On Computing the Average Distance for Some Chordal-Like Graphs

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

Abstract

The Wiener index of a graph G is the sum of all its distances. Up to renormalization, it is also the average distance in G. The problem of computing this parameter has different applications in chemistry and networks. We here study when it can be done in truly subquadratic time (in the size $n + m$ of the input) on n-vertex m-edge graphs. Our main result is a complete answer to this question, assuming the Strong Exponential-Time Hypothesis (SETH), for all the hereditary subclasses of chordal graphs. Interestingly, the exact same result also holds for the diameter problem. The case of non-hereditary chordal subclasses happens to be more challenging. For the chordal Helly graphs we propose an intricate $\tilde{O}(m^{3/2})$-time algorithm for computing the Wiener index, where m denotes the number of edges. We complete our results with the first known linear-time algorithm for this problem on the dually chordal graphs. The former algorithm also computes the median set.

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

Keywords and phrases Wiener index, Graph diameter, Hardness in P, Chordal graphs, Helly graphs

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.44

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 the fine-grained complexity of computing the average distance in a graph, a fundamental distance problem. For any undefined graph terminology, see [4, 20]. Unless stated otherwise, we only consider graphs that are simple, loopless, unweighted undirected, and more importantly connected. Let $G = (V, E)$ be such a graph. Throughout the paper, let $n = |V|$ and $m = |E|$. The distance between two vertices $x, y \in V$ equals the minimum number of edges on a xy-path in G. We denote it by $d_G(x, y)$, or simply by $d(x, y)$ whenever the graph G is clear from the context. The Wiener index of a graph G is $W(G) = \sum_{x,y \in V} d(x, y)$. Let also $diam(G) = \max_{x,y \in V} d(x, y)$ be the diameter of G. Note that $diam(G)$ and $\frac{1}{n(n-1)} W(G)$ represent the maximum and average distances in G. Although we focus in this work on computing $W(G)$, as it turns out, this problem is closely related to computing $diam(G)$. One of our objectives with this paper is to make clearer the connection between both problems.

The study of both parameters has applications in the fields of network optimization and analysis. For instance, delays are amongst the main causes of QoS degradation in a network. Roughly, if we further assume the network is subject to uniformly distributed demand, then we can approximate delays in the networks by distances in the underlying graph. In particular, with this interpretation in mind, the diameter and the (normalized) Wiener index would correspond to the maximum and average delays in the network, respectively. On a different note, for the analysis of social networks, and of more general complex networks with a core-periphery structure, various centrality indices have been introduced in order to measure the importance of a node. One of them, the so-called eccentricity centrality [34],

© Guillaume Ducoffe;
licensed under Creative Commons License CC-BY 4.0

is tighly related to the diameter: in fact, with respect to this centrality measurement, the most peripheral vertices are exactly the diametral vertices (whose eccentricity equals the diameter). In the same way, the Wiener index is related to the closeness centrality [34]. In chemistry, relations were also shown between some quantities of molecules and the Wiener index of their chemical graph representation [42].

By a straightforward reduction to All-Pairs Shortest-Paths (APSP), the Wiener index and the diameter of a graph can both be computed in $\mathcal{O}(nm)$ time. Somehow, this is optimal, even for sparse graphs. Indeed, assuming the Strong Exponential-Time Hypothesis (SETH) [32], one cannot decide in $\tilde{\mathcal{O}}(n^{2-\epsilon})$ time, for any $\epsilon > 0$, whether a graph with $\tilde{\mathcal{O}}(n)$ edges has diameter either 2 or 3 [39]. The former implies that one cannot compute either the Wiener index of such graph in $\tilde{\mathcal{O}}(n^{2-\epsilon})$ time, for any $\epsilon > 0$ [12]. If we allow approximation algorithms, then the situation is completely different. Indeed, while for any fixed ε we can compute an $(1+\varepsilon)$-approximation of the Wiener index in almost linear time [27], a long line of recent works has ruled out such possibility for the diameter problem, establishing various trade-off between the allowed running-time and the best possible approximation factor [1, 5, 6, 18, 36]. In what follows, we only consider exact computations, but on restricted graph classes rather than on general graphs.

Let us call an algorithm truly subquadratic if it runs in $\tilde{\mathcal{O}}(n^a m^b)$ time, for some non negative a, b such that $a + b < 2$. On sparse graphs, such running time becomes $\tilde{\mathcal{O}}(n^{a+b})$ which for the Wiener index and the diameter problem is ruled out by SETH. For dense graphs, the running time becomes $\tilde{\mathcal{O}}(n^{a+2b})$, that may be worse than the classic $\mathcal{O}(nm) = \mathcal{O}(n^3)$-time algorithm for APSP if $a + 2b > 3$. Therefore, so as to avoid this caveat, we are especially interested in running times in $\tilde{\mathcal{O}}(n^a m)$, for some non negative $a < 1$. Algorithms with such running times are known, for the diameter problem, on many graph classes [22, 24]. However, so far, we lack a good picture about the (non)existence of truly subquadratic-time algorithms for the Wiener index and the diameter problems within special graph classes. Indeed, the systematic study of the (non)existence of such algorithms is quite recent, motivated by the hardness results obtained in [7, 39]. Most prior works were about the (non)existence of almost linear-time algorithms, a much more restricted case [17]. We here make progress toward getting such good picture for the subclasses of chordal graphs.

Recall that, in what follows, most of our results apply to the Wiener index. What is remarkable, we think, is that many recent results for the Wiener index were obtained as a byproduct of similar results for the diameter [12, 14, 26]. Said otherwise, many SETH-hardness results for the diameter also apply to the Wiener index and, conversely, many truly subquadratic-time algorithms for computing the diameter can be modified in order to also compute the Wiener index (although this is not the case for all of them, *e.g.*, see [2, 24, 28]). It would be interesting to identify relevant graph classes where the complexity of the Wiener index and the diameter problem are different. One of our results in the paper, obtained for the subclass of chordal Helly graphs, may be a first step in this direction.

Our results. A graph is *chordal* if it has no induced cycle of length at least four. We here propose linear-time and truly subquadratic-time algorithms for computing the Wiener index, on subclasses of chordal graphs and related graph classes. First, in Sec. 2, we consider dually chordal graphs (*a.k.a.*, the clique-graphs of chordal graphs). The former are not chordal graphs in general, but they generalize strongly chordal graphs, and so, directed path graphs and interval graphs [11]. A linear-time algorithm for computing the diameter of dually chordal graphs was presented in [9] (it was recently extended to the computation of all eccentricities, see [21]). We propose a simple linear-time algorithm for computing the Wiener index in this class of graphs (Theorem 1). Doing so, we obtain the first linear-time algorithm for computing the Wiener index on strongly chordal graphs, and on even larger chordal subclasses such as doubly chordal graphs. Then, in Sec. 3, we design a general divide-and-conquer method on the clique-tree of any chordal graph in order to compute its Wiener index. We give applications of this method in Sec. 4 and 5. Our main result is proved in Sec. 4, where under SETH we completely characterize the hereditary subclasses of chordal graphs for which we can compute the diameter, resp. the Wiener index, in truly subquadratic time. These subclasses turn out to be the same and they can be characterized via a VC-dimension argument or, in more graph-theoretic terms, as those subclasses excluding at least one split graph. See our Theorem 7 for details. Doing so, we get a simple criterion for the existence of truly subquadratic-time algorithms, both for the Wiener index and the diameter, on *many* subclasses of chordal graphs that have been considered in the literature (Corollary 13). Our characterization for the diameter problem follows from several previous works, although to our best knowledge it has not been observed before. Our main technical contribution in Sec. 4 is to prove that the same characterization also holds for the Wiener index. Finally, in Sec. 5, we end up studying the Wiener index for chordal Helly graphs, a prominent non-hereditary subclass of chordal graphs. – Recall that a graph is Helly if any family of pairwise intersecting balls (of arbitrary centers and radii) have a nonempty common intersection. – We state a few open questions in Sec. 6.

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

2 Warm-up: Maximum neighbourhood orderings

Let $G = (V, E)$ be a graph. Recall that for a vertex v, $N(v) = \{u \in V \mid uv \in E\}$ denotes its open neighbourhood, while we call $N[v] = N(v) \cup \{v\}$ its closed neighbourhood. A maximum neighbour of a vertex v is some $u \in N[v]$ (possibly, $u = v$) such that $N[w] \subseteq N[u]$ for every $w \in N[v]$. We call a graph G dually chordal if its vertex-set can be totally ordered as $(v_1, v_2, \ldots, v_n)$ so that, for every $1 \leq i \leq n$, vertex v_i has a maximum neighbour in the induced subgraph $G \setminus \{v_1, v_2, \ldots, v_{i-1}\}$ [11]. Such ordering is sometimes called a MNO (Maximum Neighbourhood Ordering), and it can be computed in linear time [9]. In what follows, we implicitly use the fact that, if a vertex v has a maximum neighbour $u \neq v$, then $G \setminus v$ is an isometric (distance-preserving) subgraph of G.

▶ **Theorem 1.** *The Wiener index of a dually chordal graph can be computed in linear time.*

Proof. Fix a MNO $(v_1, v_2, \ldots, v_n)$ for G. Let $G_0 = G$ and, for every $1 \leq i < n$, let $G_i = G \setminus \{v_1, v_2, \ldots, v_i\}$. We observe that, for a vertex in G_{i-1} to be its own maximum neighbour, it must be a universal vertex. Since such a universal vertex can always be chosen last in a MNO of G_{i-1}, we assume from now on that every v_i, $i < n$ has a maximum neighbour $u_i \neq v_i$ in G_{i-1}. In what follows, we scan the ordering once in order to define some variables S_i and functions $\pi_i := V(G_i) \to \mathbb{N}$. Then, we reverse scan the MNO to compute, for every i and every $u \in V(G_i)$,

$$D_i(u) = S_i + \sum_{w \in V(G_i)} \pi_i(w) \cdot d(u, w).$$

Initially, let $S_0 = 0$ and, for every $v \in V$, let $\pi_0(v) = 1$. Doing so, we ensure that at the end of the algorithm we have $W(G) = \sum_{v \in V} D_0(v)$. Then, let us assume S_{i-1} and π_{i-1} to be known, for some $i > 0$. Let $u_i \in N_{G_{i-1}}(v_i)$ have maximum degree in the (isometric) subgraph G_{i-1}. Note that by maximality of $|N_{G_{i-1}}(u_i)|$, this vertex u_i must be a maximum neighbour of v_i in G_{i-1}. We set $S_i = S_{i-1} + \pi_{i-1}(v_i)$, $\pi_i(u_i) = \pi_{i-1}(u_i) + \pi_{i-1}(v_i)$ and $\pi_i(w) = \pi_{i-1}(w)$ for every other $w \in V(G_i) \setminus \{u_i\}$.

We now reverse scan the MNO. Clearly, $D_{n-1}(v_n) = S_{n-1}$. Let us assume the values $D_i(u)$ to be known. We set:

$$D_{i-1}(v_i) = D_i(u_i) + n - 2\pi_{i-1}(v_i) - \left(\sum_{w \in N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_{i-1}(w) \right)$$

and we proceed to the following update for every $w \in N_{G_{i-1}}(v_i) \setminus \{u_i\}$: $D_{i-1}(w) = D_i(w) - \pi_{i-1}(v_i)$. Indeed, as it shall become clearer in the remainder of our proof, this update is because these are the only vertices $x \in V(G_i)$ for which we do not have $d_{G_{i-1}}(v_i, x) = d_{G_{i-1}}(u_i, x) + 1$. For every other vertex $x \in V(G_{i-1})$, $D_{i-1}(x) = D_i(x)$.

All the above operations can be performed in total $\sum_i \mathcal{O}(|N_{G_{i-1}}(v_i)|) = \mathcal{O}(m+n)$ time. Furthermore, if all values $D_i(x)$ are correctly computed, then we get $D_0(x) = \sum_{y \in V} d(x,y)$. In particular, $W(G) = \sum_{x \in V} D_0(x)$. Let us assume in what follows all the values $D_i(x)$ to be correctly computed, for some $i > 0$. Since u_i is a maximum neighbour of v_i we have $d(v_i, x) = d(u_i, x) + 1$ for every $x \notin N_{G_{i-1}}[v_i]$. In particular:

$$\begin{aligned}
D_{i-1}(x) &= S_{i-1} + \sum_{y \in V(G_{i-1})} \pi_{i-1}(y) \cdot d(y,x) \\
&= S_{i-1} + \pi_{i-1}(v_i) \cdot d(x, v_i) + \pi_{i-1}(u_i) \cdot d(x, u_i) + \sum_{y \in V(G_i) \setminus \{u_i\}} \pi_{i-1}(y) \cdot d(y,x) \\
&= S_{i-1} + \pi_{i-1}(v_i) \cdot (d(x,u_i) + 1) + \pi_{i-1}(u_i) \cdot d(x,u_i) + \sum_{y \in V(G_i) \setminus \{u_i\}} \pi_i(y) \cdot d(y,x) \\
&= S_{i-1} + \pi_{i-1}(v_i) + (\pi_{i-1}(v_i) + \pi_{i-1}(u_i)) \cdot d(x,u_i) + \sum_{y \in V(G_i) \setminus \{u_i\}} \pi_i(y) \cdot d(y,x) \\
&= S_i + \sum_{y \in V(G_i)} \pi_i(y) \cdot d(y,x) = D_i(x).
\end{aligned}$$

In the same way (using $d(u_i, u_i) = 0$),

$$\begin{aligned}
D_{i-1}(u_i) &= S_{i-1} + \pi_{i-1}(v_i) + \sum_{y \in V(G_{i-1}) \setminus \{v_i\}} \pi_{i-1}(y) \cdot d(y, u_i) \\
&= S_i + \sum_{y \in V(G_i)} \pi_i(y) \cdot d(y, u_i) \\
&= D_i(u_i).
\end{aligned}$$

However, for every $w \in N_{G_{i-1}}(v_i) \setminus \{u_i\}$, $\pi_{i-1}(v_i) \cdot d(v_i, w) = \pi_{i-1}(v_i)$ is counted twice in $D_i(w)$: once in S_i, and once in $\pi_i(u_i) \cdot d(u_i, w) = \pi_i(u_i) = \pi_{i-1}(u_i) + \pi_{i-1}(v_i)$. In particular, we obtain $D_{i-1}(w) = D_i(w) - \pi_{i-1}(v_i)$.

We are left proving that $D_{i-1}(v_i)$ is correctly computed. By induction, $\forall j,\ \sum_x \pi_j(x) = n$. Then, we have:

$$\begin{aligned}
D_{i-1}(v_i) &= S_{i-1} + \sum_y \pi_{i-1}(y) \cdot d(y, v_i) = S_i + \left(\sum_y \pi_{i-1}(y) \cdot d(y, v_i) \right) - \pi_{i-1}(v_i) \\
&= S_i + \left(\sum_{w \in N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_{i-1}(w) + \sum_{y \notin N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_{i-1}(y) \cdot (d(y, u_i) + 1) \right) - \pi_{i-1}(v_i) \\
&= S_i + \left(\sum_{y \notin N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_{i-1}(y) \cdot d(y, u_i) \right) + \left(\sum_{y \neq v_i} \pi_{i-1}(y) \right) - \pi_{i-1}(v_i)
\end{aligned}$$

$$= S_i + \left(\sum_{y \notin N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_i(y) \cdot d(y, u_i) \right) + n - 2\pi_{i-1}(v_i)$$

$$= S_i + \left(\sum_{y} \pi_i(y) \cdot d(y, u_i) \right) - \left(\sum_{w \in N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_{i-1}(w) \right) + n - 2\pi_{i-1}(v_i)$$

$$= D_i(u_i) - \left(\sum_{w \in N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_{i-1}(w) \right) + n - 2\pi_{i-1}(v_i). \qquad \blacktriangleleft$$

A vertex v is a median if it minimizes $\sum_{u \in V} d(u, v)$. The median set of a graph G contains all its medians. With the same proof as for Theorem 1, we obtain:

▶ **Corollary 2.** *The median set of a dually chordal graph can be computed in linear time.*

Finally, a graph G is doubly chordal if it is both chordal and dually chordal [37]. Note that doubly chordal graphs properly contain the strongly chordal graphs, and so, the directed path graphs and the interval graphs.

▶ **Corollary 3.** *The Wiener index and the median set of a doubly chordal graph (and so, of a strongly chordal graph, resp. directed path graph, resp. interval graph) can be computed in linear time.*

We refer to [19] for a previous linear-time algorithm for the interval graphs. In contrast to our own algorithm, the former is taking an interval representation of the graph as input.

3 A framework for chordal graphs

We introduce a general method for computing the Wiener index of chordal graphs. We recall that a graph is called a split graph if its vertex-set can be bi-partitioned into a clique and a stable set [29]. In the Split-Weighted-Wiener problem, we are given as input a tuple $(\mathcal{P}(V), E', \alpha, \beta)$ where, for some split graph $H = (V, E)$:

- $\mathcal{P}(V) = (K, S^1, S^2, \ldots, S^c)$ is a partition of the vertex-set V, where K is a clique and $S := \bigcup_{i=1}^{c} S^i$ is a stable set.
- $E' = \{uv \mid u \in K, v \in S\} \subseteq E$.
- $\alpha, \beta : V \to \mathbb{N}_{\geq 1}$ are weight functions.

We call H the underlying input split graph.

The output is equal to $\sum_{i \neq j} \sum_{x \in S^i, y \in S^j} [\beta(y)\alpha(x) + \beta(x)\beta(y)d(x, y) + \beta(x)\alpha(y)]$.

▶ **Theorem 4.** *There is an $\tilde{\mathcal{O}}(m + n)$-time reduction from computing the Wiener index on a chordal graph G to the* Split-Weighted-Wiener *problem on some instances $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$. Furthermore, each underlying split H_k is obtained from some induced subgraph of G by removing the edges with their both ends in the same group S^i_k of the partition.*

We need to introduce a few additional notions and related intermediate results.

First, recall that a clique-tree of a graph G is a tree T of which the nodes are the maximal cliques of G, and such that for every vertex v the set of all the maximal cliques that contain v induces a connected subtree. It is known that G is chordal if and only if it has a clique-tree [13, 30, 41] and, furthermore, a clique-tree can be computed in linear time [40]. – See also [3, 10] and the references therein. – We may see a clique-tree T as a node-weighted tree where, for any maximal clique C, $w(C) = |C|$. Then, let $w(T) := \sum_C w(C)$. For a chordal graph, $w(T) = \mathcal{O}(n + m)$ [3].

For a set S and a vertex x, let us define $Pr(x,S) = \{y \in S \mid d(x,y) = d(x,S)\}$. Let also $I(x,y) = \{z \in V \mid d(x,y) = d(x,z) + d(z,y)\}$ for every vertices x and y. The following two results will be useful in our proofs:

▶ **Lemma 5** ([15]). *In a chordal graph G, if C is a clique and $x \notin C$, then there exists a vertex $g(x) \in \bigcap\{I(x,y) \mid y \in Pr(x,C)\}$ that is adjacent to all vertices from $Pr(x,C)$. This vertex $g(x)$ is sometimes called a gate of x.*

▶ **Lemma 6** ([22]). *If T is a clique-tree of a chordal graph G then, for every (not necessarily maximal) clique C of G, for every $v \notin C$ we can compute $d_G(v,C)$ and a corresponding gate v^* in total $\mathcal{O}(w(T))$ time, where $w(T)$ denotes the sum of cardinalities of all the maximal cliques of G.*

Finally, 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 [33]. Furthermore, we can compute a centroid in $\mathcal{O}(n)$ time by dynamic programming (*e.g.*, see [31]).

Proof of Theorem 4. Up to additional $\mathcal{O}(n+m)$-time pre-processing, we may assume each input graph G to be given under the form of a clique tree T. Our reduction is recursive. Consider first the following two base cases:

- **Case $|V(T)| = 1$.** Then, G is a clique, and we have $W(G) = n(n-1)$.
- **Case $|V(T)| = 2$.** Then, G is the union of two intersecting cliques X and X'. In particular, $diam(G) = 2$, and so (see [12]), we have $W(G) = 2n(n-1) - 2m$. Note that $n = |X|+|X'|-|X\cap X'|$ and $2m = |X|\cdot(|X|-1)+|X'|\cdot(|X'|-1)-|X\cap X'|\cdot(|X\cap X'|-1)$ (computable by scanning once the maximal cliques of G).

From now on, $|V(T)| \geq 3$. We compute X a centroid of T. Let $T_1, T_2, \ldots, T_c$ be the subtrees of $T \setminus \{X\}$. For each i, let G_i be induced by the vertices contained in at least one node of T_i. By the properties of a clique-tree, each G_i is an isometric subgraph of G.

(1) Computation of the $W(G_i)$'s. We apply our reduction to $G_1, G_2, \ldots, G_c$ (encoded by their respective clique-trees $T_1, T_2, \ldots, T_c$). Doing so (throughout one-to-many reductions to the Split-Weighted-Wiener problem), we computed their respective Wiener indices $W(G_1), W(G_2), \ldots, W(G_c)$.

(2) Computation of a first (non definitive) instance $(\mathcal{P}, E', \alpha, \beta)$. We apply Lemma 6 in order to compute, for every $v \notin X$, $d(v,X)$ and a gate $g(v)$, whose existence is ensured by Lemma 5. For every i, let $U_i = \{g(v_i) \mid v_i \in V(G_i) \setminus X\}$. We set, for every $u_i \in U_i$:

$$\alpha(u_i) = \sum\{d(v_i,X) - 1 \mid g(v_i) = u_i\}, \text{ and } \beta(u_i) = \#\{v_i \in V(G_i) \setminus X \mid g(v_i) = u_i\}.$$

We consider the following instance $(\mathcal{P}, E', \alpha, \beta)$ where we have:

- $\mathcal{P} = (X, U_1, U_2, \ldots, U_c)$;
- $E' = E \cap (X \times V \setminus X)$ (edges between X and the neighbours of X);
- and α, β as they were previously defined in the proof.

We claim that by solving the Split-Weighted-Wiener problem on this above instance, one computes the sum of all distances $d(v_i,v_j)$ for $v_i \in V(G_i) \setminus X$, $v_j \in V(G_j) \setminus X$ and $i \neq j$. Indeed, let $v_i \in V(G_i) \setminus X$ and $v_j \in V(G_j) \setminus X$, for some $i \neq j$. Every v_iv_j-path crosses X. Furthermore, since X is a clique, there is a shortest v_iv_j-path such that the vertex of X closest to v_i (resp., to v_j) is in $Pr(v_i,X)$ (resp., in $Pr(v_j,X)$). As a result, we have $d(v_i,v_j) = d(v_i,g(v_i)) + d(g(v_i),g(v_j)) + d(g(v_j),v_j)$. For v_i fixed, we get:

$$\begin{aligned}\sum_{v_j \in V(G_j)\setminus X} d(v_i,v_j) &= \sum_{v_j \in V(G_j)\setminus X} \left(d(v_i,g(v_i)) + d(g(v_i),g(v_j)) + d(g(v_j),v_j)\right)\\ &= \sum_{u_j \in U_j} \sum_{v_j \mid g(v_j)=u_j} \left(d(v_i,g(v_i)) + d(g(v_i),u_j) + d(u_j,v_j)\right)\\ &= \sum_{u_j \in U_j} \left(\beta(u_j)\cdot\left(d(v_i,g(v_i)) + d(g(v_i),u_j)\right) + \alpha(u_j)\right).\end{aligned}$$

If we sum the above over all the v_i's, we obtain:

$$\begin{aligned}\sum_{v_i \in V(G_i)\setminus X}\sum_{v_j \in V(G_j)\setminus X} d(v_i,v_j) &= \sum_{v_i \in V(G_i)\setminus X}\sum_{u_j \in U_j} \left(\beta(u_j)\cdot\left(d(v_i,g(v_i)) + d(g(v_i),u_j)\right) + \alpha(u_j)\right)\\ &= \sum_{u_i \in U_i}\sum_{u_j \in U_j} \left(\beta(u_j)\cdot\left(\alpha(u_i) + \beta(u_i)d(u_i,u_j)\right) + \beta(u_i)\alpha(u_j)\right)\\ &= \sum_{u_i \in U_i}\sum_{u_j \in U_j} \left(\beta(u_j)\alpha(u_i) + \beta(u_j)\beta(u_i)d(u_i,u_j) + \beta(u_i)\alpha(u_j)\right).\end{aligned}$$

(3) Computation of a reduced instance. The problem with the above instance $(\mathcal{P}, E', \alpha, \beta)$ is that, in order to fit with our claimed running time for the reduction, we further need to have $|E'| = \mathcal{O}(w(T))$, that may not be the case in general. Thus, we need to reduce the instance. For that, let $U := \bigcup_i U_i$. For every $u \in U$, there is a maximal clique that contains $\{u\} \cup (N(u) \cap X)$. Thus, in order to relate u with its neighbours in X, it suffices to compute amongst all maximal cliques containing u one X_u maximizing $|X_u \cap X|$. We can do so, in $\mathcal{O}(w(T))$ time, as follows:

- We scan all the maximal cliques $X' \neq X$ in order to compute $|X' \cap X|$.
- We order the maximal cliques $X' \neq X$ by non increasing value of $|X' \cap X|$. It can be done by using, *e.g.*, counting sort.
- We scan all the ordered maximal cliques X'. Initially, all the vertices are left unmarked. When scanning a maximal clique X', we set $X_u = X'$ for every $u \in X' \cap U$ unmarked. Then, we mark all vertices in X'.

Let $u, u' \in U$ be such that $X_u = X_{u'}$. Since u and u' are adjacent, there is an i such that $u, u' \in U_i$. Since furthermore, $N(u) \cap X = N(u') \cap X$ (these vertices are twins in the underlying split graph H), we may remove u' from U and update $\alpha(u), \beta(u)$ as follows: $\alpha(u) := \alpha(u) + \alpha(u')$, $\beta(u) = \beta(u) + \beta(u')$. Doing so until it can no more be done, we end up with a smaller set U^* such that no two vertices $u \in U^*$ are associated to the same maximal clique X_u. For every i, let us replace U_i in the above instance for SPLIT-WEIGHTED-WIENER by the subset $S_i = U_i \cap U^*$. Then, for $E^* = \{uv \mid u \in U^*, v \in X\}$ we obtain $|E^*| < \sum_{u \in U^*} |X_u| = \mathcal{O}(w(T))$, as desired.

From now on, we assume to be given $W_0 := \sum_{i \neq j} \sum_{v_i \in V(G_i)\setminus X, v_j \in V(G_j)\setminus X} d(v_i,v_j)$ (*i.e.*, by the above reduction to one instance of SPLIT-WEIGHTED-WIENER).

(4) Computation of the sum of distances between X and $V \setminus X$. Next, we compute $W_X := \sum_{x \in X, v \notin X} d(x,v)$. For that, recall that we computed the set $U = \bigcup_i U_i$ of all the gates, and the weight functions α, β. Furthermore, for each $x \in X$ and index i we have:

$$\sum_{v_i \in V(G_i)\setminus X} d(x,v_i) = \sum_{v_i \in V(G_i)\setminus X} \left(d(x,g(v_i)) + d(g(v_i),v_i)\right) = \sum_{u_i \in U_i} \left(\beta(u_i)d(x,u_i) + \alpha(u_i)\right).$$

Note that, for every $u \in U$, $d(u,x) \in \{1,2\}$. Hence, we get:

$$\sum_{v \notin X} d(x,v) = 2 \cdot \left(\sum_{u \in U} \beta(u) \right) - \left(\sum_{u \in N(x) \cap U} \beta(u) \right) + \left(\sum_{u \in U} \alpha(u) \right).$$

The two sums $\sum_{u \in U} \beta(u)$ and $\sum_{u \in U} \alpha(u)$ can be pre-computed in $\mathcal{O}(|U|) = \mathcal{O}(w(T))$ time. Therefore, in order to compute W_X, we are left computing $\sum_{u \in N(x) \cap U} \beta(u)$ for every $x \in X$.

- For each $x \in X$, let $\gamma(x) = 0$ (at the end of the procedure, we shall have $\gamma(x) = \sum_{u \in N(x) \cap U} \beta(u)$.
- We root T arbitrarily and we start a BFS from the root.
- When we reach some maximal clique X' during the search, we further assume to have access to its intersection $Y = X' \cap p(X')$ with its parent node ($Y = \emptyset$ if X' is the root). For every $x \in (X \cap X') \setminus Y$, we increment $\gamma(x)$ by $\sum_{u \in U \cap X'} \beta(u)$. However, in order to avoid overcounting, for every $x \in X \cap Y$, we increment $\gamma(x)$ by $\sum_{u \in (U \cap X') \setminus Y} \beta(u)$.
- After processing each X', we scan all the maximal cliques X'' that are children nodes of X' in order to compute $X' \cap X''$.

Since each maximal clique is scanned $\mathcal{O}(1)$ times, the total running time is in $\mathcal{O}(w(T))$.

(5) A formula for computing $W(G)$. At this point of the reduction, we are almost done for computing $W(G)$. However, if we sum all the partial estimates computed so far, there are a few distances overcounted. Specifically, let $Y_i = X \cap V(G_i)$. We have:

$$\begin{aligned} W(G) = W_0 &+ \left(\sum_{i=1}^{c} W(G_i) \right) + |X| \cdot (|X| - 1) + 2W_X \\ &- \sum_{i=1}^{c} \left(|Y_i| \cdot (|Y_i| - 1) + 2 \cdot \sum_{y_i \in Y_i} \sum_{v_i \in V(G_i) \setminus X} d(v_i, y_i) \right) \end{aligned}$$

Each set Y_i above can be computed as the intersection between X and the unique maximal clique $X_i \in V(T_i) \cap N_T(X)$. Furthermore, for $y_i \in Y_i$ fixed, we have:

$$\sum_{v_i \in V(G_i) \setminus X} d(v_i, y_i) = 2 \cdot \left(\sum_{u_i \in U_i} \beta(u_i) \right) - \left(\sum_{u_i \in N(y_i) \cap U_i} \beta(u_i) \right) + \left(\sum_{u_i \in U_i} \alpha(u_i) \right).$$

Hence, we are left computing $\gamma_i(y_i) = \sum_{u_i \in U_i \cap N(y_i)} \beta(u_i)$ for every $y_i \in Y_i$. This can be done in $\mathcal{O}(w(T_i))$ time, by using the same procedure as for computing the values $\gamma(x)$, but restricted to the clique-subtree T_i. Since all the T_i's are disjoint, the total running time is still in $\mathcal{O}(w(T))$.

Complexity. Since we use in our reduction a centroid decomposition of the clique-tree T, there are $\mathcal{O}(\log |V(T)|) = \mathcal{O}(\log n)$ recursive stages. At each recursive stage, we proceed on disjoint clique-subtrees T'. Therefore, each recursive stage takes $\mathcal{O}(w(T))$ time (excluding the calls to an oracle solving SPLIT-WEIGHTED-WIENER). The total running time of the reduction, excluding the calls of the oracle, is in $\mathcal{O}(w(T) \log n) = \mathcal{O}(m \log n)$. ◀

We shall use the heavy machinery presented above in the next two sections.

4 Application: Hereditary subclasses of chordal graphs

Recall that a class of graphs is called hereditary if it is stable by induced subgraph. The complexity of the diameter problem has been studied for many hereditary subclasses of chordal graphs [9, 17, 25, 22, 23, 38]. For the special case of the interval graphs, a linear-time algorithm for computing the Wiener index is also known [19] (which we extended to the strongly chordal graphs in Sec. 2). However, even for the hereditary subclass of split graphs, under SETH there is no truly subquadratic algorithm for computing the diameter nor the Wiener index. In this section, we exactly characterize the hereditary subclasses of chordal graphs for which such algorithms exist (conditioned on SETH).

Recall that a hypergraph is a pair $\mathcal{H} = (X, \mathcal{R})$ such that each element of $\mathcal{R}$ (called a hyperedge) is a subset of X (the elements of X are called vertices, by analogy to graphs). A vertex-subset $Y \subseteq X$ is shattered by $\mathcal{H}$ if, for any possible subset $Z \subseteq Y$, there exists an $e \in \mathcal{R}$ such that $e \cap Y = Z$. The VC-dimension of $\mathcal{H}$ is the largest cardinality of a shattered subset. For a graph, its VC-dimension is the VC-dimension of its neighbourhood hypergraph $\mathcal{N}(G) = (V, \{N[v] \mid v \in V\})$. Finally, a class of graphs has bounded VC-dimension if there exists a constant d such that every graph in the class has VC-dimension at most d.

▶ **Theorem 7.** *Under SETH, for any hereditary subclass $\mathcal{C}$ of chordal graphs, the following statements are equivalent:*

1. *There is a truly subquadratic algorithm for computing the Wiener index within $\mathcal{C}$.*
2. *There is a truly subquadratic algorithm for computing the diameter within $\mathcal{C}$.*
3. *There is a truly subquadratic algorithm for deciding if a graph in $\mathcal{C}$ has diameter ≤ 2.*
4. *$\mathcal{C}$ does not contain all the split graphs.*
5. *$\mathcal{C}$ has bounded VC-dimension.*

The above theorem follows from previous works in the literature, and a new result on our own in Sec. 4.1. Specifically, we will use the following lemmas in our proof:

▶ **Lemma 8** ([7]). *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 $c(\varepsilon)\log n$.*

▶ **Lemma 9** ([22]). *If $\mathcal{C}$ is a subclass of chordal graphs of bounded VC-dimension, then there exists a randomized truly subquadratic-time algorithm for computing the diameter of the graphs in $\mathcal{C}$.*

The next result of Bousquet et al. [8] shows that for any hereditary chordal subclass, either Lemma 8 or Lemma 9 can be applied.

▶ **Lemma 10** ([8]). *Let $\mathcal{C}$ be a hereditary class. If $\mathcal{C}$ has infinite VC-dimension, then $\mathcal{C}$ must contain either all the bipartite graphs, or all the co-bipartite graphs, or all the split graphs.*

We also prove a quantitative version of Lemma 10, for chordal graphs. Note that the worst-case running time of the algorithms presented in Lemma 9 and in Sec. 4.1 depends on the largest VC-dimension of a graph in the class $\mathcal{C}$.

▶ **Lemma 11.** *If H is a split graph, then every H-free chordal graph has VC-dimension at most $|V(H)| - 1$.*

In Sec. 4.1, we prove that:

▶ **Theorem 12.** *If $\mathcal{C}$ is a subclass of chordal graphs of bounded VC-dimension, then there exists a deterministic truly subquadratic-time algorithm for computing the Wiener index of the graphs in $\mathcal{C}$.*

We are finally ready to prove our main result:

Proof of Theorem 7. Let $G \in \mathcal{C}$ be arbitrary. It is known [12] that $W(G) \leq 2n(n-1) - 2m$ if and only if $diam(G) \leq 2$. Therefore, (1) $\Longrightarrow$ (3). We also have (2) $\Longrightarrow$ (3). Since the diameter of a split graph is at most three, we get by Lemma 8 that (3) $\Longrightarrow$ (4). Furthermore, since not all bipartite graphs and co-bipartite graphs are chordal, by Lemma 10 we get that (4) $\Longrightarrow$ (5). Finally, by Lemma 9 we have (5) $\Longrightarrow$ (2), and by Theorem 12 we have (5) $\Longrightarrow$ (1). ◀

▶ **Corollary 13.** *The following subclasses of chordal graphs admit truly subquadratic algorithms for the Wiener index and the diameter problem: chordal bull-free graphs, chordal claw-free graphs, block graphs, interval graphs [38], strongly chordal graphs [9], directed path graphs [17], undirected path graphs [22], chordal dominating pair graphs [25], hereditary Helly graphs [22], k-separator chordal graphs [35], chordal graphs of bounded interval number, chordal graphs of bounded asteroidal number [25].*

To our best knowledge, our results are new for chordal bull-free graphs, chordal claw-free graphs, k-separator chordal graphs and chordal graphs of bounded interval number, both for the Wiener index and the diameter problem (references to prior works are given in the statement of Corollary 13). For the Wiener index only, our results are also new for the subclasses of strongly chordal graphs (see also Sec. 2 for a faster algorithm), directed path graphs, undirected path graphs, chordal dominating pair graphs, hereditary Helly graphs and chordal graphs of bounded asteroidal number. This above listing is far from exhaustive.

4.1 Sketch Proof of Theorem 12

In what follows, let $G = (V, E) \in \mathcal{C}$. By Theorem 4, computing the Wiener index of G can be reduced in $\tilde{\mathcal{O}}(m + n)$ time to solving the Split-Weighted-Wiener problem on some instances $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$. Let d be the maximum VC-dimension of a graph in $\mathcal{C}$. We prove below that each instance $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$ can be solved in $\tilde{\mathcal{O}}(|E'_k| \cdot |V_k|^{1-\varepsilon_d})$ time, where ε_d is a constant that only depends on d. Note that $\sum_k |E'_k| = \tilde{\mathcal{O}}(m+n)$ since it is the total running time of our reduction. Furthermore, since by Theorem 4 each V_k is a subset of V, $\max_k |V_k| = \mathcal{O}(n)$. Hence, our result below implies an $\tilde{\mathcal{O}}(\sum_k |E'_k| \cdot |V_k|^{1-\varepsilon_d}) = \tilde{\mathcal{O}}(mn^{1-\varepsilon_d})$ running time in order to compute $W(G)$.

For the remainder of the proof, let $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$ be fixed. Recall (see Sec. 3) $\mathcal{P}(V_k) = (K_k, S_k^1, S_k^2, \ldots, S_k^{c_k})$ with K_k a clique of G. Let $S_k = \bigcup_j S_k^j = V_k \setminus K_k$. By Theorem 4, $E'_k = E(G) \cap (K_k \times S_k)$ (*i.e.*, there is no edge added or removed between K_k and S_k compared to G). We start with a simple observation:

▶ **Lemma 14.** *Let $G = (V, E)$ have VC-dimension at most d, and let $X, Y \subseteq V$. The hypergraph $\mathcal{H} = (X, \{N_G[y] \cap X \mid y \in Y\})$ also has VC-dimension at most d.*

Proof. Any subset shattered by $\mathcal{H}$ is shattered by the neighbourhood hypergraph of G. ◀

We apply Lemma 14 to $X = S_k$ and $Y = K_k$. Let $\mathcal{H}_k$ be the corresponding hypergraph.

A spanning path of a hypergraph $\mathcal{H}$ is a total order of its vertex-set. The stabbing number of such spanning path is the least t such that every hyperedge of $\mathcal{H}$ is the union of at most t intervals onto the path. The following result is based on a prior work of Chazelle and Welzl about range queries [16]:

▶ **Lemma 15** ([24]). *For every $d > 0$, there exists a constant $\varepsilon_d \in (0;1)$ such that in $\tilde{\mathcal{O}}(m+n^{2-\varepsilon_d})$ deterministic time, for every n-vertex hypergraph $\mathcal{H} = (X, \mathcal{R})$ of VC-dimension at most d and size $m = \sum_{e\in\mathcal{R}} |e|$, we can compute a spanning path of stabbing number $\tilde{\mathcal{O}}(n^{1-\varepsilon_d})$. Moreover, $\varepsilon_d = \frac{1}{2^{d+1}[c(d+1)-1]+1}$ for some constant $c > 2$.*

Apply Lemma 15 to $\mathcal{H}_k$. Doing so, for every $u \in K_k$, $N_H(u) \cap S_k$ is the union of $\tilde{\mathcal{O}}(|S_k|^{1-\varepsilon_d})$ intervals of the resulting spanning path of $\mathcal{H}_k$. Then, for every $s \in S_k$, define $N_k^2(s)$ to be the set of all vertices in S_k at distance two from s in the underlying split graph $(K_k \cup S_k, E'_k \cup (K_k \times K_k))$. We have that $N_k^2(s)$ is the union of $\tilde{\mathcal{O}}(|N_G(s) \cap K_k| \cdot |S_k|^{1-\varepsilon_d})$ intervals of the spanning path computed for $\mathcal{H}_k$.

One more ingredient is needed in our proof. Consider a set Q of 2-dimensional points. Each point $(x, y) \in Q$ is assigned some weight $f(x, y)$. A box is the Cartesian product of two intervals (we also allow intervals that are infinite, semi-finite, or reduced to a singleton). Note that each box defines a rectangle in the plane (possibly, a line or a point if some intervals are reduced to a singleton). A (counting) range query asks, for a given box, the sum of the weights of all points in Q that are contained into this rectangle.

▶ **Lemma 16** ([43]). *Let Q be a set of 2-dimensional points. After a pre-processing in $\mathcal{O}(|Q| \log |Q|)$ time, one can answer any range query in $\mathcal{O}(\log |Q|)$ time.*

Let $\sigma_k : S_k \to \{1, 2, \ldots, |S_k|\}$ be the mapping of S_k to the nodes of the spanning path. For each $1 \le i \le c_k$ and $s_i \in S_k^i$, we create a point $(\sigma_k(s_i), i)$ with weight $f(\sigma_k(s_i), i) = \beta_k(s_i)$. Let Q_k be the resulting 2-dimensional point-set. We apply Lemma 16 in order to compute, for each i and $s_i \in S_k^i$, the weighted sum $\Phi(s_i) = \sum\{\beta_k(s') \mid s' \in N_k^2(s_i) \setminus S_i^k\}$; indeed, for any fixed s_i, this operation can be reduced to $\tilde{\mathcal{O}}(|N_G(s_i) \cap K_k| \cdot |S_k|^{1-\varepsilon_d})$ range queries, by using the interval representation of $N_k^2(s_i)$.

Finally, for every j, define $\alpha^j = \sum_{s'\in S_k^j} \alpha_k(s')$ and $\beta^j = \sum_{s'\in S_k^j} \beta_k(s')$. Let also $\alpha^* = \sum_{j=1}^{c_k} \alpha^j$ and $\beta^* = \sum_{j=1}^{c_k} \beta^j$. All the values α^j, β^j, $1 \le j \le c_k$ and α^*, β^* can be pre-computed in total $\mathcal{O}(|S_k|)$ time. Since in the underlying split graph H_k, the distance $d(s, s')$ between two different vertices $s, s' \in S_k$ in the stable set is either two or three, we may rewrite the output of Split-Weighted-Wiener as follows:

$$\sum_{i\neq j} \sum_{s\in S_k^i, s'\in S_k^j} \left[\beta_k(s')\alpha_k(s) + \beta_k(s)\beta_k(s')d(s,s') + \beta_k(s)\alpha_k(s')\right] =$$

$$\sum_i \sum_{s\in S_k^i} \left[(\beta^* - \beta^i)\cdot\alpha_k(s) + \beta_k(s)\cdot\left(\sum_{s'\notin S_k^i} \beta_k(s')d(s,s')\right) + \beta_k(s)\cdot(\alpha^* - \alpha^i)\right]$$

$$= \left[\sum_i (\beta^* - \beta^i)\alpha^i\right] + \left[\sum_i \sum_{s\in S_k^i} \beta_k(s)\cdot\left(\sum_{s'\notin S_k^i} \beta_k(s')d(s,s')\right)\right] + \left[\sum_i \beta^i(\alpha^* - \alpha^i)\right]$$

$$= \left[\sum_i (\beta^* - \beta^i)\alpha^i\right] + \left[\sum_i \sum_{s\in S_k^i} \beta_k(s)\cdot\left(3(\beta^* - \beta^i) - \Phi(s)\right)\right] + \left[\sum_i \beta^i(\alpha^* - \alpha^i)\right]$$

$$= \left[\sum_i (\beta^* - \beta^i)(\alpha^i + 3\beta^i)\right] - \left[\sum_{s\in S_k} \beta_k(s)\Phi(s)\right] + \left[\sum_i \beta^i(\alpha^* - \alpha^i)\right].$$

Now, given the pre-computed values $\alpha^i, \beta^i, \alpha^*, \beta^*$ and $\Phi(s)$, we can compute the desired output in additional $\mathcal{O}(|S_k|)$ time. ◀

5 Application: Chordal Helly graphs

We end up our study investigating the complexity of the Wiener index on non-hereditary subclasses of chordal graphs. The chordal Helly graphs are a prominent such subclass. Indeed, they are a strict generalization of doubly chordal graphs, and so, of strongly chordal graphs, interval graphs, etc. Recently, a linear-time algorithm for computing the diameter of this subclass of graphs was proposed [22].

▶ **Theorem 17.** *There is an $\tilde{\mathcal{O}}(m^{3/2})$-time algorithm for computing the Wiener index of chordal Helly graphs.*

Proof. Let $G = (V, E)$ be a chordal Helly graph. We apply Theorem 4 in order to reduce the computation of $W(G)$ to solving the Split-Weighted-Wiener problem on some instances $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$. Write $\mathcal{P}(V_k) = (K_k, S_k^1, S_k^2, \ldots, S_k^{c_k})$. We also know from Theorem 4 that K_k is a clique of G. In what follows, we present an algorithm for solving the instance $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$ in $\mathcal{O}(|K_k| \cdot |E'_k|)$ time. Doing so, we can compute $W(G)$ in $\sum_k \mathcal{O}(|K_k| \cdot |E'_k|)$ time. Since we further have $\sum_k |E'_k| = \tilde{\mathcal{O}}(n+m)$ (time of the reduction of Theorem 4) and $\max_k |K_k| = \mathcal{O}(m^{1/2})$ because each subset K_k is a clique, we get a running time in $\tilde{\mathcal{O}}(m^{3/2})$.

Throughout the remainder of the proof, let $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$ be fixed. Let also $S_k = \bigcup_{i=1}^{c_k} S_k^i = V_k \setminus K_k$. We write $s \sim s'$ if there exists an i such that $s, s' \in S_k^i$. Then, our goal is to compute the following value:

$$\Psi_k := \sum \{\beta_k(s)\beta_k(s') \mid s \not\sim s' \text{ and } d(s, s') = 2\}.$$

Indeed, let us define $\alpha^i = \sum_{s \in S_k^i} \alpha_k(s)$ and $\beta^i = \sum_{s \in S_k^i} \beta_k(s)$ for every i. Similarly, let $\alpha^* = \sum_i \alpha^i$ and $\beta^* = \sum_i \beta^i$. All these values can be pre-computed in total $\mathcal{O}(|S_k|) = \mathcal{O}(|E'_k|)$ time. Then, the desired output $\sum_{i \neq j} \sum_{s \in S_k^i, s' \in S_k^j} [\beta_k(s')\alpha_k(s) + \beta_k(s)\beta_k(s')d(s,s') + \beta_k(s)\alpha_k(s')]$ can be rewritten as:

$$\sum_i (\beta^* - \beta^i)\alpha^i + \left(\sum_{i \neq j} \sum_{s \in S_k^i, s' \in S_k^j} \beta_k(s)\beta_k(s')d(s,s') \right) + \sum_i \beta^i(\alpha^* - \alpha^i).$$

Since in the underlying split graph, the distance between two distinct vertice in the stable set S_k is either two or three, we get:

$$\sum_{i \neq j} \sum_{s \in S_k^i, s' \in S_k^j} \beta_k(s)\beta_k(s')d(s,s') = 3 \left(\sum_i \beta^i(\beta^* - \beta^i) \right) - \Psi_k.$$

The algorithm. Let us define the adjacency lists $N_k(u) = \{s \in S_k \mid us \in E'_k\}$, for every $u \in K_k$. In the same way, let us define the adjacency lists $N_k(s) = \{u \in K_k \mid us \in E'_k\}$, for every $s \in S_k$. We set initially $\Psi_k := 0$. Then, let $K_k = (u_1, u_2, \ldots, u_{|K_k|})$ be totally ordered. We consider each $u_i \in K_k$ sequentially, and in order from $i = 1$ to $i = |K_k|$. At step i, let us define for every $u' \in K_k$ the subset $N_{k,i}(u') := N_k(u') \setminus \left(\bigcup_{j<i} N_k(u_j) \right)$. For each $s \in S_k \setminus \left(\bigcup_{j<i} N_k(u_j) \right)$, let t be such that $s \in S_k^t$. We select a vertex $u' \in N_k(s)$ such that $|(N_{k,i}(u') \cap N_{k,i}(u_i)) \setminus S_k^t|$ is maximized. Then, we increment Ψ_k by:

$$\begin{cases} \beta_k(s) \cdot \sum\{\beta_k(s') \mid s' \in (N_{k,i}(u_i) \cap N_{k,i}(u')) \setminus S_k^t\} \text{ if } s \in N_{k,i}(u_i) \\ 2\beta_k(s) \cdot \sum\{\beta_k(s') \mid s' \in (N_{k,i}(u_i) \cap N_{k,i}(u')) \setminus S_k^t\} \text{ if } s \notin N_{k,i}(u_i). \end{cases}$$

Correctness. We first need to observe that $N_{k,1}(u_1)$ $(= N_k(u_1))$, $N_{k,2}(u_2), \ldots,$ $N_{k,i}(u_i), \ldots$ is a partition of S_k. Therefore in order to prove correctness of the algorithm, it suffices to prove that at each step i, we increment Ψ_k by twice the sum of all $\beta_k(x)\beta_k(y)$ with $x \in N_{k,i}(u_i)$, $y \in S_k \setminus \left(\bigcup_{j<i} N_k(u_j)\right)$, $x \not\sim y$ and $d(x,y) = 2$. For that, let $s \in S_k \setminus \left(\bigcup_{j<i} N_k(u_j)\right)$ be arbitrary and such that $s \in S_k^t$ for some t. We prove next that for the vertex u' selected for s, all the vertices in $N_{k,i}(u_i) \setminus S_k^t$ and at distance two from s are also in $N_{k,i}(u')$. In particular, all the desired pairs (x,y) are enumerated: twice if $x, y \in N_{k,i}(u_i)$, and only once otherwise, thus proving correctness of our above formula for incrementing Ψ_k.

By maximality of u', it is sufficient to prove the existence of a vertex $u^* \in N_k(s)$ such that all the vertices of $N_{k,i}(u_i) \setminus S_k^t$ that are at distance two from s are also contained into $N_{k,i}(u^*)$. We do so by reasoning on the whole graph G. Specifically, let $\mathcal{F} = \{N_G[x] \mid x = s \text{ or } (\ x \in N_{k,i}(u_i) \setminus S_k^t \text{ and } d(s,x) = 2\ )\}$. The balls in $\mathcal{F}$ pairwise intersect. Therefore, by the Helly property, all balls in $\mathcal{F}$ contain some common vertex z. We claim that $z \in K_k$. Note that it will prove the existence of the desired vertex u^* because in such a case we can always choose $u^* = z$. It follows from the proof of Theorem 4 that S_k^t and $S_k \setminus S_k^t$ are in separate connected components of $G \setminus K_k$, thus immediately proving the claim.

Implementation and Complexity. As a starter, we observe that all the lists $N_k(u)$, for $u \in K_k$ (resp., all the lists $N_k(s)$, for $s \in S_k$) can be constructed in $\mathcal{O}(|E'_k|)$ time. Throughout the algorithm, we maintain an $|S_k|$-size array `A`, whose entries are indexed by S_k and are initialized to 0 (at the end of the algorithm, we have for all $s \in N_{k,i}(u_i)$ that `A`$[s]= i$). We also store two auxiliary matrices of dimensions $|K_k| \times (c_k + 1)$, denoted by `Int` and `Sum`, whose entries are indexed by $K_k \times \{0, 1, 2, \ldots, c_k\}$ and of which we ensure that all entries equal 0 at the beginning of any step i. Finally, we find more convenient for certain operations to maintain a stack `Stack` and a boolean $|K_k|$-size array `InStack` whose entries are indexed by K_k; before each step, we ensure that the stack is emptied and that all entries in `InStack` are set to False. Note that all the above data structures can be constructed in $\mathcal{O}(|S_k| + |K_k| \cdot c_k) = \mathcal{O}(|K_k| \cdot |S_k|) = \mathcal{O}(|K_k| \cdot |E'_k|)$ time.

We proceed as follows during any step i. First, we scan $N_k(u_i)$ and, for every $s \in N_k(u_i)$ such that `A`$[s]= 0$, we set `A`$[s]= i$. Furthermore, if we set `A`$[s]= i$, then we add every $u' \in N_k(s)$ in `Stack` (we use the auxiliary array `InStack` in order to avoid adding twice a vertex). Then, we consider each vertex in `Stack` sequentially, until we emptied the stack. For every vertex u' considered, we scan the list $N_k(u')$. If $s \in N_k(u')$ is such that `A`$[s]= i$ (equivalently, if $s \in N_{k,i}(u_i)$) then, we increment `Int`$[u'][0]$ by one (intersection size with $N_{k,i}(u_i)$). Similarly, we increment `Sum`$[u'][0]$ by $\beta_k(s)$. For the unique t such that $s \in S_k^t$, we also increment `Int`$[u'][t]$ by one and `Sum`$[u'][t]$ by $\beta_k(s)$, respectively. Finally, we apply our above formula in order to increment Ψ_k:

1. For every $s \in N_k(u_i)$, if `A`$[s]= i$, then let t be the unique index such that $s \in S_k^t$. We increment Ψ_k by $\beta_k(s) \times ($`Sum`$[u_i][0] -$ `Sum`$[u_i][t])$.
2. For every $s \in S_k$ such that `A`$[s]= 0$, let also t be the unique index such that $s \in S_k^t$. We scan the list $N_k(s)$ in order to find some u' maximizing `Int`$[u'][0] -$ `Int`$[u'][t]$. Then, we increment Ψ_k by $2\beta_k(s) \times ($`Sum`$[u'][0] -$ `Sum`$[u'][t])$.

The whole step only takes $\mathcal{O}(|E'_k|)$ time because each adjacency list is scanned $\mathcal{O}(1)$ times. ◀

Although it is a truly subquadratic algorithm (in the size $n + m$ of the input), our algorithm does not perform better than the classic $\mathcal{O}(nm)$-time algorithm for general graphs if $m = \Theta(n^2)$. This is in sharp contrast with our results in Sec. 4, for hereditary chordal subclasses, where all our algorithms run in $\tilde{\mathcal{O}}(n^a m)$ time, for some $a < 1$. It would be very interesting to improve the running time of Theorem 17, and to bring it much closer to the linear-time complexity of the diameter problem on this subclass of graphs.

6 Open problems

We left open whether there are there relevant graph classes where the complexity of the Wiener index and the diameter are different. In particular, are both problems subquadratic equivalent? Another interesting question is whether we could compute the Wiener index of Helly graphs in truly subquadratic time. In this paper, we only managed to find such algorithm for the subclasses of dually chordal graphs and chordal Helly graphs.

References

1 A. Backurs, L. Roditty, G. Segal, V. Vassilevska Williams, and N. Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 267–280, 2018.

2 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.

3 J. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*, pages 1–29. ORNL, 1993.

4 John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory*, volume 244 of *Graduate Texts in Mathematics*. Springer-Verlag London, 2008.

5 E. Bonnet. 4 vs 7 sparse undirected unweighted Diameter is SETH-hard at time $n^{4/3}$. Technical report, arXiv, 2021. `arXiv:2101.02312`.

6 E. Bonnet. Inapproximability of diameter in super-linear time: Beyond the 5/3 ratio. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 17:1–17:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.STACS.2021.17`.

7 Michele Borassi, Pierluigi Crescenzi, and Michel Habib. Into the square: On the complexity of some quadratic-time solvable problems. *Electronic Notes in Theoretical Computer Science*, 322:51–67, April 2016. `doi:10.1016/j.entcs.2016.03.005`.

8 Nicolas Bousquet, Aurélie Lagoutte, Zhentao Li, Aline Parreau, and Stéphan Thomassé. Identifying codes in hereditary classes of graphs and vc-dimension. *SIAM Journal on Discrete Mathematics*, 29(4):2047–2064, 2015.

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

10 A. Brandstädt, V. Chepoi, and F.F. Dragan. Tree-structured graphs. In *Handbook of Graph Theory, Combinatorial Optimization, and Algorithms*. CRC Press, 2016.

11 A. Brandstädt, F.F. Dragan, V. Chepoi, and V. Voloshin. Dually chordal graphs. *SIAM Journal on Discrete Mathematics*, 11(3):437–455, 1998.

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

13 P. Buneman. A characterisation of rigid circuit graphs. *Discrete Mathematics*, 9(3):205–212, 1974.

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

15 G. Chang and G. Nemhauser. The k-domination and k-stability problems on sun-free chordal graphs. *SIAM Journal on Algebraic Discrete Methods*, 5(3):332–345, 1984.

16 B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete & Computational Geometry*, 4(5):467–489, 1989.

17 D. Corneil, F. F. Dragan, M. Habib, and C. Paul. Diameter determination on restricted graph families. *Discrete Applied Mathematics*, 113(2-3):143–166, 2001.

18 M. Dalirrooyfard and N. Wein. Tight Conditional Lower Bounds for Approximating Diameter in Directed Graphs. In *53rd Annual ACM Symposium on Theory of Computing (STOC)*, 2021. To appear.

19 P. Dankelmann. Computing the average distance of an interval graph. *Information Processing Letters*, 48(6):311–314, 1993.

20 Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2010. 4*th* edition. doi:10.1007/978-3-662-53622-3.

21 F.F. Dragan, G. Ducoffe, and H.M. Guarnera. Fast deterministic algorithms for computing all eccentricities in (hyperbolic) Helly graphs, 2021. To appear. Technical report available on arXiv (arXiv:2102.08349).

22 G. Ducoffe and Feodor F.F. Dragan. A story of diameter, radius, and (almost) Helly property. *Networks*, 77(3):435–453, 2021.

23 G. Ducoffe, M. Habib, and L. Viennot. Fast diameter computation within split graphs. In *COCOA*, pages 155–167. Springer, 2019.

24 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.

25 Guillaume Ducoffe. Around the diameter of AT-free graphs. *CoRR*, abs/2010.15814, 2020. arXiv:2010.15814.

26 Guillaume Ducoffe. Optimal diameter computation within bounded clique-width graphs. *CoRR*, abs/2011.08448, 2020. arXiv:2011.08448.

27 David Eppstein and Joseph Wang. Fast approximation of centrality. *Journal of Graph Algorithms and Applications*, 8(1):39–45, 2004. doi:10.7155/jgaa.00081.

28 Jacob Evald and Søren Dahlgaard. Tight hardness results for distance and centrality problems in constant degree graphs. Technical Report arXiv:1609.08403, ArXiv, 2016.

29 S. Foldes and P.L. Hammer. Split graphs having Dilworth number two. *Canadian Journal of Mathematics*, 29(3):666–672, 1977.

30 F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.

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

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

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

34 Dirk Koschützki, Katharina Anna Lehmann, Leon Peeters, Stefan Richter, Dagmar Tenfelde-Podehl, and Oliver Zlotowski. Centrality indices. In *Network Analysis*, pages 16–61. Springer, 2005. doi:10.1007/978-3-7091-0741-6_9.

35 P.S. Kumar and C.V. Madhavan. Minimal vertex separators of chordal graphs. *Discrete Applied Mathematics*, 89(1-3):155–168, 1998.

36 R. Li. Settling SETH vs. Approximate Sparse Directed Unweighted Diameter (up to (NU)NSETH). In *53rd Annual ACM Symposium on Theory of Computing (STOC)*, 2021. To appear.

37 M. Moscarini. Doubly chordal graphs, Steiner trees, and connected domination. *Networks*, 23(1):59–69, 1993.

38 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.

39 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.

40 R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on computing*, 13(3):566–579, 1984.

41 J. R. Walter. *Representations of rigid cycle graphs.* PhD thesis, Wayne State University, Department of Mathematics, 1972.

42 H. Wiener. Structural determination of paraffin boiling points. *Journal of the American chemical society*, 69(1):17–20, 1947.

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