When Can Graph Hyperbolicity be Computed in Linear Time?

Till Fluschnik · Christian Komusiewicz ·
George B. Mertzios · André Nichterlein ·
Rolf Niedermeier · Nimrod Talmon

Abstract Hyperbolicity is a distance-based measure of how close a given graph is to being a tree. Due to its relevance in modeling real-world networks, hyperbolicity has seen intensive research over the last years. Unfortunately, the best known algorithms used in practice for computing the hyperbolicity number of an n-vertex graph have running time $O(n^4)$. Exploiting the framework of parameterized complexity analysis, we explore possibilities for "linear-time FPT" algorithms to compute hyperbolicity. For example, we show that hyperbolicity can be computed in $2^{O(k)} + O(n + m)$ time (where m and k denote the number of edges and the size of a vertex cover in the input graph, respectively) while at the same time, unless the Strong Exponential Time Hypothesis (SETH) fails, there is no $2^{o(k)} \cdot n^{2-\varepsilon}$-time algorithm for every $\varepsilon > 0$.

T. Fluschnik, A. Nichterlein, R. Niedermeier
Algorithmics and Computational Complexity, Fakultät IV, TU Berlin, Germany,
E-mail: {till.fluschnik, andre.nichterlein, rolf.niedermeier}@tu-berlin.de

C. Komusiewicz
Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Germany,
E-mail: komusiewicz@informatik.uni-marburg.de

G. B. Mertzios
Department of Computer Science, Durham University, UK,
E-mail: george.mertzios@durham.ac.uk

N. Talmon
Ben-Gurion University, Be'er Sheva, Israel,
E-mail: nimrodtalmon77@gmail.com

1 Introduction

Gromov hyperbolicity [21] is a popular attempt to capture and measure how *metrically* close a graph is to being a tree. The study of hyperbolicity is motivated by the fact that many real-world graphs are tree-like from a distance metric point of view [2, 4]. This is due to the fact that many of these graphs (including Internet application networks or social networks) possess certain geometric and topological characteristics. Hence, for many applications (cf., e.g., Borassi et al. [4]), including the design of efficient algorithms, it is useful to know the hyperbolicity of a graph. The hyperbolicity of a graph is a nonnegative number δ; the smaller δ is, the more tree-like the graph is; in particular, $\delta = 0$ means that the graph metric indeed is a tree metric. Typical hyperbolicity values for real-world graphs are below five [2]. Notably, the graph parameter treewidth—measuring tree-likeness in a non-metrical way—is unrelated to the hyperbolicity of a graph.

Hyperbolicity can be defined via a four-point condition: Considering a size-four subset $\{a, b, c, d\}$ of the vertex set of a graph, one takes the (nonnegative) difference between the two largest of the three sums $\overline{ab}+\overline{cd}$, $\overline{ac}+\overline{bd}$, and $\overline{ad}+\overline{bc}$, where $\overline{uv}$ denotes the length of a shortest path between vertices u and v in the given graph. The hyperbolicity is the maximum of these differences over all size-four subsets of the vertex set of the graph. For an n-vertex graph, this characterization of hyperbolicity directly implies a simple (brute-force) $O(n^4)$-time algorithm to compute its hyperbolicity. It has been observed that this running time is too slow for computing the hyperbolicity of large graphs as occurring in applications [2, 4, 5, 18]. On the theoretical side, it was shown that relying on some (rather impractical) matrix multiplication results, one can improve the upper bound to $O(n^{3.69})$ [18]. Moreover, roughly quadratic lower bounds are known [5, 18]. It is also known [12] that the problem of deciding whether a graph is 1-hyperbolic and the problem of deciding whether a graph contains a cycle of length four either both admit an $O(n^{3-\epsilon})$-time algorithm, for some $\epsilon > 0$, or neither does. In practice, however, the best known algorithm still has an $O(n^4)$-time worst-case bound but uses several clever tricks when compared to the straightforward brute-force algorithm [4]. Indeed, empirical studies suggest an $O(mn)$ running time, where m is the number of edges in the graph. Furthermore, there are heuristics for computing the hyperbolicity of a given graph [8]. Cohen et al. [9] study computing the hyperbolicity with a given clique-decomposition. In this context, they prove that computing the hyperbolicity of the subgraphs induced by the parts of the clique-decomposition yields a 1-additive approximation. Moreover, they prove that the hyperbolicity of an outerplanar graph can be computed in linear time.

The guiding principle of this work is to explore the possibility of faster algorithms for hyperbolicity in relevant special cases. More specifically, introducing some graph parameters, we investigate whether one can compute hyperbolicity in linear time when these parameters take small values. In other words, we employ the framework of parameterized complexity analysis (so far mainly used for studying NP-hard problems) applied to the polynomial-time

Table 1 Summary of our algorithmic results. Herein, k denotes the parameter and n and m denote the number of vertices and edges, respectively. RR abbreviates "Reduction Rule" used in the respective result. The symbol [†] stands for "together with Reduction Rule 1".

Parameter	Running time		
covering path number	$O(k(n+m)) + k^4(\log n)^{O(1)}$	[Thm. 1]	[RR 2][†]
feedback edge number	$O(k(n+m)) + k^4(\log n)^{O(1)}$	[Thm. 2]	[RR 2][†]
number of $\geq$ 3-degree vertices	$O(k^2(n+m)) + k^8(\log n)^{O(1)}$	[Thm. 3]	[RR 3][†]
vertex cover number	$O(n+m) + 2^{O(k)}$	[Thm. 4]	[RR 4][†]
distance to cographs	$O(4^{4k} \cdot k^7 \cdot (n+m))$	[Thm. 7]	

solvable hyperbolicity problem. In this sense, we follow the recently proposed framework of systematically studying "FPT in P" [20, 27]. Indeed, other than for NP-hard problems (where parameterized complexity is typically applied), for some parameters we achieve not only an exponential dependence on the parameter but also polynomial ones. Note, however, that for the metric parameters diameter or hyperbolicity, such linear-time algorithms are unlikely for any dependency on the parameter [5].

In this paper, we aim for developing *linear-time parameterized algorithms* for computing the hyperbolicity of a graph, that is, algorithms with running times of the form $f(k) \cdot |x|$, where $|x|$ denotes the input size, and $f(k)$ is some computable function only depending on the parameter k (referred to as L-FPT running time). A linear-time parameterized algorithm is *polynomial* if $f(k)$ is some polynomial in k (referred to as PL-FPT running time).

Our contributions. Table 1 summarizes our main results. On the positive side, for a number of natural graph parameters we can attain L-FPT running times. Our "positive" graph parameters here are the following:

– the *covering path* number, that is, the minimum number of paths where only the endpoints have degree greater than two and which cover all vertices;
– the *feedback edge* number, that is, the minimum number of edges to delete to obtain a forest;
– the number of graph vertices of *degree at least three*;
– the *vertex cover* number, that is, the minimum number of vertices needed to cover all edges in the graph;
– the *distance to cographs*, that is, the minimum number of vertices to delete to obtain a cograph.[1]

On the negative side, we prove for the parameter vertex cover number k that we cannot hope for any $2^{o(k)}n^{2-\epsilon}$-time algorithm unless the Strong Ex-

---

[1] Cographs are the graphs without induced P_4s [10]. The parameter distance to cographs is upper-bounded by the parameter distance to cluster graph [15] and thus also by the parameter vertex cover number.

ponential Time Hypothesis[2] (SETH) fails. We also obtain a "quadratic-time FPT" lower bound with respect to the parameter *maximum vertex degree*, again assuming SETH. We point out that our lower-bound result regarding vertex cover implies lower bounds for many other well-known graph parameters as feedback vertex set and pathwidth, which can be much smaller than the vertex cover number. As one representative of this group of parameters—being unrelated to the latter two mentioned parameters—we studied vertex-deletion distance to cographs. Our remaining three parameters, namely covering path number, number of ≥ 3-degree vertices, and feedback edge number, are unrelated to the vertex cover number and can be arbitrarily larger than the vertex cover number (consider any biclique $K_{2,n}$, any biclique $K_{3,n}$, and any cycle C_n, respectively).

Finally, we show that computing the hyperbolicity is at least as hard as computing a *size-four independent set* in a graph. It is conjectured that computing size-four independent sets needs $\Omega(n^3)$ time [31].

2 Preliminaries and Basic Observations

We write $[n] := \{1, \ldots, n\}$ for every $n \in \mathbb{N}$. For a function $f : X \to Y$ and $X' \subseteq X$ we set $f(X') := \{f(x) \mid x \in X'\}$.

Parameterized Complexity. A parameterized problem is a set of instances $(x, k) \in \Sigma^* \times \mathbb{N}$, where Σ denotes a finite alphabet. A *kernelization* is a polynomial-time algorithm that maps any instance (x, k) to an equivalent instance (x', k') (the *kernel*) such that $|x'| + k' \leq f(k)$ for some computable function f.

Graph theory. All graphs within this work are undirected. Let $G = (V, E)$ be an undirected graph. We define $|G| = |V| + |E|$. For $W \subseteq V$, we denote by $G[W]$ the graph *induced* by W. We use $G - W := G[V \setminus W]$ to denote the graph obtained from G by deleting the vertices of $W \subseteq V$. For $F \subseteq E$, we denote by $G - F$ the graph with vertex set V and edge set $E \setminus F$. A *path* $P = (v_1, \ldots, v_k)$ in G is a tuple of distinct vertices in V such that $\{v_i, v_{i+1}\} \in E$ for all $i \in [k-1]$; we say that such a path P has *endpoints* v_1 and v_k, we call the other vertices of P *inner vertices*, and we say that P is a v_1-v_k path. We denote by $\overline{ab}$ the length of a shortest a-b path if such a path exists; otherwise, that is, if a and b are in different connected components, we define $\overline{ab} := \infty$. Let $P = (v_1, \ldots, v_k)$ be a path and v_i, v_j two vertices on P. We denote by $\overline{v_i v_j}|_P$ the distance of v_i to v_j on the path P, that is, $\overline{v_i v_j}|_P = |j - i|$. For two vertices $a, b \in V$, a vertex set $S \subseteq V \setminus \{a, b\}$ is an a-b *separator* if there is no a-b path in $G - S$.

For a graph G we denote by $V_G^{\geq 3}$ the set of vertices of G that have degree at least three.
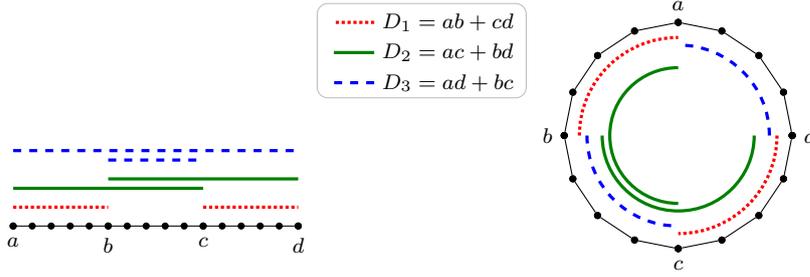

Fig. 1 Two illustrative examples for the four-point condition. *Left*: A path P_{16} with 16 vertices; *Right*: A cycle C_{16} with 16 vertices. Indeed, $\delta(P_{16}) = 0$, here indicated by the equally largest two distance sums D_2 (solid) and D_3 (dashed). The situation changes in the case of a C_{16} (that is, the P_{16} with one additional edge connecting the endpoints). We have $\delta(C_{16}) = 8$, here indicated by the largest distance sum D_2 (solid) and the two equally smallest distance sums D_1 (dotted) and D_2 (dashed).

Hyperbolicity. Let $G = (V, E)$ be graph and $a, b, c, d \in V$. We define $D_1 := \overline{ab} + \overline{cd}$, $D_2 := \overline{ac} + \overline{bd}$, and $D_3 := \overline{ad} + \overline{bc}$ (referred to as *distance sums*). Moreover, we define $\delta(a, b, c, d) := |D_i - D_j|$ if $D_k \leq \min\{D_i, D_j\}$, for pairwise distinct $i, j, k \in \{1, 2, 3\}$. If any two vertices of the quadruple $\{a, b, c, d\}$ are not connected by a path, then we set $\delta(a, b, c, d) := 0$.[3] The *hyperbolicity* of $G = (V, E)$ is defined as $\delta(G) := \max_{a,b,c,d \in V}\{\delta(a, b, c, d)\}$. We refer to Figure 1 for two illustrative examples. Note that by our definition, if G is not connected, then $\delta(G)$ computes the maximal hyperbolicity over all connected components of G. We say that the graph is δ-*hyperbolic* for some $\delta \in \mathbb{N}$ if it has hyperbolicity at most δ. That is, a graph is δ-hyperbolic[4] if for each 4-tuple $a, b, c, d \in V$ we have

$$\overline{ab} + \overline{cd} \leq \max\{\overline{ac} + \overline{bd}, \overline{ad} + \overline{bc}\} + \delta.$$

Formally, the HYPERBOLICITY problem is defined as follows.

HYPERBOLICITY

Input: An undirected graph $G = (V, E)$ and a positive integer δ.
Question: Is G δ-hyperbolic?

The following lemmas will be useful later. For any quadruple $\{a, b, c, d\}$, Lemma 1 upper-bounds $\delta(a, b, c, d)$ by twice the distance between any pair of vertices of the quadruple. Lemma 2 considers graphs for which the hyperbolicity equals the diameter. Lemma 3 is used in the proof of Reduction Rule 1.

Lemma 1 ([8, Lemma 3.1]) $\delta(a, b, c, d) \leq 2 \cdot \min_{u \neq v \in \{a,b,c,d\}}\{\overline{uv}\}$

An implicit proof of the following lemma is given by Mitsche and Pralat [28]. We provide a direct proof of our particular statement.

---

[3] This case is often left undefined in the literature. Our definition, however, allows to consider also disconnected graphs.

[4] Note that there is also a slightly different definition where graphs that we call δ-hyperbolic are called 2δ-hyperbolic [8, 28]; we follow the definition of Brinkmann et al. [7].

Lemma 2 *Let G be a graph with diameter h and $\delta(G) = h$. Then for each quadruple $a, b, c, d \in V(G)$ with $\delta(a, b, c, d) = h$, it holds that exactly two disjoint pairs are at distance h and all the other pairs are at distance $h/2$.*

Proof Let $a, b, c, d \in V(G)$ be an arbitrary but fixed quadruple with $\delta(a, b, c, d) = h$. Without loss of generality, assume $D_1 = \overline{ab} + \overline{cd}$ and $D_1 \geq \max\{D_2, D_3\}$. By Lemma 1 we have $\min_{u \neq v \in \{a,b,c,d\}}\{\overline{uv}\} \geq h/2$, and hence $\max\{D_2, D_3\} \geq h/2 + h/2 = h$. It follows that $h = D_1 - \max\{D_2, D_3\} \leq D_1 - h$ and thus $D_1 \geq 2h$. Since G is of diameter h, we get $\overline{ab} = \overline{cd} = h$. Moreover, $\max\{D_2, D_3\} = h$ and together with $\min_{u \neq v \in \{a,b,c,d\}}\{\overline{uv}\} \geq h/2$, we obtain that each other distance equals $h/2$. □

The following lemma immediately follows from a result due to Cohen et al. [9, Theorem 5].

Lemma 3 *Let $G = (V, E)$ be a graph with $|V| > 4$ and with a vertex $v \in V$ such that the number of connected components in $G - \{v\}$ is larger than in G. Let $A_1, \ldots, A_\ell$ be the components in $G - \{v\}$. Then there is an $i \in [\ell]$ such that $\delta(G) = \delta(G - V(A_i))$.* □

Lemma 3 gives rise to the following degree-1 data reduction rule.

Reduction Rule 1 As long as there are more than four vertices, remove vertices of degree at most one.

Lemma 4 *Reduction Rule 1 is sound and can be exhaustively applied in linear time.*

Proof The soundness of Reduction Rule 1 follows immediately from Lemma 3. To apply Reduction Rule 1 in linear time do the following. First, as long as there are more than four vertices, delete degree-zero vertices. Second, collect all vertices with degree at most one in linear time in a list L. Then, as long as there are more than four vertices, iteratively delete degree-one vertices and put their neighbor in L if it has degree at most one after the deletion. Each iteration can be applied in constant time. Thus, Reduction Rule 1 can be applied in linear time. □

We call a graph *reduced* if Reduction Rule 1 is not applicable.

3 Polynomial Linear-Time Parameterized Algorithms

In this section, we provide polynomial linear-time parameterized algorithms with respect to the parameters feedback edge number and number of vertices with degree at least three.

To this end, we first introduce an auxiliary parameter, the *minimum maximal path cover number* (formally defined below), for which we also describe a polynomial linear-time parameterized algorithm.

Building upon this result, for the parameter feedback edge number we then show that, after applying Reduction Rule 1, the number of maximal paths can be upper-bounded by a polynomial of the feedback edge number. This implies a polynomial linear-time parameterized algorithm for the feedback edge number as well. For the parameter number of vertices with degree at least three, we introduce an additional data reduction rule to achieve that the number of maximal paths is upper-bounded in a polynomial of this parameter. Again, this implies an algorithm with PL-FPT running time.

Minimum maximal path cover number. We first give the definition of maximal paths and then discuss graphs that can be covered by maximal paths.

Definition 1 (Maximal path) Let G be a graph and P be a path in G. Then, P is a *maximal path* if the following hold: (1) P contains at least two vertices; (2) all inner vertices of P have degree two in G; (3) both endpoints of P have degree at least three in G.

We will be interested in the minimum number of maximal paths needed to cover the vertices of a given graph; we call this number the *minimum maximal path cover number*. A *pending cycle* in a graph is an induced cycle in G with at most one vertex of degree larger than two. A pending cycle with no vertex of degree larger than two is called *isolated*. While not all graphs can be covered by maximal paths (e.g., edgeless graphs), graphs which have minimum degree two and contain no pending cycles can be covered by maximal paths (this follows by, e.g., a greedy algorithm which iteratively starts a path with an arbitrary uncovered vertex and exhaustively extends it arbitrarily; since there are no isolated cycles and the minimum degree is two, we are bound to eventually hit at least one vertex of degree three).

Reduction Rule 2 Let $\mathcal{I} = (G, \delta)$ be an instance of HYPERBOLICITY with $C \subseteq G$ being a pending cycle of G. If $\delta(C) > \delta$, then return that I is a no-instance, and otherwise, delete from G all vertices $v \in V(C)$ with $\deg(v) = 2$ and their incident edges.

The correctness of Reduction Rule 2 follows immediately from Lemma 3. We refer to a reduced graph G with no pending cycles as *cycle-reduced*. Based on Reduction Rule 2, we have the following.

Lemma 5 *There is a linear-time algorithm that given an instance $\mathcal{I} = (G, \delta)$ of HYPERBOLICITY, either decides $\mathcal{I}$ or computes an instance (G', δ) equivalent to $\mathcal{I}$ and a set $\mathcal{P}(G')$ such that $G' \subseteq G$ is cycle-reduced and $\mathcal{P}(G') \subseteq \mathcal{P}(G)$ is the set of all maximal paths in G' of length at least three.*

Proof First, we apply Reduction Rule 1 to have a graph with no vertices of degree at most one. Next, we employ the linear-time algorithm by Bentert et al. [3, Lemma 2] towards computing the set of all pending cycles and the set of maximal paths. Herein, instead of storing the pending cycles, in each

iteration of the algorithm where a pending cycle C_{4p+q} is found, we apply Reduction Rule 2. Note that for a cycle C_{4p+q} with $p \in \mathbb{N}$ and $q \in \{0, 1, 2, 3\}$ it holds true that $\delta(C_{4p+q}) = 2p$ if $q \in \{0, 2, 3\}$, and $\delta(C_{4p+q}) = 2p - 1$ otherwise (that is, if $q = 1$) [25]. If the deletion causes a vertex to have degree one, then, starting at this vertex, we iteratively delete vertices of degree one along the path (as long as there are more than four vertices). Then we continue.   □

Based on the linear-time approximation algorithm given in the next lemma, we assume in the following that we are given a maximal path cover.

Lemma 6 *There is a linear-time 2-approximation algorithm for the minimum maximal path cover number for cycle-reduced graphs.*

Proof The algorithm operates in two phases. In the first phase, we employ Lemma 5 to obtain set of all maximal paths of length at least three.

The second phase begins when all vertices of degree two are already covered. In the second phase, ideally we would find a matching between those uncovered vertices of degree at least three. To get a 2-approximation we arbitrarily select a vertex of degree at least three, view it as a path of length one, and arbitrarily extend it until it is maximal. This finishes the description of the linear-time algorithm.

For correctness of the first phase, the crucial observation is that each vertex of degree two has two be covered by at least one path. For the second phase, the factor 2 follows since each maximal path can cover at most two vertices of degree at least three.   □

Now we are ready to present a polynomial linear-time parameterized algorithm for HYPERBOLICITY with respect to the minimum maximal path cover number.

Theorem 1 *Let $G = (V, E)$ be a cycle-reduced graph and k be its minimum maximal path cover number. Then, HYPERBOLICITY can be solved in $O(k(n + m)) + k^4(\log n)^{O(1)}$ time.*

Proof We use Lemma 6 to get a set $\mathcal{P}$ of at most $2k$ maximal paths which cover G. By initiating a breadth-first search from each of the endpoints of those maximal paths, we can compute the pairwise distances between those endpoints in $O(k(n + m))$ time. Thus, for the rest of the algorithm we assume that we can access the distances between any two vertices which are endpoints of those maximal paths in constant time.

Let (a, b, c, d) be a quadruple such that $\delta(a, b, c, d) = \delta(G)$. Since the set $\mathcal{P}$ covers all vertices of G, each vertex of a, b, c, and d belongs to some path $P \in \mathcal{P}$. Since $|\mathcal{P}| \le 2k$, there are $O(k^4)$ possibilities to assign the vertices a, b, c, and d to paths in $\mathcal{P}$. For each possibility we compute the maximum hyperbolicity respecting the assignment, that is, we compute the positions of the vertices on their respective paths that maximize $\delta(a, b, c, d)$. We achieve this by formulating an integer linear program (ILP) with a constant number of variables and constraints whose coefficients have value at most n.

To this end, denote with $P_a, P_b, P_c, P_d \in \mathcal{P}$ the paths containing a, b, c, d, respectively. We assume for now that these paths are different and discuss subsequently the case that one path contains at least two vertices from a, b, c, d. Let a_1 and a_2 ($b_1, b_2, c_1, c_2, d_1, d_2$) be the endpoints of P_a (P_b, P_c, P_d, respectively). Furthermore, denote by m_P the length of a path $P \in \mathcal{P}$, that is, the number of its edges. Without loss of generality assume that $D_1 \geq D_2 \geq D_3$. We now compute the positions of the vertices on their respective paths that maximize $D_1 - D_2$ by solving an ILP. Recall that $\overline{v_1 v}|_{P_v}$ denotes the distance of v to v_1 on P_v. Thus, $\overline{v_1 v}|_{P_v} + \overline{v v_2}|_{P_v} = m_{P_v}$, and $\overline{v_1 v}|_{P_v} \geq 0$ and $\overline{v v_2}|_{P_v} \geq 0$. The following is a compressed description of the ILP containing the minimum function. We describe below how to remove it.

$$
\begin{aligned}
\text{maximize:} \quad & D_1 - D_2 \\
\text{subject to:} \quad & D_1 = \overline{ab} + \overline{cd} \\
& D_2 = \overline{ac} + \overline{bd} \\
& D_3 = \overline{ad} + \overline{bc} \\
& D_1 \geq D_2 \geq D_3 \\
\forall x \in \{a,b,c,d\}: \quad & m_{P_x} = \overline{x_1 x}|_{P_x} + \overline{x x_2}|_{P_x} \quad (1)
\end{aligned}
$$

$$
\forall x, y \in \{a,b,c,d\}: \quad \overline{xy} = \min \left\{ \begin{array}{l} \overline{x_1 x}|_{P_x} + \overline{x_1 y_1} + \overline{y_1 y}|_{P_y}, \\ \overline{x_1 x}|_{P_x} + \overline{x_1 y_2} + \overline{y y_2}|_{P_y}, \\ \overline{x x_2}|_{P_x} + \overline{x_2 y_1} + \overline{y_1 y}|_{P_y}, \\ \overline{x x_2}|_{P_x} + \overline{x_2 y_2} + \overline{y y_2}|_{P_y} \end{array} \right\} \quad (2)
$$

First, observe that the ILP obviously has a constant number of variables: $\overline{xy}$, for all $x, y \in \{a,b,c,d\}$, $x \neq y$, $\overline{x x_j}|_{P_x}$, for all $x \in \{a,b,c,d\}$ and $j \in \{1,2\}$, and D_j, $j \in \{1,2,3\}$. The only constant coefficients are $\overline{x_i y_j}$ for $x, y \in \{a,b,c,d\}$ and $i,j \in \{1,2\}$ and obviously have value at most $n-1$. To remove the minimization function in Equation (2), we use another case distinction: We simply try all possibilities of which value is the smallest one and adjust the ILP accordingly. For example, for the case that for some $x, y \in \{a,b,c,d\}$ the minimum in Equation (2) is $\overline{x x_1}|_{P_x} + \overline{x_1 y_1} + \overline{y_1 y}|_{P_y}$, we replace this equation by the following:

$$
\begin{aligned}
\overline{xy} &= \overline{x_1 x}|_{P_x} + \overline{x_1 y_1} + \overline{y_1 y}|_{P_y} \\
\overline{xy} &\leq \overline{x_1 x}|_{P_x} + \overline{x_1 y_2} + \overline{y y_2}|_{P_y} \\
\overline{xy} &\leq \overline{x x_2}|_{P_x} + \overline{x_2 y_1} + \overline{y_1 y}|_{P_y} \\
\overline{xy} &\leq \overline{x x_2}|_{P_x} + \overline{x_2 y_2} + \overline{y y_2}|_{P_y}
\end{aligned}
$$

There are four possibilities of which value is the smallest one, and we have to consider each of them independently for each of the $\binom{4}{2} = 6$ pairs. Hence, for each assignment of the vertices a, b, c, and d to paths in $\mathcal{P}$, we need to solve $4 \cdot 6 = 24$ different ILPs in order to remove the minimization function. Since each ILP has a constant number of variables and constraints, this takes $L^{O(1)}$ time

where $L = O(\log n)$ is the total size of the ILP instance (for example by using the algorithm of Lenstra [26]).

It remains to discuss the case that at least two vertices of a, b, c, and d are assigned to the same path $P \in \mathcal{P}$. As a deputy, we show the changes for the case that a, b, and c are mapped to $P_a \in \mathcal{P}$. The adjustments for the other cases can be done in a similar fashion. We assume without loss of generality that the vertices a_1, a, b, c, a_2 appear in this order in P (allowing $a = a_1$ and $c = a_2$). The objective function as well as the first four lines of the ILP remain unchanged. Equation (1) is replaced with the following:

$$m_{P_a} = \overline{a_1 a}|_{P_a} + \overline{ab}|_{P_a} + \overline{bc}|_{P_a} + \overline{ca_2}|_{P_a}$$
$$m_{P_d} = \overline{d_1 d}|_{P_d} + \overline{dd_2}|_{P_a}$$

To ensure that Equation (2) works as before, we add the following:

$$\overline{aa_2}|_{P_a} = \overline{ab}|_{P_a} + \overline{bc}|_{P_a} + \overline{ca_2}|_{P_a}$$
$$\overline{b_1 b}|_{P_b} = \overline{a_1 a}|_{P_a} + \overline{ab}|_{P_a}$$
$$\overline{bb_2}|_{P_b} = \overline{bc}|_{P_a} + \overline{ca_2}|_{P_a}$$
$$\overline{c_1 c}|_{P_c} = \overline{a_1 a}|_{P_a} + \overline{ab}|_{P_a} + \overline{bc}|_{P_a}$$
$$\overline{cc_2}|_{P_c} = \overline{ca_2}|_{P_a}$$

$\square$

Feedback edge number. We next present a polynomial linear-time parameterized algorithm with respect to the parameter feedback edge number k. The idea is to show that a graph that is cycle-reduced contains $O(k)$ maximal paths.

Theorem 2 HYPERBOLICITY *can be solved in* $O(k(n + m)) + k^4 (\log n)^{O(1)}$ *time, where k is the feedback edge number.*

Proof The first step of the algorithm is to apply the algorithm of Lemma 5. After this step, if the input instance is not yet decided, we can assume our input graph to be cycle-reduced.

Denote by $X \subseteq E$ a minimum feedback edge set for the cycle-reduced graph $G = (V, E)$ and observe that $|X| = k$. We will show that the minimum maximal path cover number of G is $O(k)$. More precisely, we show the slightly stronger claim that the number of maximal paths in G is $O(k)$.

Observe that all vertices in G have degree at least two since G is cycle-reduced. Thus, every leaf of $G - X$ is incident with at least one feedback edge which implies that there are at most $2k$ leaves in $G - X$. Moreover, since $G - X$ is a forest, the number of vertices with degree at least three in $G - X$ is at most the number of leaves in $G - X$ and thus at most $2k$. This implies that the number of maximal paths in $G - X$ is at most $2k$ (each maximal path corresponds to an edge in the forest obtained from $G - X$ by contracting all degree-two vertices).

We now show that number of maximal paths in G is linear in k by showing that an insertion of an edge into any graph H increases the number of maximal paths by at most three. First, note that each edge can be part of at most one maximal path in any graph. If each endpoint of the edge to insert is of degree two in H, then the number of maximal paths increases by three. This is indeed the maximal case since in the case that at least one endpoint is of degree at least three or at most one the insertion increases the number of maximal paths by at most two. Thus G contains at most $5k$ maximal paths. The statement of the theorem now follows from Theorem 1.   □

Number of vertices with degree at least three. We finally show a polynomial linear-time parameterized algorithm with respect to the number k of vertices with degree three or more. To this end, we use the following data reduction rule additionally to the linear-time algorithm of Lemma 5 to upper-bound the number of maximal paths in the graph by $O(k^2)$ (with the goal to make use of Theorem 1).

Reduction Rule 3 Let $G = (V, E)$ be a graph, $u, v \in V_G^{\geq 3}$ be two vertices of degree at least three, and $\mathcal{P}_{uv}$ be the set of maximal paths in G with endpoints u and v. Let $\mathcal{P}_{uv}^9 \subseteq \mathcal{P}_{uv}$ be the set containing the shortest path, the four longest even-length paths, and the four longest odd-length paths in $\mathcal{P}_{uv}$. If $\mathcal{P}_{uv} \setminus \mathcal{P}_{uv}^9 \neq \emptyset$, then delete in G all inner vertices of the paths in $\mathcal{P}_{uv} \setminus \mathcal{P}_{uv}^9$.

Lemma 7 *Reduction Rule 3 is sound and can be exhaustively applied in linear time.*

Proof We first prove the running time. In linear time we compute the set $V_G^{\geq 3}$ of all vertices with degree at least three. Then for each $v \in V_G^{\geq 3}$ we do the following. Starting from v, we perform a modified breadth-first search that stops at vertices in $V_G^{\geq 3}$. Let $R(V_G^{\geq 3}, v)$ denote the visited vertices and edges. Observe that $R(V_G^{\geq 3}, v)$ consists of v, some degree-two vertices, and all vertices of $V_G^{\geq 3}$ that can be reached from v via maximal paths in G. Furthermore, with the breadth-first search approach we can also compute for all $u \in R(V_G^{\geq 3}, v) \cap V_G^{\geq 3}$ with $u \neq v$ the number of maximal paths between u and v and their respective lengths. Then, in time linear in $|R(V_G^{\geq 3}, v)|$, we remove the paths in $\mathcal{P}_{uv} \setminus \mathcal{P}_{uv}^9$ for all $u \in R(V_G^{\geq 3}, v) \cap V_G^{\geq 3}$. Thus, we can apply Reduction Rule 3 for each $v \in V_G^{\geq 3}$ in $O(|R(V_G^{\geq 3}, v)|)$ time. Altogether, the running time is

$$O(\sum_{v \in V_G^{\geq 3}} |R(V_G^{\geq 3}, v)|) = O(n + m)$$

where the equality follows from the fact each edge and each maximal path in G is visited twice by the modified breadth-first search.

We now prove the soundness of the data reduction rule. To this end, let $G = (V, E)$ be the input graph, let $P \in \mathcal{P}_{uv} \setminus \mathcal{P}_{uv}^9$ be a maximal path from u to v whose inner vertices are removed by the application of the data reduction rule,
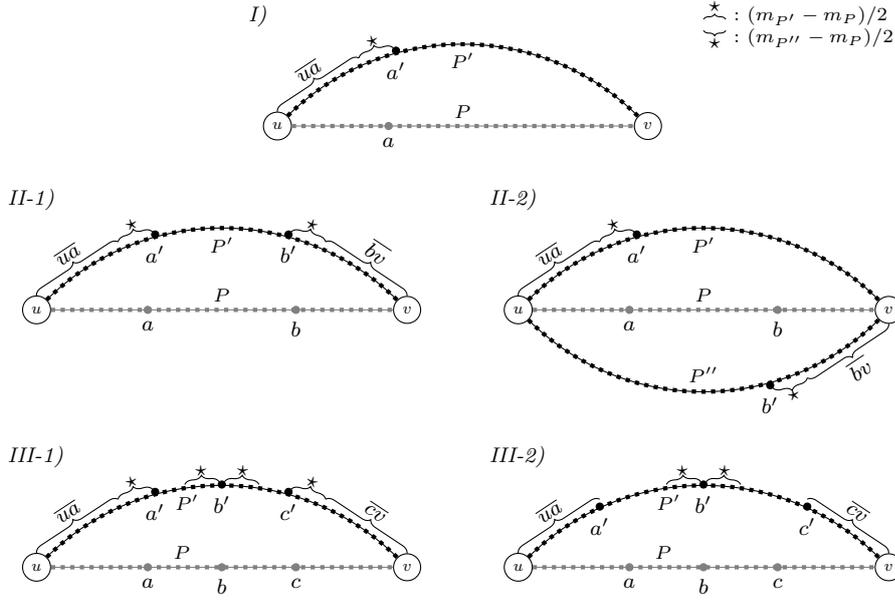
Fig. 2 Illustrations to the cases *I–III* and the subcases therein in the proof of Lemma 7.

and let $G' = (V', E')$ be the resulting graph. We show that $\delta(G) = \delta(G')$. The soundness of Reduction Rule 3 follows then from iteratively applying this argument. First, observe that since $\mathcal{P}^9_{uv}$ contains the shortest maximal path of $\mathcal{P}_{uv}$, it follows that u and v have the same distance in G and G'. Furthermore, it is easy to see that each pair of vertices $w, w' \in V'$ has the same distance in G and G' (Reduction Rule 3 removes only paths and does not introduce degree-one vertices). Hence, we have that $\delta(G) \geq \delta(G')$ and it remains to show that $\delta(G) \leq \delta(G')$

Towards showing that $\delta(G) \leq \delta(G')$, let $a, b, c, d \in V$ be the four vertices defining the hyperbolicity of G, that is, $\delta(G) = \delta(a, b, c, d)$. If P does not contain any of these four vertices, then we are done. Thus, assume that P contains at least one vertex from $\{a, b, c, d\}$. (For convenience, we say in this proof that a path Q contains a vertex z if z is an inner vertex of Q.) We next do a case distinction on the number of vertices of $\{a, b, c, d\}$ that are contained in P (we refer to Figure 2 for illustrations of the cases *I–III* and subcases therein).

Case (I): P contains one vertex of $\{a, b, c, d\}$. Without loss of generality assume that P contains a. We show that we can replace a by another vertex a' in a path $P' \in \mathcal{P}^9_{uv}$ such that $\delta(a, b, c, d) = \delta(a', b, c, d)$. Since P contains a, we can choose P' as one of the four (odd/even)-length longest paths in $\mathcal{P}^9_{uv}$ such that

- $m_{P'} - m_P$ is nonnegative and even (either both lengths are even or both are odd) and

   − P' contains no vertex of $\{b, c, d\}$.

Since P is removed by Reduction Rule 3, it follows that $m_P \leq m_{P'}$. We choose a' on P' such that $\overline{ua'}|_{P'} = \overline{ua}|_P + (m_{P'} - m_P)/2$. Observe that this implies that $\overline{a'v}|_{P'} = \overline{av}|_P + (m_{P'} - m_P)/2$ and thus

$$\overline{ua}|_P - \overline{av}|_P = \overline{ua'}|_{P'} - \overline{a'v}|_{P'}.$$

Recall that

$$D_1 := \overline{ab} + \overline{cd}, \qquad D_2 := \overline{ac} + \overline{bd}, \text{ and} \qquad D_3 := \overline{ad} + \overline{bc}.$$

Denote with D_1', D_2', and D_3' the respective distance sums resulting from replacing a with a', for example $D_1' = \overline{a'b} + \overline{cd}$. Observe that by the choice of a' we increased all distance sums by the same amount, that is, for all $i \in \{1, 2, 3\}$ we have $D_i' = D_i + (m_{P'} - m_P)/2$. Since $\delta(a, b, c, d) = D_i - D_j$ for some $i, j \in \{1, 2, 3\}$, we have that

$$\delta(G') = \delta(a', b, c, d) = D_i' - D_j' = \delta(a, b, c, d) = \delta(G).$$

    Case (II): P contains two vertices of $\{a, b, c, d\}$. Without loss of generality, assume that P contains a and b but not c and d, and a is closer to u on P than b. We follow a similar pattern as in the previous case and again use the same notation. Let $P', P'' \in \mathcal{P}_{uv}^9$ be the two longest paths such that both P' and P'' do neither contain c nor d and both $m_{P'} - m_P$ and $m_{P''} - m_P$ are even. We distinguish two subcases:

    Case (II-1): D_1 is not the largest sum ($D_1 < D_2$ or $D_1 < D_3$). We replace a and b with a' and b' on P' such that $\overline{ua'}|_{P'} = \overline{ua}|_P + (m_{P'} - m_P)/2$ and $\overline{ub'}|_{P'} = \overline{ub}|_P + (m_{P'} - m_P)/2$. Thus, $D_1' = D_1$ since $\overline{ab} = \overline{a'b'}$. However, for $i \in \{2, 3\}$ we have $D_i' = D_i + (m_{P'} - m_P)/2$. Since either D_2 or D_3 was the largest distance sum, we obtain

$$\delta(G) = \delta(a, b, c, d) = D_i - D_j \leq D_i' - D_{j'}' = \delta(a', b', c, d) = \delta(G')$$

for some $i \in \{2, 3\}$, $j, j' \in \{1, 2, 3\}$, $i \neq j$, and $i \neq j'$.

    Case (II-2): D_1 is the largest sum ($D_1 \geq D_2$ and $D_1 \geq D_3$). We need another replacement strategy since we did not increase D_1 in case (*II-1*). In fact, we replace a and b with two vertices on different paths P' and P''. We replace a with a' on P' and b with b' on P'' such that $\overline{ua'}|_{P'} = \overline{ua}|_P + (m_{P'} - m_P)/2$ and $\overline{b'v}|_{P''} = \overline{bv}|_P + (m_{P''} - m_P)/2$. Observe that for $i \in \{2, 3\}$ it holds that

$$D_i' = D_i + (m_{P'} - m_P)/2 + (m_{P''} - m_P)/2.$$

Moreover, since a' and b' are on different maximal paths, we also have

$$\begin{aligned}
\overline{ab} &\leq \min_{x \in \{u, v\}} \{\overline{xa}|_P + \overline{xb}|_P\} \\
&= \min_{x \in \{u, v\}} \{\overline{xa'}|_{P'} + \overline{xb'}|_{P''}\} - \frac{m_{P'} - m_P}{2} - \frac{m_{P''} - m_P}{2} \\
&= \overline{a'b'},
\end{aligned}$$

where the last equality is due to the fact that $\{u, v\}$ forms an a'-b' separator in G'. Thus $D'_1 \geq D_1 + (m_{P'} - m_P)/2 + (m_{P''} - m_P)/2$. It follows that

$$\delta(G) = \delta(a, b, c, d) = D_1 - D_j \leq D'_1 - D'_j = \delta(a', b', c, d) = \delta(G')$$

for some $j \in \{2, 3\}$.

Case (III): P contains three vertices of $\{a, b, c, d\}$. Without loss of generality, assume that P contains a, b, and c but not d and that among a, b, c vertex a is the closest vertex to u on P and c is the closest vertex to v on P (that is, a, b, c appear in this order on P). We distinguish two subcases.

Case (III-1): $\overline{ac}|_P = \overline{ac}$. We follow a similar pattern as in case (I) and use the same notation. Again, there is a $P' \in \mathcal{P}^9_{uv}$ such that $m_{P'} - m_P$ is even and nonnegative and P' does not contain d. We replace each vertex a, b, c as in case (I), that is, for each $x \in \{a, b, c\}$ we choose x' on P' such that $\overline{ux'}|_{P'} = \overline{ux}|_P + (m_{P'} - m_P)/2$. Observe that only the distances between d and the other three vertices change. Thus, we have again for all $i \in \{1, 2, 3\}$ that $D'_i = D_i + (m_{P'} - m_P)/2$ and hence $\delta(G) = \delta(G')$.

Case (III-2): $\overline{ac}|_P > \overline{ac}$. We use again a similar strategy as in case (I) and use the same notation. Again, there is a $P' \in \mathcal{P}^9_{uv}$ such that $m_{P'} - m_P$ is even and nonnegative and P' does not contain d. We replace the vertices a, b, c with a', b', c' on P' such that

- $\overline{au} = \overline{au}|_P = \overline{a'u}|_{P'} = \overline{a'u}$,
- $\overline{cv} = \overline{cv}|_P = \overline{c'v}|_{P'} = \overline{c'v}$,
- $\overline{b'u}|_{P'} = \overline{bu}|_P + (m_{P'} - m_P)/2$, and
- $\overline{b'v}|_{P'} = \overline{bv}|_P + (m_{P'} - m_P)/2$.

Note that since $\overline{ac}|_P > \overline{ac}$, it follows that the distances not involving b (resp. b') remain unchanged, that is, $\overline{ac} = \overline{a'c'}$, $\overline{ad} = \overline{a'd}$, and $\overline{cd} = \overline{c'd}$. Furthermore, all distances involving b (resp. b') increase by $(m_{P'} - m_P)/2$, that is, $\overline{bx} = \overline{b'x} - (m_{P'} - m_P)/2$ for each $x \in \{a, c, d\}$. Thus, we have again for all $i \in \{1, 2, 3\}$ that $D'_i = D_i + (m_{P'} - m_P)/2$ and hence $\delta(G) = \delta(G')$.

Case (IV): P contains all four vertices of $\{a, b, c, d\}$. We consider two subcases.

Case (IV-1): the union of the shortest paths between these four vertices induces a path. In this case, we have $\delta(G) = 0$ and thus trivially $\delta(G) \leq \delta(G')$.

Case (IV-2): the union of the shortest paths between these four vertices induces a cycle C. As before, there is a path $P' \in \mathcal{P}^9_{uv}$ such that $m_{P'} - m_P$ is nonnegative and even. Let Q denote the shortest path on C between u and v. Observe that Q contains no vertex in $\{a, b, c, d\}$ and is present in G'. Denote by C' the cycle formed by Q and P'. Note that $|C'| \geq |C|$ since $m_{P'} \geq m_P$. Moreover, $(|C'| - |C|) \bmod 2 = 0$ since $(m_{P'} - m_P) \bmod 2 = 0$, and hence the case $C = C_{4p}$ and $C' = C_{4p+1}$ for some $p \in \mathbb{N}$ is excluded. Thus, it holds true that $\delta(C') \geq \delta(C)$. It follows that $\delta(G') \geq \delta(C') \geq \delta(C) = \delta(a, b, c, d) = \delta(G)$. □

Observe that if the graph G is reduced with respect to Reduction Rule 3 after Lemma 5 was applied, then for each pair $u, v \in V_G^{\geq 3}$ there exist at most

nine maximal paths with endpoints u and v. Thus, G contains at most $O(k^2)$ maximal paths and using Theorem 1 we arrive at the following.

Theorem 3 HYPERBOLICITY *can be solved in* $O(k^2(n + m)) + k^8(\log n)^{O(1)}$ *time, where k is the number of vertices with degree at least three.* □

4 Exponential Linear-Time Parameterized Algorithms

In this section, we provide exponential linear-time parameterized algorithms with respect to the parameters vertex cover number (Section 4.1) and vertex-deletion distance to cographs (Section 4.2).

4.1 Parameter Vertex Cover Number

A *vertex cover* of a graph $G = (V, E)$ is a subset $W \subseteq V$ of vertices of G such that each edge in G is incident to at least one vertex in W. Deciding whether a graph G has a vertex cover of size at most k is NP-complete in general [19]. There is, however, a simple greedy linear-time factor-2 approximation (see, e.g., [29]). In this section, we consider the size k of a vertex cover as the parameter. We show that we can solve HYPERBOLICITY in time linear in $|G|$, but exponential in k; further, we show that, unless SETH fails, we cannot do asymptotically better.

A Linear-Time Algorithm Parameterized by the Vertex Cover Number. We prove that HYPERBOLICITY can be solved in time linear in the size of the graph and exponential in the size k of a vertex cover. This result is based on a linear-time computable problem kernel consisting of $O(2^k)$ vertices that can be obtained by exhaustively applying the following data reduction rule.

Reduction Rule 4 If there are at least five vertices $v_1, \ldots, v_\ell \in V$, $\ell > 4$, with the same (open) neighborhood $N(v_1) = N(v_2) = \ldots = N(v_\ell)$, then delete $v_5, \ldots, v_\ell$.

We next show that the above rule is correct, can be applied in linear time, and leads to a problem kernel for the parameter vertex cover number.

Lemma 8 *Reduction Rule 4 is correct and can be applied exhaustively in linear time. Furthermore, if Reduction Rule 4 is not applicable, then the graph contains at most $k + 4 \cdot 2^k$ vertices and $O(k \cdot 2^k)$ edges, where k is the vertex cover number.*

Proof Let $G = (V, E)$ be the input graph with a vertex cover $W \subseteq V$ of size k and let $v_1, \ldots, v_\ell \in V$, $\ell > 4$, be vertices with the same open neighborhood.

First, we show that Reduction Rule 4 is correct, that is, $\delta(G[V \setminus \{v_5, \ldots, v_\ell\}]) = \delta(G)$. To see this, consider two vertices v_i, v_j with the same open neighborhood, and consider any other vertex u. The crucial observation

is that $\overline{uv_i} = \overline{uv_j}$. This means that the two vertices are interchangeable with respect to the hyperbolicity. In particular, if $v_i, v_j \in V$ have the same open neighborhood, then $\delta(v_i, x, y, z) = \delta(v_j, x, y, z)$ for every $x, y, z \in V \setminus \{v_i, v_j\}$. As the hyperbolicity is obtained from a quadruple, it is sufficient to consider at most four vertices with the same open neighborhood. We conclude that $\delta(G[V \setminus \{v_5, \ldots, v_\ell\}]) = \delta(G)$.

Next we show how to exhaustively apply Reduction Rule 4 in linear time. To this end, we apply in linear time a *partition refinement* [22] to compute a partition of the vertices into twin classes. Then, for each twin class we remove all but four (arbitrary) vertices. Overall, this can be done in linear time.

Since $|W| \leq k$, it follows that there are at most 2^k pairwise-different neighborhoods (and thus twin classes) in $V \setminus W$. Thus, if Reduction Rule 4 is not applicable, then the graph consists of the vertex cover W of size k plus at most $4 \cdot 2^k$ vertices in $V \setminus W$. Furthermore, since W is a vertex cover, it follows that the graph contains at most $k^2 + 4k \cdot 2^k$ edges.   □

With Reduction Rule 1 we can compute in linear time an equivalent instance having a bounded number of vertices. Applying to this instance the trivial $O(n^4)$-time algorithm yields the following.

Theorem 4 HYPERBOLICITY *can be computed in* $O(2^{4k} + n + m)$ *time, where* k *denotes the size of a vertex cover of the input graph.*   □

SETH-based Lower Bounds. We show that, unless the SETH breaks, the $2^{O(k)} + O(n + m)$-time algorithm of Theorem 4 cannot be improved to an algorithm even with running time $2^{o(k)} \cdot n^{2-\epsilon}$ for any $\epsilon > 0$. This also implies that, assuming SETH, there is no problem kernel with $2^{o(k)}$ vertices computable in $O(n^{2-\epsilon})$ time, i.e., the kernel obtained by applying Reduction Rule 4 cannot be improved significantly. The proof follows by a linear-time many-one reduction from the following problem:

ORTHOGONAL VECTORS

Input: Two sets $\mathbf{A}$ and $\mathbf{B}$ each containing n binary vectors of length $\ell = O(\log n)$.

Question: Are there two vectors $\mathbf{a} \in \mathbf{A}$ and $\mathbf{b} \in \mathbf{B}$ such that $\mathbf{a}$ and $\mathbf{b}$ are orthogonal, that is, such that there is no position i for which $\mathbf{a}[i] = \mathbf{b}[i] = 1$?

Williams and Yu [30] proved that, if ORTHOGONAL VECTORS can be solved in $O(n^{2-\epsilon})$ time, then SETH breaks. We provide a linear-time reduction from ORTHOGONAL VECTORS to HYPERBOLICITY where the graph G constructed in the reduction contains $O(n)$ vertices and admits a vertex cover of size $O(\log n)$ (and thus contains $O(n \cdot \log n)$ edges). The reduction then implies that, unless SETH breaks, there is no algorithm solving HYPERBOLICITY in time polynomial in the size of the vertex cover and linear in the size of the graph. We mention that Borassi et al. [5] showed that, assuming the SETH, HYPERBOLICITY cannot be solved in $O(n^{2-\epsilon})$ time. The instances constructed
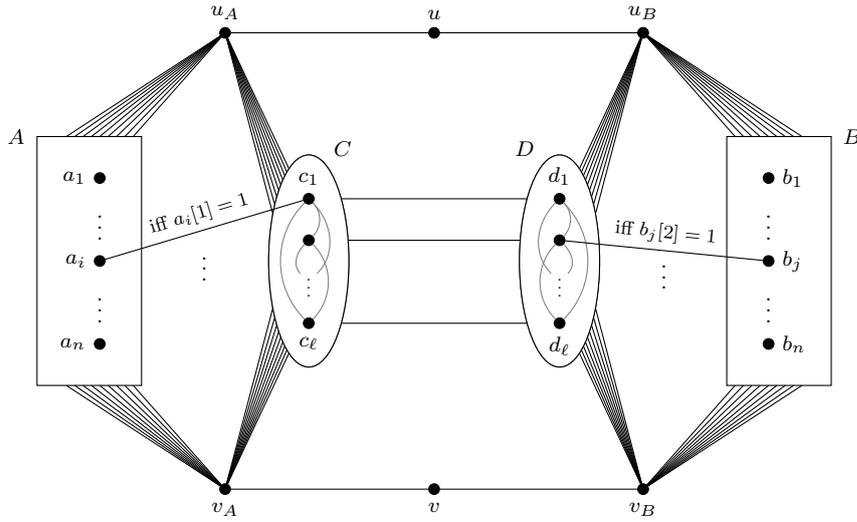
Fig. 3 Sketch of the construction described in the proof of Theorem 5. Ellipses indicate cliques, rectangles indicate independent sets. Multiple edges to an object indicate that the corresponding vertex is incident to each vertex enclosed within that object.

in their reduction, however, have a minimum vertex cover of size $\Omega(n)$. Note that our reduction is based on ideas from the reduction of Abboud et al. [1, Theorem 1.7] for the DIAMETER problem.

Theorem 5 *Assuming SETH,* HYPERBOLICITY *cannot be solved in* $2^{o(k)} \cdot n^{2-\epsilon}$ *time, even on graphs with* $O(n \log n)$ *edges, diameter four, and domination number three. Here, k denotes the vertex cover number of the input graph.*

Proof We reduce any instance $(\mathbf{A}, \mathbf{B})$ of ORTHOGONAL VECTORS to an instance (G, δ) of HYPERBOLICITY, where we construct the graph G as follows (we refer to Figure 3 for a sketch of the construction).

Make each $\mathbf{a} \in \mathbf{A}$ into a vertex a and each $\mathbf{b} \in \mathbf{B}$ into a vertex b of G, and denote these vertex sets by A and B, respectively. Add two vertices for each of the ℓ dimensions, that is, add the vertex set $C := \{c_1, \ldots, c_\ell\}$ and the vertex set $D := \{d_1, \ldots, d_\ell\}$ to G and make each of C and D a clique. Next, connect each $a \in A$ to the vertices of C in the natural way, that is, add an edge between a and c_i if and only if $\mathbf{a}[i] = 1$. Similarly, add an edge between $b \in B$ and $d_i \in D$ if and only if $\mathbf{b}[i] = 1$. Moreover, add the edge set $\{\{c_i, d_i\} \mid i \in [\ell]\}$. This part will constitute the central gadget of our construction.

Our aim is to ensure that the maximum hyperbolicity is reached for 4-tuples (a, b, c, d) such that $a \in A$, $b \in B$, and $\mathbf{a}$ and $\mathbf{b}$ are orthogonal vectors. The construction of G is completed by adding two paths (u_A, u, u_B) and (v_A, v, v_B), and making u_A and v_A adjacent to all vertices in $A \cup C$, and u_B and v_B adjacent to all vertices in $B \cup D$.

Observe that G contains $O(n)$ vertices, $O(n \cdot \log n)$ edges, and that the set $V \setminus (A \cup B)$ forms a vertex cover in G of size $O(\log n)$. Moreover, observe that G has diameter four. Note that each vertex in $A \cup B \cup C \cup D$ is at distance two to each of u and v. Moreover, v_A and v_B are at distance three to u. Analogously, u_A, u_B are at distance three to v. Furthermore u and v are at distance four. Finally, observe that $\{u_A, u_B, v\}$ forms a dominating set in G, that is, every vertex of G is either contained in or adjacent to a vertex in $\{u_A, u_B, v\}$.

We complete the proof by showing that $(\mathbf{A}, \mathbf{B})$ is a yes-instance of OR-THOGONAL VECTORS if and only if G has hyperbolicity at least $\delta = 4$.

($\Rightarrow$)   Let $(\mathbf{A}, \mathbf{B})$ be a yes-instance, and let $\mathbf{a} \in \mathbf{A}$ and $\mathbf{b} \in \mathbf{B}$ be a pair of orthogonal vectors. We claim that $\delta(a, b, u, v) = 4$. Since $\mathbf{a}$ and $\mathbf{b}$ are orthogonal, there is no $i \in [\ell]$ with $\mathbf{a}[i] = \mathbf{b}[i] = 1$ and, hence, there is no path connecting a and b only containing two vertices in $C \cup D$, and it holds that $\overline{ab} = 4$. Moreover, we know that $\overline{uv} = 4$ and that $\overline{au} = \overline{bu} = \overline{av} = \overline{av} = 2$. Thus, $\delta(a, b, u, v) = 8 - 4 = 4$, and G is 4-hyperbolic.

($\Leftarrow$)   Let $S = \{a, b, c, d\}$ be a set of vertices such that $\delta(a, b, c, d) \geq 4$. By Lemma 1, it follows that no two vertices of S are adjacent. Hence, we assume without loss of generality that $\overline{ab} = \overline{cd} = 4$. Observe that all vertices of C and D have distance at most three to all other vertices. Similarly, each vertex of $\{u_A, v_A, u_B, v_B\}$ has distance at most three to all other vertices. Consider for example u_A. By construction, u_A is a neighbor of all vertices in $A \cup C \cup \{u\}$ and, hence, u_A has distance at most two to v_A and to all vertices in D. Thus, u_A has distance at most three to v, B, u_B and v_B and therefore to all vertices of G. The arguments for v_A, u_B, and v_B are symmetric.

It follows that $S \subseteq A \cup B \cup \{u, v\}$, and therefore at least two vertices in S are from $A \cup B$. Thus, assume without loss of generality that a is contained in A. By the previous assumption, we have that $\overline{ab} = 4$. This implies that $b \in B$ and $\mathbf{a}$ and $\mathbf{b}$ are orthogonal vectors, as every other vertex in $V \setminus B$ is at distance three to a and each $b' \in B$ with $\mathbf{b}'$ being non-orthogonal to $\mathbf{a}$ is at distance three to a. Hence, $(\mathbf{A}, \mathbf{B})$ is a yes-instance. $\square$

We remark that, with the above reduction, the hardness also holds for the variants in which we fix one vertex (u) or two vertices (u and v). The reduction also shows that approximating the hyperbolicity of a graph within a factor of $4/3 - \epsilon$ cannot be done in strongly subquadratic time or in PL-FPT running time with respect to the vertex cover number.

Next, we adapt the above reduction to obtain the following hardness result on graphs of bounded maximum degree.

Theorem 6 *Assuming SETH,* HYPERBOLICITY *cannot be solved in* $f(\Delta) \cdot n^{2-\epsilon}$ *time, where* Δ *denotes the maximum degree of the input graph.*

Proof We reduce any instance $(\mathbf{A}, \mathbf{B})$ of ORTHOGONAL VECTORS to an instance (G, δ) of HYPERBOLICITY as follows.

We use the following notation. For two sets of vertices X and Y with $|X| = |Y|$, we say that we introduce *matching paths* if we connect the vertices in X

with the vertices in Y with paths with no inner vertices from $X \cup Y$ such that for each $x \in X$, x is connected to exactly one $y \in Y$ via one path and for each $y \in Y$, y is connected to exactly one $x \in X$ via one path.

Let G' be the graph obtained from the graph constructed in the proof of Theorem 5 after deleting all edges. For each x_A, $x \in \{u, v\}$, add two binary trees, $T_{x_A}^A$ with n leaves and height at most $\lceil \log n \rceil$, and $T_{x_A}^C$ with ℓ leaves and height at most $\lceil \log \ell \rceil$. Connect each tree root by an edge with x_A. Next introduce matching paths between A and the leaves of $T_{x_A}^A$ such that each shortest path connecting a vertex in A with x_A is of length $h := 2(\lceil \log(n) \rceil + 1) + 1$. Similarly, introduce matching paths between C and the leaves of $T_{x_A}^C$ such that each shortest path connecting a vertex in C with x_A is of length h. Apply the same construction for x_B, $x \in \{u, v\}$, B, and D.

For $x \in A \cup B$, we denote by $|x|_1$ the number of 1's in the corresponding binary vector $\mathbf{x}$. Moreover, for $c_i \in C$, we denote by $|c_i|$ the number of vectors in A with a 1 as its ith entry. For $d_i \in D$, we denote by $|d_i|$ the number of vectors in B with a 1 as their ith entry.

For each vertex $a \in A$, add a binary tree with $|a|_1$ leaves and height at most $\lceil \log |a|_1 \rceil$ and connect its root by an edge with a. For each $i \in [\ell]$, add a binary tree with $|c_i|$ leaves and height at most $\lceil \log |c_i| \rceil$ and connect its root by an edge with c_i. Next, construct matching paths between the leaves of all binary trees introduced for the vertices in A on the one hand, and the leaves of all binary trees introduced for the vertices in C on the other hand, such that the following holds: (i) for each $a \in A$ and $c_i \in C$, there is a path only containing the vertices of the corresponding binary trees if and only if $\mathbf{a}[i] = 1$, and (ii) each of these paths is of length exactly h. Apply the same construction for B and D.

Next, for each $i \in [\ell]$, add a binary tree with $\ell - 1$ leaves and height at most $\lceil \log(\ell - 1) \rceil$ and connect its root by an edge with c_i. Finally, add paths between the leaves of all binary trees introduced in this step such that (i) each leaf is incident to exactly one path, (ii) for each $i, j \in [\ell]$, $i \neq j$, there is a path only containing the vertices of the corresponding binary trees, and (iii) each of these paths is of length exactly h. Apply the same construction for D.

Finally, for each $i \in [\ell]$, connect c_i with d_i via a path of length h. Moreover, for $x \in \{u, v\}$, connect x_A with x and x with x_B each via a path of length h. This completes the construction of G. Observe that the number of vertices in G is at most the number of vertices in the graph obtained from G' by replacing each edge with paths of length h. As G' contains $O(n \log n)$ edges, the number of vertices in G is in $O(n \log^2 n)$. Finally, observe that the vertices in $C \cup D$ are the vertices of maximum degree which is five.

Next, we discuss the distances of several vertices in the constructed graph. Observe that u and v are at distance $4h$. For $x \in \{u, v\}$, the distance between x and x_A or x_B is h, and the distance between x_A and x_B is $2h$. The distance from any $c \in C$ to any $d \in D$ is at least h and at most $2h$. Moreover, the distance between any $a \in A$ and $b \in B$ is at least $3h$ and at most $4h$.

Claim 1 For any $a \in A$ and $b \in B$, $\overline{ab} = 4h$ if and only if $\mathbf{a}$ and $\mathbf{b}$ are orthogonal.

Proof (of Claim 1) ($\Leftarrow$)   Let $\mathbf{a}$ and $\mathbf{b}$ be orthogonal. Suppose that there is a shortest path P between a and b of length smaller than $4h$. Observe that any shortest path between a and b containing u or v is of length $4h$. Hence, P contains vertices in $C \cup D$. As the shortest paths from a to C, C to D, and D to b are each of length h, the only shortest path containing vertices in $C \cup D$ of length smaller than $4h$ is of the form (a, c_i, d_i, b) for some $c_i \in C$ and $d_i \in D$ (recall that the shortest path between any two vertices in C or in D is of length h). Hence, $\mathbf{a}$ and $\mathbf{b}$ have both a 1 as their ith entry, and thus are not orthogonal. This contradicts the fact that $\mathbf{a}$ and $\mathbf{b}$ form a solution. It follows that $\overline{ab} = 4h$.

($\Rightarrow$)   Let $\mathbf{a}$ and $\mathbf{b}$ be not orthogonal. Then there is an $i \in [\ell]$ such that $a[i] = b[i] = 1$. Hence, there is a path (a, c_i, d_i, b) of length $3h < 4h$. $\diamond$

Let $M := A \cup B \cup C \cup D \cup \{x, x_A, x_B \mid x \in \{u, v\}\}$. So far, we know that the only vertices that can be at distance $4h$ are those in $A \cup B \cup \{u, v\}$.

Consider any vertex $p \in V(G) \setminus M$. Then p is contained in a shortest path between two vertices x and y in M at distance h. Moreover, $\max\{\overline{px}, \overline{py}\} =: h' < h$. Let P_x^Y denote the set of inner vertices of the shortest path connecting x and x_Y, for $x \in \{u, v\}$, $Y \in \{A, B\}$. Moreover, let $M^* := \{p \in P_x^Y \mid x \in \{u, v\}, Y \in \{A, B\}\}$. We first discuss the case where $p \in M^*$. By symmetry, let $p \in P_u^A$. Observe that for $q \in P_v^B$ with $\overline{vq} = \overline{up}$ holds $\overline{pq} = 4h$.

Let $p \notin M \cup M^*$. We claim that for all vertices $q \in V(G)$ it holds that $\overline{pq} < 4h$. Suppose not, so that there is some $q \in V(G)$ with $\overline{pq} \geq 4h$. Observe that q is not contained in a shortest path between x and y. It follows that $\overline{xq} \geq 4h - h' > 3h$ or $\overline{yq} \geq 4h - h' > 3h$. Let $z \in \{x, y\}$ denote the vertex of minimal distance among the two, and let $\bar{z}$ denote the other one. Note that since h is odd, the distances to z and $\bar{z}$ are different.

Case 1: $q \in M$. Then $z, q \in A \cup B$, where z and q are not both contained in A or B. Recall that $p \notin M \cup M^*$ and, hence, the case $z, q \in \{u, v\}$ is not possible. By symmetry, assume $z \in A$ and $q \in B$. As $\overline{zq} > 3h$, it follows that $\bar{z} = c_i \in C$ for some $i \in [\ell]$ with $1 = \mathbf{z}[i] \neq \mathbf{q}[i]$, or $\bar{z} \in \{u_A, v_A\}$. Hence, the distance of $\bar{z}$ to q is at most the distance of z to q, contradicting the choice of z.

Case 2: $q \notin M$. Then q is contained in a shortest path between two vertices $x', y' \in M$ of length h. Moreover, $\max\{\overline{qx'}, \overline{qy'}\} =: h'' < h$. Consider a shortest path between p and q and notice that it must contain z and $z' \in \{x', y'\}$. It holds that $\overline{zz'} \geq 4h - h' - h'' > 2h$. By symmetry, assume $z \in A$, and $z' \in D \cup \{u_B, v_B\}$ (recall that $p \notin M \cup M^*$). Then $\bar{z}$ is in $C \cup \{u_A, v_A\}$, and hence of shorter distance to q, contradicting the choice of z.

We proved that $\overline{pq} < 4h$ for all $p \in V(G) \setminus (M \cup M^*)$, $q \in V(G)$. We conclude that the vertex set $A \cup B \cup \{u, v\} \cup M^*$ is the only set containing vertices at distance $4h$. Moreover, G is of diameter $4h$.

We claim that $(\mathbf{A}, \mathbf{B})$ is a yes-instance of ORTHOGONAL VECTORS if and only if G has hyperbolicity at least $\delta = 4h$.

($\Rightarrow$)   Let $\mathbf{a} \in \mathbf{A}$ and $\mathbf{b} \in \mathbf{B}$ be orthogonal. We claim that $\delta(a, b, u, v) = 4h$. Observe that $\overline{uv} = 4h$, and that $\overline{ab} = 4h$ by Claim 1. The remaining distances are $2h$ by construction, and hence $\delta(G) = \delta(a, b, u, v) = 4h$.

($\Leftarrow$)   Let $\delta(G) = 4h$ and let w, x, y, z be a quadruple with $\delta(w, x, y, z) = 4h$. By Lemma 2, we know that there are exactly two pairs of distance $4h$ and, hence, $\{w, x, y, z\} \subseteq A \cup B \cup \{u, v\} \cup M^*$. We claim that $|\{w, x, y, z\} \cap (M^* \cup \{u, v\})| \leq 2$. By Lemma 2, we know that, out of w, x, y, z, there are exactly two pairs at distance $4h$ and all other pairs have distance $2h$. Assume that $|\{w, x, y, z\} \cap M^* \cup \{u, v\}| \geq 3$. Then, at least two vertices are in $P_v^A \cup P_v^B \cup \{v\}$ or in $P_u^A \cup P_u^B \cup \{u\}$. Observe that any two vertices in $P_v^A \cup P_v^B \cup \{v\}$ or in $P_u^A \cup P_u^B \cup \{u\}$ are at distance smaller than $2h$, but this contradicts the choice of the quadruple. It follows that $|\{w, x, y, z\} \cap M^* \cup \{u, v\}| \leq 2$. We may thus assume without loss of generality that $w, x \in A \cup B$. As each vertex in A is at distance smaller than $3h$ to any vertex in $A \cup \{u, v\} \cup M^*$, it follows that the other vertex is in B. Applying Claim 1, we have that w and x are at distance $4h$ if and only if $\mathbf{w}$ and $\mathbf{x}$ are orthogonal; hence, the statement of the theorem follows. $\square$

4.2 Parameter Distance to Cographs

In this section we describe a *linear-time parameterized* algorithm for HYPER-BOLICITY parameterized by the vertex deletion distance k to cographs; that is, we present an algorithm with linear dependence on the input size but arbitrary dependence on the parameter (to which we refer to as L-FPT). A graph is a cograph if and only if it is P_4-free. Given a graph G we can determine in linear time whether it is a cograph and return an induced P_4 if this is not the case. This implies that in $O(k \cdot (m + n))$ time we can compute a set $X \subseteq V$ of size at most $4k$ such that $G - X$ is a cograph.

A further characterization is that a cograph can be obtained from graphs consisting of one single vertex via unions and joins [6].
– A *union* of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $V_1 \cap V_2 = \emptyset$ is the graph $(V_1 \cup V_2, E_1 \cup E_2)$.
– A *join* of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the graph $(V_1 \cup V_2, E_1 \cup E_2 \cup \{\{v_1, v_2\} \mid v_1 \in V_1, v_2 \in V_2\})$.
The union of t graphs and the join of t graphs are defined by taking successive unions or joins, respectively, of the t graphs in an arbitrary order. Each cograph G can be associated with a rooted cotree T_G. The leaves of T_G are the vertices of V. Each internal node of T_G is labeled either as a union or join node. For node v in T_G, let $L(v)$ denote the leaves of the subtree rooted at v. For a union node v with children $u_1, \ldots, u_t$, the graph $G[L(v)]$ is the union of the graphs $G[L(u_i)]$, $1 \leq i \leq t$. For a join node v with children $u_1, \ldots, u_t$, the graph $G[L(v)]$ is the join of the graphs $G[L(u_i)]$, $1 \leq i \leq t$.

The cotree of a cograph can be computed in linear time [11]. In a subroutine in our algorithm for HYPERBOLICITY we need to solve the following variant of SUBGRAPH ISOMORPHISM.

COLORED INDUCED SUBGRAPH ISOMORPHISM

Input: An undirected graph $G = (V, E)$ with a vertex-coloring $\gamma : V \to \mathbb{N}$ and an undirected graph $H = (W, F)$, where $|W| = k$, with a vertex-coloring $\chi : W \to \mathbb{N}$.

Question: Is there a vertex set $S \subseteq V$ such that there is an isomorphism f from $G[S]$ to H such that $\gamma(v) = \chi(f(v))$ for all $v \in S$?

Informally, the condition that $\gamma(v) = \chi(f(v))$ means that every vertex is mapped to a vertex of the same color. We say that such an isomorphism *respects the colorings*. As shown by Damaschke [14], INDUCED SUBGRAPH ISOMORPHISM on cographs is NP-complete. Since this is the special case of COLORED INDUCED SUBGRAPH ISOMORPHISM where all vertices in G and H have the same color, COLORED INDUCED SUBGRAPH ISOMORPHISM is also NP-complete (containment in NP is obvious). In the following, we show that on cographs COLORED INDUCED SUBGRAPH ISOMORPHISM can be solved by an L-FPT algorithm when the parameter k is the order of H.

Lemma 9 COLORED INDUCED SUBGRAPH ISOMORPHISM *can be solved in $O(3^k(n + m))$ time in cographs where $k = |V(H)|$.*

Proof We use dynamic programming on the cotree $T_G =: T$. Herein, we assume that for each internal node v there is an arbitrary (but fixed) ordering of its children; the ith child of v is denoted by $c_i(v)$ and the set of leaves in the subtrees rooted at the first i children of v is denoted by $L_i(v)$. We fill a three-dimensional table D with entries of the type $D[v, i, X]$ where v is a node of the cotree with at least i children and $X \subseteq W$ is a subset of the vertices of the pattern H. The entry $D[v, i, X]$ has value 1 if $(G[L_i(v)], \gamma|_{L_i(v)}, H[X], \chi|_X)$ is a yes-instance of COLORED INDUCED SUBGRAPH ISOMORPHISM, that is, there is a subgraph isomorphism from $G[L_i(v)]$ to $H[X]$ that respects the coloring. Otherwise, the entry has value 0. Thus, $D[v, \deg_T(v) - 1, X]$ has value 1 if and only if there is an induced subgraph isomorphism from $G[L_i(v)]$ to $H[X]$. After the table is completely filled, the instance is a yes-instance if and only if $D[r, \deg_T(r), W]$ has value 1 where r is the root of the cotree. We initialize the table for leaf vertices v by setting

$$D[v, 0, X] = \begin{cases} 1, & \text{if either } X = \emptyset \text{ or } X = \{u\} \text{ with } \gamma(v) = \chi(u), \\ 0, & \text{otherwise.} \end{cases}$$

For each inner node v and $i = 1$, we have the following recurrence:

$$D[v, 1, X] = \begin{cases} 1, & \text{if } D[c_1(v), \deg(c_1(v)) - 1, X] = 1, \\ 0, & \text{otherwise.} \end{cases}$$

For $i \geq 2$, the following recurrences hold for union and join nodes. For union nodes, the table D is filled by the following recurrence:

$$D[v, i, X] = \begin{cases} 1, & \exists X' \subseteq X : D[v, i-1, X'] = 1 \\ & \wedge \ D[c_i(v), \deg(c_i(v)) - 1, X \setminus X'] = 1 \\ & \wedge \ \text{there are no edges between } X' \text{ and } X \setminus X' \text{ in } H, \\ 0, & \text{otherwise.} \end{cases}$$

For join nodes, the table D is filled by the following recurrence:

$$D[v, i, X] = \begin{cases} 1, & \exists X' \subseteq X : D[v, i-1, X'] = 1 \\ & \wedge \ D[c_i(v), \deg(c_i(v)) - 1, X \setminus X'] = 1 \\ & \wedge \ \text{every } q \in X' \text{ is adjacent to every } p \in X \setminus X' \text{ in } H, \\ 0, & \text{otherwise.} \end{cases}$$

The correctness of the recurrence can be seen as follows for the union nodes. First assume that there is a color-respecting induced subgraph isomorphism from $G[L_i(v)]$ to $H[X]$. Then there is a set $S \subseteq L_i(v)$ such that there is a color-respecting isomorphism f from $G[S]$ to $H[X]$. If $i = 1$, then $D[c_1(v), \deg(c_1(v)) - 1, X] = 1$, and hence, $D[v, 1, X]$ evaluates correctly to 1. Otherwise, if $i \geq 2$, let $S' := S \cap L_{i-1}(v)$ be the set of vertices that are from S and from $L_{i-1}(v)$ which implies that $S \setminus S' = S \cap L(c_i(v))$. Since v is a union node, there are no edges between S' and $S \setminus S'$ in G. Let $X' := f(S')$ and $X \setminus X' = f(S \setminus S')$ denote the image of S' and $S \setminus S'$, respectively. Since f is an isomorphism there are no edges between X' and $X \setminus X'$. Moreover, since restricting a color-respecting isomorphism $f : G[S] \to H[X]$ to a subset S' gives a color-respecting isomorphism from $f : G[S'] \to H[f(S')]$ we have that $D[v, i-1, X']$ and $D[c_i(v), \deg(c_i(v)) - 1, X \setminus X']$ have value 1. Hence, there is a case such that the recurrence evaluates correctly to 1.

Conversely, if the recurrence evaluates to 1, then the conditions in the recurrence (about the existence of X') imply a color-respecting induced subgraph isomorphism from $G[L_i(V)]$ to $H[X]$. Thus, the table is filled correctly for union nodes. The correctness of the recurrence for join nodes follows by symmetric arguments.

The running time is upper-bounded as follows. The cotree has size $O(n+m)$ and thus, there are $O((n+m) \cdot 2^k)$ entries in the table. For each $X \subseteq W$, filling the entries of a particular table entry is done by considering all subsets of X, thus the overall number of evaluations is $O(3^k \cdot (n+m))$. $\square$

We now turn to the algorithm for HYPERBOLICITY on graphs that can be transformed into cographs by at most k vertex deletions. The final step of this algorithm is to reduce HYPERBOLICITY to the following problem:

DISTANCE-CONSTRAINED 4-TUPLE

Input: An undirected graph $G = (V, E)$ and six integers $d_{\{a,b\}}$, $d_{\{a,c\}}$, $d_{\{a,d\}}$, $d_{\{b,c\}}$, $d_{\{b,d\}}$, and $d_{\{c,d\}}$,

Question: Is there a set $S \subseteq V$ of four vertices and a bijection $f : S \to \{a, b, c, d\}$ such that for each $x, y \in S$ we have $\overline{xy} = d_{\{f(x), f(y)\}}$?

We prove that DISTANCE-CONSTRAINED 4-TUPLE is L-FPT when parameterized by the vertex deletion distance k to cographs.

Lemma 10 DISTANCE-CONSTRAINED 4-TUPLE *can be solved in* $O(4^{4k} \cdot k \cdot (n+m))$ *time if* $G - X$ *is a cograph for some* $X \subseteq V(G)$ *with* $|X| \le k$.

Proof Let $G = (V, E)$ be the input graph and $X \subseteq V$, $|X| \le k$, such that $G - X$ is a cograph. Without loss of generality, let $X = \{x_1, \dots, x_k\}$.

In a preprocessing step, we classify the vertices of each connected component in $G[V \setminus X]$ according to the length of shortest paths to vertices in X such that all internal vertices of the shortest path are in $V \setminus X$.

More precisely, for a vertex $v \in V \setminus X$ in a connected component C_v of $G - X$, the *type* t_v of v is a length-k vector containing the distance of v to each vertex x_i of X in $G[C_v \cup \{x_i\}]$. That is, $t_v[i]$ equals the distance from v to $x_i \in X$ within the graph $G[C_v \cup \{x_i\}]$. Since the diameter of $G[C_v]$ is at most two, $t_v[i] \in \{1, 2, 3, \infty\}$. Thus, the number of distinct types in G is at most 4^k. For simplicity of notation, for every type t we denote by v_t an arbitrary vertex such that $t_v = t$.

Observation 1 Let u and v be two vertices of the same type, that is, $t_u = t_v$. Then for each vertex w in $G - (C_u \cup C_v)$, we have $\overline{uw} = \overline{vw}$.

Observation 2 Given two vertices u and v such that $C_u \ne C_v$, we can compute $\overline{uv}$ in $O(k)$ time when the distance between each $u \in X$ and each $v \in V \setminus X$ can be retrieved in $O(1)$ time.

The dominating part of the running time of the preprocessing is the computation of the vertex types which can be performed in $O(k \cdot (n + m))$ time as follows. Create a length-k vector for each of the at most n vertices of $V \setminus X$ and initially set all entries to ∞. Then compute for each $x_i \in X$, the graph $G - (X \setminus \{x_i\})$ in $O(n + m)$ time. In this graph perform a breadth-first search from x_i to compute the distances between x_i and each vertex $v \in V \setminus X$ that is in the same connected component as x_i. This distance is exactly the one in the graph $G[C_v \cup \{x_i\}]$. Thus, for all vertices that reach x_i, the ith entry in their type vector is updated. Afterwards, each vertex has the correct type vector.

After this preprocessing, the algorithm proceeds as follows by restricting the choice of vertices for the 4-tuple.

(1) First, branch into all $O(k^4)$ cases of taking a subset $X' \subseteq X$ of size at most four. (We will assume that $X' = S \cap X$.)
(2) For each such $X' \subseteq X$, branch into the different cases for the types of vertices in $S \setminus X'$. That is, consider all multisets M_T of size $|S \setminus X'| = 4 - |X'|$ over the universe of all types. (There are 4^k types and thus at most 4^{4k} cases for each $X' \subseteq X$.)
(3) For each such $X' \subseteq X$ and multiset M_T, branch into all cases of matching the vertices in $\{a, b, c, d\}$ to the vertices in X' and types in M_T (branch into all "bijections" f between $X' \cup M_T$ and $\{a, b, c, d\}$). (There are at most 4! cases.)

(4) For each such branch, branch into the different possibilities to assign the types in M_T to connected components of $G-X$. That is, create one branch for each partition of the multiset M_T and assume in this branch that two types are in the same connected component if and only if they are in the same set of the partition of M_T. The current partition is called the *component partition* of the branch.

We now check whether there is a solution to the DISTANCE-CONSTRAINED 4-TUPLE instance that fulfills the additional assumptions made in the above branches. To this end, for each pair of vertices $x, y \in X'$, check whether $\overline{xy} = d_{\{f(x),f(y)\}}$. Now, for each vertex $x \in X'$ and each type $t \in M_T$, check whether $\overline{xv_t} = d_{\{f(x),f(v_t)\}}$, where v_t is an arbitrary vertex of type t. Observe that this is possible since, by Observation 1 the distance between X and any vertex of type t is the same in G. Next, for pair of types $t, t' \in M_T$ such that the branch assumes that t and t' do not lie in the same connected component of $G - X$, check whether $\overline{v_t v_{t'}} = d_{\{f(v_t),f(v_{t'})\}}$. Again, this is possible due to Observation 1.

The remaining problem is thus to determine whether the types of M_T can be assigned to vertices in such a way that

– for each pair of types $t, t' \in M_T$ the assigned vertices are in the same connected component of $G - X$ if and only if it is constrained to be in the same type of connected component in the current branch, and
– for each pair of types $t, t' \in M_T$ such that their assigned vertices v_t and $v_{t'}$ are constrained to be in the same connected component, we need to ensure that $\overline{v_t v_{t'}} = d_{\{f(v_t),f(v_{t'})\}}$.

We solve this problem by a reduction to COLORED INDUCED SUBGRAPH ISO-MORPHISM. Observe that, since $G - X$ is a cograph, for each pair u and v of vertices in the same connected component of $G - X$, the distance between u and v is 2 if and only if they are not adjacent. With this observation, the reduction works as follows. Let $j \leq 4$ denote the number of distinct connected components of $G - X$ that shall contain at least one type of M_T. Now, for each connected component C of $G - X$ add one further vertex v_C, make v_C adjacent to all vertices of C and call the resulting graph G'. Now color the vertices of $G - X$ as follows. The additional vertices of each connected component receive the color 0. Next, for each vertex type t in $G - X$ introduce one color and assign this color to each vertex of type t. Call the vertices with color 0 the *component-vertices* and all other vertices the *type-vertices*. To complete the construction of the input instance, we build H as follows. Add j vertices of color 0. Then add a vertex for each type t of M_T and color it with the color corresponding to its type. As in G', call the vertices with color 0 *component-vertices* and all other vertices *type-vertices*. Add edges between the component-vertices and the type-vertices in such a way that every type-vertex is adjacent to one vertex of color 0 and two type-vertices are adjacent to the same color-0 vertex if and only if they are constrained to be in the same connected component. Finally, if two type-vertices are constrained to be in the same connected component and have distance 1 in G, then add an edge between

them, otherwise add no edge between them. This completes the construction of H. The instance of COLORED INDUCED SUBGRAPH ISOMORPHISM consists of G' and H and of the described coloring. We now claim that this instance is a yes-instance if and only if there is a solution to the DISTANCE-CONSTRAINED 4-TUPLE instance that fulfills the constraints of the branch.

If the instance has a solution, then the subgraph isomorphism ϕ from $G'[S]$ to H corresponds to a selection of types from j connected components since H contains neighbors of j component-vertices. Moreover, in H, and thus in $G'[S]$, every type-vertex is adjacent to exactly one component-vertex and thus the component-vertices define a partition of the type-vertices of $G'[S]$ that is, due to the construction of H, exactly the component partition of M_T. Selecting the type-vertices of S and assigning them to $\{a, b, c, d\}$ as specified by f gives, together with the selected vertices of X, a special 4-tuple Q. Observe that Q fulfills all constraints of the branch except for the conditions on the distances between the type-vertices of the same component. Now for two vertices u and v of S in the same connected component of $G - X$, the distance is 1 if they are adjacent and 2 otherwise. Due to the construction of H, and the fact that ϕ is an isomorphism, the distance is thus 1 if $d_{\{f(u),f(v)\}} = 1$ and 2 if $d_{\{f(u),f(v)\}} = 2$. Thus, if the COLORED INDUCED SUBGRAPH ISOMORPHISM instance is a yes-instance, so is the DISTANCE-CONSTRAINED 4-TUPLE instance. The converse direction follows by the same arguments.

The running time can be seen as follows. The preprocessing can be performed in $O(k(n+m))$ time, as described above. Then, the number of branches is $O(4^{4k})$: the only time when the number of created branches is not constant is when the types of the vertices in the 4-tuple are constrained or when the 4-tuple vertices are fixed to belong to X. In the worst case, we have $X' = \{a, b, c, d\} \cap X = \emptyset$, that is, $S \subseteq V \setminus X$ and for all four vertices of S one has to branch in total into 4^{4k} cases to fix the types. In each branch, the algorithm first checks the conditions on all distances except for the distances between vertices of the same parts of the component partition. This can be done in $O(k)$ time for each of these distance. Afterwards, the algorithm builds and solves the COLORED INDUCED SUBGRAPH ISOMORPHISM in $O(n + m)$ time. Altogether, this gives the claimed running time bound.   □

Lemma 10 at hand, we can solve HYPERBOLICITY as follows: For every graph with vertex deletion distance k to cographs it holds true that the distance between every two vertices is upper-bounded by a linear function in k. Hence, we can check all $O(k^6)$ possibilities for distances between two vertices by solving the respective instances of DISTANCE-CONSTRAINED 4-TUPLE each in L-FPT time. Altogether, we get the following.

Theorem 7 HYPERBOLICITY *can be solved in $O(4^{4k} \cdot k^7 \cdot (n+m))$ time, where k is the vertex deletion distance of G to cographs.*

Proof Let $G = (V, E)$ be the input graph and $X \subseteq V$, $|X| \leq k$, such that $G - X$ is a cograph and observe that X can be computed in $O(4^k \cdot (n + m))$ time. Since every connected component of $G - X$ has diameter at most two, the

maximum distance between any pair of vertices in the same component of G is at most $4k + 2$: any shortest path between two vertices u and v visits at most k vertices in X, at most three vertices between every pair of vertices x and x' from X and at most three vertices before encountering the first vertex of X and at most three vertices before encountering the last vertex of X.

Consequently, for the 4-tuple (a, b, c, d) that maximizes $\delta(a, b, c, d)$, there are $O(k^6)$ possibilities for the pairwise distances between the four vertices. Thus, we may compute whether there is a 4-tuple such that $\delta(a, b, c, d) = \delta$ by checking for each of the $O(k^6)$ many 6-tuples of possible pairwise distances of four vertices in G whether there are four vertices in G with these six pairwise distances and whether this implies $\delta(a, b, c, d) \geq \delta$. The latter check can be performed in $O(1)$ time, and the first is equivalent to solving DISTANCE-CONSTRAINED 4-TUPLE which can be done in $O(4^{4k} \cdot k \cdot (n + m))$ time by Lemma 10. The claimed overall running time follows. □

5 Reduction from 4-Independent Set

In this section, we provide a further relative lower bound for HYPERBOLICITY. Specifically, we prove that, if the running time is measured in terms of n, then HYPERBOLICITY is at least as hard as 4-INDEPENDENT SET, that is, the problem of finding an independent set of size four in a graph. The currently best running time for this problem is $O(n^{3.257})$ [16, 31]. Hence, any improvement on the running time of HYPERBOLICITY which breaks this bound (e.g., an algorithm running in $o(n^3)$ time), would also yield a substantial improvement for the 4-INDEPENDENT SET problem.

To this end, we reduce from a 4-partite (or 4-colored) variant of the INDEPENDENT SET problem. The standard reduction [17] from INDEPENDENT SET to MULTICOLORED INDEPENDENT SET shows that this 4-colored variant has the same asymptotic running time lower bound as 4-INDEPENDENT SET.

Theorem 8 *Any algorithm solving* HYPERBOLICITY *in* $O(n^c)$ *time for some constant* $c \geq 2$ *yields an* $O(n^c)$-*time algorithm solving* 4-INDEPENDENT SET.

Proof Let $G = (V = V_1 \uplus V_2 \uplus V_3 \uplus V_4, E)$ be an instance of the 4-COLORED-INDEPENDENT SET problem. Assume an arbitrary order on the vertices of V_i, that is, $V_i = \{v_1^i, \ldots, v_{n_i}^i\}$, where $n_i := |V_i|$, for each $1 \leq i \leq 4$. We construct a graph G', initially being the empty graph, as follows (we refer to Figure 4 for an illustration).

- Add the vertex sets X_1, X_2, X_3, and X_4, where $X_i = \{x_1^i, \ldots, x_{n_i}^i\}, 1 \leq i \leq 4$. We say that x_j^i corresponds to the vertex $v_j^i \in V_i$ in V, for each $1 \leq i \leq 4$, $1 \leq j \leq n_i$. Introduce a copy X_i' of each X_i and further copies Y_1, Z_1 of X_1 and Y_2, Z_2 of X_2. Make each X_i and each copy of each X_i a clique. We say that the jth vertex of some copy of X_i *corresponds to the jth vertex* of X_i and hence corresponds to the jth vertex in V_i.
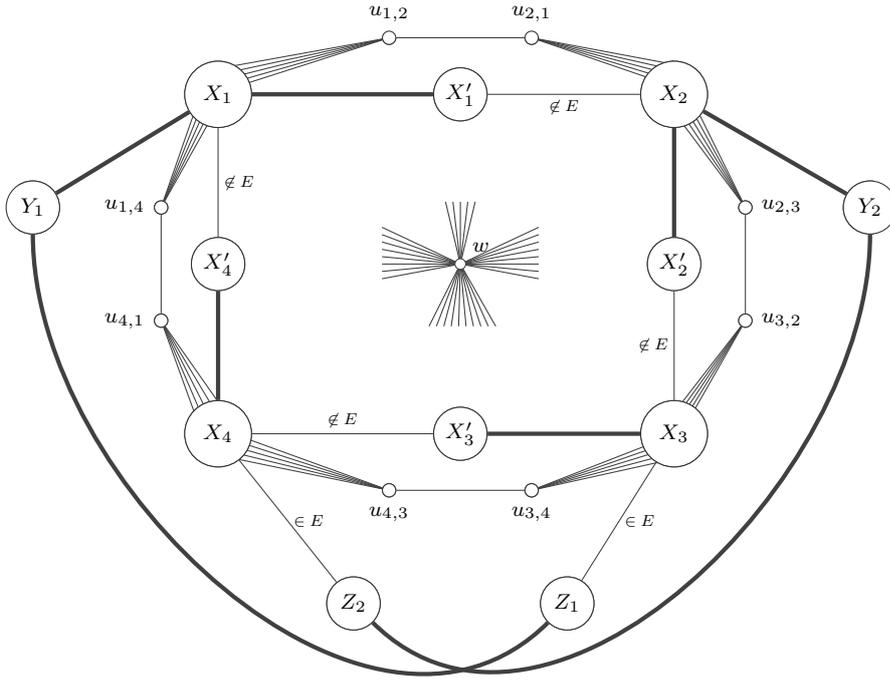- For each vertex in X_i introduce an edge to its corresponding vertex in X_i'.

Fig. 4 An illustrative sketch of the graph constructed in the proof of Theorem 8. Encircled vertices correspond to cliques. A thick edge represents a matching between copy-vertices. An edge labeled "$\in E$" ("$\notin E$") represent incidences corresponding to present (not present) edges in the original graph. The vertex w is incident with all vertices except vertices contained in X_i for all $1 \le i \le 4$.

- For $i \in \{1,2,3\}$, introduce an edge between a vertex in X_i' and a vertex in X_{i+1} if their corresponding vertices in V are *not* adjacent in G. Introduce edges between vertices in X_4' and X_1 analogously.
- For $i \in \{1,2\}$, introduce edges for corresponding vertices between X_i and Y_i, and between Y_i and Z_i.
- For $i \in \{1,2\}$, introduce an edge between a vertex in Z_i and a vertex in X_{i+2} if their corresponding vertices in V *are* adjacent in G.
- Introduce a set $U := \{u_{1,2}, u_{2,1}, u_{2,3}, u_{3,2}, u_{3,4}, u_{4,3}, u_{1,4}, u_{4,1}\}$ of eight further vertices and call the vertices in U the *connection vertices*.
- Introduce the edges $\{u_{i,j}, u_{j,i}\}$ for each $u_{i,j}, u_{j,i} \in U$ and connect each vertex in X_i with $u_{i,j}$ for all j such that $u_{i,j} \in U$ via an edge.
- Finally, add the vertex w and connect w via an edge with all vertices except the vertices in X_i, $1 \le i \le 4$.

This finishes the construction of $G' = (V', E')$. Observe that $|V(G')| = 2 \cdot |V(G)| + 2 \cdot (n_1 + n_2) + 9$. Moreover, observe that the diameter of G' is four. To see this, observe that w has distance at most two to each vertex in G'. Assuming that there exists at least one pair of vertices $a \in V_1$ and $c \in V_3$ such that $\{a, c\} \notin E$ (as otherwise G is a trivial no-instance), the distance

between a and c is exactly 4. We prove that $\delta(G') = 4$ if and only if G has an independent set of size 4.

($\Leftarrow$) Let $\{a, b, c, d\}$ be a colored-independent set of size four in G, and let without loss of generality $a \in V_1$, $b \in V_2$, $c \in V_3$, and $d \in V_4$. Let $a' \in X_1$, $b' \in X_2$, $c' \in X_3$, and $d' \in X_4$ be the corresponding vertices in G'. We show that $\delta(a', b', c', d') = 4$.

First, we show that $\overline{a'b'} = 2$. As no vertex in X_1 is adjacent to any vertex in X_2, we have $\overline{a'b'} \geq 2$. As a and b are not adjacent in G, they have a common neighbor in X_1' by construction of G'. It follows that $\overline{a'b'} = 2$. By a symmetric argument, we conclude that $\overline{b'c'} = \overline{c'd'} = \overline{d'a'} = 2$.

Further, we show that $\overline{a'c'} = 4$. As each vertex in G' is at distance two to vertex w, it follows that $\overline{a'c'} \leq 4$. Moreover, all the neighbors of a' are in $X_1' \cup Y_1 \cup X_4' \cup \{u_{1,2}, u_{1,4}\}$ and all the neighbors of c' are in $Z_1 \cup X_2' \cup X_3' \cup \{u_{3,2}, u_{3,4}\}$. Thus, to have a distance of at most three, a neighbor of a' must be adjacent to a neighbor of c'. By construction, this is only possible if the unique neighbor a'_Y of a' in Y_1 is adjacent to a neighbor of c' in Z_1. The unique neighbor $a'_Z \in Z_1$ of $a'_Y \in Y_1$ is, however, not adjacent to c' since a and c are not adjacent in G. It follows that $\overline{a'c'} = 4$. By a symmetric argument, we conclude that $\overline{b'd'} = 4$. Altogether this implies $\delta(a', b', c', d') = (4 + 4) - (2 + 2) = 4$.

($\Rightarrow$) Let $S := \{a', b', c', d'\} \subseteq V(G')$ be a vertex set such that $\delta(a', b', c', d') = 4$. We show that these vertices correspond to four vertices in G forming a colored independent set in G.

First, by Lemma 1, we have $4 = \delta(a', b', c', d') \leq 2 \cdot \min_{u \neq v \in S}\{\overline{uv}\}$, and hence no two vertices in S are adjacent. Now, assume without loss of generality that $\overline{a'c'} + \overline{b'd'}$ is the largest sum among the distances. Since the diameter of G' is four, by Lemma 2 we have that $\overline{a'c'} = \overline{b'd'} = 4$ and $\overline{a'b'} = \overline{b'c'} = \overline{c'd'} = \overline{d'a'} = 2$.

This already implies that $w \notin S$ as w has distance at most two to all other vertices in G'. Moreover, let $X := X_1 \cup X_2 \cup X_3 \cup X_4$. Since all vertices in $V' \setminus X$ are adjacent to w, each of them is at distance at most three to all other vertices in G'. Thus, $S \subseteq X$, but as S forms an independent set in G', there are no two vertices of X_i, $1 \leq i \leq 4$, in S (recall that each X_i forms a clique in G'). Assume without loss of generality that $a' \in X_1$, $b' \in X_2$, $c' \in X_3$, and $d' \in X_4$. Let $a \in V_1$, $b \in V_2$, $c \in V_3$, and $d \in V_4$ be the vertices in G corresponding to a', b', d', and d', respectively. Finally, by the construction of G' together with $\overline{a'b'} = \overline{b'c'} = \overline{c'd'} = \overline{d'a'} = 2$ and $\overline{a'c'} = \overline{b'd'} = 4$, it follows that $\{a, b, c, d\}$ forms a colored-independent set in G. □

6 Conclusion

To efficiently compute the hyperbolicity number of a given graph, parameterization sometimes may help. In this respect, perhaps our practically most promising results relate to the $O(k(n+m)) + k^4(\log n)^{O(1)}$ running times (for the parameters covering path number and feedback edge number, see Table 1).

Note that they asymptotically improve on the standard algorithm with running time $O(n^4)$ when $k \in O(n^{1-\varepsilon})$ for any $\varepsilon > 0$. Moreover, our linear-time data reduction rules (Reduction Rules 1, 2, 3, and 4) may be of independent practical interest. On the lower bound side, together with the work of Abboud et al. [1] our SETH-based lower bound with respect to the parameter vertex cover number is among few known "exponential lower bounds" for a polynomial-time solvable problem.

Recently, Coudert et al. [13] proved polynomial linear-time parameterized algorithms for computing the hyperbolicity number with respect to the parameters modular-width, split-width, neighborhood diversity, and P_4-sparseness (we omit the definitions of the parameters and refer to their paper). We remark that, on the one hand, modular-width, split-width, and neighborhood diversity can be exponentially large in the vertex cover number. On the other hand, P_4-sparseness is incomparable with the vertex cover number.

As to future work, we particularly point to the following open questions. First, we left open whether there is an L-FPT algorithm exploiting the parameter feedback vertex number for computing the hyperbolicity number. Second, for the parameter vertex cover number we have an SETH-based exponential lower bound for the parameter function in any L-FPT algorithm. This does not, however, imply that it is impossible to achieve a polynomial parameter dependence when asking for algorithms with running time factors such as $O(n^2)$ or $O(n^3)$.

References

1. A. Abboud, V. Vassilevska Williams, and J. R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proc. 27th SODA*, pages 377–391. SIAM, 2016. 17, 30

2. M. Abu-Ata and F. F. Dragan. Metric tree-like structures in real-world networks: an empirical study. *Networks*, 67(1):49–68, 2016. 2

3. M. Bentert, A. Dittmann, L. Kellerhals, A. Nichterlein, and R. Niedermeier. Towards improving Brandes' algorithm for betweenness centrality. *CoRR*, abs/1802.06701, 2018. URL http://arxiv.org/abs/1802.06701. Extended abstract to appear in *Proc. of 29th ISAAC*. 7

4. M. Borassi, D. Coudert, P. Crescenzi, and A. Marino. On computing the hyperbolicity of real-world graphs. In *Proc. 23rd ESA*, volume 9294 of *LNCS*, pages 215–226. Springer, 2015. 2

5. M. Borassi, P. Crescenzi, and M. Habib. Into the square: On the complexity of some quadratic-time solvable problems. *Electronic Notes in Theoretical Computer Science*, 322:51–67, 2016. 2, 3, 16

6. A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: a Survey*, volume 3 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, 1999. 21

7. G. Brinkmann, J. H. Koolen, and V. Moulton. On the hyperbolicity of chordal graphs. *Annals of Combinatorics*, 5(1):61–69, 2001. 5

8. N. Cohen, D. Coudert, and A. Lancin. On computing the Gromov hyperbolicity. *ACM Journal of Experimental Algorithmics*, 20:1.6:1–1.6:18, 2015. 2, 5

9. N. Cohen, D. Coudert, G. Ducoffe, and A. Lancin. Applying clique-decomposition for computing Gromov hyperbolicity. *Theoretical Computer Science*, 690:114–139, 2017. 2, 6

10. D. G. Corneil, H. Lerchs, and L. S. Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163–174, 1981. 3

11. D. G. Corneil, Y. Perl, and L. K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985. 21

12. D. Coudert and G. Ducoffe. Recognition of C_4-free and 1/2-hyperbolic graphs. *SIAM Journal on Discrete Mathematics*, 28(3):1601–1617, 2014. 2

13. D. Coudert, G. Ducoffe, and A. Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. In *Proc. of 29th SODA*, pages 2765–2784. SIAM, 2018. 30

14. P. Damaschke. Induced subgraph isomorphism for cographs is NP-complete. In *Proc. 16th WG*, volume 484 of *LNCS*, pages 72–78. Springer, 1991. 22

15. M. Doucha and J. Kratochvíl. Cluster vertex deletion: A parameterization between vertex cover and clique-width. In *Proc. 37th MFCS*, volume 7464 of *LNCS*, pages 348–359. Springer, 2012. 3

16. F. Eisenbrand and F. Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1-3):57–67, 2004. 27

17. M. Fellows, D. Hermelin, F. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009. 27

18. H. Fournier, A. Ismail, and A. Vigneron. Computing the Gromov hyperbolicity of a discrete metric space. *Information Processing Letters*, 115 (6-8):576–579, 2015. 2

19. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979. 15

20. A. C. Giannopoulou, G. B. Mertzios, and R. Niedermeier. Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs. *Theoretical Computer Science*, 689:67–95, 2017. 3

21. M. Gromov. Hyperbolic groups. In *Essays in Group Theory, MSRI Publ., vol. 8*, pages 75–263. Springer New York, 1987. 2

22. M. Habib and C. Paul. A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010. 16

23. R. Impagliazzo and R. Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. 4

24. R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63 (4):512–530, 2001. 4

25. J. H. Koolen and V. Moulton. Hyperbolic bridged graphs. *European Journal of Combinatorics*, 23(6):683–699, 2002. 8

26. H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983. 10

27. G. B. Mertzios, A. Nichterlein, and R. Niedermeier. The power of linear-time data reduction for maximum matching. In *Proc. 42nd MFCS*, volume 83 of *LIPIcs*, pages 46:1–46:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. 3

28. D. Mitsche and P. Pralat. On the hyperbolicity of random graphs. *The Electronic Journal of Combinatorics*, 21(2):P2.39, 2014. 5

29. C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982. 15

30. R. Williams and H. Yu. Finding orthogonal vectors in discrete structures. In *Proc. 25th SODA*, pages 1867–1877. SIAM, 2014. 16

31. V. V. Williams, J. R. Wang, R. Williams, and H. Yu. Finding four-node subgraphs in triangle time. In *Proc. 26th SODA*, pages 1671–1680. SIAM, 2015. 4, 27