Isometric Embeddings in Trees and Their Use in Distance Problems

Guillaume Ducoffe
National Institute of Research and Development in Informatics, Bucharest, Romania
University of Bucharest, Romania

Abstract

We present powerful techniques for computing the diameter, all the eccentricities, and other related distance problems on some geometric graph classes, by exploiting their "tree-likeness" properties. We illustrate the usefulness of our approach as follows:

- We propose a subquadratic-time algorithm for computing all eccentricities on partial cubes of bounded lattice dimension and isometric dimension $\mathcal{O}(n^{0.5-\varepsilon})$. This is one of the first positive results achieved for the diameter problem on a subclass of partial cubes beyond median graphs.
- Then, we obtain almost linear-time algorithms for computing all eccentricities in some classes of face-regular plane graphs, including benzenoid systems, with applications to chemistry. Previously, only a linear-time algorithm for computing the diameter and the center was known (and an $\tilde{\mathcal{O}}(n^{5/3})$-time[1] algorithm for computing all the eccentricities).
- We also present an almost linear-time algorithm for computing the eccentricities in a polygon graph with an additive one-sided error of at most 2.
- Finally, on any cube-free median graph, we can compute its absolute center in almost linear time. Independently from this work, Bergé and Habib have recently presented a linear-time algorithm for computing all eccentricities in this graph class (*LAGOS'21*), which also implies a linear-time algorithm for the absolute center problem.

Our strategy here consists in exploiting the existence of some embeddings of these graphs in either a system or a product of trees, or in a single tree but where each vertex of the graph is embedded in a subset of nodes. While this may look like a natural idea, the way it can be done efficiently, which is our main technical contribution in the paper, is surprisingly intricate.

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

Keywords and phrases Tree embeddings, Range queries, Centroid decomposition, Heavy-path decomposition, Diameter, Radius and all Eccentricities computations

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.43

Funding *Guillaume Ducoffe*: This work was supported by project PN-19-37-04-01 "New solutions for complex problems in current ICT research fields based on modelling and optimization", funded by the Romanian Core Program of the Ministry of Research and Innovation (MCI) 2019–2022.

1 Introduction

This paper is about classic location problems on graphs and metric spaces. Although we focus on unweighted undirected graphs in what follows, it should be clear to the reader that most of our results could also be applied to discrete metric spaces. For standard graph terminology, see [12, 33]. The distance in a graph $G = (V, E)$ between two vertices $u, v \in V$ equals the minimum number of edges in a uv-path, and is denoted $d_G(u, v)$. For any vertex u in G, let $e_G(u) := \max_{v \in V} d_G(u, v)$ be its eccentricity. The diameter and the radius of G are the maximum and minimum eccentricities of a vertex. We denote the former and the latter

[1] The $\tilde{\mathcal{O}}(\cdot)$ notation suppresses poly-logarithmic factors.

by $diam(G)$ and $rad(G)$, respectively. The center of G contains all vertices with minimum eccentricity. We study the problem of computing all eccentricities in Sec. 2. Finally, to any graph G we can associate a continuous metric space (X_G, d): obtained from G by replacing every edge with a unit-length geodesic. Motivated by the problem of locating the "switching center" in a communication network, Hakimi's absolute center problem consists in, being given G, computing all points of X_G that minimize their largest distance to a vertex of G [59]. We investigate this problem in Sec. 5.

Related work. There is a naive algorithm for computing all the eccentricities, resp. the absolute center, in an n-vertex m-edge graph in total $\mathcal{O}(nm)$ time. Conversely, assuming the so-called Strong-Exponential-Time Hypothesis (SETH), if an algorithm computes all the eccentricities in a graph, or even just the diameter, in $\mathcal{O}(n^a m^b)$ time, then we must have $a+b \geq 2$ [70]. In what follows, by *truly subquadratic* we mean $\mathcal{O}(n^a m^b)$ time where $a+b<2$. The problem of finding truly subquadratic algorithms for the diameter problem on some special graph classes has been addressed in many papers [1, 14, 15, 17, 18, 23, 30, 31, 42, 46, 47, 45, 51, 55, 69]. There are comparatively much fewer results for the absolute center problem, *e.g.*, see [60, 63, 65] for previous results on trees and cacti.

As it is often the case, such problems become much easier for the graphs with a suitable "tree-like" representation such as: bounded tree-width graphs [17] and bounded clique-width graphs [44]. In the context of Metric Graph Theory (hereafter called MGT), one natural way to define tree-likeness of a graph is by using embeddings in trees. Recall that, if (X, d_X) and (Y, d_Y) are metric spaces, then an embedding is simply an injective function $\varphi : X \to Y$. Its distortion is the least $\alpha \geq 1$ s.t., for all $x, x' \in X$, we have $\alpha^{-1} d_X(x,x') \leq d_Y(\varphi(x), \varphi(x')) \leq \alpha d_X(x,x')$. The stretch is the least $\beta \geq 0$ s.t., for all $x, x' \in X$, we have $|d_X(x,x') - d_Y(\varphi(x), \varphi(x'))| \leq \beta$. An isometric embedding is one s.t. $\alpha = 1$, or equivalently $\beta = 0$. A quasi isometric embedding is one such that $\alpha = \mathcal{O}(1)$, or even better $\beta = \mathcal{O}(1)$. If we are given an isometric embedding of a graph in a tree (resp., a quasi isometric embedding), then we can solve exactly (resp., approximately) the diameter problem in linear time. Unfortunately, the graph classes to which this nice result can be applied are rather restricted. *E.g.*, the graphs that can be *isometrically* embedded in a *weighted* tree are exactly the block graphs [4, 61]. See also [5, 9] for an efficient recognition of tree metrics. More generally, all the metric spaces that embed in a tree with constant distortion have a bounded Gromov hyperbolicity: a polynomial-time computable parameter from geometric group theory that is inspired by the four-point characterization of tree metrics [58]. While many real-life networks are known to have bounded hyperbolicity [2], this is *not* the case for important classes in MGT such as: *median graphs* (even of dimension at most two), *Helly graphs* (even of strong isometric dimension at most two) and ℓ_1*-graphs*. We refer to [3] for their respective definitions (median graphs are defined in Sec. 5).

Thus, we need to consider stronger notions of tree embeddings, or embeddings in more complicated "tree-like" spaces. Recall that the *Cartesian product* of graphs G_1, G_2, denoted by $G_1 \square G_2$, has vertex-set $V(G_1) \times V(G_2)$ and edge-set $\{(u_1,u_2)(u_1,v_2) \mid u_2v_2 \in E(G_2)\} \cup \{(u_1,u_2)(v_1,u_2) \mid u_1v_1 \in E(G_1)\}$. In [21], Chepoi studied the benzenoid systems: a subclass of planar graphs with applications to chemistry, and he proved that they can be isometrically embedded in linear time in the Cartesian product of three trees. In a subsequent work [23], Chepoi et al. used this nice property in order to compute the diameter, and even the center, of these graphs in linear time. However, they also needed certain "total monotonic" property to hold for the distance-matrix of the graphs considered. To our best knowledge, the problem of computing all the eccentricities within benzenoid systems in almost linear time has been open until our work. Furthermore, we point out that the existence of similar embeddings as the one in [23], in a product or a system of constantly many trees, has been thoroughly

investigated for many graph classes [8, 6, 7, 25, 20, 35, 36, 37, 38, 39, 40, 41, 48, 68]. While it is NP-hard in general to decide whether such an embedding exists [8], we note that several of these above previous works have presented almost linear-time algorithms for this problem on some special cases.

Our results. We first prove that, for any fixed k, we can compute all eccentricities in a graph in quasi linear time if it is isometrically embedded in the Cartesian product of k trees (Theorem 1). This improves on Chepoi et al. [23] since we needn't any additional assumption on the distance-matrix and we solve a more general problem than just computing the diameter or the center. We apply this very general result to the following geometric graph classes:

- the triangular systems and hexagonal systems (*a.k.a.*, benzenoids), see Sec. 2.2;
- and the graphs of bounded lattice dimension, that can be isometrically embedded in the product of constantly many paths (with an additional technical assumption needed for computing the embedding), see Sec. 2.1.

Our actual result for Theorem 1 is even more general than what we stated above, since it can also be applied to other types of tree embeddings. If the embedding is not isometric but it has $\mathcal{O}(1)$ stretch or distortion, then our above approach leads to approximation algorithms for computing all the eccentricities. In particular,

- we prove that for any fixed k, all eccentricities in a k-polygon graph can be approximated in almost linear time within an additive one-sided error of at most 2, see Sec. 2.3.

In the second part of the paper, we consider a different type of tree embedding: from a graph G to a single tree T, but where each vertex u of G is mapped to a *subset* of nodes $U \subseteq V(T)$. This is similar to the notion of clan embedding, introduced in [52]. It leads us to solve a more general eccentricity problem on trees, that may be of independent interest for k-facility location problems [64, 73]. This is our second main technical contribution in the paper. More precisely, given an n-node tree T, the eccentricity of a node-subset U is defined as $e_T(U) = \max_{v \in V(T)} \min_{u \in U} d_T(v, u)$.

- We prove that after an $\mathcal{O}(n \log n)$-time pre-processing, the eccentricity of any node-subset U can be computed in $\mathcal{O}(|U| \log^2 n)$ query time. See Theorem 9. These running times are optimal up to polylogarithmic factors.

Let $k \in \mathbb{N}$ and $\varepsilon \in (0; 1)$ be arbitrary. We observe that, combined to Theorem 1 in [52], our result implies a data structure for computing, for any graph G and any vertex-subset U (up to pre-processing), a $16k$-approximation of $e_G(U)$ (resp., an $\tilde{\mathcal{O}}\left(\frac{\log n}{\varepsilon}\right)$-approximation) in expected $\tilde{\mathcal{O}}(|U| n^{1/k})$ time (resp., in expected $\tilde{\mathcal{O}}((1+\varepsilon) \cdot |U|)$ time). See Sec. 5.1.

We give another application of Theorem 9, in the context of distance and routing labelling schemes. Indeed, a successful approach for computing such schemes with small label sizes on geometric graph classes is as follows. Roughly, the vertices of a graph get partitioned recursively into convex subgraphs. At each step, the vertices of such convex subgraphs are projected to their respective boundary, so as to hit all shortest-paths to vertices in nearby convex subgraphs of the decomposition. Then, by mimicking the recursive construction of the labels, if the projections to the boundaries have some nice properties then, it becomes possible to compute all eccentricities, as well as to solve other related distance problems. It turns out that such embeddings are often *tree-like*, in the sense of either Theorem 1 or Theorem 9. We illustrate this with the case of cube-free median graphs (formally defined in Sec. 5). The cube-free median graphs properly contain the partial double trees (which can be isometrically embedded in the Cartesian product of two trees), and if their maximum degree is Δ then, they can be isometrically embedded in the Cartesian product of $\Delta^{\mathcal{O}(1)}$

trees [25]. However, in general they cannot be embedded in the product of constantly many trees, and so, Theorem 1 cannot be applied. We overcome this issue by relying on a recursive partition scheme as above, where the boundaries induce *isometric trees*, see [26]. Doing so,

- we present the first almost linear-time algorithm for computing the absolute center of cube-free median graphs, see Theorem 10. Our approach can also be used in order to compute all eccentricities in this graph class.

Recently, Bergé and Habib have presented a linear-time algorithm for computing all eccentricities within the median graphs of bounded dimension (generalizing cube-free median graphs) [11]. A reviewer observed that the absolute center of a median graph is always: either its center if it is an independent set, or the middle points of all edges with their both ends in the center. As a result, we can also compute the absolute center of cube-free median graphs in linear time. That being said, we think that our alternative algorithm, although it is slightly slower, has the potential to be generalized to other graph classes beyond the median graphs. Indeed, to prove Theorem 10, we use some deep structural properties obtained recently for cube-free median graphs [26]. We point out that very similar properties have been obtained in the past for completely unrelated graph classes, such as planar graphs of non-positive combinatorial curvature [24, 29, 28]. Thus, there is room for generalizing our approach far beyond the cube-free median graphs.

Since both Theorems 1 and 9 are very general, we expect them to find applications beyond the classes studied in the paper.

Organization of the paper. In Sec. 2 we state Theorem 1, then we summarize our results obtained for subclasses of partial cubes, planar graphs and circle graphs. We postpone the proof of Theorem 1 to Sec. 3, due to its technicality. Our algorithms in Sec. 3 have an exponential dependency on the number k of trees considered. In Sec. 4, we give simple *conditional lower bounds* showing their near optimality. Finally, in Sec. 5, we address the problem of computing the eccentricity of k-subsets of nodes in a tree, and its application to the absolute center problem for cube-free median graphs.

Due to lack of space, several proofs are omitted from the following technical sections.

2 Eccentricity computation in some geometric graph classes

This section is devoted to the applications of our Theorem 1, which we formally state in what follows. We defined the Cartesian product of two graphs in Sec. 1. The *strong product* of graphs G_1, G_2, denoted by $G_1 \boxtimes G_2$, is a supergraph of the Cartesian product with additional edge-set $\{(u_1,u_2)(v_1,v_2) \mid u_1v_1 \in E(G_1),\ u_2v_2 \in E(G_2)\}$. Finally, an embedding of a graph $G=(V,E)$ in a *system* of trees $T_1, T_2, \ldots, T_k$ is defined as k projections $\varphi_i : V \to V(T_i),\ 1 \le i \le k$. Then, the distortion of this embedding (resp. its stretch) is defined as the least α s.t., $\forall x,y \in X,\ \alpha^{-1}d(x,y) \le \min_i d_{T_i}(\varphi_i(x),\varphi_i(y)) \le \alpha d(x,y)$ (resp., as the least β s.t. $\forall x,y \in X,\ |d(x,y) - \min_i d_{T_i}(\varphi_i(x),\varphi_i(y))| \le \beta$). We consider quasi-isometric embeddings of graphs and metric spaces in: Cartesian products of trees, strong products of trees, and systems of trees.

▶ **Theorem 1.** *Let $G=(V,E)$ be a graph and let $T_1, T_2, \ldots, T_k$ be a collection of k trees, where $N := \sum_{i=1}^{k} |V(T_i)|$.*

1. *If we are given an isometric embedding of G in either the system or the Cartesian product of these k trees, then we can compute all eccentricities in G in $\mathcal{O}(2^{\mathcal{O}(k \log k)}(N+n)^{1+o(1)})$ time*[2].

[2] See [17, Lemma 5] for a more careful analysis of the running time.

2. *If we are given an embedding of G with distortion α (resp., with stretch β) in either the system or the Cartesian product of these k trees, then we can compute an α^2-approximation (resp., an $+2\beta$-approximation) of all eccentricities in G in $\mathcal{O}(2^{\mathcal{O}(k \log k)}(N+n)^{1+o(1)})$ time. It can be improved to an α-approximation (resp., a $+\beta$-approximation) if the embedding only has one-sided distance errors.*
3. *All the results above also hold if we are given an embedding of G in the strong product of these k trees, with an improved runtime in $\mathcal{O}(N+kn)$.*

See Sec. 3 for a sketch of the proof. We left open whether our approach could be generalized to, say, the direct product or the layered cross product [50] of constantly many trees. In what follows, we apply Theorem 1 to several geometric graph classes.

2.1 Partial cubes

The lattice dimension of a graph is the smallest k such that it isometrically embeds in the Cartesian product of k paths. This parameter only exists for partial cubes, *a.k.a.*, isometric subgraphs of hypercubes (where, by a hypercube, we mean a Cartesian product of edges). The isometric dimension of a partial cube is the least τ such that it isometrically embeds in the τ-dimensional hypercube.

▶ **Theorem 2.** *All eccentricities in an n-vertex partial cube with lattice dimension k and isometric dimension τ can be computed in $\mathcal{O}((\tau^2 + 2^{\mathcal{O}(k \log k)}) \cdot n^{1+o(1)})$ time.*

This is truly subquadratic provided $k = o(\log n)$ and $\tau = \mathcal{O}(n^{0.5-\varepsilon})$, for some $\varepsilon > 0$.

Proof. Let $G = (V, E)$ be a partial cube. It is well-known that E can be partitioned in so-called θ-classes, where each class represents a dimension of the smallest hypercube in which G can be isometrically embedded [34, 75]. Furthermore, given an edge, we can compute its θ-class in linear time (see Sec. 3 in [49]). Therefore, we can isometrically embed G in a smallest hypercube in $\tilde{\mathcal{O}}(\tau n)$ (here, we implicitly use that G only has $\tilde{\mathcal{O}}(n)$ edges, see Lemma 2 in [49]). Being given such an embedding, Eppstein's algorithm computes an embedding of G in the Cartesian product of a least number of paths in $\mathcal{O}(\tau^2 n)$ time [48]. Then, we are done applying Theorem 1 to the resulting embedding. ◀

2.2 Triangular and hexagonal systems

A *triangular system* is a subgraph of the regular triangular grid which is induced by the vertices lying on a simple circuit and inside the region bounded by this circuit. Similarly, a hexagonal system (*a.k.a.*, benzenoid) is a subgraph of the regular hexagonal grid bounded by a simple circuit. Improving on two prior works [21, 23], but at the price of a slightly slower running time, we prove that:

▶ **Theorem 3.** *All eccentricities in an n-vertex triangular system can be computed in $\tilde{\mathcal{O}}(n)$ time. All eccentricities in an n-vertex hexagonal system can be computed in $\tilde{\mathcal{O}}(n)$ time.*

Proof. Every hexagonal system can be isometrically embedded in the Cartesian product of three trees, in linear time [21]. Then, we are done applying Theorem 1. The same holds for triangular system, but it is a scale embedding: all distances in the tree are multiplied by two [23]. As it shall become clearer in Sec. 3.2, our framework easily accommodates to this more general type of embeddings. ◀

2.3 Polygon graphs

A circle graph is the intersection graph of chords in a cycle. For every $k \geq 2$, a k-polygon graph is the intersection graph of chords in a convex k-polygon where the ends of each chord lie on two different sides. Note that the k-polygon graphs form an increasing hierarchy of all the circle graphs. We obtain:

▶ **Theorem 4.** *All eccentricities in an n-vertex m-edge k-polygon graph can be computed in $\mathcal{O}(2^{\mathcal{O}(k)}(n+m)^{1+o(1)})$ time, within an additive one-sided error of at most two.*

Proof. Every k-polygon graph can be embedded in a system of $2\log_{3/2} k + 7$ spanning trees with stretch two. Furthermore, such a system can be computed in linear time, if the graph is given together with its intersection model [36]. It was observed in [72] that an intersection model can be computed in $\mathcal{O}(4^k(n+m)\alpha(n+m))$ time, where $\alpha(\cdot,\cdot)$ denotes the Ackermann inverse function. We are now done applying Theorem 1 to the system (for $k' = \mathcal{O}(\log k)$). ◀

3 Proof of Theorem 1

We devote this section to the proof of our first main result in the paper (Theorem 1). In Sec. 3.1, we reduce to a more general query-answering problem on trees. We then solve this problem in Sec. 3.2.

3.1 Reductions

It turns out that *all* three types of embeddings that are considered in Theorem 1 can be reduced to the design of an efficient data structure for some abstract problem over a system of trees. In what follows, $\odot$ denotes a binary associative operation over the nonnegative real numbers (*e.g.*, the addition, minimum or maximum of two numbers).

▶ **Problem 1** ($\odot$-Eccentricities)**.**
Global Input: *A system $(T_i)_{1\leq i\leq k}$ of trees, and a subset $S \subseteq \prod_{i=1}^{k} V(T_i)$.*
Query Input: $v = (v_1, v_2, \ldots, v_k) \in \prod_{i=1}^{k} V(T_i)$
Query Output: $e_{\odot}(v,S) := \max\{d_{T_1}(s_1,v_1) \odot d_{T_2}(s_2,v_2) \odot \ldots \odot d_{T_k}(s_k,v_k) \mid (s_1,s_2,\ldots,s_k) \in S\}$.

Due to lack of space, formal reductions are omitted from the paper. Let us only sketch one of them (we stress that all these reductions are pretty similar to each other). Specifically, let φ be an isometric embedding of a graph $G = (V,E)$ (or of a discrete metric space) to the Cartesian product of the trees $T_1, T_2, \ldots, T_k$. We set $S := \varphi(V)$, and then, for every $u \in V$ we obtain: $e_+(\varphi(u)) = \max_{v\in V} \sum_{i=1}^{k} d_{T_i}(\varphi_i(u), \varphi_i(v)) = \max_{v\in V} d_{\square_{i=1}^{k} T_i}(\varphi(u), \varphi(v))$, where $\square$ stands for the Cartesian product and, for every $1 \leq i \leq k$, φ_i denotes the projection of φ to $V(T_i)$. In particular, $e_+(\varphi(u))$ is equal to the eccentricity of u. Similarly, we can reduce the eccentricity problem in systems and strong products of trees to min- and max-Eccentricities, respectively. We also stress that being given a *quasi* isometric embedding, our above approach leads to approximation algorithms for computing all the eccentricities.

3.2 A range-query framework

Our main result in this section is as follows:

▶ **Theorem 5.** *For every $(T_i)_{1\leq i\leq k}$ and S, let $N := \sum_{i=1}^{k} |V(T_i)|$. We can solve* min-ECCENTRICITIES *(resp., +-*ECCENTRICITIES*) with $\mathcal{O}(2^{\mathcal{O}(k\log k)}(N+|S|)^{1+o(1)})$ pre-processing time and $\mathcal{O}(2^{\mathcal{O}(k\log k)}(N+|S|)^{o(1)})$ query time.*

We need to introduce two useful tools. First, let V be a set of k-dimensional points, and let $f : V \to \mathbb{R}$. A *box* is the Cartesian product of k intervals. A range query asks, given a box $\mathcal{R}$, for a point $\overrightarrow{p} \in V \cap \mathcal{R}$ s.t. $f(\overrightarrow{p})$ is maximized. Up to some pre-processing in $\mathcal{O}(|V|\log^{k-1}|V|)$ time, such query can be answered in $\mathcal{O}(\log^{k-1}|V|)$ time [74]. The corresponding data structure is called a k-dimensional range tree. Furthermore, note that for any $\varepsilon > 0$, $\forall x > 0$, $\log^k x \leq 2^{\mathcal{O}(k\log k)}x^{\varepsilon}$ [17].

Second, for an n-node tree $T = (V, E)$, a *centroid* is a node whose removal leaves subtrees of order at most $n/2$. A classic theorem from Jordan asserts that such node always exists [62]. Furthermore, we can compute a centroid in $\mathcal{O}(n)$ time by dynamic programming (*e.g.*, see [56]). A *centroid decomposition* of T is a rooted tree T', constructed as follows. If $|V(T)| \leq 1$, then $T' = T$. Otherwise, let c be a centroid. Let $T'_1, T'_2, \ldots, T'_p$ be centroid decompositions for the subtrees $T_1, T_2, \ldots, T_p$ of $T \setminus \{c\}$. We obtain T' from $T'_1, T'_2, \ldots, T'_p$ by adding an edge between c and the respective roots of these rooted subtrees, choosing c as the new root. Note that we can compute a centroid decomposition in $\mathcal{O}(n)$ time [32]. We rather use the folklore $\mathcal{O}(n\log n)$-time algorithm since, for any node v, we also want to store its path $P(v)$, in T', until the root, and the distances $d_T(v, c_i)$, in T, for any $c_i \in P(v)$.

Sketch proof of Theorem 5. Due to lack of space, we only give a proof for the case $\odot = +$. During a pre-processing phase, we compute a centroid decomposition for each tree T_i, $1 \leq i \leq k$ separately, in total $\mathcal{O}(N\log N)$ time. Then, we iterate over the elements $s = (s_1, s_2, \ldots, s_k) \in S$. For every $1 \leq i \leq k$, let $P(s_i)$ be the path of s_i to the root into the centroid decomposition T'_i computed for T_i. We consider all possible k-sequences $c = (c_1, c_2, \ldots, c_k)$ s.t., $\forall 1 \leq i \leq k$, $c_i \in P(s_i)$. Since the length of each path $P(s_i)$ is in $\mathcal{O}(\log N)$, there are $\mathcal{O}(\log^k N)$ possibilities. W.l.o.g., all nodes have a unique identifier. Throughout the remainder of the proof, we identify the nodes with their identifiers, thus treating them as numbers. For any sequence c, we create an $2k$-dimensional point $\overrightarrow{p}_c(s)$, as follows: for every $1 \leq i \leq k$, the $(2i-1)^{th}$ and $2i^{th}$ coordinates are equal to c_i and the unique neighbour of c_i onto the c_is_i-path in T'_i, respectively (if $c_i = s_i$, then we may set both coordinates equal to s_i). The construction of all these $\mathcal{O}(|S|\log^k N)$ points takes time $\mathcal{O}(k|S|\log^k N)$. We include these points $\overrightarrow{p}_c(s)$, with an associated value $f(\overrightarrow{p}_c(s))$ (to be specified later in the proof) in some $2k$-dimensional range tree. It takes $\mathcal{O}((|S|\log^k N)\log^{2k-1}(|S|\log^k N)) = \mathcal{O}(2^{\mathcal{O}(k\log k)}(N+|S|)^{1+o(1)})$ time.

Then, in order to answer a query, let $v = (v_1, v_2, \ldots, v_k)$ be the input. As before, for every $1 \leq i \leq k$, let $P(v_i)$ be the path of v_i to the root into T'_i. We iterate over all the k-sequences $c = (c_1, c_2, \ldots, c_k)$ s.t., $\forall 1 \leq i \leq k$, $c_i \in P(v_i)$. Let $S_c \subseteq S$ contain every $(s_1, s_2, \ldots, s_k)$ s.t. $\forall 1 \leq i \leq k$, c_i is the least common ancestor of v_i and s_i in T'_i. The subsets S_c partition S, and so, $e_+(v, S) = \max_c e_+(v, S_c)$. We are left explaining how to compute $e_+(v, S_c)$ for any fixed c. For that, we observe that c_i is the least common ancestor of v_i and s_i in T'_i if and only if $c_i \in P(s_i) \cap P(v_i)$, and either $c_i \in \{v_i, s_i\}$ or the two neighbours of c_i onto the c_iv_i-path and c_is_i-path in T'_i are different. In the latter case, let us denote by u_i the neighbour of c_i onto the c_iv_i-path in T'_i. We can check these above conditions, for every $1 \leq i \leq k$, with the following constraints, over the $2k$-dimensional points $\overrightarrow{p} = (p_1, p_2, \ldots, p_{2k})$ constructed during the pre-processing phase: $\forall 1 \leq i \leq k$, $p_{2i-1} = c_i$ and if $c_i \neq v_i$, $p_{2i} \neq u_i$. Since $p_{2i} \neq u_i$ is equivalent to $p_{2i} \in (-\infty, u_i) \cup (u_i, +\infty)$, each inequality can be replaced by two range constraints over the same coordinate. In

particular, since there are $\leq k$ such inequalities, we can transform these above constraints into $\mathcal{O}(2^k)$ range queries. Therefore, we can output a point $\overrightarrow{p}_c(s)$, for some $s \in S_c$, maximizing $f(\overrightarrow{p}_c(s))$, in $\mathcal{O}(2^k \log^{2k-1}(|S| \log^k N)) = \mathcal{O}(2^{\mathcal{O}(k \log k)}(N + |S|)^{o(1)})$ time. Let $f(\overrightarrow{p}_c(s)) = \sum_{i=1}^{k} d_{T_i}(s_i, c_i)$. Since we have $s \in S_c$, $d_{T_i}(s_i, v_i) = d_{T_i}(s_i, c_i) + d_{T_i}(c_i, v_i)$ [54]. As a result: $e_+(v, S_c) = f(\overrightarrow{p}_c(s)) + \sum_{i=1}^{k} d_{T_i}(c_i, v_i)$. There are $\mathcal{O}(\log^k N)$ possible c, and so, the final query time is in $\mathcal{O}(2^{\mathcal{O}(k \log k)}(N + |S|)^{o(1)})$.

For the case when $\odot = \min$, we need points with $\mathcal{O}(k)$ more coordinates in order to correctly identify some index i s.t. $d_{T_i}(v_i, s_i) = \min_{1 \leq j \leq k} d_{T_j}(v_j, s_j)$. This is a similar trick as the one used in [1, 17, 19]. ◀

In contrast to Theorem 5, the distance between two vertices in the strong product of k trees equals the maximum distance between their k respective projections. Hence, we can process each tree of the system separately, and we obtain:

▶ **Lemma 6.** *For every $(T_i)_{1 \leq i \leq k}$ and S, let $N := \sum_{i=1}^{k} |V(T_i)|$. We can solve* max-Eccentricities *with $\mathcal{O}(N + k|S|)$ pre-processing time and $\mathcal{O}(k)$ query time.*

Proof. For every $1 \leq i \leq k$, let $\varphi_i : S \to V(T_i)$ be the projection of S to T_i. We stress that the projections $\varphi_i(S)$, $1 \leq i \leq k$, can be computed in total $\mathcal{O}(k|S|)$ time. Then, we iteratively remove from T_i the leaves that are not in $\varphi_i(S)$. Let $T'_1, T'_2, \ldots, T'_k$ be the k subtrees resulting from this above pre-processing. Note that, for every $1 \leq i \leq k$, $T_i \setminus T'_i$ is a forest whose each subtree can be rooted at some node adjacent to a leaf of T'_i, and so, adjacent to a node of $\varphi_i(S)$. For every node $v_i \in V(T_i) \setminus V(T'_i)$, let $\phi(v_i)$ be the unique leaf of T'_i s.t. the subtree of $T_i \setminus T'_i$ that contains v_i also contains a neighbour of $\phi(v_i)$. We compute, and store, the distance $d_{T_i}(v_i, \phi(v_i))$. Since, for doing so, we only need to perform breadth-first searches on disjoint subtrees, the total running time of this step is in $\mathcal{O}(N)$. Finally, we compute, for $1 \leq i \leq k$, all eccentricities in T'_i. Again, this can be done in total $\mathcal{O}(N)$ time (*e.g.*, see [22, 43]). This concludes the pre-processing phase.

In order to answer a query, let us consider some input $v = (v_1, v_2, \ldots, v_k)$. Our key insight here is that we have:

$$e_{\max}(v, S) = \max\{\max_{1 \leq i \leq k} d_{T_i}(v_i, s_i) \mid (s_1, s_2, \ldots, s_k) \in S\} = \max_{1 \leq i \leq k} \max\{d_{T_i}(v_i, s_i) \mid s_i \in \varphi_i(S)\}.$$

Therefore, in order to compute $e_{\max}(v, S)$ in $\mathcal{O}(k)$ time, it suffices to compute $\max\{d_{T_i}(v_i, s_i) \mid s_i \in \varphi_i(S)\}$ in $\mathcal{O}(1)$ time for every $1 \leq i \leq k$. If $v_i \in V(T'_i)$ then, since $\varphi_i(S) \subseteq V(T'_i)$ and furthermore all the leaves of T'_i are in $\varphi_i(S)$, we get $\max\{d_{T_i}(v_i, s_i) \mid s_i \in \varphi_i(S)\} = e_{T'_i}(v_i)$. Otherwise, $\max\{d_{T_i}(v_i, s_i) \mid s_i \in \varphi_i(S)\} = d_{T_i}(v_i, \phi(v_i)) + e_{T'_i}(\phi(v_i))$. ◀

Theorem 1 now follows from our reductions in Sec. 3.1 combined with Theorem 5 and Lemma 6.

4 Hardness results

We complete the positive results of Theorem 1 with two conditional lower bounds. Both theorems follow from a "SETH-hardness" result in order to compute the diameter of *split graphs* with a logarithmic[3] clique-number [13], and from the observation that every split graph with a maximal clique K embeds in any system of $|K|$ shortest-path trees rooted at the vertices of K.

[3] The logarithmic upper bound is not explicitly stated in [13], but it easily follows from the sparsification lemma applied to k-SAT.

▶ **Theorem 7.** *For any $\varepsilon > 0$, there exists a $c(\varepsilon)$ s.t., under SETH, we cannot compute the diameter of n-vertex graphs in $\mathcal{O}(n^{2-\varepsilon})$ time, even if we are given as input an isometric embedding of the graph in a system of at most $c(\varepsilon)\log n$ spanning trees. In particular, under SETH, there is no data structure for* min-Eccentricities *with $\mathcal{O}(2^{o(k)}(N+|S|)^{1-o(1)})$ pre-processing time and $\mathcal{O}(2^{o(k)}(N+|S|)^{o(1)})$ query time, where $N := \sum_{i=1}^{k} |V(T_i)|$.*

We believe our Theorem 7 to be important since the embeddings of graphs in systems of tree spanners are well-studied in the literature [35, 36, 37, 38, 39, 40, 41].

▶ **Theorem 8.** *For any $\varepsilon > 0$, there exists a $c(\varepsilon)$ s.t., under SETH, we cannot compute the diameter of n-point metric spaces in $\mathcal{O}(n^{2-\varepsilon})$ time, even if we are given as input an isometric embedding of the space in a Cartesian product of at most $c(\varepsilon)\log n$ tree factors. In particular, under SETH, there is no data structure for* +-Eccentricities *with $\mathcal{O}(2^{o(k)}(N+|S|)^{1-o(1)})$ pre-processing time and $\mathcal{O}(2^{o(k)}(N+|S|)^{o(1)})$ query time, where $N := \sum_{i=1}^{k} |V(T_i)|$.*

Proof. Recall that for any $\varepsilon > 0$, there exists a $c(\varepsilon)$ s.t., under SETH, we cannot compute the diameter in $\mathcal{O}(n^{2-\varepsilon})$ time on the split graphs of order n and clique-number at most $c(\varepsilon)\log n$ [13]. Let $G = (K \cup I, E)$ be a split graph, where K and I are a clique and a stable set, respectively. If it is not given, such a bipartition of $V(G)$ can be computed in linear time [57]. For every $u \in K$, we construct a tree T'_u, and an embedding φ_u of G into the latter, as follows. We start from a single-node tree, to which we map vertex u. Then, for every $v \in N_G(u)$, we add a leaf into the tree, adjacent to the image of u, to which we map the vertex v. We add another node u^* in T'_u, that is also adjacent to the image of u (note that u^* is *not* the image of a vertex of G). Finally, for every vertex $v \in V(G) \setminus N_G[u]$, we add a leaf node, to which we map vertex v, that we connect to u^* by a path of length two. Let $\varphi : V(G) \to V(\square_{u\in K} T'_u)$ be s.t., for every $v \in V(G)$, $\varphi(v) = (\varphi_u(v))_{u\in K}$. The metric space considered is $(\varphi(V(G)), d)$, where d is the sub-metric induced by $d_{\square_{u\in K} T'_u}$ (distances in the Cartesian product). Indeed, let $v, v' \in V(G)$. For every $u \in K$, by construction we have $diam(T'_u) = 4$, and so, $d_{T'_u}(\varphi_u(v), \varphi_u(v')) \leq 4$. Specifically, if $u \in \{v, v'\}$ then $d_{T'_u}(\varphi_u(v), \varphi_u(v')) \leq 3$; if $v, v' \in N_G(u)$ then $d_{T'_u}(\varphi_u(v), \varphi_u(v')) = 2$; otherwise, $d_{T'_u}(\varphi_u(v), \varphi_u(v')) = 4$. Altogether combined, if $d_G(v, v') = 3$ (in particular, $v, v' \in I$) then we get $d(\varphi(v), \varphi(v')) = 4|K|$, otherwise we get $d(\varphi(v), \varphi(v')) \leq 3 + 4(|K| - 1) = 4|K| - 1$. ◀

5 One-to-many tree embeddings

We end up discussing a different type of tree-like embedding than in Sec. 2. First we present an algorithm for computing all eccentricities being given such an embedding (Sec. 5.1). We then propose, in Sec 5.2, an application of our result to the absolute center problem in a subclass of median graphs.

5.1 Computation of subset-eccentricities

We now address the problem of computing the eccentricity of *subsets* of nodes, in *one* tree:

▶ **Theorem 9.** *Let T be any n-node tree, and let $\alpha : V(T) \to \mathbb{R}$. After a pre-processing in $\mathcal{O}(n \log n)$ time, for any subset U of nodes and function $\beta : U \to \mathbb{R}$, we can compute the value $e_{T,\alpha}(U, \beta) = \max_{v\in V(T)} \min_{u\in U}(\alpha(v) + d_T(v, u) + \beta(u))$ in $\mathcal{O}(|U| \log^2 n)$ time.*

In particular, after an $\mathcal{O}(n \log n)$-time pre-processing we can compute $e_T(U)$ in $\mathcal{O}(|U| \log^2 n)$ time by setting $\alpha(v) = 0$ for every node $v \in V(T)$ and $\beta(u) = 0$ for every node $u \in U$. The proof of Theorem 9 is technical, and we omit it due to lack of space. It is based

on a completely different processing method of the tree than in Sec. 3.2: using heavy-path decomposition [71] and the local computation of so-called Cartesian trees for maximum range queries [53].

Here is a possible application of our Theorem 9 for general graphs. By a clan embedding of a graph $G = (V, E)$ in a tree T, we mean a one-to-many embedding $S : V \to 2^{V(T)}$ such that, to each vertex $v \in V$, we also associate a leader $\chi(v) \in S_v$ in its corresponding node-subset of T. This embedding must further satisfy $d_T(S_u, S_v) \geq d_G(u, v)$ for every $u, v \in V$. The distortion of a clan embedding can be defined as $t := \max_{u \neq v} d_T(\chi(u), S_v)/d_G(u, v)$.

For each node x of T, we set $\alpha(x)$ to 0 if $x = \chi(u)$ for some vertex $u \in V$; otherwise, we set $\alpha(x)$ to some large enough *negative* value. Then, for any subset $U \subseteq V$, we can apply Theorem 9 to $\bigcup_{u \in U} S_u$ in order to compute a t-approximation of $e_G(U)$. We refer to [52] for various trade-offs between the size of the subsets S_u and the resulting distortion t.

5.2 Application to a distance-labelling scheme

We devote our last section to cube-free median graphs. Recall that a graph $G = (V, E)$ is called *median* if, for any triple $x, y, z \in V$, there exists a unique vertex c that is simultaneously on some shortest xy-, yz- and zx-paths. This class is ubiquitous in Theoretical Computer Science. Indeed, the median graphs are exactly the 1-skeletons of CAT(0) cube complexes [58], the domains of event structures [67] and the solution sets of 2-SAT formulas [66], among many characterizations. The *dimension* of a median graph G is the largest $d \geq 1$ such that G contains a d-cube (hypercube of dimension d) as an induced subgraph. In particular, the median graphs of dimension 1 are exactly the trees. The median graphs of dimension at most 2, *a.k.a.*, *cube-free median graphs*, have already received some attention in the literature [7, 16, 25, 26, 27].

In what follows, we abusively call eccentricity of a point $x \in X_G$, denoted by $e_G(x)$, its maximum distance to a vertex of G.

▶ **Theorem 10.** *There is an $\tilde{\mathcal{O}}(n)$-time algorithm for computing the absolute center of n-vertex cube-free median graphs. More generally, the algorithm encodes in $\mathcal{O}(n)$ space the eccentricity of all the points of X_G.*

The remainder of this section is devoted to the proof of Theorem 10. Given any point $x \in X_G$, if $W \subseteq V$ then, let $e_G(x, W) = \max_{w \in W} d(x, w)$. We observe that:

▶ **Lemma 11.** *For a bipartite graph $G = (V, E)$ and an edge $uv \in E$, let $W_{u,v} = \{w \in V \mid d(u, w) < d(v, w)\}$. If x is a point of the edge uv such that $d(u, x) = t$, then, we have $e(x) = \max\{t + e(u, W_{u,v}), 1 - t + e(v, W_{v,u})\}$.*

Proof. We have $V = W_{u,v} \cup W_{v,u}$ because G is bipartite. Let $w \in W_{u,v}$. Every xw-path going by vertex v would have length $1 - t + d(v, w) = 2 - t + d(u, w) > t + d(u, w)$, and therefore, it cannot be a shortest xw-path. In the same way, let $w \in W_{v,u}$. Every xw-path going by vertex u would have length $t + d(u, w) = 1 + t + d(v, w) > 1 - t + d(v, w)$, and therefore, it cannot be a shortest xw-path either. ◀

As a result of Lemma 11, we are left computing the values $e(u, W_{u,v})$ and $e(v, W_{v,u})$ for every edge uv. Since the cube-free median graphs are sparse [26], there are only $\mathcal{O}(n)$ values to be stored. Furthermore, for median graphs, the subsets $W_{u,v}$ are called half-spaces and they induce convex subgraphs. Thus, in a way, we could reduce the absolute center problem to the computation of all eccentricities in various convex subgraphs of the input. However, the number of subgraphs to be considered is in general super-constant, even for the cube-free

median graphs. Since by Lemma 11, a point in the absolute center must be either a vertex or at the middle of an edge, we could also reduce the absolute center problem to computing all eccentricities in the subdivision of G. Unfortunately, median graphs are not closed by taking subdivisions. We take an alternative approach in what follows.

For that, we first recall some notions and results from [26]. These results are specific to cube-free median graphs but, as we pointed it out in Sec. 1, similar structural decomposition theorems were proved in [24, 29, 28] for completely unrelated geometric graph classes.

In what follows, let $G = (V, E)$ be a cube-free median graph. A *centroid* is any vertex minimizing the sum of its distances to all other vertices. Recently, it was shown that a centroid in a median graph can be computed in linear time [10]. So, let $c \in V$ be a centroid.

A subgraph H of G is *gated* if, for every $v \in V \setminus V(H)$, there exists a $v^* \in V(H)$ s.t., $\forall u \in V(H),\ d_G(u, v) = d_G(u, v^*) + d_G(v^*, v)$. We define the *fibers* $F(x) = \{x\} \cup \{v \in V(G \setminus H) \mid x \text{ is the gate of } v \text{ in } H\}$. The fibers $F(x),\ x \in V(H)$ partition the vertex-set of G, and each induces a gated subgraph [26].

For any $z \in V$, the *star* $St(z)$ of z is the subgraph of G induced by all edges and squares of G incident to z. Any such star $St(z)$ is gated and, if furthermore $z = c$, every fiber $F(x),\ x \in St(c)$ contains at most $|V|/2$ vertices [26].

A fiber $F(x)$ of the star $St(c)$ is a *panel* if $x \in N_G(c)$, and a *cone* otherwise. We say that two fibers $F(x), F(y)$ are *neighboring* if there exists an edge with an end in $F(x)$ and the other end in $F(y)$. If two fibers are neighboring then one must be a panel and the other must be a cone; furthermore, a cone has two neighboring panels [26]. Two fibers are 2-*neighboring* if they are cones adjacent to the same panel. Finally, two fibers that are neither neighboring nor 2-neighboring are called *separated*.

The subset of vertices in $F(x)$ with a neighbour in $F(y)$ is denoted by $\partial_y F(x)$ (with the understanding that $\partial_y F(x) = \emptyset$ when $F(x), F(y)$ are not neighboring). The *total boundary* of $F(x)$ is defined as $\partial^* F(x) = \cup_y \partial_y F(x)$.

For a set of vertices A, an *imprint* of a vertex u is a vertex $a \in A$ such that there is no vertex of A (but a itself) on any shortest au-path. A subgraph H of G is *quasigated* if every vertex of $V(G \setminus H)$ has at most two imprints. It is known that for each fiber $F(x)$ of a star $St(z)$, the total boundary $\partial^* F(x)$ is an isometric quasigated tree [26].

▶ **Lemma 12** ([26]). *Let $G = (V, E)$ be a cube-free median graph, let $c \in V$ be a centroid, and let $F(x), F(y)$ be two fibers of the star $St(c)$. The following hold for every $u \in F(x),\ v \in F(y)$.*

- *If $F(x)$ and $F(y)$ are separated, then $d_G(u, v) = d_G(u, c) + d_G(c, v)$;*
- *If $F(x)$ and $F(y)$ are neighboring, $F(x)$ is a panel and $F(y)$ is a cone, then let u_1, u_2 be the two (possibly equal) imprints of u on the total boundary $\partial^* F(x)$, and let v^* be the gate of v in $F(x)$. We have $d_G(u, v) = \min\{d_G(u, u_1) + d_G(u_1, v^*) + d_G(v^*, v), d_G(u, u_2) + d_G(u_2, v^*) + d_G(v^*, v)\}$;*
- *If $F(x)$ and $F(y)$ are 2-neighboring, then let $F(w)$ be the panel neighboring $F(x)$ and $F(y)$. Let u^* and v^* be the gates of u and v in $F(w)$. Then $d_G(u, v) = d_G(u, u^*) + d_G(u^*, v^*) + d_G(v^*, v)$.*

Sketch proof of Theorem 10. We compute for each edge $uv \in E$ two values, denoted by $f_G(u, v)$ and $f_G(v, u)$, so that: $f_G(u, v) \leq e_G(u, W_{u,v}),\ f_G(v, u) \leq e_G(v, W_{v,u})$ and, for each point x of the edge such that $d(u, x) = t$, $e_G(x) = \max\{t + f_G(u, v), 1 - t + f_G(v, u)\}$. If $E = \{uv\}$ then, we set $f_G(u, v) = f_G(v, u) = 0$. Otherwise, we compute a centroid c and we use an algorithmic procedure from [26] in order to compute in $\mathcal{O}(n)$ time the vertex-set and the edge-set of all fibers $F(x),\ x \in St(c)$. We enumerate all the fibers $F(x),\ x \in St(c)$, and we compute $D(x) = \max_{v \in F(x)} d_G(v, c)$. It can be done in total $\mathcal{O}(n)$ time by performing a

BFS rooted at c. Also, for each $x \in St(c)$, let G_x be induced by $F(x)$. Since, $F(x)$ is gated, and so convex, G_x is a cube-free median graph. In particular, we can apply our algorithm recursively on it in order to compute the values $f_{G_x}(\cdot,\cdot)$ associated to its edges. Then, we apply the following steps (some of which not being detailed due to lack of space):

Step 1. We consider all edges with their both ends in $St(c)$ and we compute the corresponding values $f_G(\cdot,\cdot)$. Roughly, it can be done in total $\mathcal{O}(n \log n)$ time if we order the vertices of $St(c)$ by non-decreasing $D(\cdot)$ value, and if we keep track for each $x \in St(c)$ of its neighbours in the star. To illustrate this, consider for example some edge xc. Then, $W_{x,c} = \bigcup\{F(y) \mid y \in N[x] \cap (St(c) \setminus \{c\})\}$, and so it becomes possible to compute $f_G(x,c), f_G(c,x)$ from the $D(\cdot)$ values. In the same way for an edge xy, where $x \in N(c), y \in St(c) \setminus N[c]$, we have $W_{y,x} = F(y) \cup F(x')$ where $F(x), F(x')$ are the two panels neighboring $F(y)$, and so, we can proceed as in the previous case for computing $f_G(x,y), f_G(y,x)$. Thus from now on, we only consider edges with at least one end not in $St(c)$.

Step 2. We consider each remaining edge uv sequentially, where $d(u,c) < d(v,c)$. Let $x, y \in St(c)$ satisfy $u \in F(x),\ v \in F(y)$. Let A be the union of all the fibers $F(z)$ s.t. $F(z)$ is separated from both $F(x)$ and $F(y)$. By Lemma 12, we have $A \subseteq W_{u,v}$. We want to compute $\max_{w \in A} d_G(u,w)$. There are three cases: $x = y$ and $F(x)$ is a panel; $x = y$ and $F(x)$ is a cone; $x \neq y$. Due to lack of space, we only detail the first case (the other two cases can be handled with similar techniques). Specifically, let $z \neq x$ be such that $F(z)$ is a panel and $D(z)$ is maximized. It can be computed in constant-time assuming the panels were ordered during a pre-processing phase. Recall that two panels are always separated. Therefore, the maximum distance between u and a vertex of A in a panel is equal to $d(u,c) + D(z)$. The case of separated cones is more complicated, and it requires some pre-processing. Specifically, we assign to all the fibers $F(z)$ pairwise different positive numbers, that we abusively identify with the vertices z of the star $St(c)$. Then, we enumerate the cones $F(z'),\ z' \in St(c) \setminus N_G[c]$. Let $F(z_1), F(z_2)$ be the two panels neighboring $F(z')$. We create a point $\overrightarrow{p}(z') = (z_1, z_2)$, to which we associate the value $f(\overrightarrow{p}(z')) = D(z')$. To complete the pre-processing, we put these points and their associated values in a 2-dimensional range tree, that takes $\mathcal{O}(n \log n)$ time. Now, to compute the maximum value between u and a vertex of A in a cone, it suffices to compute a point $\overrightarrow{p}(z') = (p_1, p_2)$ such that:

$$\begin{cases} p_1 \neq x,\ p_2 \neq x \\ f(\overrightarrow{p}(z')) \text{ is maximized for these above properties.} \end{cases}$$

Indeed, this maximum distance is exactly $d_G(u,c) + D(z') = d_G(u,c) + f(\overrightarrow{p}(z'))$. Furthermore, since each inequality gives rise to two disjoint intervals to which the corresponding coordinate must belong, a point $\overrightarrow{p}(z')$ as above can be computed using four range queries. It takes $\mathcal{O}(\log n)$ time. As a result, the total running time of this step is in $\mathcal{O}(n \log n)$.

Step 3. We consider each edge uv with at least one end not in $St(c)$ and such that $u \in F(x)$ for some panel $F(x)$. In what follows, for any $z, z' \in St(c)$, we write $z \sim z'$ if $F(z), F(z')$ are neighboring. Let $B = \bigcup\{F(y) \mid x \sim y \text{ and } v \notin F(y)\}$. Intuitively, what we try to do at this step is to compute $e(u, W_{u,v} \cap B)$ and $e(v, W_{v,u} \cap B)$. There are two cases: either $v \notin F(x)$ or $v \in F(x)$. In both cases, we reduce our computations to some suitable eccentricity problem on the tree $T = \partial^* F(x)$. Next, we detail the case $v \in F(x)$, which is the one to which we need to apply Theorem 9 (the case $v \notin F(x)$ is solved by using a rather standard dynamic

programming approach on the tree T). For each node $z \in V(T)$, let $\alpha(z)$ be the maximum distance between z and any vertex w in a neighboring cone $F(y)$ of which z is the gate in $F(x)$ (with the understanding that, if no vertex has z as its gate, then $\alpha(z)$ is a sufficiently large *negative* value, say $\alpha(z) = -|V(T)|$). All these nodes weights can be pre-computed in $\mathcal{O}(\sum_{y|x\sim y} |F(y)|)$ time. Furthermore, note that a cone is neighboring two panels [26]. As a result, if we consider each panel $F(x)$ sequentially then, we scan each cone only twice, and the total running-time of this pre-computation phase is in $\mathcal{O}(n)$.

We compute $e_G(u, B)$, $e_G(v, B)$ and $e_G(\{u, v\}, B) =^{def} \max_{w\in B} d_G(w, \{u, v\})$. For that, let $u_1, u_2 \in V(T)$ be the two (possibly equal) imprints of u, and similarly let $v_1, v_2 \in V(T)$ be the imprints of v. Set $\beta(u_1) = d_G(u, u_1)$, $\beta(u_2) = d_G(u, u_2)$ and in the same way $\beta(v_1) = d_G(v, v_1)$, $\beta(v_2) = d_G(v, v_2)$. Since T is isometric, then it follows from the distance formulae in Lemma 12 that the values to be computed are exactly $e_{T,\alpha}(\{u_1, u_2\}, \beta)$, $e_{T,\alpha}(\{v_1, v_2\}, \beta)$ and $e_{T,\alpha}(\{u_1, v_1, u_2, v_2\}, \beta)$. By Theorem 9, the latter can be computed in $\mathcal{O}(\log^2 n)$ time, up to some initial pre-processing of T in $\mathcal{O}(|V(T)| \log |V(T)|) = \mathcal{O}(|F(x)| \log |F(x)|)$ time. Overall, the running-time of this step is in $\mathcal{O}(n \log^2 n)$.

Let $e_G(u, B) = p_1$, $e_G(v, B) = p_2$ and $e_G(\{u, v\}, B) = p$. W.l.o.g., $p_1 \leq p_2$. It implies $p_2 = p + 1$. Then, $e_G(u, B \cap W_{u,v}) = p$. In the same way, if $p_1 = p + 1$ then we also have $e_G(v, B \cap W_{v,u}) = p$. From now on, we assume $p_1 < p + 1$. In particular, $p_1 = p$. But then, $e_G(v, B \cap W_{v,u}) \leq p - 1$, and therefore we needn't compute this value (*i.e.*, because we always have $t + p \geq 1 - t + e_G(v, B \cap W_{v,u})$ for any $t \in (0; 1)$).

Step 4. Finally, consider each edge uv with at least one end not in $St(c)$ and such that $v \in F(y)$ for some cone $F(y)$. Let $C = \bigcup\{F(y') \mid F(y)$ and $F(y')$ are either neighboring or 2-neighbouring, and $u \notin F(y')\}$. We would like to compute $e_G(u, W_{u,v} \cap C)$ and $e_G(v, W_{v,u} \cap C)$. There are two cases: either $u \in F(y)$ or $u \notin F(y)$. Consider the case $u \in F(y)$ (the other case can be dealt with similarly). We consider each x s.t. $F(x)$ is a panel neighboring $F(y)$ sequentially (there are only two such x). Let C_x contain $F(x)$ and all cones of C that are neighboring $F(x)$. We now consider $T = \partial^* F(x)$ which we assume to be pre-processed as during the previous Step 3. Let $u^*, v^* \in V(T)$ be the respective gates of u, v. Indeed, by Lemma 12 there is always a shortest-path between u (resp., v) and any vertex of C_x that goes by u^* (resp., by v^*). This part is solved through a delicate case analysis which depends on: some values $f_{G_x}(\cdot, \cdot)$ (obtained by applying our algorithm recursively to G_x) and some eccentricity functions computed for T during Step 3.

Overall, each fiber contains at most $n/2$ vertices, and so there are $\mathcal{O}(\log n)$ recursive stages. Since a stage runs in $\mathcal{O}(n \log^2 n)$ time (the bottleneck being Step 3), the total running time for computing all the values $f_G(\cdot, \cdot)$ is in $\mathcal{O}(n \log^3 n)$. ◀

Open problem. To which other graph classes can our framework in this section be applied? A good candidate could be the planar graphs of non positive combinatorial curvature [24], and especially the trigraphs [29]. To our best knowledge, it is open whether the eccentricities in a trigraph can be computed in almost linear time (there exists a linear-time algorithm for computing the diameter and the center [53]).

References

1 A. Abboud, V. Vassilevska Williams, and J. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the twenty-seventh annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 377–391. SIAM, 2016.

2 Muad Abu-Ata and Feodor F. Dragan. Metric tree-like structures in real-world networks: an empirical study. *Networks*, 67(1):49–68, 2016. doi:10.1002/net.21631.

3 H. Bandelt and V. Chepoi. Metric graph theory and geometry: a survey. *Contemporary Mathematics*, 453:49–86, 2008.

4 H. Bandelt and H. Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41(2):182–208, 1986.

5 H.-J. Bandelt. Recognition of tree metrics. *SIAM Journal on Discrete Mathematics*, 3(1):1–6, 1990.

6 H.-J. Bandelt, V. Chepoi, and D. Eppstein. Combinatorics and geometry of finite and infinite squaregraphs. *SIAM Journal on Discrete Mathematics*, 24(4):1399–1440, 2010.

7 H.-J. Bandelt, V. Chepoi, and D. Eppstein. Ramified rectilinear polygons: coordinatization by dendrons. *Discrete & Computational Geometry*, 54(4):771–797, 2015.

8 H.-J. Bandelt and M. van De Vel. Embedding topological median algebras in products of dendrons. *Proceedings of the London Mathematical Society*, 3(3):439–453, 1989.

9 V. Batagelj, T. Pisanski, and J. Simoes-Pereira. An algorithm for tree-realizability of distance matrices. *International Journal of Computer Mathematics*, 34(3-4):171–176, 1990.

10 L. Bénéteau, J. Chalopin, V. Chepoi, and Y. Vaxès. Medians in median graphs and their cube complexes in linear time. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

11 P. Bergé and M. Habib. Diameter in linear time for constant-dimension median graphs. In *XI Latin and American Algorithms, Graphs and Optimization Symposium (LAGOS 2021)*, 2021. To appear.

12 J. A. Bondy and U. S. R. Murty. *Graph theory*. Springer, 2008.

13 M. Borassi, P. Crescenzi, and M. Habib. Into the square: On the complexity of some quadratic-time solvable problems. *Electronic Notes in Theoretical Computer Science*, 322:51–67, 2016.

14 M. Borassi, P. Crescenzi, and L. Trevisan. An axiomatic and an average-case analysis of algorithms and heuristics for metric properties of graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 920–939. SIAM, 2017.

15 A. Brandstädt, V. Chepoi, and F. Dragan. The algorithmic use of hypertree structure and maximum neighbourhood orderings. *Discrete Applied Mathematics*, 82(1-3):43–77, 1998.

16 B. Bresar, S. Klavzar, and R. Skrekovski. On cube-free median graphs. *Discrete Mathematics*, 307(3):345–351, 2007.

17 K. Bringmann, T. Husfeldt, and M. Magnusson. Multivariate Analysis of Orthogonal Range Searching and Graph Distances. *Algorithmica*, pages 1–24, 2020.

18 S. Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Transactions on Algorithms (TALG)*, 15(2):1–38, 2018.

19 S. Cabello and C. Knauer. Algorithms for graphs of bounded treewidth via orthogonal range searching. *Computational Geometry*, 42(9):815–824, 2009.

20 C. Cheng. A poset-based approach to embedding median graphs in hypercubes and lattices. *Order*, 29(1):147–163, 2012.

21 V. Chepoi. On distances in benzenoid systems. *Journal of chemical information and computer sciences*, 36(6):1169–1172, 1996.

22 V. Chepoi, F. Dragan, M. Habib, Y. Vaxès, and H. Alrasheed. Fast approximation of eccentricities and distances in hyperbolic graphs. *Journal of Graph Algorithms and Applications*, 23(2):393–433, 2019.

23 V. Chepoi, F. Dragan, and Y. Vaxès. Center and diameter problems in plane triangulations and quadrangulations. In *Symposium on Discrete Algorithms (SODA'02)*, pages 346–355, 2002.

24 V Chepoi, F Dragan, and Y Vaxes. Distance and routing problems in plane graphs of non-positive curvature. *J. Algorithms*, 61:1–30, 2006.

25 V. Chepoi and M. Hagen. On embeddings of CAT(0) cube complexes into products of trees via colouring their hyperplanes. *Journal of Combinatorial Theory, Series B*, 103(4):428–467, 2013.

26 V. Chepoi, A. Labourel, and S. Ratel. Distance and routing labeling schemes for cube-free median graphs. *Algorithmica*, pages 1–45, 2020.

27 V. Chepoi and D. Maftuleac. Shortest path problem in rectangular complexes of global nonpositive curvature. *Computational Geometry*, 46(1):51–64, 2013.

28 Victor Chepoi, Arnaud Labourel, and Sébastien Ratel. Distance labeling schemes for K_4-free bridged graphs. In *International Colloquium on Structural Information and Communication Complexity*, pages 310–327. Springer, 2020.

29 Victor Chepoi, Y Vaxes, and FR Dragan. Distance-based location update and routing in irregular cellular networks. In *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*, pages 380–387. IEEE, 2005.

30 D. Coudert, G. Ducoffe, and A. Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. *ACM Transactions on Algorithms (TALG)*, 15(3):1–57, 2019.

31 P. Damaschke. Computing giant graph diameters. In *International Workshop on Combinatorial Algorithms (IWOCA)*, pages 373–384. Springer, 2016.

32 D. Della Giustina, Ni. Prezza, and R. Venturini. A new linear-time algorithm for centroid decomposition. In *String Processing and Information Retrieval*, pages 274–282. Springer International Publishing, 2019.

33 R. Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

34 D Ž Djoković. Distance-preserving subgraphs of hypercubes. *Journal of Combinatorial Theory, Series B*, 14(3):263–267, 1973.

35 F. Dragan and M. Abu-Ata. Collective additive tree spanners of bounded tree-breadth graphs with generalizations and consequences. *Theoretical Computer Science*, 547:1–17, 2014.

36 F. Dragan, D. Corneil, E. Köhler, and Y. Xiang. Collective additive tree spanners for circle graphs and polygonal graphs. *Discrete Applied Mathematics*, 160(12):1717–1729, 2012.

37 F. Dragan, Y. Xiang, and C. Yan. Collective Tree Spanners for Unit Disk Graphs with Applications. *Electronic Notes in Discrete Mathematics*, 32:117–124, 2009.

38 F. Dragan and C. Yan. Collective tree spanners in graphs with bounded parameters. *Algorithmica*, 57(1):22–43, 2010.

39 F. Dragan, C. Yan, and D. Corneil. Collective Tree Spanners and Routing in AT-free Related Graphs. *Journal of Graph Algorithms and Applications*, 10(2):97–122, 2006.

40 F. Dragan, C. Yan, and I. Lomonosov. Collective tree spanners of graphs. *SIAM Journal on Discrete Mathematics*, 20(1):240–260, 2006.

41 F. Dragan, C. Yan, and Y. Xiang. Collective additive tree spanners of homogeneously orderable graphs. In *Latin American Symposium on Theoretical Informatics*, pages 555–567. Springer, 2008.

42 G. Ducoffe. A New Application of Orthogonal Range Searching for Computing Giant Graph Diameters. In *Symposium on Simplicity in Algorithms (SOSA)*, 2019.

43 G. Ducoffe. Easy computation of eccentricity approximating trees. *Discrete Applied Mathematics*, 260:267–271, 2019.

44 G. Ducoffe. Optimal diameter computation within bounded clique-width graphs. Technical report, arXiv, 2020. `arXiv:2011.08448`.

45 G. Ducoffe and F.F. Dragan. A story of diameter, radius and (almost) helly property. *Networks*, 2021. To appear.

46 G. Ducoffe, M. Habib, and L. Viennot. Fast diameter computation within split graphs. In *International Conference on Combinatorial Optimization and Applications*, pages 155–167. Springer, 2019.

47 G. Ducoffe, M. Habib, and L. Viennot. Diameter computation on H-minor free graphs and graphs of bounded (distance) VC-dimension. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1905–1922. SIAM, 2020.

48 D. Eppstein. The lattice dimension of a graph. *European Journal of Combinatorics*, 26(5):585–592, 2005.

49 D. Eppstein. Recognizing Partial Cubes in Quadratic Time. *Journal of Graph Algorithms and Applications*, 5(2):269–293, 2011.

50 S. Even and A. Litman. Layered cross product—A technique to construct interconnection networks. In *Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures*, pages 60–69, 1992.

51 A. Farley and A. Proskurowski. Computation of the center and diameter of outerplanar graphs. *Discrete Applied Mathematics*, 2(3):185–191, 1980.

52 A. Filtser and H. Le. Clan embeddings into trees, and low treewidth graphs. In *53rd Annual ACM Symposium on Theory of Computing (STOC 2021)*. ACM, 2021. To appear.

53 H. Gabow, J. Bentley, and R. Tarjan. Scaling and related techniques for geometry problems. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 135–143, 1984.

54 C. Gavoille, D. Peleg, S. Pérennes, and R. Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004.

55 P. Gawrychowski, H. Kaplan, S. Mozes, M. Sharir, and O. Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\tilde{O}(n^{5/3})$ time. In *Symposium on Discrete Algorithms (SODA)*, pages 495–514. SIAM, 2018.

56 A. Goldman. Optimal center location in simple networks. *Transportation science*, 5(2):212–221, 1971.

57 M. Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier, 2004.

58 Mikhaïl Gromov. Hyperbolic groups. In *Essays in group theory*, pages 75–263. Springer, 1987. doi:10.1007/978-1-4613-9586-7_3.

59 S.L. Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations research*, 12(3):450–459, 1964.

60 G.Y. Handler. Minimax location of a facility in an undirected tree graph. *Transportation Science*, 7(3):287–293, 1973.

61 E. Howorka. On metric properties of certain clique graphs. *Journal of Combinatorial Theory, Series B*, 27(1):67–74, 1979.

62 C. Jordan. Sur les assemblages de lignes. *J. Reine Angew. Math*, 70(185):81, 1869.

63 O. Kariv and S.L. Hakimi. An algorithmic approach to network location problems. I: The p-centers. *SIAM Journal on Applied Mathematics*, 37(3):513–538, 1979.

64 S. Khuller and Y. Sussmann. The capacitated k-center problem. *SIAM Journal on Discrete Mathematics*, 13(3):403–418, 2000.

65 Y.-F. Lan, Y.-L. Wang, and H. Suzuki. A linear-time algorithm for solving the center problem on weighted cactus graphs. *Information Processing Letters*, 71(5-6):205–212, 1999.

66 H. M. Mulder and A. Schrijver. Median graphs and Helly hypergraphs. *Discrete Mathematics*, 25(1):41–50, 1979.

67 M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, part I. *Theoretical Computer Science*, 13(1):85–108, 1981.

68 R. Nowakowski and I. Rival. The smallest graph variety containing all paths. *Discrete Mathematics*, 43(2-3):223–234, 1983.

69 S. Olariu. A simple linear-time algorithm for computing the center of an interval graph. *International Journal of Computer Mathematics*, 34(3-4):121–128, 1990.

70 L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing (STOC)*, pages 515–524, 2013.

71 D. Sleator and R. Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983.

72 L. Stewart and R. Valenzano. On polygon numbers of circle graphs and distance hereditary graphs. *Discrete Applied Mathematics*, 248:3–17, 2018.

73 S. Ting. A linear-time algorithm for maxisum facility location on tree networks. *Transportation Science*, 18(1):76–84, 1984.

74 D. Willard. New data structures for orthogonal range queries. *SIAM Journal on Computing*, 14(1):232–253, 1985.

75 P.M. Winkler. Isometric embedding in products of complete graphs. *Discrete Applied Mathematics*, 7(2):221–225, 1984.