The power of interval models for computing graph centralities

Guillaume Ducoffe

The power of interval models for computing graph centralities

Guillaume Ducoffe[1,2]

[1]National Institute for Research and Development in Informatics, Romania
[2]University of Bucharest, Romania

Abstract

Food webs, some scheduling problems and DNA molecules all have in common a "linear structure" which can be captured through the idealized model of *interval graphs* (intersection graphs of intervals on a line). However, real data is prone to errors and noise, thus raising the question of whether the algorithmic results obtained for the interval graphs could be extended to more realistic models of "almost interval" graphs. We address this question in the context of computing the *vertex eccentricities*, one of the most studied centrality indices in order to determine the relative importance of nodes in a network. We give a positive answer for the interval+kv graphs and a negative one (assuming plausible complexity hypotheses) for the graphs of bounded interval number. In particular, we present an almost linear-time algorithm for computing all vertex eccentricities in an interval+kv graph, for any fixed k, thus improving on the recent quadratic-time algorithm of (Bentert & Nichterlein, 2022) for this problem.

1 Introduction

We study the feasibility of computing centrality indices in a network, a fundamental task in Network analysis in order to determine the relative importance of every unit (the larger the centrality of a node is, the more important it should be). As is standard in such theoretical investigations, we represent a network by a graph. For any undefined graph notions and terminology in what follows, see [BM08]. Unless stated otherwise, all graphs considered are undirected, simple (*i.e.*, without loops or multiple edges), unweighted and connected. Let $G = (V, E)$ be an arbitrary graph. The distance between two vertices u and v (sometimes called their "hop distance") equals the minimum number of edges on a uv-path. We denote it in what follows by $d_G(u, v)$, or simply $d(u, v)$ if graph G is clear from the context. For communication networks in a broad sense (*e.g.*, telecommunication networks, online social networks but also large-scale brain networks), the distance $d(u, v)$ may be regarded as the delay for transmitting a message with respective sender and recipient u and v. Let the graph centrality of vertex v be defined as $1/e_G(v)$, where $e_G(v) = \max_{u \in V} d_G(u, v)$ [HH95]. The value $e_G(v)$ is also called the eccentricity of vertex v. Note that there also exist other centrality indices than the one we discussed above [DSP18].

Distances in graphs play an important role in Location theory. For instance, in order to broadcast a message, it is desirable to minimize the maximum distance of a node to the source. In this respect, an optimal location for sending such a message would be at a vertex of minimum eccentricity or, equivalently, one of maximum centrality. The center of a graph is the subset of all its vertices of minimum eccentricity. In light of its aforementioned applications to networks, the problem of computing the graph center, or even better, all vertex eccentricities, has received considerable attention. It is well-known that for every n-vertex m-edge graph we can compute all vertex eccentricities in $\mathcal{O}(nm)$ time, simply by running a breadth-first search from every vertex. This algorithm is not practical for huge complex networks such as Facebook, with hundreds of millions of nodes and billions of links, for which even using a massively parallel implementation of BFS it takes hours to complete [BBR$^+$12]. The existence of a faster algorithm for computing all vertex eccentricities was open for decades, until it was solved in the negative by [RVW13]. Specifically, they proved a surprising connection between this problem and the Boolean Satisfiability problem, an important problem in

electronic design automation and many other areas in computer science. The Boolean Satisfiability problem asks whether a given logic formula is satisfiable. The Strong Exponential-Time Hypothesis (SETH) posits that we cannot solve this problem in $\mathcal{O}(2^{cN})$ time, for any $c < 1$, where N is the number of variables [IP01]. Assuming the SETH, we cannot compute the diameter (*i.e.*, the maximum eccentricity of a vertex) in $\mathcal{O}(n^{2-\varepsilon})$ time, for any $\varepsilon > 0$, even for n-vertex graphs with only $n^{1+o(1)}$ edges [RVW13].

This above hardness result only tells us that it is unlikely we can improve centrality computations for all the graphs. However, with real data at hands, it is sometimes possible to break this quadratic barrier (in the number m of edges) by exploiting some underlying structure in the data. An interval graph is a graph $G = (V, E)$ the vertices of which can be mapped to closed intervals on the real line, in such a way that two vertices $u, v \in V$ are adjacent in if and only if their respective intervals I_u, I_v intersect. Such a mapping is called an interval model of graph G. We refer to Figure 1 for an illustration.

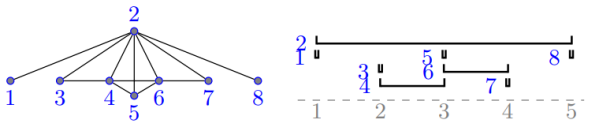

Figure 1: An interval graph along with a corresponding interval model [Cao21].

The first introduction of interval graphs is credited to Hajös and Benzer in the 1950's. This class of graphs has played a pivotal role in refining our understanding of the linear structure of DNA molecules [Ben59]. The interval graphs were used as a mathematical model for food webs, *a.k.a.* consumer-resource systems [Coh78], and ever since they found further applications in job scheduling in industry [BNBYF+01]. The first linear-time algorithm for the recognition of these graphs, and the construction of an interval model, was quite complicated due to its use of PQ-trees, an intricate tree-based data structure [BL76]. Since then, simpler linear-time algorithms were found based on alternative characterizations of these graphs [Hsu92, HMPV00, COS98].

As far as we are concerned in this paper, we can compute in linear time all vertex eccentricities in an interval graph (and so, all graph centralities) with a very simple algorithm [Ola90]. We start presenting a different algorithm for computing all vertex eccentricities in an interval graph (Theorem 1). It runs in $\mathcal{O}(m + n \log^3 n)$ time on n-vertex, m-edge interval graphs. Although it is slightly slower than the state-of-the-art algorithm for this problem, what makes this algorithm interesting is that it can be generalized to a larger class of "almost" interval graphs. We discuss about two different extensions of interval graphs in the paper. First, in his seminal work, Benzer considered 145 mutant strains of a bacteria-infecting virus, T4, for which he experimentally uncovered an interval graph structure on 144 of the 145 strains. These results imply that interval graphs are a suitable model for the interactions between *most* fragments of genetic materials in some viruses, but not necessarily for *all* of it. Let us call a graph an interval+kv graph if it can be made interval by removing at most k vertices. Second, subsequent works have evidenced that many genes are better represented by a collection of unbroken sequences of nucleotides on the DNA strand rather than just by one interval [Cha81]. Let us define a t-interval representation of a graph G as a mapping of its vertices to subsets of at most t intervals, so that two vertices are adjacent in G if and only if some of their respective intervals intersect. The interval number of a graph is the least t such that it admits a t-interval representation.

Until this paper, the fastest known algorithm in order to compute all vertex eccentricities in an interval+kv graph with n vertices was running in $\mathcal{O}(kn^2)$ time, that is only interesting if the number of edges is at least $\omega(n)$ [BN22]. The fine-grained complexity of computing all vertex eccentricities within graphs of bounded interval number was left as an open problem [DHV21].

1.1 Contributions

Our main result is an almost linear-time algorithm for computing all vertex eccentricities in an interval+kv graph, for any fixed k (Theorem 2). Specifically, if the input graph has n vertices and m edges, then for any positive value of ε, we can upper bound the running time of the algorithm by an $\mathcal{O}(2^{\delta k}(n+m)^{1+\varepsilon})$, for some constant δ depending on ε. The exponential dependency on k is shown to be necessary assuming the SETH (Lemma 7). For that we combine, it seems for the first time, the properties of interval models with some range queries techniques used in previous works [CK09] and originating from database computing [Ben79].

However, in contrast to this above positive result, we revisit a known construction in the literature [ED16] in order to show that assuming the SETH, the diameter of n-vertex m-edge graphs with interval number two cannot be computed in $\mathcal{O}(n^{1-\varepsilon}m)$ time, for any positive value ε (Theorem 3).

1.2 Comparison with previous works

There is a growing literature on the relations between range queries techniques and faster centralities computation algorithms [AVWW16, BHM20, CK09, DHV22]. Applications of these techniques to the computation of all shortest paths intersecting a bounded number of vertices were proposed [Duc22a]. This scenario is relevant for transportation networks, where most shortest paths intersect a few "hubs" in the network [CHKZ03]. Other applications of range queries techniques to graph classes with small diameter and an interval-like representation were proposed [DHV21]. To the best of our knowledge, our work is the first to combine both approaches. Some authors also have studied the relations between faster centralities computation and other techniques from Computational Geometry, such as network Voronoi diagrams [Cab18].

The only previous algorithm for centralities computation in an interval+kv graph [BN22] builds on the existence of an optimal quadratic-time algorithm for computing all distances in an interval graph [RMPR92]. Therefore, for every fixed k, their algorithm requires quadratic work space. By contrast, our algorithm builds on range queries techniques in order to store a compact version of the distance matrix of an interval graph, which only requires quasi linear work space. The design of compact distance encodings for interval graphs has predated our paper [GP08]. We here propose a different encoding than in previous works, see Lemma 4, which looks easier to incorporate within our main algorithm.

1.3 Organization of the paper

In Section 2, we first present an almost linear-time algorithm for computing all eccentricities in an interval graph. This algorithm is then extended to the interval+kv graphs, for any fixed k, in Section 3. In Section 4, we show that assuming the SETH, the existence of a faster algorithm for this problem can be ruled out for the graphs of interval number two. We end up discussing a few open questions in Section 5.

2 Interval models and graph centralities

Throughout the whole section, we assume dealing with an n-vertex interval graph $G = (V, E)$, represented as an interval model $I(G) = (I_v)_{v \in V}$. For each vertex $v \in V$, its interval I_v is encoded as the ordered pair (a_v, b_v) of its two end-points, that we can alway assume to take integer values between 1 and $2n$.

2.1 Representation of balls as intervals

The ball of center v and radius ℓ contains all vertices that are at distance at most ℓ from v. Formally, $N^\ell_G[v] = \{u \in V \mid d_G(u, v) \leq \ell\}$. The following folklore results on interval graphs are proved for completeness of the paper:

Lemma 1. *In any interval model of a graph G, for each vertex v and integer ℓ, the union $\bigcup\{I_u \mid u \in N^\ell_G[v]\}$ of all intervals representing a vertex at distance at most ℓ from v is itself an interval, denoted by $I_\ell(v)$.*

Proof. We prove the property by induction on ℓ. If $\ell = 0$, then we have $N_G^0[v] = \{v\}$, and therefore, $I_0(v) = I_v$ is an interval. From now on let $\ell > 0$. By the induction hypothesis, $I_{\ell-1}(v)$ is an interval. Let $u \in N_G^\ell[v]$ be such that a_u is minimized. In the same way, let $w \in N_G^\ell[v]$ be such that b_w is maximized. Since $u, w \in N_G^\ell[v]$, there exist vertices $u', w' \in N_G^{\ell-1}[v]$ such that $d_G(u, u'), d_G(w, w') \leq 1$. In particular, $I_u \cap I_{u'} \neq \emptyset$ and $I_w \cap I_{w'} \neq \emptyset$, that implies $I_u \cap I_{\ell-1}(v) \neq \emptyset$ and $I_w \cap I_{\ell-1}(v) \neq \emptyset$. Therefore, $I_\ell(v) = I_u \cup I_{\ell-1}(v) \cup I_w$ is also an interval. □

In general, not all vertices u such that $I_u \subseteq I_\ell(v)$ belong to $N_G^\ell[v]$. For instance, we may have $I_u \subseteq I_{u'}$ for some $u' \in N_G^\ell[v]$, and u' is on a shortest uv-path. However, we have the following slightly weaker property:

Lemma 2. *Let u and v be distinct vertices in an interval graph G. We have that $d_G(u, v) \leq \ell$ if and only if $I_u \cap I_{\ell-1}(v) \neq \emptyset$ (resp., $I_{\ell-1}(u) \cap I_v \neq \emptyset$).*

Proof. Let us first assume $d_G(u, v) \leq \ell$. Let $u' \in N_G^{\ell-1}[v] \cap N_G^1[u]$ (possibly, $u' = u$). We have that $I_u \cap I_{\ell-1}(v) \supseteq I_u \cap I_{u'} \neq \emptyset$. Conversely, let us now assume $I_u \cap I_{\ell-1}(v) \neq \emptyset$. Let $u' \in N_G^{\ell-1}[v]$ satisfy $I_u \cap I_{u'} \neq \emptyset$. We have $d_G(u, v) \leq d_G(u, u') + d_G(u', v) \leq 1 + (\ell - 1) = \ell$. □

Corollary 1. *Let u and v be distinct vertices in an interval graph G and let $\ell > j \geq 0$ be integers. We have that $d_G(u, v) \leq \ell$ if and only if $I_j(u) \cap I_{\ell-j-1}(v) \neq \emptyset$.*

Proof. Clearly, $d_G(u, v) \leq \ell$ if and only if there exists a vertex $u' \in N_G^j[u]$ such that $d_G(u', v) \leq \ell - j$. By Lemma 2, the latter is equivalent to have $I_{u'} \cap I_{\ell-j-1}(v) \neq \emptyset$. We are done as $I_j(u) = \bigcup\{I_{u'} \mid u' \in N_G^j[u]\}$. □

2.2 Fast computation of the balls

For each vertex v and integer $\ell \geq 0$, let $a_\ell(v), b_\ell(v)$ denote the end-points of the interval $I_\ell(v)$ (defined in the previous Section 2.1). Next, we present a data structure in order to efficiently compute any interval $I_\ell(v)$ (encoded by the ordered pair of its two end-points). We start with an easy observation:

Lemma 3. *Being given an interval model $I(G)$ for some n-vertex graph $G = (V, E)$ and an integer $p \geq 0$, we can compute all intervals $I_{2^j-1}(v), I_{2^j}(v)$, for every vertex v and every $0 \leq j \leq p$, in total $\mathcal{O}(np)$ time.*

Proof. We consider all values j sequentially. Let us assume to be given the intervals $I_{2^j-1}(v)$ (for $j = 0$, $I_{2^0-1}(v) = I_0(v) = I_v$ is already part of the interval model, and for $j > 0$ we may assume these intervals to be computed at the previous step $j - 1$ of the algorithm). By Lemma 2,

$$a_{2^j}(v) = \min\{a_u \mid I_u \cap I_{2^j-1}(v) \neq \emptyset\} = \min\{a_u \mid a_{2^j-1}(v) \leq b_u\}.$$

We scan the points of the model, from left to right, and we do as follows for each $i \in \{1, 2, \ldots, 2n\}$:

1. *Case $i = a_u$ for some u:* We insert a_u at the end of some auxiliary list L. At any moment during the scan, L contains all left points, in order, of all intervals I_u that are already started but not yet closed.
2. *Case $i = a_{2^j-1}(v)$ for some v:* We simply set $a_{2^j}(v)$ as the head a_u of list L.
3. *Case $i = b_u$ for some u:* We remove a_u from L. It can be done in $\mathcal{O}(1)$ time if we keep a pointer to its position in L at the time of its insertion to this list.

The overall running time is in $\mathcal{O}(n)$. In the same way,

$$b_{2^j}(v) = \max\{b_u \mid a_u \leq b_{2^j-1}(v)\}.$$

We can compute all values $b_{2^j}(v)$ in $\mathcal{O}(n)$ time by scanning the model from right to left and replacing $a_u, a_{2^j-1}(v), b_u$ by $b_u, b_{2^j-1}(v), a_u$ in the above procedure. Then, by Corollary 1,

$$a_{2^{j+1}-1}(v) = \min\{a_{2^j-1}(u) \mid a_{2^j-1}(v) \leq b_{2^j-1}(u)\}$$

and

$$b_{2^{j+1}-1}(v) = \max\{b_{2^j-1}(u) \mid a_{2^j-1}(u) \leq b_{2^j-1}(v)\}.$$

We can compute all values $a_{2^{j+1}-1}(v), b_{2^{j+1}-1}(v)$ in $\mathcal{O}(n)$ time by replacing the I_u's by the $I_{2^j-1}(u)$'s in the above procedures. □

In practice, it suffices to set $p = \lceil \log n \rceil$. We now prove that in order to compute any interval $I_\ell(u)$, it suffices to store all intervals $I_{2^j-1}(v), I_{2^j}(v)$ (pre-computed by applying Lemma 3) in some suitable data structure. A 2-range tree stores a static collection P of 2-dimensional points on which we can perform the following queries: Given as inputs $\langle \alpha, \beta, \gamma, \delta \rangle$, we can either output

$$\min\{a \mid (a,b) \in P \cap ([\alpha;\beta] \times [\gamma;\delta])\} \text{ (min-query)}$$

or

$$\max\{b \mid (a,b) \in P \cap ([\alpha;\beta] \times [\gamma;\delta])\} \text{ (max-query)}.$$

Other types of queries can be also supported (see Section 3 for an example). If we need to store n points, then we can construct a 2-range tree in $\mathcal{O}(n \log n)$ time, so that any query can be answered in $\mathcal{O}(\log n)$ time [Cha90].

Lemma 4. *Being given an interval model $I(G)$ for some n-vertex graph $G = (V, E)$, after a pre-processing in $\mathcal{O}(n \log^2 n)$ time we can compute any interval $I_\ell(u)$ in $\mathcal{O}(\log^2 n)$ time.*

Proof. First, we apply Lemma 3 for $p = \lceil \log n \rceil$. For every $0 \leq j \leq p$, we construct a 2-range tree in order to store all the points $(a_{2^j-1}(v), b_{2^j-1}(v))$. In doing so, we construct $\mathcal{O}(\log n)$ 2-range trees, that can be done in $\mathcal{O}(n \log^2 n)$ time. Then, let us assume that we want to compute $I_\ell(u)$, for some u and ℓ. We may assume that $\ell \neq 0$ (else, $I_0(u) = I_u$ is already part of the model $I(G)$). Let us write $\ell = \sum_{i=1}^{q} 2^{j_i}$ for some $p \geq j_1 > j_2 > \ldots > j_q \geq 0$. We construct the intervals $I_{\ell_t}(u)$ sequentially, where $\ell_t = \sum_{i=1}^{t} 2^{j_i}$ for every $1 \leq t \leq q$. Since ℓ_1 is a power of two, we already pre-computed $I_{\ell_1}(u)$ for any u. From now on, we assume that $q > t > 1$. By Corollary 1,

$$a_{\ell_{t+1}}(u) = \min\{a_{2^{j_{t+1}}-1}(w) \mid a_{\ell_t}(u) \leq b_{2^{j_{t+1}}-1}(w)\}$$

and

$$b_{\ell_{t+1}}(u) = \max\{b_{2^{j_{t+1}}-1}(w) \mid a_{2^{j_{t+1}}-1}(w) \leq b_{\ell_t}(u)\}.$$

In particular, we can compute $a_{\ell_{t+1}}(u), b_{\ell_{t+1}}(u)$ in $\mathcal{O}(\log n)$ time with two queries on the j_{t+1}^{th} 2-range tree. Overall, the total time for computing $I_{\ell_q}(u) = I_\ell(u)$ is in $\mathcal{O}(q \log n) = \mathcal{O}(\log^2 n)$. □

2.3 The algorithm

Theorem 1. *All eccentricities in an interval graph $G = (V, E)$ with n vertices and m edges can be computed in $\mathcal{O}(m + n \log^3 n)$ time (resp., in $\mathcal{O}(n \log^3 n)$ time if an interval model is given in advance).*

Proof. First, we compute in $\mathcal{O}(m + n)$ time an interval model $I(G)$. We add all points (a_w, b_w), for every vertex $w \in V$, in a 2-range tree. Then, we apply Lemma 4. Given some positive values $(\ell_u)_{u \in V}$, we present an algorithm in order to determine for each vertex u separately whether $e_G(u) \leq \ell_u$. Indeed, doing so we can compute all eccentricities by applying $\mathcal{O}(\log n)$ times this algorithm, performing on our way n simultaneous binary searches. By Lemma 2 we have $e_G(u) \leq \ell_u$ if and only if $I_{\ell_u - 1}(u) \cap I_w \neq \emptyset$ for every vertex $w \in V$. In order to check whether it is the case, we apply Lemma 4 in order to compute the interval $I_{\ell_u-1}(u)$. Then, we test for the existence of an interval I_w such that either $b_w < a_{\ell_u-1}(u)$ or $a_w > b_{\ell_u-1}(u)$. It can be done in $\mathcal{O}(\log n)$ time with two queries on our 2-range tree. Overall, our intermediate algorithm runs in $\mathcal{O}(\log^2 n)$ time per vertex, hence in $\mathcal{O}(n \log^2 n)$ time, and therefore the total running time in order to compute all vertex eccentricities (being given $I(G)$) is in $\mathcal{O}(n \log^3 n)$. □

3 Generalization to almost interval graphs

The purpose of this section is to generalize Theorem 1 to the interval+kv graphs, for any fixed k. We first need the following result:

Lemma 5 ([Cao16]). *Let $G = (V, E)$ be an n-vertex m-edge graph. For any k, we can decide whether G is an interval+kv graph in $\mathcal{O}(6^k(n+m))$ time. If yes, we can compute in the same time a k-subset S such that $G \setminus S$ is an interval graph.*

In general, the output interval graph $G \setminus S$ may be disconnected, even if G is connected. However, we observe that the techniques used for Theorem 1 can be applied to every connected component of $G \setminus S$ separately, with no overhead in the total running time. As for the vertices in S, we may compute their eccentricities directly (and in fact, all the distances $d_G(s, v)$, for every vertices $s \in S$ and $v \in V$), in total $\mathcal{O}(km)$ time. The main issue to be resolved is to determine, for every two vertices $u, v \in V \setminus S$, whether some shortest path is fully into $V \setminus S$ (in which case, we can apply Theorem 1), or all their shortest paths go by S.

A k-range tree is a data structure storing a static set of k-dimensional points, on which we can perform the following counting queries: given lower and upper bounds α_i and β_i for every dimension i, $1 \leq i \leq k$, return the number of points $(x_1, x_2, \ldots, x_k)$ in the collection so that $\alpha_i \leq x_i \leq \beta_i$ for every $1 \leq i \leq k$. Other types of queries can be also supported.

Lemma 6 ([BHM20]). *Let $B(n, k) = \binom{\lceil \log n \rceil + k}{k}$. Being given n points, we can construct a k-range tree in $\mathcal{O}(k^2 B(n, k)n)$ time, in such a way that each query can be answered in $\mathcal{O}(2^k B(n, k))$ time. Moreover, for every $\varepsilon > 0$, there exists a $\delta > 0$ such that $B(n, k) = 2^{\delta k} n^{\varepsilon}$.*

We combine Lemma 6 and Theorem 1 in order to derive our main result in the paper:

Theorem 2. *All eccentricities in an interval+kv graph $G = (V, E)$, with n vertices and m edges, can be computed in $\mathcal{O}(6^k(m + kB(n, k+2)n \log n))$ time.*

Proof. We first apply Lemma 5, that results in a k-subset S and an interval graph $H = G \setminus S$. Then, we execute a breadth-first search from every vertex $s \in S$, thus computing the distances $d_G(v, s)$ for every vertex v. It takes $\mathcal{O}(km)$ time. In doing so, we computed the eccentricities $e_G(s)$ for every vertex $s \in S$. We also apply Lemma 4 to H. Now, in order to compute all remaining eccentricities, as already noticed in the proof of Theorem 1, it suffices to call $\mathcal{O}(\log n)$ times an algorithm solving the following decision problem: being given positive values ℓ_u, $u \in V \setminus S$, decide for each vertex u separately whether $e_G(u) \leq \ell_u$. For that, for each vertex u we start computing the interval $I_{\ell_u - 1}(u)$ which represents $N_H^{\ell_u - 1}[u]$. It can be done in $\mathcal{O}(n \log^2 n)$ time by applying Lemma 4. Furthermore, we may assume without loss of generality that $\ell_u \geq \max\{d_G(u, s) \mid s \in S\}$ for every vertex u.

1. For every vertex u, we compute the number $n_0(u)$ of vertices in $N_H^{\ell_u}[u]$. By Lemma 2, these are exactly the vertices w such that $I_w \cap I_{\ell_u - 1}(u) \neq \emptyset$. Therefore, in order to compute all values $n_0(u)$, we first add all points (a_u, b_u) in a 2-range tree, in $\mathcal{O}(n \log n)$ time. Then, for each vertex u we compute the number of such points (a_w, b_w) such that either $a_w \leq a_{\ell_u - 1}(u) \leq b_w$ or $a_{\ell_u - 1}(u) < a_w \leq b_{\ell_u - 1}(u)$, that is exactly $n_0(u)$. It only requires two counting queries per vertex.

2. Let $S = \{s_1, s_2, \ldots, s_k\}$ be arbitrarily ordered. For every $1 \leq i \leq k$ and for every vertex u, we compute the number $n_i(u)$ of vertices w in $N_G^{\ell_u}[u] \setminus \left(N_H^{\ell_u}[u] \cup S\right)$ such that: $d_G(u, s_i) + d_G(s_i, w) \leq \ell_u$; $d_G(u, s_j) + d_G(s_j, w) > \ell_u$ for every $1 \leq j < i$. For that, for every vertex u, we create a $(k+2)$-dimensional point
$$\overrightarrow{p_u} = (a_u, b_u, d_G(u, s_1), d_G(u, s_2), \ldots, d_G(u, s_k)),$$
that we add in some $(k+2)$-range tree. It takes $\mathcal{O}(k^2 B(n, k+2)n)$ time. Now, for every $1 \leq i \leq k$ and for every vertex u, we count the number of such points $\overrightarrow{p_w}$ that satisfy:

- Either $b_w < a_{\ell_u - 1}(u)$ or $a_w > b_{\ell_u - 1}(u)$;
- $d_G(s_j, w) > \ell_u - d_G(u, s_j)$ for every $1 \leq j < i$;
- $d_G(s_i, w) \leq \ell_u - d_G(u, s_i)$;

That is exactly $n_i(u)$. The last two constraints impose each of the last k coordinates of the point to belong to some fixed range, while the first constraint enforces the two first coordinates to belong to either of two disjoint ranges. As a result, we can compute $n_i(u)$ with two counting queries. It takes $\mathcal{O}(2^k B(n, k+2))$ time per vertex, and it needs to be done k times (once per index i).

For every vertex u, the number of vertices in $N_G^{\ell_u}[u] \setminus S$ is exactly $\sum_{i=0}^{k} n_i(u)$. Hence, in order to decide whether $e_G(u) \leq \ell_u$, it suffices to check whether this number is equal to $n - k$. □

We note that the dependency on k is exponential. We end up this section by showing this is unavoidable, assuming the SETH. Recall for what follows that the diameter of a graph is its maximum eccentricity.

Lemma 7. *Assuming the SETH, we cannot compute the diameter of interval+kv graphs, with n vertices and $n^{1+o(1)}$ edges, in $\mathcal{O}(2^{o(k)} n^{2-\varepsilon})$ time for any $\varepsilon > 0$.*

Proof. A split graph is a graph $G = (K \cup S, E)$ that can be vertex-partitioned into a clique K and a stable set S. Assuming the SETH, we cannot compute the diameter of n-vertex split graphs G in $\mathcal{O}(n^{2-\varepsilon})$ time, for any $\varepsilon > 0$, even if $|K| = k = \mathcal{O}(\log n)$ [AVWW16, BCH16]. Note that in this situation, G only has $\mathcal{O}(nk + k^2) = n^{1+o(1)}$ edges. Furthermore, G is an interval+kv graph because it suffices to remove all vertices in the clique K in order to obtain an edgeless graph, a special case of interval graph. □

4 SETH-hardness results

We now prove that the existence of a faster algorithm for centralities computation is unlikely for the graphs of bounded interval number, even if the latter is only two. For that, we use the following Orthogonal Vector problem (OV): being given two set families A and B over a common universe C, decide whether there exist sets $a \in A,\ b \in B$ such that $a \cap b = \emptyset$.

Lemma 8 ([AVWW16])**.** *Assuming the SETH, for every $\varepsilon > 0$, there is some $c > 0$ such that we cannot solve OV in $\mathcal{O}(n^{2-\varepsilon})$ time, even if $|A| = |B| = n$ and $|C| \leq c \cdot \log n$.*

The following construction was inspired by Lemma 8. It transforms any instance (A, B, C) of OV, with $|A| = |B| = n$ and $|C| = \mathcal{O}(\log n)$, into a graph $G\langle A, B, C\rangle$, as follows:

- For every $a \in A$, we add a balanced binary rooted tree T^a with root r_a and $|C|$ leaves, indexed by the elements in C. Similarly, for every $b \in B$, we add a balanced binary rooted tree T^b with root r_b and $|C|$ leaves, also indexed by the elements in C.
- For every $c \in C$, we add two balanced binary rooted trees $T^{c,A}$ and $T^{c,B}$, with n leaves each, that are indexed by the sets in A and B respectively, and a common root vertex r_c.
- We add two more trees T^A, T^B, with n leaves each, that are indexed by the sets in A and B respectively.
- Fix some $p = \omega(\log n)$ for the remainder of the construction. For each $a \in A$ and $c \in a$, we add a path $P_{a,c}$ of length p between the leaves of indices c and a in $T^a, T^{c,A}$ respectively. Similarly, for each $b \in B$ and $c \in b$, we add a path $P_{b,c}$ of length p between the leaves of indices c and b in $T^b, T^{c,B}$ respectively.
- Finally, for every $a \in A$ we add a path $P_{a,A}$ of length p between r_a and the leaf of index a in T^A. We further add a path P_a of length p between r_a and some new node. In the same way, for every $b \in B$ we add a path $P_{b,B}$ of length p between r_b and the leaf of index b in T^B. We also add a path P_b of length p between r_b and some new node.

All graphs $G\langle A, B, C\rangle$ can be constructed in $n^{1+o(1)}$ time; furthermore, assuming the SETH, we cannot compute their diameter in $\mathcal{O}(n^{2-\varepsilon})$ time, for any $\varepsilon > 0$ [ED16]. We prove the following additional property of these graphs:

Lemma 9. *Every graph $G\langle A, B, C\rangle$ has interval number at most two.*

Proof. We start with the following useful construction. Namely, we construct a 2-interval representation for any rooted tree T as follows. For every internal node x we create disjoint intervals $I^1_{y_1}, I^1_{y_2}, \ldots, I^1_{y_d}$ for its children $y_1, y_2, \ldots, y_d$ and then we create the interval $I^0_x \supseteq \bigcup_{i=1}^{d} I^1_{y_i}$ for x. Let us call this model the canonical 2-representation of tree T. All nodes in this canonical 2-representation have exactly two intervals, except the root and the leaves which have only one interval each.

The graph $G\langle A, B, C\rangle$ can be vertex-covered by the following collection of rooted trees:

- $T^a \cup P_a \cup P_{a,A} \cup \{P_{a,c} \mid c \in a\}$, for every $a \in A$, with root r_a;
- $T^b \cup P_b \cup P_{b,B} \cup \{P_{b,c} \mid c \in b\}$, for every $b \in B$, with root r_b;
- $T^{c,A} \cup T^{c,B}$, for every $c \in C$, with root r_c;
- T^A, T^B.

We now take the union of the canonical 2-representations of all these trees above. Every node that appears in only one tree is associated to at most two intervals. The only nodes appearing in multiple trees are: the leaves of T^A (each appears as a leaf in the tree rooted at r_a, for one set $a \in A$), the leaves of T^B (each appears as a leaf in the tree rooted at r_b, for one set $b \in B$), the leaves of $T^{c,A}$ for every $c \in C$ (each appears as a leaf in the tree rooted at r_a, for at most one set $a \in A$) and the leaves of $T^{c,B}$ for every $c \in C$ for every $b \in B$ (each appears as a leaf in the tree rooted at r_b, for at most one set $b \in B$). In particular, each such node only appears in two trees, and as a leaf, therefore it is associated to two intervals. As a result, the union of the canonical 2-representations of all trees above is a 2-interval representation of $G\langle A, B, C\rangle$. □

Overall, the following result is derived by the combination of [ED16] with Lemma 9:

Theorem 3. *Assuming the SETH, we cannot compute the diameter of n-vertex graphs in $\mathcal{O}(n^{2-\varepsilon})$ time, for any $\varepsilon > 0$, even if they only have $n^{1+o(1)}$ edges and interval number at most two.*

5 Conclusion and open perspectives

In this paper, we completely settled the complexity of computing all vertex eccentricities (and therefore, all graph centralities) within the interval+kv graphs for every fixed k and within the graphs of bounded interval number. Let us briefly summarize what is known insofar for other important generalizations of the interval graphs:

Boxicity. The boxicity of a graph $G = (V, E)$ is the least integer k such that there exist interval graphs $G_1, \ldots, G_k$ with same vertex-set V and so that E is exactly the set of all common edges to $E(G_1), \ldots, E(G_k)$. Assuming the SETH, the diameter of n-vertex graphs with $n^{1+o(1)}$ edges cannot be computed in $\mathcal{O}(n^{2-\varepsilon})$ time, for any $\varepsilon > 0$, even if the graphs considered are intersection graphs of axis-parallel line segments [BKK+22]. Such graphs have boxicity at most two [HNZ91].

Track number. The track number of a graph $G = (V, E)$ is the least integer k such that there exist interval graphs $G_1, \ldots, G_k$ with same vertex-set V and so that E is exactly the union of all edges in $E(G_1), \ldots, E(G_k)$. Every bounded-degree graph also has bounded track-number [KD94], and therefore a faster algorithm for computing all vertex eccentricities in these graphs is unlikely to exist [ED16]. However, either proving or disproving the existence of such faster algorithms for the special case of graphs with track number *two* seems to be open.

Interval probe graphs. Last, we call a graph $G = (V, E)$ an interval probe graph if for some bipartition $P \cup N$ of its vertices, where N is an independent set, we can make G an interval graph by only adding edges between vertices in N. It follows from [She99] that every interval probe graph $G = (P \cup N, E)$ has asteroidal number at most $|N| + 2$. Therefore, we can compute its diameter in $\mathcal{O}(|N|^3 \cdot m^{3/2})$ time, where m is the number of edges [Duc22b]. However, whether we can remove the dependency on N is an open problem.

Acknowledgement. This work was supported by a grant of the Ministry of Research, Innovation and Digitalization, CCCDI - UEFISCDI, project number PN-III-P2-2.1-PED-2021-2142, within PNCDI III.

References

[AVWW16] 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 Symposium on Discrete Algorithms (SODA)*, pages 377–391. SIAM, 2016.

[BBR+12] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna. Four degrees of separation. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 33–42. ACM, 2012.

[BCH16] 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.

[Ben59] S. Benzer. On the topology of the genetic fine structure. *Proceedings of the National Academy of Sciences*, 45(11):1607–1620, 1959.

[Ben79] J.L. Bentley. Decomposable searching problems. *Information Processing Letters*, 8(5):244–251, 1979.

[BHM20] K. Bringmann, T. Husfeldt, and M. Magnusson. Multivariate Analysis of Orthogonal Range Searching and Graph Distances. *Algorithmica*, pages 1–24, 2020.

[BKK+22] K. Bringmann, S. Kisfaludi-Bak, M. Künnemann, A. Nusser, and Z. Parsaeian. Towards Sub-Quadratic Diameter Computation in Geometric Intersection Graphs. In *38th International Symposium on Computational Geometry (SoCG 2022)*, volume 224 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

[BL76] K.S. Booth and G.S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976.

[BM08] J. A. Bondy and U. S. R. Murty. *Graph theory*. Springer, 2008.

[BN22] M. Bentert and A. Nichterlein. Parameterized complexity of diameter. *Algorithmica*, pages 1–27, 2022.

[BNBYF+01] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM (JACM)*, 48(5):1069–1090, 2001.

[Cab18] S. Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Transactions on Algorithms*, 15(2):21, 2018.

[Cao16] Y. Cao. Linear recognition of almost interval graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 1096–1115. SIAM, 2016.

[Cao21] Y. Cao. Recognizing (Unit) Interval Graphs by Zigzag Graph Searches. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 92–106. SIAM, 2021.

[Cha81] P. Chambon. Split genes. *Scientific American*, 244(5):60–71, 1981.

[Cha90] B. Chazelle. Lower bounds for orthogonal range searching: I. the reporting case. *Journal of the ACM (JACM)*, 37(2):200–212, 1990.

[CHKZ03] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick. Reachability and distance queries via 2-hop labels. *SIAM Journal on Computing*, 32(5):1338–1355, 2003.

[CK09] S. Cabello and C. Knauer. Algorithms for graphs of bounded treewidth via orthogonal range searching. *Computational Geometry*, 42(9):815–824, 2009.

[Coh78] J.E. Cohen. *Food webs and niche space*. Princeton University Press, 1978.

[COS98] D.G. Corneil, S. Olariu, and L. Stewart. The ultimate interval graph recognition algorithm? In *SODA*, volume 98, pages 175–180. SIAM, 1998.

[DHV21] G. Ducoffe, M. Habib, and L. Viennot. Fast Diameter Computation within Split Graphs. *Discrete Mathematics & Theoretical Computer Science*, 23, 2021.

[DHV22] 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.

[DSP18] K. Das, S. Samanta, and M. Pal. Study on centrality measures in social networks: a survey. *Social network analysis and mining*, 8(1):1–11, 2018.

[Duc22a] G. Ducoffe. Eccentricity queries and beyond using hub labels. *Theoretical Computer Science*, 930:128–141, 2022.

[Duc22b] G. Ducoffe. Obstructions to faster diameter computation: Asteroidal sets. In *IPEC*, pages 15:1–15:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

[ED16] J. Evald and S. Dahlgaard. Tight Hardness Results for Distance and Centrality Problems in Constant Degree Graphs. Technical Report arXiv:1609.08403, ArXiv, 2016.

[GP08] C. Gavoille and C. Paul. Optimal distance labeling for interval graphs and related graph families. *SIAM Journal on Discrete Mathematics*, 22(3):1239–1258, 2008.

[HH95] P. Hage and F. Harary. Eccentricity and centrality in networks. *Social networks*, 17(1):57–63, 1995.

[HMPV00] M. Habib, R. McConnell, C. Paul, and L. Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234(1-2):59–84, 2000.

[HNZ91] I.B.-A. Hartman, I. Newman, and R. Ziv. On grid intersection graphs. *Discrete Mathematics*, 87(1):41–52, 1991.

[Hsu92] W.-L. Hsu. A simple test for interval graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 11–16. Springer, 1992.

[IP01] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[KD94] N. Kumar and N. Deo. Multidimensional interval graphs. *Congressus Numerantium*, 102:45–56, 1994.

[Ola90] 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.

[RMPR92] R. Ravi, M.V. Marathe, and C. Pandu Rangan. An optimal algorithm to solve the all-pair shortest path problem on interval graphs. *Networks*, 22(1):21–35, 1992.

[RVW13] 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*, pages 515–524, 2013.

[She99] L. Sheng. Cycle free probe interval graphs. *Congressus Numerantium*, 140:33–42, 1999.