The power of interval models for computing graph centralities

Guillaume DUCOFFE

National Institute for Research and Development in Informatics – ICI Bucharest, Romania
guillaume.ducoffe@ici.ro

University of Bucharest, Faculty of Mathematics and Computer Science, 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. This question is addressed 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. A positive answer is given for the interval $+kv$ graphs and a negative one (assuming plausible complexity hypotheses) for the graphs of bounded interval number. In particular, an almost linear-time algorithm for computing all vertex eccentricities in an interval$+kv$ graph, for any fixed k, is presented, thus improving on the recent quadratic-time algorithm of (Bentert & Nichterlein, 2022) for this problem.

Keywords: interval graphs, interval number, distance to triviality, diameter, eccentricities computation, SETH-hardness.

Puterea modelelor de interval în problema calculării indicilor de centralitate

Rezumat: Modelarea "structurii liniare" care este prezentă în mai multe sisteme, precum rețelele trofice, problemele de alocare a resurselor și ADN-ul, are la bază *grafurile de interval* (grafuri intersecție a unor intervale de pe o linie dreapta). Totodată, având în vedere că mulțimile de date reale pot conține greșeli și zgomot, este nevoie de a extinde rezultatele algoritmice obținute pentru grafurile de interval la unele clase de grafuri mai generale. Această problematică este abordată în contextul calculării *excentricităților*, datorită importanței acestora privind clasificarea nodurilor unei rețele. O astfel de generalizare este obținută de la grafurile de interval până la grafurile interval$+kv$ (obținute prin adăugarea a k noduri într-un graf de interval). În special, un algoritm care rulează în timp cvasiliniar, pentru orice valoare k fixată, este propus. Până în prezent, timpul de rulare al celui mai bun algoritm pentru această problema era cvadratic în numărul de noduri (Bentert & Nichterlein, 2022). Cu toate aceasta, este demonstrat sub unele ipoteze standard de complexitate că o astfel de generalizare nu este posibilă pentru grafurile al cărui număr de interval este limitat la o constantă.

Cuvinte cheie: grafuri de interval, număr de interval, algoritmi adaptivi, calcularea excentricităților, ipoteza timpului exponențial puternic.

1. Introduction

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, is studied (the larger the centrality of a node is, the more important it should be). As is standard in such theoretical investigations, a network is represented by a graph. For any undefined graph notions and terminology in what follows, see (Bondy & Murty, 2008). 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. It is denoted 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 $\frac{1}{e_G(v)}$,

https://doi.org/10.33436/v32i4y202203

where $e_G(v) = \max_{u \in V} d_G(u, v)$ (Hage & Harary, 1995). The value $e_G(v)$ is also called the eccentricity of vertex v. There also exist other centrality indices than the one discussed above (Das et al., 2018).

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 centre 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 centre, or even better, all vertex eccentricities, has received considerable attention. An introduction to all-pairs shortest paths computation can be found in (Balan, 1992). In particular, it is well-known that for every n-vertex m-edge graph all vertex eccentricities can be computed in $O(nm)$ 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 (Backstrom et al., 2012). The existence of a faster algorithm for computing all vertex eccentricities was open for decades, until it was solved in the negative by (Roditty & Vassilevska Williams, 2013). 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 this problem cannot be solved in $O(2^{cN})$ time, for any $c < 1$, where N is the number of variables (Impagliazzo & Paturi, 2001). Assuming the SETH, the diameter (*i.e.*, the maximum eccentricity of a vertex) cannot be computed in $O(n^{2-\epsilon})$ time, for any $\epsilon > 0$, even for n-vertex graphs with only $n^{1+o(1)}$ edges (Roditty & Vassilevska Williams, 2013).

This above hardness result only implies that an improved algorithm for centrality computations is unlikely to exist 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. The reader is referred to Figure 1 for an illustration. Other well-structured graph classes and their applications are surveyed in (Lupu, 2001).

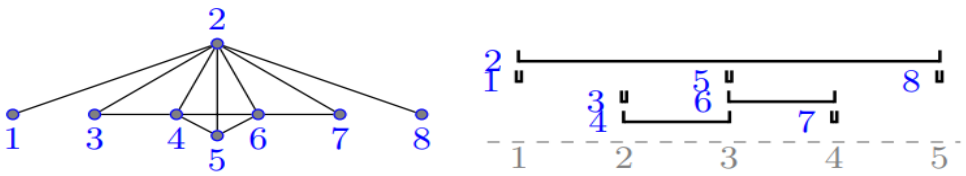


Figure 1. An interval graph along with a corresponding interval model (Cao, 2021)

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 (Benzer, 1959). The interval graphs were used as a mathematical model for food webs, a.k.a. consumer-resource systems (Cohen, 1978), and ever since they found further applications in job scheduling in industry (Bar-Noy et al., 2001). 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 (Booth & Lueker, 1976). Since then, simpler linear-time algorithms were found based on alternative characterizations of these graphs (Hsu, 1992), (Habib et al., 2000), (Corneil et al., 1998).

As far as this paper is concerned, all vertex eccentricities in an interval graph (and so, all graph centralities) can be computed in linear time with a very simple algorithm (Olariu, 1990). A different algorithm for computing all vertex eccentricities in an interval graph is presented (Theorem 1). It runs in $O(m + n \cdot log^3 n)$ 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. Two different extensions of interval

graphs are discussed 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 a graph be called 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 (Chambon, 1981). 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 $O(kn^2)$ time, which is only interesting if the number of edges is at least $\omega(n)$ (Bentert & Nichterlein, 2022). The fine-grained complexity of computing all vertex eccentricities within graphs of bounded interval number was left as an open problem (Ducoffe et al., 2021).

1.1. Contributions

The main result in the paper 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, the running time of the algorithm can be upper bounded by an $O(2^{\delta k}(n+m)^{1+\epsilon})$, 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 (Cabello & Knauer, 2009) and originating from database computing (Bentley, 1979).

However, in contrast to this above positive result, a known construction in the literature is revisited (Dahlgaard & Evald, 2016) in order to show that assuming the SETH, the diameter of n-vertex m-edge graphs with interval number two cannot be computed in $O(n^{1-\epsilon}m)$ 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 (Abboud et al., 2016), (Bringmann et al., 2020), (Cabello & Knauer, 2009), (Ducoffe et al., 2022). Applications of these techniques to the computation of all shortest paths intersecting a bounded number of vertices were proposed (Ducoffe, 2022a). This scenario is relevant for transportation networks, where most shortest paths intersect a few "hubs" in the network (Cohen et al., 2003). Other applications of range queries techniques to graph classes with small diameter and an interval-like representation were proposed (Ducoffe et al., 2021). To the best of our knowledge, this 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 (Cabello, 2018).

The only previous algorithm for centralities computation in an interval $+kv$ graph (Bentert & Nichterlein, 2022) builds on the existence of an optimal quadratic-time algorithm for computing all distances in an interval graph (Ravi et al., 1992). Therefore, for every fixed k, their algorithm requires quadratic work space. By contrast, the presented 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 this paper (Gavoille & Paul, 2008). A different encoding than in previous works is proposed, see Lemma 4, which looks easier to incorporate within the main algorithm.

1.3. Organization of the paper

Section 2 first presents 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. Section 4 shows that assuming the SETH, the existence of a faster algorithm for this problem can be ruled out for the graphs of interval number two. A few open questions are finally discussed in Section 5.

2. Interval models and graph centralities

Throughout the whole section, an n-vertex interval graph $G = (V, E)$ is considered, which is 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 can always be assumed to take integer values between 1 and $2n$.

2.1. Representation of balls as intervals

The ball of centre v and radius ℓ contains all vertices that are at distance at most ℓ from v. Formally, $N_G^\ell[v] = \{ u \in V \mid d_G(u, v) \leq \ell \}$. 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 $\cup\{I_u \mid u \in N_G^\ell[v]\}$ of all intervals representing a vertex at distance at most ℓ from v is itself an interval, denoted by $I_\ell(v)$.

Proof. The property is proven by induction on ℓ. If $\ell = 0$, then $N_G^0[v] = \{v\}$, 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, there may exist vertices u, u' such that $I_u \subseteq I_{u'}$, $u' \in N_G^\ell[v]$, and u' is on a shortest uv-path. However, the following slightly weaker property is true:

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. If $d_G(u, v) \leq \ell$, then there exists a vertex $u' \in N_G^{\ell-1}[v] \cap N_G^1[u]$ (possibly, $u' = u$) such that $I_u \cap I_{\ell-1}(v) \supseteq I_u \cap I_{u'} \neq \emptyset$. Conversely, if $I_u \cap I_{\ell-1}(v) \neq \emptyset$ then let $u' \in N_G^{\ell-1}[v]$ satisfy $I_u \cap I_{u'} \neq \emptyset$. In this situation, $d_G(u, v) \leq d_G(u, u') + d_G(u', v) \leq 1 + (\ell - 1) = \ell$. ■

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 $I_j(u) = \cup\{ I_{u'} \mid u' \in N_G^j[u]\}$. ■

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, a data structure is presented 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 $p \geq 0$, all intervals $I_{2^j-1}(v), I_{2^j}(v)$, for every vertex v and every $0 \leq j \leq p$, can be computed in total $O(np)$ time.

Proof. All values j are considered sequentially. Let the intervals $I_{2^j-1}(v)$ be given (for $j = 0$, $I_{2^0-1}(v) = I_0(v) = I_v$ is already part of the interval model, and for $j > 0$ these intervals are computed at the previous step $j - 1$ of the algorithm). By Lemma 2,

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

The points of the model are scanned, from left to right, and each point $i \in \{1,2,\ldots,2n\}$ is processed as follows:

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 $O(1)$ 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 $O(n)$. In the same way,

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

All values $b_{2^j}(v)$ can be computed in $O(n)$ 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

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

All values $a_{2^{j+1}-1}(v), b_{2^{j+1}-1}(v)$ can be computed in $O(n)$ 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 $p = \lceil \log n \rceil$. It is now proven 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 the following queries can be performed: Given as inputs $\langle \alpha, \beta, \gamma, \delta \rangle$, 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 there are n points that need to be stored, then a 2-range tree can be constructed in $O(n \log n)$ time, so that any query can be answered in $O(\log n)$ time (Chazelle, 1990).

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

Proof. First, Lemma 3 is applied 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, $O(\log n)$ 2-range trees are

constructed, that can be done in $O(n \log^2 n)$ time. Then, let $I_\ell(u)$ be some interval to be computed, for some u and ℓ. It may be assumed that $\ell \neq 0$ (else, $I_0(u) = I_u$ is already part of the model $I(G)$). Let ℓ be written as $\ell = \sum_{1 \le i \le q} 2^{j_i}$ for some $p \ge j_1 > j_2 > \cdots > j_q \ge 0$. The intervals $I_{\ell_t}(u)$ are constructed sequentially, where $\ell_t = \sum_{1 \le i \le t} 2^{j_i}$ for every $1 \le t \le q$. Since ℓ_1 is a power of two, $I_{\ell_1}(u)$ is already pre-computed for any u. From now on, $q > t > 1$ is assumed. By Corollary 1,

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

and

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

In particular, the two of $a_{\ell_{t+1}}(u), b_{\ell_{t+1}}(u)$ can be computed in $O(log\, n)$ 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 $O(q \log n) = O(\log^2 n)$. ■

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 $O(m + n\, log^3 n)$ time (resp., in $O(n\, log^3 n)$ time if an interval model is given in advance).*

Proof. First, an interval model $I(G)$ is computed in $O(n + m)$ time. All points (a_w, b_w) are added, for every vertex $w \in V$, in a 2-range tree. Then, Lemma 4 is applied. Given some positive values $(\ell_u), u \in V$, an algorithm is presented in order to determine for each vertex u separately whether $e_G(u) \le \ell_u$. Indeed, doing so all eccentricities can be computed by applying $O(log\, n)$ times this algorithm, performing n simultaneous binary searches. By Lemma 2, $e_G(u) \le \ell_u$ 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, Lemma 4 is applied 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 $O(log\ n)$ time with two queries on our 2-range tree. Overall, the intermediate algorithm runs in $O(log^2 n)$ time per vertex, hence in $O(n\, log^2 n)$ time, and therefore the total running time in order to compute all vertex eccentricities (being given $I(G)$) is in $O(n\, log^3 n)$. ■

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. *(Cao, 2016) 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 $O(6^k(n + m))$ 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, it can be observed 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, their eccentricities may be computed directly (and in fact, all the distances $d_G(s, v)$, for every vertices $s \in S$ and $v \in V$), in total $O(km)$ 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, Theorem 1 is applied), 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 $i, 1 \le i \le k$, return the number of points $(x_1, x_2, \ldots, x_k)$ in the collection so that $\alpha_i \le x_i \le \beta_i$ for every $1 \le i \le k$. Other types of queries can be also supported.

Lemma 6. *(Bringmann et al., 2020) Let* $B(n,k) = \binom{\lceil \log n \rceil + k}{k}$. *Being given n points, a* k *−range tree can be constructed in* $O(k^2 B(n,k)n)$ *time, in such a way that each query can be answered in* $O(2^k B(n,k))$ *time. Moreover, for every* $\epsilon > 0$, *there exists a* $\delta > 0$ *such that* $B(n,k) = 2^{\delta k} n^{\epsilon}$.

Lemma 6 is combined with 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* $O(6^k(m + kB(n,k+2)n \log n))$ *time.*

Proof. Firstly, Lemma 5 is applied, that results in a k-subset S and an interval graph $H = G \setminus S$. Then, a breadth-first search is executed from every vertex $s \in S$, thus computing the distances $d_G(v,s)$ for every vertex v. It takes $O(km)$ time. In doing so, the eccentricities $e_G(s)$ were computed for every vertex $s \in S$. Lemma 4 is applied to H. Now, in order to compute all remaining eccentricities, as already noticed in the proof of Theorem 1, it suffices to call $O(\log n)$ times an algorithm solving the following decision problem: being given positive values $\ell_u, u \in V \setminus S$, 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 $O(n \log^2 n)$ time by applying Lemma 4. Furthermore, it may be assumed, 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, the number $n_0(u)$ of vertices in $N_H^{\ell_u}[u]$ is computed. 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 $n_0(u)$, all points (a_u, b_u) are added in a 2-range tree, in $O(n \log n)$ time. Then, for each vertex u the number of such points (a_w, b_w) such that either $a_w \leq a_{\ell_u - 1}(u) \leq b_w$ or $a_{\ell_u - 1}(u) < a_w \leq b_{\ell_u - 1}(u)$ is computed, that is exactly $n_0(u)$. It only requires two counting queries per vertex.
2. Let $S = \{s_1, s_2, \dots, s_k\}$ be arbitrarily ordered. For every $1 \leq i \leq k$ and for every vertex u, the number $n_i(u)$ of vertices w in $N_G^{\ell_u}[u] \setminus (N_H^{\ell_u}[u] \cup S)$ 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 $1 \leq j < i$ is computed. For that, for every vertex u, a $(k+2)$-dimensional point is created:

   $$\overrightarrow{p_u} = \left(a_u, b_u, d_G(u,s_1), d_G(u,s_2), \dots, d_G(u,s_k)\right),$$

   that is added in some $(k+2)$ −range tree. It takes $O(k^2 B(n,k+2)n)$ time. Now, for every $1 \leq i \leq k$ and for every vertex u, 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 $1 \leq j < i$;
   - $d_G(s_i,w) \leq \ell_u - d_G(u,s_i)$;

   are counted, 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 $O(2^k B(n,k+2))$ 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 $\sum_{0 \leq i \leq k} n_i(u)$. 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. This section is concluded 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, the diameter of interval $+kv$ graphs with n vertices and $n^{1+o(1)}$ edges cannot be computed in $O(2^{o(k)}n^{2-\epsilon})$ time for any $\epsilon > 0$.*

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, the diameter of n-vertex split graphs G cannot be computed in $O(2^{o(k)}n^{2-\epsilon})$ time, for any $\epsilon > 0$, even if $|K| = k = \omega(log\, n)$ (Abboud et al., 2016), (Borassi et al., 2016). Note that in this situation, G only has $O(nk + k^2) = n^{1+o(1)}$ 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

In what follows, the existence of a faster algorithm for centralities computation is proven to be unlikely for the graphs of bounded interval number, even if the latter is only two. For that, the following Orthogonal Vector problem (OV) is used: 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. *(Abboud et al., 2016) Assuming the SETH, for every $\epsilon > 0$, there is some $c > 0$ such that OV cannot be solved in $O(n^{2-\epsilon})$ 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 $C = O(log\, n)$, into a graph $G\langle A, B, C\rangle$, as follows:

- For every $a \in A$, a balanced binary rooted tree T^a is added with root r_a and $|C|$ leaves, indexed by the elements in C. Similarly, for every $b \in B$, a balanced binary rooted tree T^b is added with root r_b and $|C|$ leaves, also indexed by the elements in C;
- For every $c \in C$, two balanced binary rooted trees $T^{c,A}$ and $T^{c,B}$ are added, with n leaves each, that are indexed by the sets in A and B respectively, and a common root vertex r_c;
- Two more trees T^A, T^B are added, 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 $c \in a$, a path $P_{a,c}$ of length p is added between the leaves of indices c and a in $T^a, T^{c,A}$ respectively. Similarly, for each $b \in B$ and $c \in b$, a path $P_{b,c}$ of length p is added between the leaves of indices c and b in $T^b, T^{c,B}$ respectively;
- Finally, for every $a \in A$ a path $P_{a,A}$ of length p is added between r_a and the leaf of index a in T^A. Furthermore, a path P_a of length p is added between r_a and some new node. In the same way, for every $b \in B$ a path $P_{b,B}$ of length p is added between r_b and the leaf of index b in T^B. A path P_b of length p is also added 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, their diameter cannot be computed in $O(n^{2-\epsilon})$ time, for any $\epsilon > 0$ (Dahlgaard & Evald, 2016). The following additional property of these graphs is proven:

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, a 2-interval representation is constructed for any rooted tree T as follows. For every internal node x disjoint intervals $I^1_{y_1}, I^1_{y_2}, \ldots, I^1_{y_d}$ are created for its children $y_1, y_2, \ldots, y_d$, and then the interval $I^0_x \supseteq \bigcup_{1 \leq i \leq d} I^1_{y_i}$ is created for x. Let this model be called 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 $c \in C$ (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 (Dahlgaard & Evald, 2016) with Lemma 9:

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

5. Conclusion and open perspectives

In this paper, the complexity of computing all vertex eccentricities (and therefore, all graph centralities) is completely settled within the interval+kv graphs for every fixed k and within the graphs of bounded interval number. What is known in so far for other important generalizations of the interval graphs is briefly summarized:

Boxicity. The boxicity of a graph $G = (V, E)$ is the least integer k such that there exist interval graphs $G_1, \dots, G_k$ with same vertex-set V and so that E is exactly the set of all common edges to $E(G_1), \dots, E(G_k)$. Assuming the SETH, the diameter of n-vertex graphs with $n^{1+o(1)}$ edges cannot be computed in $O(n^{2-\epsilon})$ time, for any $\epsilon > 0$, even if the graphs considered are intersection graphs of axis-parallel line segments (Bringmann et al., 2022). Such graphs have boxicity at most two (Hartman et al., 1991).

Track number. The track number of a graph $G = (V, E)$ is the least integer k such that there exist interval graphs $G_1, \dots, G_k$ with same vertex-set V and so that E is exactly the union of all edges in $E(G_1), \dots, E(G_k)$. Every bounded-degree graph also has bounded track-number (Kumar & Deo, 1994), and therefore a faster algorithm for computing all vertex eccentricities in these graphs is unlikely to exist (Dahlgaard & Evald, 2016). 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, a graph $G = (V, E)$ is called an interval probe graph if for some bipartition $P \dot{\cup} N$ of its vertices, where N is an independent set, G can be made an interval graph by only adding edges between vertices in N. It follows from (Sheng, 1999) that every interval probe graph $G = (P \dot{\cup} N, E)$ has asteroidal number at most $|N| + 2$. Therefore, its diameter can be computed in $O(|N|^3 \cdot m^{\frac{3}{2}})$ time, where m is the number of edges (Ducoffe, 2022b). However, whether the dependency on N can be removed is an open problem.

Acknowledgements

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

1. Abboud, A., Vassilevska Williams, V. & Wang, J. (2016). Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the 27th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (pp. 377-391).
2. Backstrom, L., Boldi, P., Rosa, M., Ugander, J. & Vigna, S. (2012). Four degrees of separation. In *Proceedings of the 4th annual ACM Web Science Conference (WebSci)* (pp. 33-42).
3. Balan, D. (1992). Implementarea unui algoritm pentru determinarea drumurilor minime într-un graf. *Revista Română de Informatică și Automatică (Romanian Journal of Information Technology and Automatic Control)*, 2(3-4), 83-84.
4. Bar-Noy, A., Bar-Yehuda, R, Freund, A., Naor, J. & Schieber, B. (2001). A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5), 1069-1090.
5. Bentert, M. & Nichterlein, A. (2022). Parameterized complexity of diameter. *Algorithmica*.
6. Bentley, J. L. (1979). Decomposable searching problems. *Information Processing Letters*, 8(5), 244-251.
7. Benzer, S. (1959). On the topology of the genetic fine structure. In *Proceedings of the National Academy of Sciences*, 45(11), (pp. 1607-1620).
8. Bondy, J. A. & Murty, U.S.R. (2008). *Graph Theory.* Springer, London.
9. Booth, K. S. & Lueker, G.S. (1976). 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.
10. Borassi, M., Crescenzi, P. & Habib, M. (2016). Into the square: On the complexity of some quadratic-time solvable problems. *Electronic Notes in Theoretical Computer Science*, 322, 51-67.
11. Bringmann, K., Husfeldt, T. & Magnusson, M. (2020). Multivariate analysis of orthogonal range searching and graph distances. *Algorithmica*, 82(8), 2292-2315.
12. Bringmann, K., Kisfaludi-Bak, S., Künnemann, M., Nusser, A. & Parsaeian, Z. (2022). Towards Sub-Quadratic Diameter Computation in Geometric Intersection Graphs. In *Proceedings of the 38th International Symposium on Computational Geometry (SoCG)* (pp. 21:1-21:16).
13. Cabello, S. (2018). Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Transactions on Algorithms*, 15(2), 1-38.
14. Cabello, S. & Knauer, C. (2009). Algorithms for graphs of bounded treewidth via orthogonal range searching. *Computational Geometry*, 42(9), 815-824.
15. Cao, Y. (2016). Linear recognition of almost interval graphs. In *Proceedings of the 27th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (pp. 1096-1115).
16. Cao, Y. (2021). Recognizing (Unit) Interval Graphs by Zigzag Graph Searches. In *Proceedings of the 4th Symposium on Simplicity in Algorithms (SOSA)* (pp. 92-106).
17. Chambon, P. (1981). Split genes. *Scientific American*, 244, 60-71.
18. Chazelle, B. (1990). Lower Bounds for Orthogonal Range Searching: I. The Reporting Case. *Journal of the ACM*, 37(2), 200-212.
19. Cohen, J. E. (1978). *Food webs and niche space*. Princeton University Press.
20. Cohen, E., Halperin, E., Kaplan, H. & Zwick, U. (2003). Reachability and distance queries via 2-hop labels. *SIAM Journal on Computing*, 32(5), 1338-1355.

21. Corneil, D. G., Olariu, S. & Stewart, L. (1998). The ultimate interval graph recognition algorithm? In *Proceedings of the 9th annual Symposium on Discrete Algorithms (SODA)* (pp. 175-180).

22. Dahlgaard, S. & Evald, J. (2016). *Tight hardness results for distance and centrality problems in constant degree graphs.* Technical report, arXiv:1609.08403.

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

24. Ducoffe, G. (2022a). Eccentricity queries and beyond using hub labels. *Theoretical Computer Science*, 930, 128-141.

25. Ducoffe, G. (2022b). Obstructions to faster diameter computation: Asteroidal sets. In *Proceedings of the 17th International Symposium on Parameterized and Exact Computation (IPEC)* (pp. 15:1-15:25).

26. Ducoffe, G., Habib, M. & Viennot, L. (2021). Fast Diameter Computation within Split Graphs. *Discrete Mathematics & Theoretical Computer Science*, 23(3), #11.

27. Ducoffe, G., Habib, M. & Viennot, L. (2022). Diameter, eccentricities and distance oracle computations on H-minor free graphs and graphs of bounded (distance) VC-dimension. *SIAM Journal on Computing*, 51(5), 1506-1534.

28. Gavoille, C. & Paul, C. (2008). Optimal distance labeling for interval graphs and related graph families. *SIAM Journal on Discrete Mathematics*, 22(3), 1239-1258.

29. Habib, M., McConnell, R., Paul, C. & Viennot, L. (2000). LexBFS and Partition Refinement, with Applications to Transitive Orientation, Interval Graph Recognition and Consecutive Ones Testing. *Theoretical Computer Science*, 234(1-2), 59-84.

30. Hage, P. & Harary, F. (1995). Eccentricity and centrality in networks. *Social networks*, 17(1), 57-63.

31. Hartman, I. B. A., Newman, I. & Ziv, R. (1991). On grid intersection graphs. *Discrete Mathematics*, 87(1), 41-52.

32. Hsu, W.-L. (1992). A simple test for interval graphs. In *Proceedings of the 19th International Workshop on Graph Theoretic Concepts in Computer Science (WG)* (pp. 11-16).

33. Impagliazzo, R. & Paturi, R. (2001). On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2), 367-375.

34. Kumar, N. & Deo, N. (1994). Multidimensional interval graphs. *Congressus Numerantium*, 102, 45-56.

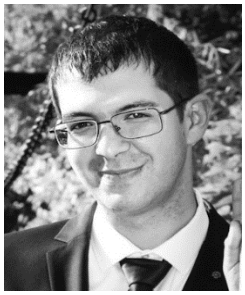
35. Lupu, C. (2001). Rețele ortogonale. *Revista Română de Informatică și Automatică (Romanian Journal of Information Technology and Automatic Control)*, 11(2), 28-36.

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

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

38. Roditty, L. & Vassilevska Williams, V. (2013). Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the 45th annual ACM Symposium on Theory of Computing (STOC)* (pp. 515-524).

39. Sheng, L. (1999). Cycle free probe interval graphs. *Congressus Numerantium*, 140, 33-42.



Guillaume DUCOFFE holds the position of Research Scientist at the National Institute for Research and Development in Informatics – ICI Bucharest, Romania. He is also an Associate Professor at the Faculty of Mathematics and Computer Science of the University of Bucharest. His main research area is Graph Theory, of which he studies combinatorial, metric and algorithmic aspects that are related to problems in Network Analysis. He has co-authored more than 60 papers in top scientific journals and conferences.

Guillaume DUCOFFE ocupă funcția de cercetător științific la Institutul Național de Cercetare-Dezvoltare în Informatică - ICI București, România. Este, de asemenea, conferențiar universitar la Facultatea de Matematică și Informatică a Universității din București. Principalul său domeniu de cercetare este teoria grafurilor, din care studiază aspecte combinatorii, metrice și algoritmice care sunt în relații cu probleme de analiză a rețelelor. Este coautor a peste 60 de lucrări în reviste și conferințe științifice de top.