem 5, we connect the vertices w_1 and w_2 by a path which leaves $B_1(q_1)$, but stays inside $B_1(v_1)$. (Introducing a spike which leaves $B_1(q_1)$ and returns to $B_1(q_1)$, see Fig. 12). We draw the spike such that its peak is arbitrarily close to the intersection of a straight line through the edge e_1 and the 1-circle around v. When enlarging ε, the point q_1 moves along e_1 toward v. We need to compute the smallest value δ_{min}, such that $B_1(v)$ is completely contained in $B_{1+\delta_{min}}(q_1)$. For any value $\delta < \delta_{min}$, there exists a drawing of the spikes, such that the characteristics of the NAE-Clause gadget still hold, e.g., there is no placement of v allowing an all-equal-labeling.

Note that δ_{min} equals the distance from q_1 to v, when q_1 is at distance $1 + \delta_{min}$ to w_1. Let q' be the position of q_1 for $\delta = 0$ and let d be the distance between q' and q_1. Then we have $\tan(30°) = \frac{\delta_{min}+d}{1} = \delta_{min} + d$. Furthermore, we have $d = \sqrt{(1+\delta_{min})^2 - 1}$ and therefore $\delta_{min} = \tan(30°) - \sqrt{(1+\delta_{min})^2 - 1}$, which solves to $\delta_{min} = \frac{1}{4} - \frac{1}{4\sqrt{3}} \approx 0.10566$. The factor by which ε can be multiplied is greater than $1 + \delta_{min}$ for all other gadgets. Thus, δ_{min} is the critical value for the whole construction and the theorem follows. □

4.2. Deciding the strong graph distance in exponential time

In the following, we assume that we have already run the first three steps of Algorithm 1. In particular, we already computed all valid ε-placements of the vertices of G_1. Recall, that this can be done in $\mathcal{O}(n_1 \cdot m_2^2)$ time.

A brute-force method to decide the directed strong graph distance is to iterate over all possible combinations of valid vertex placements, which takes $O(m_1 \cdot m_2^{n_1})$ time. Another approach is to decompose G_1 into faces and merge the substructures bottom-up.

First, we remove all tree-like substructures of G_1 and map these as described in the proof of Lemma 6. Next, we decompose the remainder of G_1 into chordless cycles, where a chord is a maximal path in G_1 incident to two faces, see Fig. 15. We map the parts of G_1 from bottom up, deciding in each step if we can map two adjacent cycles and all the nested substructures of the cycles simultaneously. To do so, we start with storing all combinations of placements of endpoints of a chord -which separates two faces (chordless cycles)- allowing us to map the two faces simultaneously. We prune all placements which are not part of any valid combination. In the following steps, for each placement C_u of an endpoint u of a chord and each valid combination c of nested chords computed in the previous step, we run one graph exploration. For each placement C_v of the other endpoint v of the chord, which allows to map both cycles simultaneously, we store a new combination consisting of C_u, c and C_v. We prune all placements of u where we cannot reach a valid placement of v by using any of the previous computed combinations. Furthermore, we prune those placements of v which are never reached by any graph exploration. If the list of placements gets empty for one vertex, we can conclude, that the graph distance is greater than ε. Conversely, if we find a valid combination of placements of the endpoints of the chord in the last step, we can conclude that we can map the whole graph G_1 as we guarantee in each step that all substructures can be mapped, too.

Theorem 7. *For plane graphs, the strong graph distance can be decided in $O(Fm_2^{2F-1})$ time and $O(m_2^{2F-1})$ space, where F is the number of faces of G_1.*

Proof. Each graph exploration takes $O(m_2)$ time and in each node we have to run $O(m_2k)$ explorations, where k is the number of valid combinations of endpoint placements from previously investigated chords. As the tree has a depth of $\log(F)$, we have

$$\frac{F}{2}m_2^2 + \frac{F}{4}m_2^2m_2^4 + \cdots + m_2^2\left(m_2^{2^{\log(F)-2}}\right)^2 = F\sum_{i=1}^{\log(F)}\frac{1}{2^i}m_2^{2^{i+1}-2}$$

as the total running time of the graph explorations. We can upper bound this term as follows:

$$F\sum_{i=1}^{\log(F)}\frac{1}{2^i}m_2^{2^{i+1}-2} \le F\sum_{i=1}^{\log(F)}m_2^{2^{i+1}-2} \le F\sum_{i=1}^{2^{\log(F)+1}-2}m_2^i$$
$$= F\sum_{i=1}^{2F-2}m_2^i = F\frac{m_2^{2F-1}-m_2}{m_2-1} \in O\left(Fm_2^{2F-1}\right),$$

where the second equality uses

$$\sum_{i=1}^{n}a^i = \frac{a(a^n-1)}{a-1},$$

for $a \in \mathbb{R}$. In the first step, we have to store $O(m_2^2)$ combinations for each two faces we want to map simultaneously. Let k be the number of combinations in the previous step. Then we have to store up to $k^2m_2^2$ combinations in the next step. This results in storing $O(m_2^{2F-1})$ combinations in the root node of the decomposition. □

Thus, this method is superior to the brute-force method if $2F-1 \le n_1$.

4.3. Approximation for plane graphs

For plane graphs, Algorithm 1 yields an approximation depending on the angle between the edges for deciding the strong graph distance. The decision is based on the existence of valid placements. Therefore, the runtime is the same as stated in Theorem 4.

Theorem 8. *Let $G_1 := (V_1, E_1)$ and $G_2 := (V_2, E_2)$ be plane graphs. Assume that for all adjacent vertices $v_1, v_2 \in V_1$, $B_\varepsilon(v_1)$ and $B_\varepsilon(v_2)$ are disjoint. Let α_v be the smallest angle between two edges of G_1 incident to vertex v with $deg(v) \ge 3$, and let $\alpha := \frac{1}{2}\min_{v\in V_1}(\alpha_v)$. If there exists at least one valid ε-placement for each vertex of G_1, then $\vec{\delta}_G(G_1, G_2) \le \frac{1}{\sin(\alpha)}\varepsilon$.*

Proof. Let α be the smallest angle between two edges incident to a vertex v with degree at least three and let $C_1, C_2, \ldots, C_j$ be the valid placements of v for a given distance value ε. Furthermore, let V_{C_i} be the set of vertices of C_i. It can be easily shown that for a larger distance value of $\hat{\varepsilon} \ge \frac{1}{\sin(\alpha)}\varepsilon$, there exist vertices $v_1, v_2, \ldots, v_k$, embedded inside $B_{\hat{\varepsilon}}$, such that the subgraph $C = (V', E')$, where $V' = \bigcup_{i=1}^{j} V_{C_i} \cup \{v_1\} \cup \{v_2\} \cup \cdots \cup \{v_k\}$ and $E' = \{\{uw\} \in E_2 | u \in V', w \in V'\}$ is connected, see Fig. 16 (a). Note that this property is not true if $B_\varepsilon(v_1) \cap B_\varepsilon(v_2) \ne \emptyset$ for two adjacent vertices $v_1, v_2 \in V_1$: Fig. 16 (b) shows an example with $\alpha = 45°$ where the two placements do not merge inside $B_{\frac{\varepsilon}{\sin(45°)}}(v)$. However, with the condition $B_\varepsilon(v_1) \cap B_\varepsilon(v_2) = \emptyset$, there is only one valid $\frac{1}{\sin(\alpha)}\varepsilon$-placement C for each vertex with degree at least three. Furthermore, every valid ε placement is a valid $\frac{1}{\sin(\alpha)}\varepsilon$-placement. Now, a path P of G_1 starting at a vertex v with $deg(v) \ge 3$ and ending at a vertex w with $deg(w) \ne 2$, with vertices of degree two in the interior of P, can be mapped as described in the proof of Lemma 6. For two paths which start and/or end at a common vertex v, v is mapped to the same placement as there is only one valid $\frac{1}{\sin(\alpha)}\varepsilon$-placement of v. This ensures that each edge of G_1 is mapped correctly. □

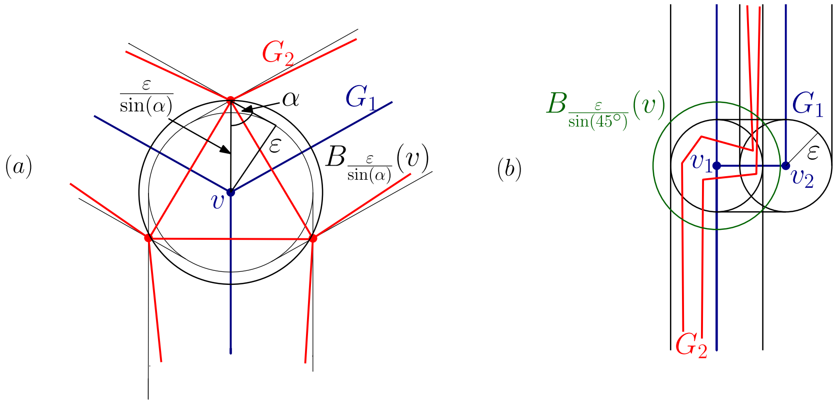
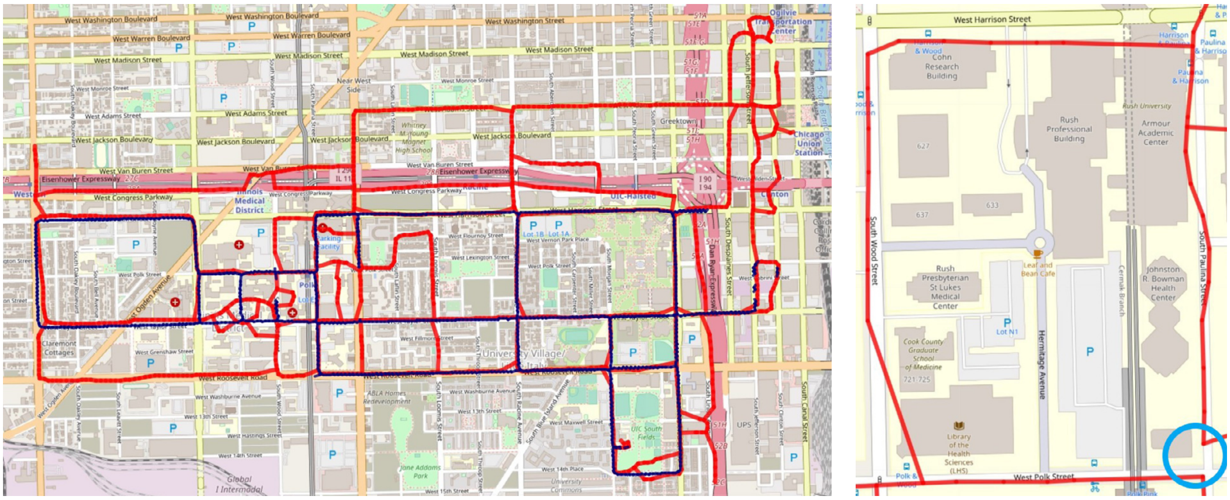


Fig. 16. Illustration of the proof of Theorem 8. (a) Three valid ε-placements form one connected component inside a slightly larger ball around v. (b) In case of short edges, the placements remain unconnected also inside a larger surrounding of v.



(a) Two partial map reconstructions of Chicago.

(b) Different topology.

Fig. 17. Two reconstructed road map graphs R (red) and B (blue), overlayed on the underlying ground truth road map G from OpenStreetMap.

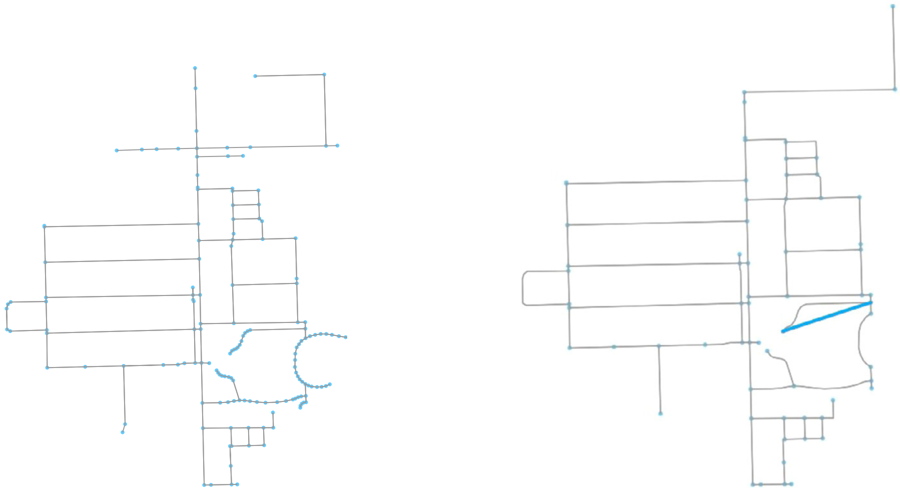
5. Experiments on road networks

In the last decade several algorithms have been developed for reconstructing maps from the trajectories of entities moving on the network [4,5]. This naturally asks to assess the quality of such reconstruction algorithms. Recently, Duran et al. [17] compared several of these algorithms on hiking data, and found that inconsistencies often arise due to noise and low sampling of the input data, for example unmerged parallel roads or the addition of short off-roads.

When assessing the quality of a network reconstruction from trajectory data, several aspects have to be taken into account. Two important aspects are the *geometric* and *topological error* of the reconstruction. Another important aspect is the *coverage*, i.e., how much of the network is reconstructed from the data. We believe our measures to be well suited for assessing the geometric error while still maintaining connectivity information.

We have used the weak graph distance for measuring the distance between different reconstructions and a ground truth, as well as a simplification of part of the road network of Chicago. Fig. 17 (a) shows two reconstructed road map graphs R (red) and B (blue), overlayed on the underlying ground truth road map G from OpenStreetMap. The reconstruction R in red resulted from Ahmed et al.'s algorithm [6], whereas the reconstruction B in blue from Davies et al.'s [16] algorithm. Our directed graph distance from B to G is 25 meters, and from R to G it is 90 meters. This reflects the local geometric error of the reconstructions (note that it does not evaluate the difference in coverage). Fig. 17 (b) shows an example where the topology of R and G differs (blue circle), affecting for instance navigation significantly. Our measure captures this difference. Although the reconstruction approximates the geometry well, our measure computes a directed distance of 200 m from G (restricted to the part covered by R) to R.

Fig. 18 shows a partial road network of Chicago at different resolutions. Both maps show vertices as blue dots. The actual map is shown in grey in both figures. However the considered data stores only the positions of the vertices and the



(a) Partial road network of Chicago. (b) Simplified partial road network of Chicago.

Fig. 18. The graph G in (a) is a higher resolution map, while the graph H in (b) is a lower resolution map that represents the road segment geometries with fewer vertices.

edges between two vertices as straight lines. The map on the left with the higher resolution approximates curves by a high number of short straight-line edges, whereas the data corresponding to the right figure consist of only one straight-line edge for each street, as illustrated by the added blue edge in the figure. The geometric error of this simplification can be measured by computing the graph distance between the two graphs representing the road network at different scales of resolution. Our approach yields a distance of 22 meters between the graphs. It is the maximum Fréchet distance between a street that is approximated by (many) small straight line edges and the corresponding simplified street that is represented by only one straight line edge. The data is extracted from *OpenSteetMap* using the *OSMnx* Python library.

6. Conclusion

We developed new distances for comparing straight-line embedded graphs and presented efficient algorithms for computing these distances for several variants of the problem, as well as proving NP-completeness for other variants. Our distance measures are natural generalizations of the Fréchet distance and the weak Fréchet distance to graphs, without requiring the graphs to be homeomorphic. Although graphs are more complicated objects than curves, the runtimes of our algorithms are comparable to those for computing the Fréchet distance between polygonal curves. The described algorithmic approach outputs the existence of a valid mapping of one graph onto the other graph and we described how to obtain a specific valid mapping for the types of graphs where the computation problem is feasible. However, a valid mapping is not unique in general and might not express the local similarity accurately as the distances are bottleneck distances. To address this issue one can for instance formulate further optimization criteria for a graph mapping. Two approaches, one that minimizes the sum of the distances between edges and paths and a lexicographic approach are described in [14]. If deciding the (directed) weak distance for plane graph that are not spike-free is an NP-complete problem or if it can be answered in polynomial time remains as an open question. On the contrary, it would be interesting if one could state further geometric restrictions such that the (directed) strong graph distance can be decided and computed in polynomial for graphs that fulfill these restrictions. Furthermore, a large-scale comparison of our approach with existing graph similarity measures is left for future work.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

We would like to thank the anonymous reviewers for the helpful comments, in particular correcting our proof of Lemma 7.

References

[1] Pankaj K. Agarwal, Kyle Fox, Abhinandan Nath, Anastasios Sidiropoulos, Yusu Wang, Computing the Gromov-Hausdorff distance for metric trees, ACM Trans. Algorithms 14 (2) (2018) 24.
[2] Mahmuda Ahmed, Brittany T. Fasy, Kyle S. Hickmann, Carola Wenk, Path-based distance for street map comparison, in: ACM Transactions on Spatial Algorithms and Systems, 2015, 28 pages.
[3] Mahmuda Ahmed, Brittany Terese Fasy, Carola Wenk, Local persistent homology based distance between maps, in: 22nd ACM SIGSPATIAL GIS, 2014, pp. 43–52.
[4] Mahmuda Ahmed, Sophia Karagiorgou, Dieter Pfoser, Carola Wenk, A comparison and evaluation of map construction algorithms using vehicle tracking data, GeoInformatica 19 (3) (2015) 601–632.
[5] Mahmuda Ahmed, Sophia Karagiorgou, Dieter Pfoser, Carola Wenk, Map Construction Algorithms, Springer, 2015.
[6] Ahmed Mahmuda, Carola Wenk, Constructing street networks from gps trajectories, in: Proceedings of the 20th Annual European Conference on Algorithms, ESA'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 60–71.
[7] Hugo A. Akitaya, Maike Buchin, Bernhard Kilgus, Stef Sijben, Carola Wenk, Distance measures for embedded graphs, in: Lu Pinyan, Guochuan Zhang (Eds.), 30th International Symposium on Algorithms and Computation (ISAAC 2019), in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 149, 2019, 55.
[8] Helmut Alt, Alon Efrat, Günter Rote, Carola Wenk, Matching planar maps, J. Algorithms 49 (2) (2003) 262–283.
[9] Helmut Alt, Michael Godau, Computing the Fréchet distance between two polygonal curves, Int. J. Comput. Geom. Appl. 5 (1&2) (1995) 75–91.
[10] Ayser Armiti, Michael Gertz, Geometric graph matching and similarity: a probabilistic approach, in: ACM International Conference Proceeding Series, 2014.
[11] James Biagioni, Jakob Eriksson, Inferring road maps from global positioning system traces: survey and comparative evaluation, Transp. Res. Rec. 2291 (2012) 61–71.
[12] Karl Bringmann, Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails, in: IEEE 55th Annual Symposium on Foundations of Computer Science, 2014, pp. 661–670.
[13] Kevin Buchin, Maike Buchin, Christian Knauer, Günter Rote, Carola Wenk, How difficult is it to walk the dog?, in: Proc. 23rd European Workshop on Computational Geometry (EWCG), 2007, pp. 170–173.
[14] Maike Buchin, Bernhard Kilgus, Distance measures for embedded graphs - optimal graph mappings, in: Proceedings of the 36th European Workshop on Computational Geometry (EuroCG), 2020, pp. 218–225.
[15] Otfried Cheong, Joachim Gudmundsson, Hyo-Sil Kim, Daria Schymura, Fabian Stehn, Measuring the similarity of geometric graphs, in: International Symposium on Experimental Algorithms, 2009, pp. 101–112.
[16] Jonathan J. Davies, Alastair R. Beresford, Andy Hopper, Scalable, distributed, real-time map generation, IEEE Pervasive Comput. 5 (4) (2006) 47–54.
[17] David Duran, Vera Sacristán, Rodrigo I. Silveira, Map construction algorithms: an evaluation through hiking data, in: Proceedings of the 5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, MobiGIS '16, 2016, pp. 74–83.
[18] David Eppstein, Subgraph isomorphism in planar graphs and related problems, in: Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '95, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1995, pp. 632–640.
[19] Sophia Karagiorgou, Dieter Pfoser, On vehicle tracking data-based road network generation, in: 20th ACM SIGSPATIAL GIS, 2012, pp. 89–98.