

Algorithms for Distance Problems in Continuous Graphs

Sergio Cabello ✉ 

Faculty of Mathematics and Physics, University of Ljubljana, Slovenia
Institute of Mathematics, Physics and Mechanics, Ljubljana, Slovenia

Delia Garijo ✉ 

University of Seville, Spain

Antonia Kalb ✉ 

Technical University of Dortmund, Germany

Fabian Klute ✉ 

Universitat Politècnica de Catalunya, Barcelona, Spain

Irene Parada ✉ 

Universitat Politècnica de Catalunya, Barcelona, Spain

Rodrigo I. Silveira ✉ 

Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract

We study the problem of computing the diameter and the mean distance of a continuous graph, i.e., a connected graph where all points along the edges, instead of only the vertices, must be taken into account. It is known that for continuous graphs with m edges these values can be computed in roughly $O(m^2)$ time. In this paper, we use geometric techniques to obtain subquadratic time algorithms to compute the diameter and the mean distance of a continuous graph for two well-established classes of sparse graphs. We show that the diameter and the mean distance of a continuous graph of treewidth at most k can be computed in $O(n \log^{O(k)} n)$ time, where n is the number of vertices in the graph. We also show that computing the diameter and mean distance of a continuous planar graph with n vertices and F faces takes $O(nF \log n)$ time.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases diameter, mean distance, continuous graph, treewidth, planar graph

Digital Object Identifier 10.4230/LIPIcs.WADS.2025.13

Related Version *Full Version*: <https://arxiv.org/abs/2503.07769> [11]

Funding *Sergio Cabello*: Funded in part by the Slovenian Research and Innovation Agency (P1-0297, N1-0218, N1-0285). Funded in part by the European Union (ERC, KARST, project number 101071836). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Delia Garijo: Partially supported by grants PID2019-104129GB-I00 and PID2023-150725NB-I00 funded by MICIU/AEI/10.13039/501100011033.

Fabian Klute: Partially supported by grants PID2019-104129GB-I00 and PID2023-150725NB-I00 funded by MICIU/AEI/10.13039/501100011033.

Irene Parada: Serra Hùnter Fellow. Partially supported by grants PID2019-104129GB-I00 and PID2023-150725NB-I00 funded by MICIU/AEI/10.13039/501100011033.

Rodrigo I. Silveira: Partially supported by grants PID2019-104129GB-I00 and PID2023-150725NB-I00 funded by MICIU/AEI/10.13039/501100011033.



© Sergio Cabello, Delia Garijo, Antonia Kalb, Fabian Klute, Irene Parada, and Rodrigo I. Silveira; licensed under Creative Commons License CC-BY 4.0

19th International Symposium on Algorithms and Data Structures (WADS 2025).

Editors: Pat Morin and Eunjin Oh; Article No. 13; pp. 13:1–13:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Graph parameters dealing with distances provide fundamental information on the graph. The *diameter*, defined as the maximum distance between any two vertices of a graph, and the *mean distance*, which gives the average of all those distances, are natural concepts of great importance in real-world applications. While the diameter gives the maximum eccentricity in the graph, the mean distance provides a measure of its compactness, and is closely related to the *sum of the pairwise distances* of the graph and the well-known *Wiener index*.¹

Computing the diameter and the sum of the pairwise distances of a given graph G is a central problem in algorithmic graph theory. A straightforward algorithm is to perform Dijkstra's algorithm from each vertex, allowing to compute both parameters in $O(nm + n^2 \log n)$ time, for n and m the number of vertices and edges of G , respectively. Given the high computational cost of this approach, considerable effort has been invested in developing faster algorithms, especially for sparse graphs. It turns out that the general problem is notably difficult. In 2013, it was showed that in sparse graphs, no $O(n^{2-\epsilon})$ -time algorithm can distinguish between diameter 2 and 3, unless the Strong Exponential Time Hypothesis fails [29]. From the proof, one can deduce the same conditional lower bound for computing the sum of the pairwise distances of the graph (see also [9]). This justifies the vast amount of ongoing research on identifying classes of sparse graphs for which these parameters can actually be computed in subquadratic time. Currently, such classes include graphs of bounded treewidth [1, 8, 12], graphs of bounded distance VC dimension [18, 26], median graphs [4, 5], and planar graphs [9, 22].

In this work, we tackle the challenge of subquadratic diameter and mean distance computation for *continuous graphs* (these objects are also called *metric graphs* in other areas closer to analysis [19, 27]). Our main motivation arises from *geometric graphs*. A geometric graph is an undirected graph where each vertex is a two-dimensional point, and each edge is a straight line segment between the corresponding two points. These graphs naturally arise in applications involving geographic information, such as road or river networks. The main characteristic of geometric graphs is that every point on each edge is considered part of the graph. Therefore, the graph can be considered an infinite point set. The diameter of a geometric graph is the maximum distance taken over its infinitely many pairs of points. The class of continuous graphs is actually more general than geometric graphs, and is formally defined in Section 2. Distances in continuous graphs, especially the diameter, have received a lot of interest recently, mainly in the context of augmentation problems [2, 13–16, 20, 21, 23, 24]; see also [3] for results on the mean distance in the context of geometric analysis. Another well-known related problem about distances in graphs, also with a continuous aspect, is the computation of the absolute center of a graph, originally proposed by Hakimi [25].

The diameter and the mean distance of a continuous graph with m edges can be computed in roughly $O(m^2)$ time [13, 16, 21]². For the diameter, this follows from the fact that, in a continuous graph, any pair of points attaining the diameter, called *diametral pair*, consists of either: (i) two vertices, (ii) two points on distinct non-pendant edges,³ or (iii) a pendant vertex and a point on a non-pendant edge [13, Lemma 6] (see Figure 1). Regarding the mean distance, one can show that it is given by a weighted sum of the mean distances of all ordered pairs of edges, which can be obtained in constant time, once the distance matrix of the vertices of the graph has been computed [21].

¹ The sum of the pairwise distances of a graph is the sum of distances between all ordered pairs of vertices, and half of this value is the Wiener index. This topological index has been studied extensively.

² The algorithm to compute the diameter in [13] is for plane geometric graphs, but it also applies here.

³ An edge $uv \in E(G)$ is *pendant* if either u or v is a *pendant* vertex (i.e., has degree 1).



■ **Figure 1** Types of diametral pairs of a continuous graph.

However, for sparse graphs, one hits again a quadratic running time barrier. Algorithms for diameter in discrete graphs do not carry over to continuous graphs, except in few situations (e.g., if there are only $O(1)$ different edge weights, then $O(1)$ Steiner points can be added to each edge, so that the diameter coincides with that of the continuous graph), and the same conditional lower bound of the discrete setting holds for the continuous case (one can reduce the continuous case to the discrete case by simply adding enough long paths to the graph.)

The main challenge for continuous graphs is that the techniques that have successfully worked to speed-up the computation of the diameter and the sum of the pairwise distances for discrete graphs do not seem to easily extend. The most similar setting to ours is perhaps that of planar graphs, for which recently the first subquadratic algorithms were discovered [9, 22]. These works use Voronoi diagrams in planar graphs to compute those values in the discrete setting. However, it is not clear whether they can be adapted to the continuous setting. More precisely, for a fixed source vertex and a fixed subgraph H of a graph G , they compute the Voronoi diagram of H using some additive weights. As the source moves, the additive weights defining the Voronoi diagram change, and the Voronoi diagrams change. Tracing those changes efficiently seems difficult, especially because the combinatorial structure of the Voronoi diagram may undergo important changes. Moreover, such changes can happen for several different movements of the sources. Thus, to achieve a subquadratic algorithm for planar continuous graphs, it seems that one should be able to treat those parallel changes in groups. The current technology for planar graphs does not seem ready for this.

Contributions. In this work, we present subquadratic algorithms to compute the diameter and the mean distance for two classes of sparse continuous graphs. In Sections 3 and 4, We study continuous graphs of bounded treewidth and show how to compute, respectively, its diameter and its mean distance in subquadratic time. In fact, we consider the slightly more general framework of computing the diameter $\text{diam}(\mathcal{H}, \mathcal{G})$ and the mean distance $\text{mean}(\mathcal{H}, \mathcal{G})$ of a subgraph $\mathcal{H}$ of a continuous graph $\mathcal{G}$ with respect to $\mathcal{G}$, which are the diameter and mean distance of $\mathcal{G}$ when $\mathcal{H} = \mathcal{G}$. This concept appears naturally in our algorithm during the recursion. Theorems 1 and 2 below distinguish whether the treewidth is assumed to be constant, as done in [12], or a parameter, as done in [8].

► **Theorem 1** (Theorems 11 and 16). *Let $k \geq 2$ be an integer constant. Let G be a graph with n vertices, treewidth at most k , nonnegative edge-lengths, and let $\mathcal{G}$ be the corresponding continuous graph. Let H be a subgraph of G and let $\mathcal{H} \subseteq \mathcal{G}$ be the corresponding continuous subgraph. We can compute the diameter $\text{diam}(\mathcal{H}, \mathcal{G})$ and the mean distance $\text{mean}(\mathcal{H}, \mathcal{G})$ in $O(n \log^{4k-2} n)$ time.*

► **Theorem 2** (Theorems 12 and 17). *Let G be a graph with n vertices, treewidth at most k , nonnegative edge-lengths, and let $\mathcal{G}$ be the corresponding continuous graph. Let H be a subgraph of G and let $\mathcal{H} \subseteq \mathcal{G}$ be the corresponding continuous subgraph. We can compute the diameter $\text{diam}(\mathcal{H}, \mathcal{G})$ and mean distance $\text{mean}(\mathcal{H}, \mathcal{G})$ in $n^{1+\varepsilon} 2^{O(k)}$ time, for any fixed $\varepsilon > 0$.*

Similarly to previous algorithms in the discrete setting to compute the diameter and the sum of the pairwise distances for graphs of bounded treewidth [1, 8, 12], the key technique that we use is orthogonal range searching; see also [17] for a novel application of this technique to compute the eccentricity and the distance-sum of any vertex of a directed weighted graph.

Finally, we investigate planar graphs. For any n -vertex continuous plane graph, we show how to compute the maximum eccentricity (i.e., largest distance from a point) over all points on the boundary of a face in $O(n \log n)$ time. Further, we show that the same approach can be used to compute the mean from all points of a face with respect to the continuous graph in $O(n \log n)$ time. This allows us to compute the diameter and mean of continuous planar graphs in $O(nF \log n)$ time, where F is the number of faces, which is subquadratic, if $F = o(\frac{n}{\log n})$. (By Euler's formula, F is the same for any embedding of a planar graph.)

► **Theorem 3.** *For a continuous planar graph $\mathcal{G}$ with n vertices and F faces, we can compute its diameter $\text{diam}(\mathcal{G})$ and its mean distance $\text{mean}(\mathcal{G})$ in $O(nF \log n)$ time.*

Note that some proofs and the section about planar graphs are deferred to the full version.

2 Preliminaries

2.1 Continuous Graphs

Consider an edge-weighted, connected graph $G = (V(G), E(G), \ell)$, where ℓ is a function that assigns a length $\ell(e) \geq 0$ to each edge $e \in E(G)$ (clearly, for our purposes we assume that not all edge lengths are 0). Informally, the *continuous graph* $\mathcal{G}$ defined by G is the infinite set of points determined by the vertices and edges of G , where each point on an edge is part of the graph. This idea requires to define precisely what we mean by a point on an edge.

For each edge uv of G , we take a closed segment of length $\ell(uv)$ with the usual metric and measure, and denote it $\mathcal{G}(uv)$. Moreover, for such an edge uv , we arbitrarily select one extreme of the segment $\mathcal{G}(uv)$ and denote it $\text{endp}(uv, u)$, and call $\text{endp}(uv, v)$ the other endpoint of $\mathcal{G}(uv)$. Finally, for each vertex u of G , we glue (mathematically, we identify) all points $\text{endp}(uv, u)$ over all edges uv of G incident to u ; we denote by $\mathcal{G}(u)$ the identified point. The continuous graph $\mathcal{G}$ defined by $G = (V(G), E(G), \ell)$ is the resulting space. The total length $\ell(\mathcal{G})$ of a continuous graph $\mathcal{G}$ defined by a graph G is defined as $\sum_{uv \in E(G)} \ell(uv)$.

We observe that one may think of $\mathcal{G}$ as a 1-dimensional simplicial complex where each edge uv is isometric to a segment of length $\ell(uv)$.

A point p of $\mathcal{G}$ can be specified by a triple $(uv, u, \lambda) \in E(G) \times V(G) \times [0, \ell(uv)]$, which represents the point of the segment $\mathcal{G}(uv)$ at distance λ (along $\mathcal{G}(uv)$) from the endpoint $\text{endp}(uv, u)$. The triples (uv, u, λ) and $(uv, v, \ell(uv) - \lambda)$ define the same point of $\mathcal{G}$. Similarly, for any two incident edges uv, uv' of G , the triples $(uv, u, 0)$, $(uv, v, \ell(uv))$, $(uv', u, 0)$ and $(uv', v', \ell(uv'))$ define the same point, namely $\mathcal{G}(u)$.

We have already set the notation $\mathcal{G}(uv)$ for an edge uv and $\mathcal{G}(u)$ for a vertex u . With a slight abuse of notation, we do not distinguish uv from $\mathcal{G}(uv)$ and u from $\mathcal{G}(u)$.

In general, for any subgraph H of G , we denote by $\mathcal{G}(H)$ the continuous subgraph of $\mathcal{G}$ defined by the objects of H . This is also the continuous graph defined by H . Also, when H is clear from the context, we denote such an object by $\mathcal{H}$ and talk about $\mathcal{H}$ as a subgraph of $\mathcal{G}$. For a vertex set $A \subseteq V(G)$, we use $G[A]$ to denote the subgraph of G induced by A .

A *walk* in $\mathcal{G}$ between two points $p, q \in \mathcal{G}$ is a sequence $pv_1, v_1v_2, \dots, v_{k-1}v_k, v_kq$ with $v_1v_2, \dots, v_{k-1}v_k \in E(G)$. If the first and last point coincide, it is *closed*. The length of a walk is the sum of the length of its pieces, counted with multiplicity. For any two points $p, q \in \mathcal{G}$, the distance $d_{\mathcal{G}}(p, q)$ is the minimum length over all p -to- q walks. A *shortest pq -path* is a p -to- q walk $\pi(p, q)$ in $\mathcal{G}$ such that $\ell(\pi(p, q)) = d_{\mathcal{G}}(p, q)$.

We can regard $d_{\mathcal{G}}(p, q)$ as the discrete graph-theoretic distance in a graph obtained by subdividing aa' with p as a new vertex, subdividing bb' with q as a new vertex, and setting $\ell(ap) = \lambda$, $\ell(pa') = \ell(aa') - \lambda$, $\ell(bq) = \lambda'$, and $\ell(qb') = \ell(bb') - \lambda'$.

2.2 Distance Problems

Consider a continuous graph $\mathcal{G}$ defined by a graph G and a continuous subgraph $\mathcal{H} \subseteq \mathcal{G}$ defined by a subgraph $H \subseteq G$. We are interested in distances between points of $\mathcal{H}$ using the metric given by $\mathcal{G}$. One may think of $\mathcal{G}$ as the ambient space and of $\mathcal{H}$ as the relevant subset.

The *eccentricity* of a point $p \in \mathcal{H}$ with respect to $\mathcal{H}$ is $\text{ecc}(p, \mathcal{H}, \mathcal{G}) = \max_{q \in \mathcal{H}} d_{\mathcal{G}}(p, q)$; when $\mathcal{H} = \mathcal{G}$, we just talk about the eccentricity of p in $\mathcal{G}$ and write $\text{ecc}(p, \mathcal{G})$ for $\text{ecc}(p, \mathcal{H}, \mathcal{G})$. The *diameter* of $\mathcal{H}$ with respect to $\mathcal{G}$ is defined as $\text{diam}(\mathcal{H}, \mathcal{G}) = \max_{p, q \in \mathcal{H}} d_{\mathcal{G}}(p, q)$; when $\mathcal{H} = \mathcal{G}$, we just talk about the diameter of $\mathcal{G}$ and write $\text{diam}(\mathcal{G})$ for $\text{diam}(\mathcal{G}, \mathcal{G})$. It is easy to see that $\text{diam}(\mathcal{H}, \mathcal{G}) = \max_{p \in \mathcal{H}} \text{ecc}(p, \mathcal{H}, \mathcal{G})$. The *sum of distances* of $\mathcal{H}$ in $\mathcal{G}$ is the sum of the pairwise distances in $\mathcal{H}$, using the metric from $\mathcal{G}$, that is, $\text{sumdist}(\mathcal{H}, \mathcal{G}) = \iint_{p, q \in \mathcal{H}} d_{\mathcal{G}}(p, q) dp dq$. The *mean distance* of $\mathcal{H}$ in $\mathcal{G}$ is $\text{mean}(\mathcal{H}, \mathcal{G}) = \frac{1}{\ell(\mathcal{H})^2} \text{sumdist}(\mathcal{H}, \mathcal{G})$. It is easy to see that $\text{mean}(\mathcal{H}, \mathcal{G}) = \frac{1}{\ell(\mathcal{H})} \int_{p \in \mathcal{H}} \text{mean}(p, \mathcal{H}, \mathcal{G}) dp$, where $\text{mean}(p, \mathcal{H}, \mathcal{G}) = \frac{1}{\ell(\mathcal{H})} \int_{q \in \mathcal{H}} d_{\mathcal{G}}(p, q) dq$ is the mean distance from the point $p \in \mathcal{H}$ with respect to $\mathcal{H}$ in $\mathcal{G}$. When $\mathcal{H} = \mathcal{G}$, then we just write $\text{mean}(\mathcal{G})$ for $\text{mean}(\mathcal{G}, \mathcal{G})$ and $\text{mean}(p, \mathcal{G})$ for $\text{mean}(p, \mathcal{G}, \mathcal{G})$.

2.3 Treewidth

The *treewidth* of a graph is an important parameter in algorithmic graph theory which, roughly speaking, measures how far the graph is from being a tree (a formal definition is given in [11]). Graphs of treewidth k have $O(kn)$ edges [6].

A *separation* in a graph G is a triple (A, B, S) such that $A, B, S \subset V(G)$, $A \cup B = V(G)$, $S = A \cap B$, and there is no edge incident to both $A \setminus B$ and $B \setminus A$. The elements of S in separation (A, B, S) are called *portals*. Each path in G from A to B must pass through a portal. Informally, we are interested in separations where A and B have a constant fraction of the vertices and S is small. Such separations of at most k portals exist for graphs of treewidth k , can be computed in linear time, and have the additional property that adding edges between the portals does not increase the treewidth [8, 12]; . For simplicity, we assume that S contains exactly k portals (it may happen that it has fewer). We use the notation $[k] = \{1, \dots, k\}$. We have two regimes, depending on whether we consider the treewidth constant or a parameter.

2.4 Orthogonal Range Searching

We use the notation $B(n, d) = \binom{d + \lceil \log n \rceil}{d}$ to bound the performance of some of our data structures. First we note the following asymptotic bounds.

► **Lemma 4** (Bringmann, Husfeldt, and Magnusson [8]). $B(n, d) = O(\log^d n)$ and $B(n, d) = n^{\varepsilon 2^{O(d)}}$ for each $\varepsilon > 0$.

A *rectangle* in $\mathbb{R}^d$ is the Cartesian product of d intervals (whose extremes can be included or not). The analysis of orthogonal range searching performed in [8, Section 3] (see also [28]) leads to the data structure described in the the following theorem. We use the version suggested by Cabello [10] because it adapts better to our needs. (We use $\sqcup$ to indicate that the union is between pairwise disjoint sets.)

► **Theorem 5** (Cabello [10]). *Given a set P of n points in $\mathbb{R}^d$, there is a family of sets $\mathcal{P} = \{P_j \mid j \in J\}$ and a data structure with the following properties:*

- $P_j \subseteq P$ for each $P_j \in \mathcal{P}$;
- all the sets of $\mathcal{P}$ together have $O(nd \cdot B(n, d))$ points, i.e., $\sum_{P_j \in \mathcal{P}} |P_j| = O(nd \cdot B(n, d))$;
- for each query rectangle $R \subset \mathbb{R}^d$, the data structure finds in $O(2^d B(n, d))$ time indices $J_R \subset J$ such that $|J_R| = O(2^d B(n, d))$ and $P \cap R = \bigcup_{j \in J_R} P_j$;
- the family $\mathcal{P}$ and the data structure can be computed in $O(nd \cdot B(n, d))$ time.

3 Diameter in Graphs with Bounded Treewidth

In this section, we discuss the computation of the diameter of a continuous graph with treewidth k . Note that computing the diameter of continuous trees (i.e., treewidth 1) can be reduced to the discrete setting, because in trees the diameter is always attained by two vertices. Thus, we restrict our attention to $k > 1$. As in [1, 8, 9, 12], we use orthogonal range searching to work with distance-related problems in graphs of bounded treewidth. However, because we consider continuous graphs, we have to consider pairs of edges instead of pairs of vertices, and the interaction between edges is more complex. We handle this by using more dimensions in the range searching space.

Sometimes we need to consider an orientation for each edge of G , to distinguish its vertices. We orient the edges of G arbitrarily, but keep track of the orientation. We use uv when the orientation is not relevant and (u, v) or (v, u) , depending on the orientation, when we consider it oriented. We use $E(G)$ in both cases.

3.1 Characterization of the Diameter via Walks

We start with a characterization of the diameter in the continuous setting that uses the length of walks (compare to Figure 1). For each $aa', bb' \in E(G)$, let $W(aa', bb')$ be a shortest closed walk passing through all the interior points of aa' and bb' .

► **Lemma 6.** *For each $aa', bb' \in E(G)$, $\max_{p \in aa', q \in bb'} d_G(p, q) = \frac{\ell(W(aa', bb'))}{2}$.*

The following corollary is an immediate consequence of Lemma 6.

► **Corollary 7.** *If $H \subseteq G$ defines the continuous subgraph $\mathcal{H}$ of $\mathcal{G}$, then $\text{diam}(\mathcal{H}, \mathcal{G}) = \frac{1}{2} \cdot \max\{\ell(W(aa', bb')) \mid aa', bb' \in E(H)\}$.*

For our computation, we use the following closed formula.

► **Lemma 8.** *For each $aa', bb' \in E(G)$, it holds that*

$$\ell(W(aa', bb')) = \ell(aa') + \ell(bb') + \min\{d_G(a, b) + d_G(a', b'), d_G(a, b') + d_G(a', b)\}.$$

For each pair of oriented edges $(a_0, a_1), (b_0, b_1) \in E(G)$, Lemma 8 implies that there are two possible values for $\ell(W(a_0a_1, b_0b_1))$. We say that (a_0, a_1, b_0, b_1) is of *type 1* if

$$\ell(W(a_0a_1, b_0b_1)) = \ell(a_0a_1) + \ell(b_0b_1) + d_G(a_0, b_0) + d_G(a_1, b_1),$$

and of *type 2* otherwise. For each oriented edge $(b_0, b_1) \in E(G)$ and type $\tau \in \{1, 2\}$, we define

$$\text{Type}_\tau(b_0, b_1) = \{(a_0, a_1) \in E(G) \mid (a_0, a_1, b_0, b_1) \text{ is of type } \tau\}.$$

Therefore, $(a_0, a_1) \in \text{Type}_1(b_0, b_1) \iff d_G(a_0, b_0) + d_G(a_1, b_1) \leq d_G(a_0, b_1) + d_G(a_1, b_0)$.

3.2 Diameter Across the Portals

Let (A, B, S) be a separation in G with k portals. We fix an enumeration $s_1, \dots, s_k$ of them. Let $E_A \subseteq E(G[A])$ be the edge set of the subgraph of G induced by A and $E_B \subseteq E(G[B])$, respectively; not necessarily disjoint. For each index $i \in [k]$, each vertex $a \in A$, and each vertex $b \in B$, let $\varphi(i; a, b)$ be the logic predicate that holds whenever s_i is the first portal in the enumeration that lies in some shortest path from a to b . Formally,

$$\varphi(i; a, b) = \bigwedge_{j < i} [d_G(a, b) < d_G(a, s_j) + d_G(s_j, b)] \wedge [d_G(a, b) = d_G(a, s_i) + d_G(s_i, b)].$$

It is easy to see that, for each $(a, b) \in A \times B$, there exists a unique index $i \in [k]$ where $\varphi(i; a, b)$ holds (in other words, $|\{i \in [k] \mid \varphi(i; a, b)\}| = 1$).

We extend this to the four shortest paths defined by two vertices $a_0, a_1 \in A$ and two vertices $b_0, b_1 \in B$, by defining the following predicate for all $\kappa = (i_{0,0}, i_{0,1}, i_{1,0}, i_{1,1}) \in [k]^4$:

$$\Phi(\kappa; a_0, a_1, b_0, b_1) = \Phi((i_{0,0}, i_{0,1}, i_{1,0}, i_{1,1}); a_0, a_1, b_0, b_1) = \bigwedge_{(\alpha, \beta) \in \{0,1\}^2} \varphi(i_{\alpha, \beta}; a_\alpha, b_\beta).$$

Therefore, this predicate holds if and only if, for each $\alpha, \beta \in \{0, 1\}$, the index $i_{\alpha, \beta}$ is the smallest index i with the property that s_i lies on some shortest path from a_α to b_β . As before, for each $(a_0, a_1, b_0, b_1) \in A^2 \times B^2$, there exists a unique 4-tuple $\kappa \in [k]^4$ where $\Phi(\kappa; a_0, a_1, b_0, b_1)$ holds. For each $\kappa \in [k]^4$, each $(b_0, b_1) \in E_B$, and each type τ we define

$$\Lambda_\tau(\kappa; b_0, b_1) = \max\{\ell(W(a_0 a_1, b_0 b_1)) \mid (a_0, a_1) \in E_A \cap \text{Type}_\tau(b_0, b_1) \wedge \Phi(\kappa; a_0, a_1, b_0, b_1)\}.$$

This represents the maximum length over all edges (a_0, a_1) of a type- τ walk between (b_0, b_1) and (a_0, a_1) , consistent with the portals in κ . We next discuss how to compute efficiently $\Lambda_\tau(\kappa; b_0, b_1)$ for several edges $(b_0, b_1) \in E_B$ simultaneously.

► **Lemma 9.** *Consider a fixed type $\tau \in \{1, 2\}$ and indices $\kappa \in [k]^4$. In $O(m2^{4k-3}B(m, 4k-3))$ time we can compute the values $\Lambda_\tau(\kappa; b_0, b_1)$ for all $(b_0, b_1) \in E_B$.*

Proof sketch. We sketch how to use orthogonal range searching to compute $\Lambda_\tau(\kappa; b_0, b_1)$ for a fixed $\kappa \in [k]^4$. The complete proof is in [11].

For each vertex $a \in A$ and each $i \in [k]$, we define the point $p(i; a) \in \mathbb{R}^k$ whose j -th coordinate is $p_j(i; a) = d_G(a, s_i) - d_G(a, s_j)$. For each vertex $b \in B$ and each index $i \in [k]$, we define the box $R(i; b) = I_1(i; b) \times \dots \times I_k(i; b) \subset \mathbb{R}^k$, where $I_j(i; b)$ is the interval

$$I_j(i; b) = \begin{cases} (-\infty, d_G(b, s_j) - d_G(b, s_i)) & \text{if } j < i, \\ \mathbb{R} & \text{if } j = i, \\ (-\infty, d_G(b, s_j) - d_G(b, s_i)] & \text{if } j > i. \end{cases}$$

In $p(i; a)$ and $I(i; b)$ we remove the i -th coordinate, as it does not provide any information. We use the same notation for the resulting objects, now in $\mathbb{R}^{k-1}$. It has been noted [8, 10, 12] that the point $p(i; a)$ lies in the rectangle $R(i; b)$ if and only if $\varphi(i; a, b)$ holds.

We do something similar for $\Phi(\kappa; a_0, a_1, b_0, b_1)$. For any two vertices $a_0, a_1 \in A$, we define the point $p(a_0, a_1)$ and, for any two vertices $b_0, b_1 \in B$, we make a rectangle $R(b_0, b_1)$ in $\mathbb{R}^{4k-4}$, such that $p(a_0, a_1) \in R(b_0, b_1)$ if and only if $\Phi(\kappa; a_0, a_1, b_0, b_1)$ holds.

For each edge $(a_0, a_1) \in E_A$, we extend $p(a_0, a_1)$ to $p^+(a_0, a_1)$ by an extra coordinate to distinguish between type 1 and 2. We also introduce the new rectangle $R^+(b_0, b_1)$ such that

$$\forall (b_0, b_1) \in E_B : \Lambda_\tau(\kappa; b_0, b_1) = \max\{\ell(W(a_0 a_1, b_0 b_1)) \mid p^+(a_0, a_1) \in R^+(b_0, b_1)\}.$$

We use orthogonal range searching to handle the right side of the equation. In total, we spend $O(2^{4k-3}B(m, 4k-3))$ time per edge $(b_0, b_1) \in E_B$ to compute $\Lambda_\tau(\kappa; b_0, b_1)$. ◀

We use Lemma 9 to compute the values $\Lambda_\tau(\kappa; b_0, b_1)$ for each type $\tau \in \{1, 2\}$, all $(b_0, b_1) \in E_B$ and indices $\kappa \in [k]^4$. This requires applying Lemma 9 a total of $O(k^4)$ times, and therefore we spend $O(m2^{4k-3}k^4B(m, 4k-3))$ time. Using Lemma 8, we note that

$$\max_{p \in \mathcal{G}(E_A), q \in \mathcal{G}(E_B)} d_{\mathcal{G}}(p, q) = \frac{1}{2} \cdot \max \{ \Lambda_\tau(\kappa; b_0, b_1) \mid (b_0, b_1) \in E_B, \kappa \in [k]^4, \tau \in \{1, 2\} \}$$

(for details check the proof of Lemma 10 in [11]).

► **Lemma 10.** *We can compute $\max_{p \in \mathcal{G}(E_A), q \in \mathcal{G}(E_B)} d_{\mathcal{G}}(p, q)$ in $O(m2^{4k-3}k^4B(m, 4k-3))$ time.*

► **Remark.** We are aware that some log factors can be shaved off, and that for small k one can improve the analysis slightly. However, we prefer to keep this high-level structure to keep it simpler, and parallel to the forthcoming computation of mean distance.

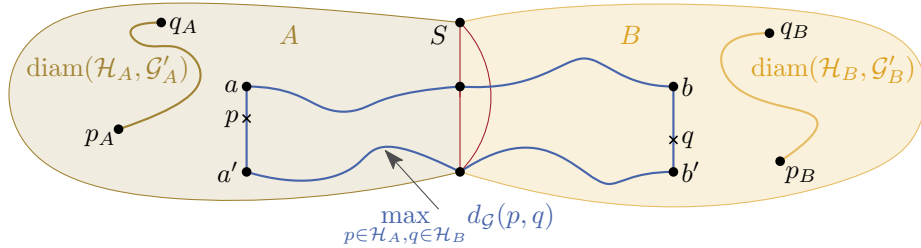
3.3 Global Diameter

Let G be a graph with n vertices and m edges defining the continuous graph $\mathcal{G}$. Let H be a subgraph of G defining a continuous graph $\mathcal{H} \subseteq \mathcal{G}$. We use a divide-and-conquer approach to compute $\text{diam}(\mathcal{H}, \mathcal{G})$.

Let (A, B, S) be a separation in G . Let G' be the graph obtained from G by adding an edge ss' with length $d_G(s, s')$ for every pair of portals $s, s' \in S$. (If the edge already exists, we redefine its length.) For $X \in \{A, B\}$, let $\mathcal{H}_X$ and $\mathcal{G}'_X$ be the continuous graphs defined by $H[X] - E(G[S])$ and $G'[X]$, respectively. The edges ss' added to G guarantee that $d_{\mathcal{G}}(p, q) = d_{\mathcal{G}'_X}(p, q)$ for any points $p, q \in \mathcal{H}_X$. (We removed $E(G[S])$ because for points on edges between portals whose length is redefined, the statement is undefined.) Therefore

$$\begin{aligned} \text{diam}(\mathcal{H}, \mathcal{G}) &= \max \left\{ \max_{p, q \in \mathcal{H}_A} d_{\mathcal{G}}(p, q), \max_{p, q \in \mathcal{H}_B} d_{\mathcal{G}}(p, q), \max_{p \in \mathcal{H}_A, q \in \mathcal{H}_B} d_{\mathcal{G}}(p, q) \right\} \\ &= \max \left\{ \max_{p, q \in \mathcal{H}_A} d_{\mathcal{G}'_A}(p, q), \max_{p, q \in \mathcal{H}_B} d_{\mathcal{G}'_B}(p, q), \max_{p \in \mathcal{H}_A, q \in \mathcal{H}_B} d_{\mathcal{G}}(p, q) \right\} \\ &= \max \left\{ \text{diam}(\mathcal{H}_A, \mathcal{G}'_A), \text{diam}(\mathcal{H}_B, \mathcal{G}'_B), \max_{p \in \mathcal{H}_A, q \in \mathcal{H}_B} d_{\mathcal{G}}(p, q) \right\} \end{aligned} \quad (1)$$

(see Figure 2). The last term can be computed by Lemma 10, which depends on the size of S .



■ **Figure 2** Visualization of the divide-and-conquer approach to compute $\text{diam}(\mathcal{G})$ (see Equation (1)).

Now we have two regimes depending on whether we want to assume that the treewidth is constant, as done in [12], or whether we want to consider the treewidth a parameter, as done in [8]. The same distinction was made in [10]. This difference affects the time to find a tree decomposition and the number of portals in a balanced separation. In both cases we use that an n -vertex graph with treewidth k has $O(kn)$ edges [6].

► **Theorem 11.** *Let $k \geq 2$ be an integer constant. Let G be a graph with n vertices, treewidth at most k , nonnegative edge-lengths, and let $\mathcal{G}$ be the corresponding continuous graph. Let H be a subgraph of G and let $\mathcal{H} \subseteq \mathcal{G}$ be the corresponding continuous subgraph. We can compute the diameter $\text{diam}(\mathcal{H}, \mathcal{G})$ in $O(n \log^{4k-2} n)$ time.*

Proof. If G has fewer than $2k = O(1)$ vertices, we compute $\text{diam}(\mathcal{H}, \mathcal{G})$ in $O(1)$ time. Otherwise, we find in linear time a separation (A, B, S) such that: $|S| \leq k$, $\frac{n}{k+1} \leq |A| \leq \frac{nk}{k+1}$. Such a separation is given, e.g., in [12]. Further, we add an edge ss' between all portals $s, s' \in S$ with length $d_G(s, s')$. This augmentation costs $O(k(m + n \log n)) = O(n \log n)$ time.

By Lemma 10, the value $\max_{p \in \mathcal{H}_A, q \in \mathcal{H}_B} d_{\mathcal{G}}(p, q)$ is computed in $O(m 2^{4k-3} k^4 B(m, 4k-3))$ time. Using that k is constant, $m = O(n)$, and Lemma 4, this time bound is $O(n \log^{4k-3} n)$.

We construct the graphs $G', G'[A], G'[B], H[A] - E(G[S])$ and $H[B] - E(G[S])$ explicitly, in $O(m) = O(n)$ time. Because adding edges between the portals of S does not increase the treewidth [12][Lemma 3], the graphs $G'[A]$ and $G'[B]$ have treewidth at most k . The values $\text{diam}(\mathcal{H}_A, \mathcal{G}'_A)$ and $\text{diam}(\mathcal{H}_B, \mathcal{G}'_B)$ are computed recursively, and we obtain $\text{diam}(\mathcal{H}, \mathcal{G})$ using Equation (1).

Since $\frac{n}{k+1} \leq |A| \leq \frac{nk}{k+1}$ and k is constant, each side of the recursion has a constant fraction of the vertices $|A| + |B| = n + k$, and the recursion depth is $O(\log n)$, leading to a total running time of $O(n \log^{4k-2} n)$. ◀

► **Theorem 12.** *Let G be a graph with n vertices, treewidth at most k , nonnegative edge-lengths, and let $\mathcal{G}$ be the corresponding continuous graph. Let H be a subgraph of G and let $\mathcal{H} \subseteq \mathcal{G}$ be the corresponding continuous subgraph. We can compute the diameter $\text{diam}(\mathcal{H}, \mathcal{G})$ in $n^{1+\varepsilon} 2^{O(k)}$ time, for any fixed $\varepsilon > 0$.*

Proof. We use the same divide-and-conquer strategy as in Theorem 11. The difference is in the properties of the separation. If G has $O(k)$ vertices, we compute $\text{diam}(\mathcal{H}, \mathcal{G})$ in $O(k^2)$ time. Otherwise, we proceed as follows.

First, we note that, given a tree decomposition of G of width k' , we can obtain in linear time a separation (A, B, S) in G with the following properties: the set S of portals for A has $k' + 1$ portals; both A and $B = (V(G) \setminus A)$ have $\Theta(n - k)$ vertices each; adding edges between the vertices of S does not increase the treewidth of the tree decomposition. See for example [6, Theorem 19]; the set S is a bag of the decomposition and thus the tree decomposition keeps being valid with the addition of edges within S .

It is shown in [7] that, for graphs of treewidth at most k , one can find a tree decomposition of width $k' = 3k + 4$ in $2^{O(k)} n \log n$ time. From this we obtain the separation (A, B, S) in G mentioned above, where $|S| \leq k' + 1 = 3k + 5$. By Lemma 10, the value $\max_{p \in \mathcal{H}_A, q \in \mathcal{H}_B} d_{\mathcal{G}}(p, q)$ is computed in $O(m 2^{O(k)} (k' + 1)^4 B(m, O(k)))$ time. Using that $m = O(kn)$ and the estimate of Lemma 4, this time bound becomes

$$O((kn) 2^{O(k)} (k' + 1)^4 \cdot B(n, O(k))) = O(n 2^{O(k)} \cdot n^\varepsilon 2^{O(k)}) = n^{1+\varepsilon} 2^{O(k)},$$

where $\varepsilon > 0$ can be chosen arbitrarily small.

To compute $\text{diam}(\mathcal{H}_A, \mathcal{G}'_A)$ and $\text{diam}(\mathcal{H}_B, \mathcal{G}'_B)$ recursively, we pass to the subproblems the tree decomposition we have computed, trimmed to the vertices of A and B , respectively. We also can adapt it to keep the tree decompositions of size $O(|A|)$ and $O(|B|)$, respectively. In this way, at any level of the recursion, we always have a set S with $k' + 1 = 3k + 5$ portals. Thus, we compute the tree decomposition only once, and then pass it to each subproblem trimmed to the relevant vertices. The recursive calls add a logarithmic factor to the total running time, which is absorbed by the polynomial term $n^{1+\varepsilon}$.

(The reason for passing the tree decomposition to the subproblems is that adding the edges between the portals in S may give a clique of size k' , which increases the treewidth of G' . Computing an approximate tree decomposition of G' anew would increase the width of the decomposition at each level. However, if we pass the tree decomposition to the subproblems, we keep the width of the decomposition bounded by $3k + 5$ at all levels of the recursion.) ◀

4 Mean Distance in Graphs with Bounded Treewidth

In this section, we discuss the computation of the mean distance in continuous graphs with treewidth k . The case of $k = 1$ corresponds to continuous trees, one of the few graph classes for which linear-time algorithms are known [21]. Thus, we focus on $k > 1$. All our efforts will be on the computation of $\text{sumdist}(\mathcal{H}, \mathcal{G})$, from which $\text{mean}(\mathcal{H}, \mathcal{G})$ can be easily computed. Again, we use orthogonal range searching, but now we need to efficiently retrieve sums of distances. We achieve this by representing the sum of distances between pairs of edges as volumes of collections of triangular prisms, which can be represented in a compact way.

4.1 Mean Distance as a Volume

We compute the sum of all distances in a continuous graph as the volume of a collection of triangular prisms. A *truncated triangular prism* is formed when a prism is sliced by a plane that is not parallel to its bases; the heights are the lengths of the three edges orthogonal to the basis. The volume of a truncated triangular prism with base $\triangle$ and heights h_1, h_2, h_3 is $\frac{1}{3} \text{area}(\triangle)(h_1 + h_2 + h_3)$.

Consider the following setting defined by a complete graph with vertex set $\{a_0, a_1, b_0, b_1\}$ and variable edge lengths, as follows: (i) a_0a_1 has variable length $y > 0$, (ii) b_0b_1 has variable length $z > 0$, (iii) for all $\alpha, \beta \in \{0, 1\}$, $a_\alpha b_\beta$ has length $x_{\alpha, \beta} \geq 0$.

Let $K = K(y, z, x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1})$ denote this graph, and let $\mathcal{K}$ denote the corresponding continuous graph. See Figure 3. We say that the 6-tuple $(y, z, x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1})$ is *compliant* if, for all $\alpha, \beta \in \{0, 1\}$ it holds $x_{\alpha, \beta} = d_K(a_\alpha, b_\beta)$. In our setting we only need to consider compliant cases. (This poses some conditions on the values that the variables can take.)

We want to understand how the total sum of distances between points on a_0a_1 and b_0b_1 ,

$$\xi(y, z, x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1}) = \iint_{p \in \mathcal{K}(a_0a_1), q \in \mathcal{K}(b_0b_1)} d_{\mathcal{K}}(p, q) dp dq,$$

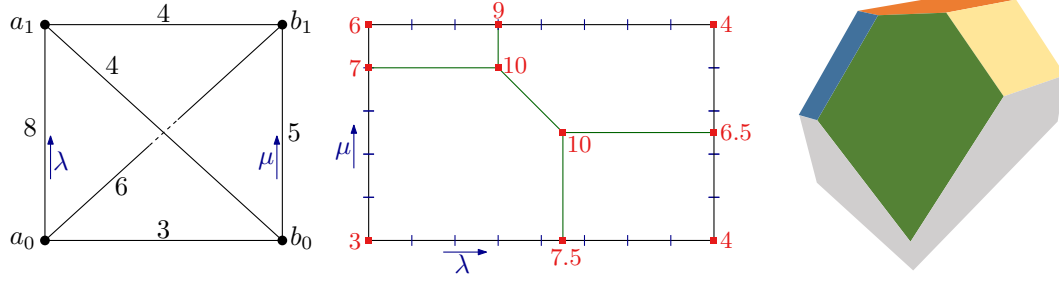
looks like. For each $\lambda \in [0, y]$, let $p(\lambda)$ be the point specified by the triple (a_0a_1, a_0, λ) , and, for each $\mu \in [0, z]$, let $q(\mu)$ be the point specified by the triple (b_0b_1, b_0, μ) . Then

$$\xi(y, z, x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1}) = \iint_{(\lambda, \mu) \in [0, y] \times [0, z]} d_{\mathcal{K}}(p(\lambda), q(\mu)) d\lambda d\mu.$$

Function $(\lambda, \mu) \mapsto d_{\mathcal{K}}(p(\lambda), q(\mu))$, defined in $[0, y] \times [0, z]$, is the lower envelope of four functions

$$x_{0,0} + \lambda + \mu, \quad x_{0,1} + \lambda + z - \mu, \quad x_{1,0} + y - \lambda + \mu, \quad x_{1,1} + y - \lambda + z - \mu.$$

Following [21], we call the graph of function $(\lambda, \mu) \mapsto d_{\mathcal{K}}(p(\lambda), q(\mu))$ a *roof*; the value ξ is the volume below the roof. The minimization diagram of this function consists of convex pieces; see Figure 3. The gradient of each function is of the form $(\pm 1, \pm 1)$. When the variable values are compliant, all four functions appear in the lower envelope, and the minimization diagram has four regions. (Some of them may contain only part of an edge of the domain.)



■ **Figure 3** Concrete example of the minimization diagram of $d_K(p(\lambda), q(\mu))$. In the center, some values of $d_K(p(\lambda), q(\mu))$ are shown in red; on the right, 3D visualization of the roofs.

► **Lemma 13.** Consider the 4-variable linear function $L(x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1}) = x_{1,0} + x_{0,1} - x_{0,0} - x_{1,1}$. There are two 6-variable polynomials $\varrho_+(\cdot), \varrho_-(\cdot)$ of degree at most three with the following property: when $(y, z, x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1})$ is compliant,

$$\xi(y, z, x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1}) = \begin{cases} \varrho_+(y, z, x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1}), & \text{if } L(x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1}) \geq 0, \\ \varrho_-(y, z, x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1}), & \text{otherwise.} \end{cases}$$

4.2 Integral Across the Portals

We reuse much of the notation and ideas from Section 3.2. As before, G is a graph with n vertices and m edges, (A, B, S) is a separation in G with exactly k portals, and we fix an enumeration of the portals as $s_1, \dots, s_k$. We also fix an orientation for the edges of G . Let $E_A \subseteq E(G[A])$ be the edge sets of the subgraph of G induced by A and $E_B \subseteq E(G[B])$, respectively; not necessarily disjoint. Our objective in this section is to compute the sum of distances between points in E_A and E_B :

$$\iint_{p \in \mathcal{G}(E_A), q \in \mathcal{G}(E_B)} d_G(p, q) dp dq = \sum_{(a_0, a_1) \in E_A} \sum_{(b_0, b_1) \in E_B} \iint_{p \in \mathcal{G}(a_0 a_1), q \in \mathcal{G}(b_0 b_1)} d_G(p, q) dp dq.$$

Define the graph $\mathcal{K} = \mathcal{K}(\ell(a_0 a_1), \ell(b_0 b_1), d_G(a_0, b_0), d_G(a_0, b_1), d_G(a_1, b_0), d_G(a_1, b_1))$, for two edges $(a_0, a_1) \in E_A, (b_0, b_1) \in E_B$.

The continuous edges $a_0 a_1$ and $b_0 b_1$ belong to $\mathcal{G}$ and $\mathcal{K}$ and, for each $p \in \mathcal{G}(a_0 a_1)$ and $q \in \mathcal{G}(b_0 b_1)$ we have $d_G(p, q) = d_K(p, q)$. It follows that

$$\iint_{p \in \mathcal{G}(a_0 a_1), q \in \mathcal{G}(b_0 b_1)} d_G(p, q) dp dq = \iint_{p \in \mathcal{K}(a_0 a_1), q \in \mathcal{K}(b_0 b_1)} d_K(p, q) dp dq.$$

It follows that our objective is to compute

$$\sum_{(a_0, a_1) \in E_A} \sum_{(b_0, b_1) \in E_B} \xi(\ell(a_0 a_1), \ell(b_0 b_1), d_G(a_0, b_0), d_G(a_0, b_1), d_G(a_1, b_0), d_G(a_1, b_1)).$$

We keep using the predicates $\varphi(i; a, b)$ and $\Phi(\kappa; a_0, a_1, b_0, b_1)$ defined in Section 3.2, where $a, a_0, a_1 \in A, b, b_0, b_1 \in B, i \in [k]$ and $\kappa = (i_{0,0}, i_{0,1}, i_{1,0}, i_{1,1}) \in [k]^4$.

Consider the polynomial L of Lemma 13. For each pair $(a_0, a_1), (b_0, b_1) \in E(G)$, we have

$$(a_0, a_1, b_0, b_1) \text{ of type 1} \iff L(d_G(a_0, b_0), d_G(a_0, b_1), d_G(a_1, b_0), d_G(a_1, b_1)) \geq 0.$$

13:12 Algorithms for Distance Problems in Continuous Graphs

Otherwise, (a_0, a_1, b_0, b_1) is of *type 2*. For each oriented edge $(b_0, b_1) \in E(G)$ and type $\tau \in \{1, 2\}$, we define

$$\text{Type}_\tau(b_0, b_1) = \{(a_0, a_1) \in E(G) \mid (a_0, a_1, b_0, b_1) \text{ of type } \tau\}. \quad (2)$$

For each $\kappa \in [k]^4$, each $(b_0, b_1) \in E_B$, and each type $\tau \in \{1, 2\}$ we define

$$\Psi_\tau(\kappa; b_0, b_1) = \sum \xi(\ell(a_0 a_1), \ell(b_0 b_1), d_G(a_0, b_0), d_G(a_0, b_1), d_G(a_1, b_0), d_G(a_1, b_1)), \quad (3)$$

where the sum ranges over all oriented edges $(a_0, a_1) \in E_A$ such that $(a_0, a_1) \in \text{Type}_\tau(b_0, b_1)$ and $\Phi(\kappa; a_0, a_1, b_0, b_1)$ holds.

Next, we show that we can efficiently compute $\Psi_\tau(\kappa; b_0, b_1)$ for all edges $(b_0, b_1) \in E_B$:

► **Lemma 14.** *Consider a fixed type $\tau \in \{1, 2\}$ and indices $\kappa \in [k]^4$. We can compute the values $\Psi_\tau(\kappa; b_0, b_1)$ for all $(b_0, b_1) \in E_B$ in $O(m2^{4k-3}B(m, 4k-3))$ time.*

► **Lemma 15.** *We can compute $\iint_{p \in \mathcal{G}(E_A), q \in \mathcal{G}(E_B)} d_{\mathcal{G}}(p, q) dp dq$ in $O(m2^{4k-3}k^4B(m, 4k-3))$ time.*

4.3 Global Mean

Let G be a graph with n vertices and m edges defining the continuous graph $\mathcal{G}$. Let H be a subgraph of G defining a continuous graph $\mathcal{H} \subseteq \mathcal{G}$. We use a divide-and-conquer approach to compute $\text{sumdist}(\mathcal{H}, \mathcal{G})$. The approach is very similar to that in Section 3.3 for the diameter, and thus we only emphasize the differences.

Let (A, B, S) be a separation in G . We use the notation defined before Theorem 11 introducing for $X \in \{A, B\}$, and the graphs $\mathcal{H}_X$ and $\mathcal{G}'_X$. We have

$$\begin{aligned} \text{sumdist}(\mathcal{H}, \mathcal{G}) = & \iint_{p \in \mathcal{G}(E_A), q \in \mathcal{G}(E_B)} d_{\mathcal{G}}(p, q) dp dq - \iint_{p, q \in \mathcal{G}(H[S])} d_{\mathcal{G}}(p, q) dp dq \\ & + \text{sumdist}(\mathcal{H}_A, \mathcal{G}'_A) + \text{sumdist}(\mathcal{H}_B, \mathcal{G}'_B). \end{aligned} \quad (4)$$

The first term can be computed using Lemma 15, which depends on $|S|$. The second term can be computed in $O(km \log n + k^2)$ time because, after computing the distances from S , it is a problem of size $O(k^2)$. Dividing $\text{sumdist}(\mathcal{H}, \mathcal{G})$ by $\ell(\mathcal{H})^2$, we obtain $\text{mean}(\mathcal{H}, \mathcal{G})$. As in Section 3.3, we have two regimes.

► **Theorem 16.** *Let $k \geq 2$ be an integer constant. Let G be a graph with n vertices, treewidth at most k , nonnegative edge-lengths, and let $\mathcal{G}$ be the corresponding continuous graph. Let H be a subgraph of G and let $\mathcal{H} \subseteq \mathcal{G}$ be the corresponding continuous subgraph. We can compute $\text{mean}(\mathcal{H}, \mathcal{G})$ in $O(n \log^{4k-2} n)$ time.*

► **Theorem 17.** *Let G be a graph with n vertices, treewidth at most k , nonnegative edge-lengths, and let $\mathcal{G}$ be the corresponding continuous graph. Let H be a subgraph of G and let $\mathcal{H} \subseteq \mathcal{G}$ be the corresponding continuous subgraph. We can compute $\text{mean}(\mathcal{H}, \mathcal{G})$ in $n^{1+\varepsilon}2^{O(k)}$ time, for any fixed $\varepsilon > 0$.*

5 Conclusion

We presented the first subquadratic algorithms to compute the diameter and the mean distance in continuous graphs, for two non-trivial graph classes. We expect that the approach for graphs parameterized by the treewidth can be adapted for computing other statistics defined by the distance between two points selected at random in a continuous subgraph $\mathcal{H} \subseteq \mathcal{G}$, like a cumulative density function (CDF) and higher moments:

for given δ , compute $\text{CDF}(\delta, \mathcal{H}, \mathcal{G}) = \frac{1}{\ell(\mathcal{H})^2} \iint_{p,q \in \mathcal{H}} \mathbb{1}[d_{\mathcal{G}}(p, q) \leq \delta] dp dq$,

median distance: $\sup \{ \delta \in \mathbb{R}_{\geq 0} \mid \text{CDF}(\delta, \mathcal{H}, \mathcal{G}) \leq 1/2 \}$,

higher moments, such as $\frac{1}{\ell(\mathcal{H})^2} \iint_{p,q \in \mathcal{H}} (d_{\mathcal{G}}(p, q))^2 dp dq$.

The main open question stemming from our work is whether our approach can be adapted to work in subquadratic time for arbitrary planar graphs. However, as already mentioned in the introduction, this requires dynamic trees with a set of suitable operations that – at the moment – seem to be out of reach.

References

- 1 A. Abboud, V. Vassilevska Williams, and J. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proc. 27th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 377–391. SIAM, 2016. doi:10.1137/1.9781611974331.ch28.
- 2 S. Won Bae, M. De Berg, O. Cheong, J. Gudmundsson, and C. Levcopoulos. Shortcuts for the circle. *Comput. Geom. Theory Appl.*, 79:37–54, 2019. doi:10.1016/J.COMGEO.2019.01.006.
- 3 L. N. Baptista, J. B. Kennedy, and D. Mugnolo. Mean distance on metric graphs. *The Journal of Geometric Analysis*, 34(137):1–25, 2024. doi:10.1007/s12220-024-01574-0.
- 4 P. Bergé, G. Ducoffe, and M. Habib. Subquadratic-time algorithm for the diameter and all eccentricities on median graphs. *Theory of Computing Systems*, 68(1):144–193, 2024. doi:10.1007/S00224-023-10153-9.
- 5 P. Bergé, G. Ducoffe, and M. Habib. Quasilinear-time eccentricities computation, and more, on median graphs. In *Proc. 36th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 1679–1704. SIAM, 2025. doi:10.1137/1.9781611978322.52.
- 6 H. L. Bodlaender. A partial k -arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.
- 7 H. L. Bodlaender, P. Grønås Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, and M. Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016. doi:10.1137/130947374.
- 8 K. Bringmann, T. Husfeldt, and M. Magnusson. Multivariate analysis of orthogonal range searching and graph distances. *Algorithmica*, 82:2292–2315, 2020. doi:10.1007/S00453-020-00680-Z.
- 9 S. Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Trans. Algorithms*, 15(2):21:1–21:38, 2019. doi:10.1145/3218821.
- 10 S. Cabello. Computing the inverse geodesic length in planar graphs and graphs of bounded treewidth. *ACM Trans. Algorithms*, 18(2):14:1–14:26, 2022. doi:10.1145/3501303.
- 11 S. Cabello, D. Garijo, A. Kalb, F. Klute, I. Parada, and R. I. Silveira. Algorithms for distance problems in continuous graphs, 2025. arXiv:2503.07769.
- 12 S. Cabello and C. Knauer. Algorithms for graphs of bounded treewidth via orthogonal range searching. *Comput. Geom. Theory Appl.*, 42(9):815–824, 2009. doi:10.1016/j.comgeo.2009.02.001.

- 13 J. Cáceres, D. Garijo, A. González, A. Márquez, M. L. Puertas, and P. Ribeiro. Shortcut sets for the locus of plane Euclidean networks. *Applied Mathematics and Computation*, 334:192–205, 2018. doi:10.1016/j.amc.2018.04.010.
- 14 J.-L. De Carufel, C. Grimm, A. Maheshwari, S. Schirra, and M. H. M. Smid. Minimizing the continuous diameter when augmenting a geometric tree with a shortcut. *Comput. Geom. Theory Appl.*, 89:101631, 2020. doi:10.1016/J.COMGEO.2020.101631.
- 15 J.-L. De Carufel, C. Grimm, A. Maheshwari, and M. H. M. Smid. Minimizing the continuous diameter when augmenting paths and cycles with shortcuts. In *Proc. 15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT’16)*, volume 53 of *LIPIcs*, pages 27:1–27:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.SWAT.2016.27.
- 16 C. E. Chen and R. S. Garfinkel. The generalized diameter of a graph. *Networks*, 12(3):335–340, 1982. doi:10.1002/net.3230120310.
- 17 G. Ducoffe. Eccentricity queries and beyond using hub labels. *Theoretical Computer Science*, 930(21):128–141, 2022. doi:10.1016/j.tcs.2022.07.017.
- 18 G. Ducoffe, M. Habib, and L. Viennot. Diameter, eccentricities and distance oracle computations on h -minor free graphs and graphs of bounded (distance) vapnik–chervonenkis dimension. *SIAM Journal on Computing*, 51(5):1506–1534, 2022. doi:10.1137/20m136551x.
- 19 L. Friedlander. Genericity of simple eigenvalues for a metric graph. *Israel Journal of Mathematics*, 146:149–156, 2005. doi:10.1007/BF02773531.
- 20 D. Garijo, A. Márquez, N. Rodríguez, and R. I. Silveira. Computing optimal shortcuts for networks. *Eur. J. Oper. Res.*, 279(1):26–37, 2019. doi:10.1016/j.ejor.2019.05.018.
- 21 D. Garijo, A. Márquez, and R. I. Silveira. Continuous mean distance of a weighted graph. *Results in Mathematics*, 78(139):1–36, 2023. doi:10.1007/s00025-023-01902-w.
- 22 P. Gawrychowski, H. Kaplan, S. Mozes, M. Sharir, and O. Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $O(n^{5/3})$ time. *SIAM J. Comput.*, 50(2):509–554, 2021. doi:10.1137/18M1193402.
- 23 J. Gudmundsson and Y. Sha. Augmenting graphs to minimize the radius. *Comput. Geom. Theory Appl.*, 114(101996):1–14, 2023. doi:10.1016/j.comgeo.2023.101996.
- 24 J. Gudmundsson and S. Wong. Improving the dilation of a metric graph by adding edges. *ACM Trans. Algorithms*, 18(3-Article 20):1–14, 2022. doi:10.1145/3517807.
- 25 S. L. Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12(3):450–459, 1964.
- 26 H. Le and C. Wulff-Nilsen. Vc set systems in minor-free (di)graphs and applications. In *Proc. 35th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 5332–5360. SIAM, 2024. doi:10.1137/1.9781611977912.192.
- 27 G. Lumer. Connecting of local operators and evolution equations on networks. In *Potential Theory Copenhagen 1979*, pages 219–234. Springer-Verlag, 1980. doi:10.1007/BFb0086338.
- 28 L. Monier. Combinatorial solutions of multidimensional divide-and-conquer recurrences. *Algorithms*, 1(1):60–74, 1980. doi:10.1016/0196-6774(80)90005-X.
- 29 L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proc. 45th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 515–524. ACM, 2013. doi:10.1145/2488608.2488673.