Distance Measures for Embedded Graphs - Optimal Graph Mappings

Maike Buchin
Department of Mathematics, Ruhr University Bochum, Bochum, Germany

Bernhard Kilgus
Department of Mathematics, Ruhr University Bochum, Bochum, Germany

Abstract

We want to compare embedded graphs at a global and a local scale. The graph distances presented in [5] compare two graphs by mapping one graph onto the other and taking the maximum Fréchet distance between edges and their mappings. For this, the graph mapping is chosen such that the bottleneck distance is minimized. Here, we present two approaches to compute graph mappings subject to additional optimization criteria in order to improve distances locally.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases graph distance; optimal mappings; Fréchet distance

Digital Object Identifier 10.4230/LIPIcs...

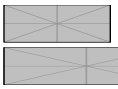
1 Introduction

Motivation We are interested in comparing two embedded graphs. There are many applications that work with graphs embedded in an Euclidean space, such as road networks. For instance, by comparing two road networks one can assess the quality of map construction algorithms [3, 4]. Recently, graph distances based on graph mappings [5] were presented suitable for this task. However, these distances are bottleneck distances and a mapping realizing the bottleneck graph distance might be far from optimal for a single edge or any subgraph. See Figure 1 for an example. In this paper, we want to improve these mappings such that they express local distances more accurately.

Related Work Several approaches have been proposed for comparing embedded graphs. These include an edit distance [8], algorithms that compare all paths [1] or random samples of shortest paths [9], traversal distance [6], and local persistent homology distance [2]. However,as argued in [5], (most of) these capture only the geometry or only the topology of the graphs. The distances presented in [5] capture both and are based on an explicit mapping between the graphs. Here, we extend these measures by looking for locally good matchings. To obtain locally good matchings between two polygonal curves, Buchin et al. introduced locally correct matchings [7] and Rote suggested lexicographic Fréchet matchings [10].

The traversal distance is a bottleneck distance but given the optimal traversals, local optimality is obtained by computing lexicographic Fréchet matchings. This approach can also be applied when computing the path-based graph distances. However, such a local traversal or path-based distance still suffers the shortcoming of the original measures, namely that it reduces the graphs to one or several paths. The local persistent homology distance allows for computing and visualization of local distances whereas expressing local distances with the edit distance is limited as some edges and vertices might be deleted during transformation.

Definition and Previous Results Here, we summarize the definition of the graph distances, the general algorithmic approach to compute the distances and the computational complexity for several settings (general graphs, planar embedded graphs, trees) as described in detail in



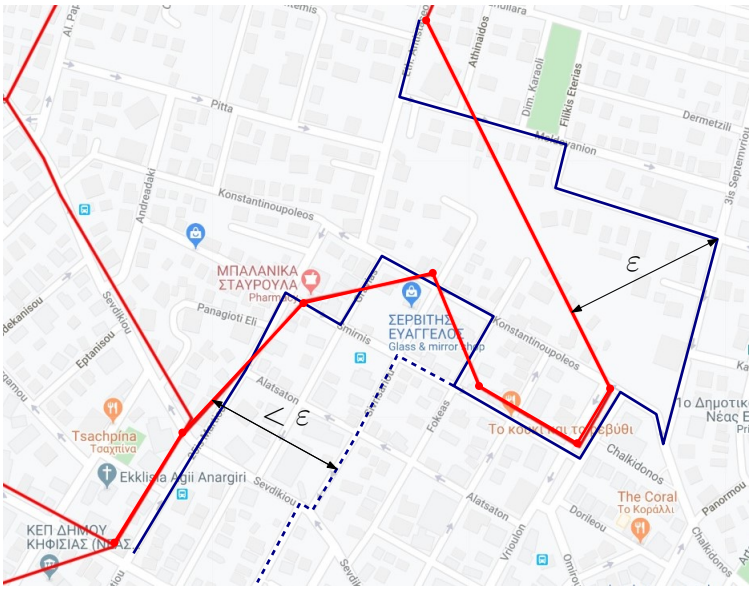


Figure 1 A partial map reconstruction R of the street map of Athens (in red). The graph distance between R and the ground truth is ε. The blue dashed and solid mappings are both valid, although the latter captures the local distance between the reconstruction and the ground truth better.

[5]. 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 in the plane) that are connected by straight-line edges. We consider a mapping $s\colon G_1 \to G_2$ that maps each vertex $v \in V_1$ to a point $s(v)$ on G_2 (not necessarily a vertex) and that maps each edge $\{u, v\} \in E_1$ to a simple path in G_2 with endpoints $s(u)$ and $s(v)$. The directed graph distances $\vec{\delta}_{(w)G}$ are defined as

$$\vec{\delta}_{(w)G}(G_1, G_2) = \inf_{s\colon G_1 \to G_2} \max_{e \in E_1} \delta_{(w)F}(e, s(e))^1,$$

where $\delta_{(w)F}$ denotes the (weak) Fréchet distance, s ranges over all graph mappings from G_1 to G_2, and e and its image $s(e)$ are interpreted as curves in the plane. These distances are not symmetric and the difference between $\vec{\delta}_{(w)G}(G_1, G_2)$ and $\vec{\delta}_{(w)G}(G_2, G_1)$ can be arbitrarily large. The undirected graph distances $\delta_{(w)G}(G_1, G_2)$ are defined as the maximum of $\vec{\delta}_{(w)G}(G_1, G_2)$ and $\vec{\delta}_{(w)G}(G_2, G_1)$. Note that in this extended abstract, we only consider optimizing the directed distances. The undirected distances can be defined analogously as the maximum of the two directions.

An *ε-placement* of a vertex v is a maximally connected component of G_2 restricted to the ε-ball $B_\varepsilon(v)$ around v. A (weak) *ε-placement* of an edge $e = \{u, v\} \in E_1$ is a path P in G_2 with endpoints on ε-placements C_u of u and C_v of v such that $\delta_{(w)F}(e, P) \leq \varepsilon$. In that case, we say that C_u and C_v are *reachable* from each other. An ε-placement C_v of 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 *reachable* from each other.

The general algorithmic approach to solve the decision problem of the directed (weak) graph distances for a given value $\varepsilon > 0$ consists of the following steps [5] **(1)** Compute all ε-placements of vertices and **(2)** of edges. Subsequently, **(3)** prune all invalid placements

[1] We use the (w)-notation to simultaneously address the weak Fréchet distance and the Fréchet distance and the weak graph distance and the graph distance, respectively.

and **(4)** decide whether $\vec{\delta}_{(w)G}(G_1, G_2) \leq \varepsilon$ based on the remaining valid ε-placements. We call a graph mapping s that realizes $\vec{\delta}_{(w)G}(G_1, G_2) \leq \varepsilon$ a (weakly) valid mapping.

Deciding the directed (weak) graph distance is NP-hard for general graphs, but we can compute the (weakly) valid ε-placements in polynomial time. If there is a vertex with no (weakly) valid ε-placement, it follows that $\vec{\delta}_{(w)G}(G_1, G_2) > \varepsilon$. Conversely, the existence of a (weakly) valid ε-placement for each vertex ensures $\vec{\delta}_{(w)G}(G_1, G_2) \leq \varepsilon$ for several cases, namely if G_1 is a tree (both graph distances) and if G_1 and G_2 are plane graphs (weak graph distances). Thus, the distances are decidable in polynomial time in these cases. Deciding whether $\vec{\delta}_G(G_1, G_2) \leq \varepsilon$ remains NP-hard, if G_1 and G_2 are plane graphs [5].

Contribution The graph distances are bottleneck distances and a valid mapping might be far from optimal for a specific edge. To improve the local distance, we introduce additional optimization criteria for the graph mappings. First observe that in contrast to the (weak) Fréchet distance for polygonal paths [7, 10], we cannot expect to find a valid mapping $s_1 \colon G_1 \to G_2$ that is locally minimal in the sense that for any other valid mapping s_2, $\delta_{(w)F}(e, s_1(e)) \leq \delta_{(w)F}(e, s_2(e))$ for each edge e of G_1. See Figure 2 for an example.

We formulate the following optimization criteria: One natural goal is to minimize the (weighted) sum of the distances between edges and their images; we denote this goal by *(weak) min-sum graph distance* and consider how to compute the (weak) min-sum graph distance for the setting where G_1 is a tree in Section 2.

Another goal is to refine the minmax optimization goal of the definition of $\vec{\delta}_{wG}(G_1, G_2)$. Intuitively, in addition to the largest value, we want to minimize the second largest value with respect to the largest value and so on. We denote this goal by *lexicographic graph distance*. Note that this approach only applies for the weak graph distance. We show how to compute the lexicographic graph distance for the setting where both graphs are planar in Section 3.

2 Min-Sum Graph Distance for Trees

Here we consider the min-sum distance where the first graph G_1 is a tree. First, we show that computing the min-sum graph distance is NP-hard when using the Fréchet distance. Then, we give an algorithm that runs in polynomial time, but restricts the choices of where to map a vertex of the first graph (which circumvents the actual NP-hardness of the problem).

The NP-hardness for the (weak) Fréchet distance as underlying distance measure for the min-sum graph distance stems from the following. Within the placement p of an edge e that is valid with respect to some threshold ε, the mapping of e onto a path in p is not unique and different choices imply a smaller or larger weight assigned to the placement. Stated differently: Along a path P in G_1 it is unclear at which points $s(P)$ should be cut

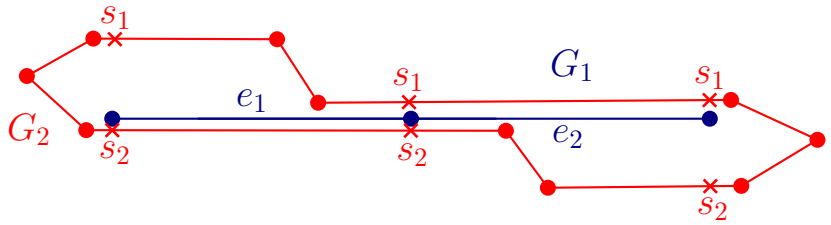


Figure 2 The graph mapping s_1 is locally optimal for e_2 but not for e_1, whereas the graph mapping s_2 is locally optimal for e_1 but not for e_2.

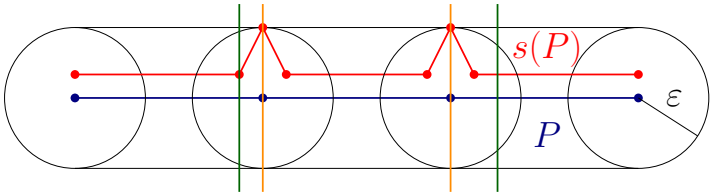


Figure 3 When cutting $c(P)$ at the intersections with the green vertical lines, both peaks are charged to the middle segment. The total weight of $c(P)$ is thus $\frac{\varepsilon}{3} + \varepsilon + \frac{\varepsilon}{3} = \frac{4}{3}\varepsilon$. However, when cutting at the intersections with the orange lines, the weight is $\varepsilon + \varepsilon + \varepsilon = 3\varepsilon$ as each edge is mapped to a path containing an ε-peak.

to minimize the sum of the weights of the (weak) Fréchet distance between the edges of P and their mappings along $s(P)$. An example of different cuts leading to a different sum is given in Figure 3. This issue also occurs when applying other maximum distances such as the Hausdorff distance.

Note that for distance measures like the continuous dynamic time warping, that sum up the distance between an edge e and a path p while traversing e and p from beginning to end, it is irrelevant where to cut $s(P)$ as the algorithm considers the sum over all weights along P. Thus, these measures are more suitable than bottleneck distances, and we conjecture that for these the distance may be computable in polynomial time.

Another aspect that should be considered in more detail for each chosen distance measure is the construction or detection of a path with minimum distance to an edge e within a placement p of e. For instance, an efficient algorithm to find a path within p with Fréchet distance to e is proposed by Alt *et al.* [6].

2.1 Algorithmic Approach

First, we compute the bottleneck distance $\varepsilon = \vec{\delta}_{(w)G}(G_1, G_2)$. Subsequently, we choose an optimal mapping based on ε. That is, we compute $\min_s \sum_{e \in E_1} dist(e, s(e))$, where $s\colon G_1 \to G_2$ ranges over all valid graph mappings with respect to ε and *dist* denotes some distance measure between an edge and a path.

Note that it may be possible to decrease this value by choosing an initial value $\varepsilon' > \vec{\delta}_{(w)G}(G_1, G_2)$. Mappings with small distance to most of the edges of G_1 are possibly declared invalid with respect to ε if the bottleneck distance for these mappings is large. Therefore, an alternative initial value can be used if we allow an (arbitrarily) large bottleneck distance for the min-sum graph distance. However, computing the min-sum graph distance with respect to ε is an optimal mapping respecting the global directed graph distance.

Description of the Search Structure For each edge $e = (u, v) \in G_1$ and each pair of ε-placements C_u, C_v, we compute the minimum (weak) Fréchet distance $\Delta(C_u, C_v)$ of e and an edge-placement of e connecting C_u and C_v[2]. We store a list L_e with entries $(C_u, C_v, \Delta(C_u, C_v))$ for each combination of vertex placements. That is, we augment step 2 of the algorithm by explicitly computing and storing the distance between an edge and its placement. This distance is defined as the minimum distance between the edge and some path within the placement. This can be done in $O(n_1 m_2^2 \log(m_2) + T)$ time and $O(n_1 m_2^2)$

[2] Here, we fix the edge placement, i.e. cutting points, before optimization which circumvents the NP-hardness. For instance, we could always cut at the closest point.

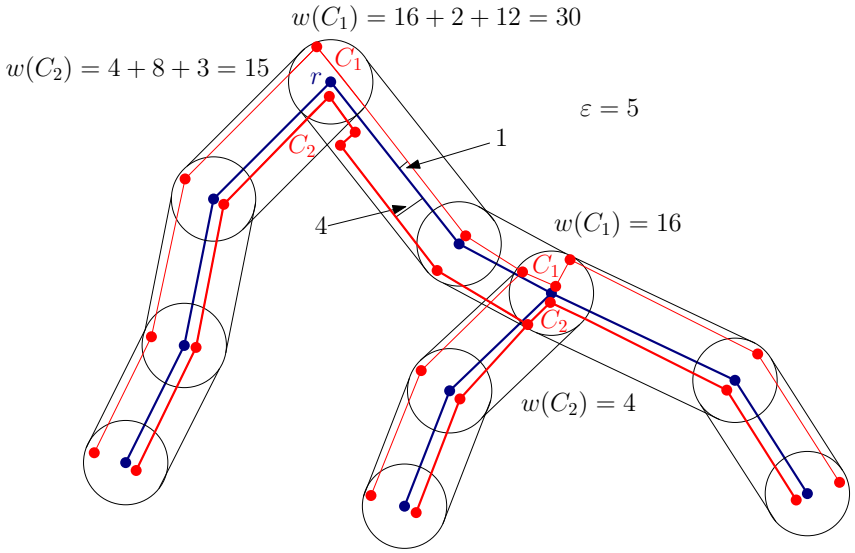


Figure 4 The weights of the vertex placements of the root r consist of the sum of the weights of the vertex placements of the children of r in $\overline{G}_1$ and the weights of the paths between the placements. The min-sum graph mapping of G_1 (in blue) onto G_2 (in red) is marked with bold lines.

space, where T is the time to compute the distances between edges and their ε-placements. We denote the set of placements of a vertex u by $P(u)$.

We will compute the min-sum graph distance *bottom-up*, maintaining the invariant that subgraphs have been optimally placed for any vertex-placement. For this, we consider G_1 as directed tree with arbitrary root r. The (abstract) *reachability graph* H of the vertex placements has one vertex for each vertex placement of a vertex of G_1. Two vertices C_u, C_v are adjacent in H if the corresponding vertices u, v of G_1 are adjacent and if the two placements are reachable from each other. The edges of H are weighted with the minimum (weak) Fréchet distance between the corresponding edge of G_1 and an edge placement connecting C_u and C_v in G_2. Note that alternatively to the Fréchet bottleneck distances one might, for instance, use the minimum area between an edge and a valid path as weights. As each vertex $u \in V_1$ has $O(m_2)$ vertex placements the vertex set V_H of H has size $O(n_1 m_2)$ and the set of edges E_H has size $O(n_1 m_2^2)$. We consider the edges of H to be directed according to the direction of the edges of G_1. Obviously, H is a directed acyclic graph (DAG).

For simplicity, let $\overline{G}_1$ be the graph obtained by replacing all paths of internal degree-2 vertices in G_1 by one edge between the start and endpoint of the path. Then, for all vertices v of $\overline{G}_1$, $deg(v) \geq 3$. See Figure 4 for a small example of a min-sum graph mapping.

Description of the Algorithm First, we set the weight $w(p) = 0$ for all placements p of vertices of $\overline{G}_1$. Let $u \in \overline{V}_1$ be a vertex whose children are all leaves. We compute

$$w(C_u) = \sum_{u' : u' \text{is child of } u} \min_{C_{u'} \in P(u')} w(C_{u'}) + w_H(C_u, C_{u'}), \tag{1}$$

where $w_H(C_u, C_{u'})$ is the weight of a minimum weight shortest path P between C_u and $C_{u'}$ in H. We store the mapping s which realizes $w(C_u)$, delete the subtree of $\overline{G}_1$ rooted at u, and proceed with the next vertex of the updated graph $\overline{G}_1$ with leaf-children only until we encounter the root r. After having processed r, the following theorem holds:

▶ **Theorem 1.** *We can compute a valid mapping s in $O(n_1 m_2^3)$ time and $O(n_1 m_2^2)$ space such that for any other valid mapping $s' : G_1 \to G_2$ we have*

$$\sum_{e \in E_1} dist(e, s'(e)) \geq \sum_{e \in E_1} dist_{(w)F}(e, s(e)).$$

Proof. The graph H can be computed in $O(n_1 m_2^2 \log(m_2))$ time and uses $O(n_1 m_2^2)$ space. The algorithm maintains the invariant of optimally mapped subtrees and thus terminates with a min-sum mapping. The complexity of the subgraph of H for computing $w_H(C_u, C_{u'})$ for all children u' of u in $\overline{G}_1$ is $O(l m_2^2)$, where l is the length of the path between u and u' in G_1. Since H is a DAG, we can compute equation (1) in $(deg(u) - 1)O(l m_2^2)$ time using topological sorting. As u has up to m_2 placements, the runtime for processing one vertex of $\overline{G}_1$ is $O(deg(u) l m_2^3)$. With $\sum_{u \in \overline{V}_1} deg(u) l = 2m_1 = 2(n_1 - 1)$, the runtime follows. ◀

▶ Remark 2. While computing the directed graph distance for planar embedded graphs is NP-hard, one can compute the directed weak graph distance in polynomial time [5]. It remains open whether the weak min-sum graph distance can be computed in polynomial time for planar embedded graphs, but we conjecture that the problem is NP-hard.

3 Lexicographic Graph Distance

Now we consider the lexicographic graph distance based on weak Fréchet distance. For this, we first formally define it.

▶ **Definition 3.** *Let $s \colon G_1 \to G_2$ be a mapping. We say that s is a* mapping realizing the lexicographic graph distance*, if it has the following property: Given an arbitrary subdivision D of the graph G_1 (with finite number l of edges). Let $e_1, e_2, \ldots, e_l$ be a numbering of the edges of D such that $\varepsilon_1 \geq \varepsilon_2 \geq \cdots \geq \varepsilon_l$, where $\varepsilon_i = \delta_{wF}(e_i, s(e_i))$. If there exists another mapping $\hat{s}$, such that $\delta_{wF}(e_i, \hat{s}(e_i)) < \delta_{wF}(e_i, s(e_i))$ for some $i \in \{1, 2, \ldots, l\}$, then there exists an index $j < i$ such that $\delta_{wF}(e_j, \hat{s}(e_j)) > \delta_{wF}(e_j, s(e_j))$.*

A mapping that realizes the lexicographic graph distance is a *lexicographic graph mapping*. Note that here we use the term lexicographic with a slight abuse of the standard notation of a lexicographic order. This is due to the fact that the entities that must be ordered differ for each mapping from G_1 to G_2. Intuitively, any mapping with a locally smaller distance in comparison with a lexicographic graph mapping increases the value of some larger weak Fréchet distance of an edge and its image.

In the following, we first assume that no pair of edges (e_1, e_2), where $e_1 \in E_1$ and e_2 in E_2, is parallel. Hence we can assume that the weak Fréchet distance between an edge $e \in E_1$ and a path $s(e)$ in G_2 is characterized by the maximum M of the maximum distance between a vertex $v \in V_2$ on $s(e)$ and the edge e and the distances of the endpoints of e and $s(e)$. Second, we assume that the length of the perpendiculars of edges of G_2 and vertices of G_1 are unique. Furthermore, we assume that M is uniquely defined by a vertex on $s(e)$ or by one of the endpoints of $s(e)$. Last, we assume that for any valid mapping s, the weak Fréchet distance between an edge e and $s(e)$ are unique. These assumptions imply:

▶ **Lemma 4.** *Let $\varepsilon = \vec{\delta}_{wG}(G_1, G_2)$. If $\delta_{wF}(e, s(e)) = \varepsilon$ for an edge $e = (u, v) \in E_1$ and a valid mapping $s \colon G_1 \to G_2$. Then, exactly one of the following cases occurs:*

- ***Case 1:*** *There is a vertex $w \in V_2$ on $s(e)$ with $dist(e, w) = \varepsilon$. Then, for each valid mapping $\hat{s} \neq s$, $\hat{s}(e)$ contains w and therefore $\delta_{wF}(e, \hat{s}(e)) = \varepsilon$.*
- ***Case 2:*** *$dist(u, s(u)) = \varepsilon$. In this case, u has exactly one valid placement.*
- ***Case 3:*** *$dist(u, s(v)) = \varepsilon$. In this case, v has exactly one valid placement.*

Description of the Algorithm First we compute the directed weak graph distance $\varepsilon = \vec{\delta}_{wG}(G_1, G_2)$ and store a copy of G_2 restricted to the ε-surrounding of e for each edge $e \in E_1$. We denote this subgraph by $G_2(e)$. In each step of the algorithm we compute $\varepsilon = \vec{\delta}_{wG}(G_1, G_2)$, where the subgraphs $G_2(e)$ are used for all graph explorations in step **(1)** and **(2)**. We consider the unique edge e with $\delta_F(e, s(e)) = \varepsilon$ and the unique point w on $s(e)$ with distance ε to e and remove this bottleneck by updating the graph as follows: Snap w to w' on e at distance ε and update all incident edges of w in $G_2(w)$ accordingly. Proceed until $\vec{\delta}_{wG}(G_1, G_2) = 0$. Figure 5 illustrates the iterations of the algorithm.

Note that the updated graphs are not necessarily plane, but planarity is not needed for computing valid placements and pruning invalid placements. To compute a mapping for two adjacent cycles, the corresponding placements in the plane graph G_2 are used.

▶ **Theorem 5.** *Given plane graphs G_1, G_2, the algorithm described above computes a lexicographic graph mapping $s\colon G_1 \to G_2$ in $O(n_1^2 n_2^2 \log(n_1 + n_2))$ time using $O(n_1 n_2)$ space.*

Proof. Let $G_2(e = (u, v))$ be the graph updated in iteration i and let $G_2(e)_{prev}$ be the graph before the update. Furthermore, let $\varepsilon = \vec{\delta}_{wG}(G_1, G_2(e))$ be the weak directed graph distance in iteration $i + 1$ and $\varepsilon_{prev} = \vec{\delta}_{wG}(G_1, G_2(e)_{prev})$. Now Lemma 4 implies that for a valid mapping $s\colon G_1 \to G_2(e)$ with respect to ε, the corresponding re-transformed mapping is a valid mapping from G_1 onto $G_2(e)_{prev}$ with respect to ε_{prev}. That is, when the algorithm terminates, we can easily compute a valid mapping from G_1 onto G_2 by a series of re-transformations of the graph G_2. The obtained mapping is a lexicographic graph mapping as in each step the algorithm identifies the current bottleneck distance. In each iteration, either a vertex of G_2 is snapped onto an edge e of G_1, or a point of an edge of G_2 is snapped onto an endpoint of e. The latter case can only happen once for each endpoint of e. Therefore, at most $n_2 + 2$ points of G_2 are snapped onto e until the distance between e and $s(e)$ is zero for any valid mapping s. Thus, after a maximum of $m_1(n_2 + 2) = O(n_1 n_2)$ iterations, the algorithm terminates. Hence, the total runtime is $O(n_1^2 n_2^2 \log(n_1 + n_2))$. ◀

▶ Remark 6. The definition and algorithmic approach cannot be directly transferred to the Fréchet distance instead of the weak Fréchet distance. The Fréchet distance depends on the specific subdivision of an edge. In general, the distance increases for larger subdivisions and is maximal between the whole edge and the corresponding path.

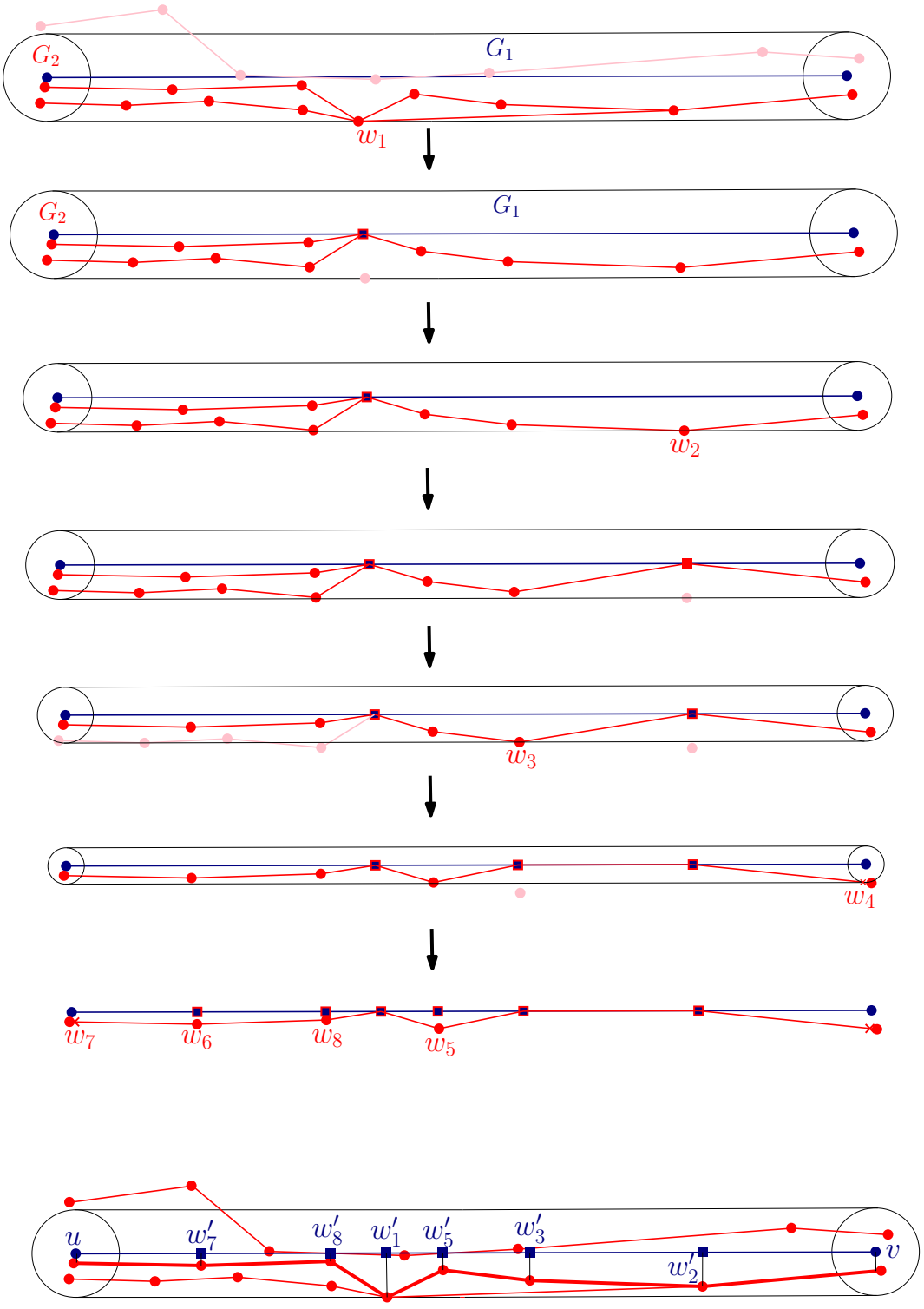


Figure 5 Iteratively updating G_2. The lexicographic graph mapping (bold line) is uniquely defined by the vertices w_i and u, v, w'_i: $s_{opt}(u) = w_7$, $s_{opt}(v) = w_4$, $s_{opt}(w'_7) = w_6$, $s_{opt}(w'_8) = w_8$, $s_{opt}(w'_1) = w_1$, $s_{opt}(w'_5) = w_5$, $s_{opt}(w'_3) = w_3$, $s_{opt}(w'_2) = w_2$, $s_{opt}(v) = w_4$.

References

1. Mahmuda Ahmed, Brittany Terese Fasy, Kyle S. Hickmann, and Carola Wenk. Path-based distance for street map comparison. *ACM Transactions on Spatial Algorithms and Systems*, 28 pages, 2015.
2. Mahmuda Ahmed, Brittany Terese Fasy, and Carola Wenk. Local persistent homology based distance between maps. In *22nd ACM SIGSPATIAL GIS*, pages 43–52, 2014.
3. Mahmuda Ahmed, Sophia Karagiorgou, Dieter Pfoser, and Carola Wenk. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica*, 19(3):601–632, 2015.
4. Mahmuda Ahmed, Sophia Karagiorgou, Dieter Pfoser, and Carola Wenk. *Map Construction Algorithms*. Springer, 2015.
5. Hugo A. Akitaya, Maike Buchin, Bernhard Kilgus, Stef Sijben, and Carola Wenk. Distance Measures for Embedded Graphs. In Pinyan Lu and Guochuan Zhang, editors, *30th International Symposium on Algorithms and Computation (ISAAC 2019)*, volume 149 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 55:1–55:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/11551`, `doi:10.4230/LIPIcs.ISAAC.2019.55`.
6. Helmut Alt, Alon Efrat, Günter Rote, and Carola Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262 – 283, 2003.
7. Kevin Buchin, Maike Buchin, Wouter Meulemans, and Bettina Speckmann. Locally correct Fréchet matchings. In *Proceedings of the 20th Annual European Conference on Algorithms*, ESA'12, pages 229–240, Berlin, Heidelberg, 2012. Springer-Verlag. URL: `http://dx.doi.org/10.1007/978-3-642-33090-2_21`, `doi:10.1007/978-3-642-33090-2_21`.
8. Otfried Cheong, Joachim Gudmundsson, Hyo-Sil Kim, Daria Schymura, and Fabian Stehn. Measuring the similarity of geometric graphs. In *International Symposium on Experimental Algorithms*, pages 101–112, 2009.
9. Sophia Karagiorgou and Dieter Pfoser. On vehicle tracking data-based road network generation. In *20th ACM SIGSPATIAL GIS*, pages 89–98, 2012.
10. Günter Rote. Lexicographic Fréchet matchings. In *Proc. 30rd European Workshop on Computational Geometry (EuroCG)*, 2014.