k-Chordal Graphs: from Cops and Robber to Compact Routing via Treewidth

Adrian Kosowski, Bi Li, Nicolas Nisse, Karol Suchan

k-Chordal Graphs: from Cops and Robber to Compact Routing via Treewidth

A. Kosowski · B. Li · N. Nisse · K. Suchan

Abstract *Cops and robber games*, introduced by Winkler and Nowakowski [41] and independently defined by Quilliot [43], concern a team of cops that must capture a robber moving in a graph. We consider the class of k-chordal graphs, i.e., graphs with no induced (chordless) cycle of length greater than k, $k \geq 3$. We prove that $k-1$ cops are always sufficient to capture a robber in k-chordal graphs. This leads us to our main result, a new structural decomposition for a graph class including k-chordal graphs.

We present a polynomial-time algorithm that, given a graph G and $k \geq 3$, either returns an induced cycle larger than k in G, or computes a *tree-decomposition* of G, each *bag* of which contains a dominating path with at most $k-1$ vertices. This allows us to prove that any k-chordal graph with maximum degree Δ has treewidth at most $(k-1)(\Delta-1)+2$, improving the $O(\Delta(\Delta-1)^{k-3})$ bound of Bodlaender and Thilikos (1997). Moreover, any graph admitting such a tree-decomposition has small hyperbolicity.

As an application, for any n-vertex graph admitting such a tree-decomposition, we propose a *compact routing scheme* using routing tables, addresses and headers of size $O(k\log\Delta+\log n)$ bits and achieving an additive stretch of $O(k\log\Delta)$. As far as we know, this is the first routing scheme with $O(k\log\Delta+\log n)$-routing tables and small additive stretch for k-chordal graphs.

Keywords Treewidth · chordality · compact routing · cops and robber games.

1 Introduction

Because of the huge size of real-world networks, an important current research effort concerns exploiting their structural properties for algorithmic purposes. Indeed, in large-scale networks, even algorithms with polynomial-time in the size of the instance may become unpractical. Therefore, it is important to design algorithms depending only quadratically or linearly on the size of the network when its topology is expected to satisfy some properties. Among these properties, the *chordality* of a graph is the length of its longest induced (i.e., chordless) cycle. The

Partially supported by programs Fondap and Basal-CMM, Anillo ACT88 and Fondecyt 11090390 (K.S.), FP7 STREP EULER (N.N.), AGAPE, ECOS-SUD, EA. AlDyNet, ANR DISPLEXITY, NCN under contract DEC-2011/02/A/ST6/00201 (A.K.).
This work has been announced at the 39th International Colloquium on Automata, Languages and Programming (ICALP 2012); the extended abstract appeared in the corresponding proceedings [35].

A. Kosowski
CEPAGE, INRIA, LaBRI, Talence, France

B. Li
Present address: Coati Project, INRIA, I3S(CNRS/UNS), Sophia Antipolis, France
Institute of Applied Mathematics, AAMS, CAS, Beijing, China
E-mail: bi.li@inria.fr

N. Nisse
Coati Project, INRIA, I3S(CNRS/UNS), Sophia Antipolis, France

K. Suchan
FIC, Universidad Adolfo Ibáñez, Santiago, Chile
WMS, AGH - University of Science and Technology, Krakow, Poland

(Gromov) *hyperbolicity* of a graph reflects how the metric (distances) of the graph is close to the metric of a tree. More precisely, a graph has hyperbolicity $\leq \delta$ if, for any $u, v, w \in V(G)$ and any shortest paths P_{uv}, P_{vw}, P_{uw} between these three vertices, any vertex in P_{uv} is at distance at most δ from $P_{vw} \cup P_{uw}$ [30]. Intuitively, in a graph with small hyperbolicity, any two shortest paths between the same pair of vertices are close to each other. Several recent works take advantage of such structural properties of large-scale networks for algorithm design (e.g., routing [36,20]). Indeed, Internet-type networks have a so-called high clustering coefficient (see e.g. [49, 42]), leading to the existence of very few long chordless cycles, whereas their low (logarithmic) diameter implies a small hyperbolicity [39].

Another way to study tree-likeness of graphs is by *tree-decompositions*. Introduced by Robertson and Seymour [44], such decompositions play an important role in design of efficient algorithms. Roughly speaking, a tree-decomposition maps each vertex of a graph to a subtree of the *decomposition tree* in a way that the subtrees assigned to adjacent vertices intersect [44,13]. The nodes of the decomposition tree are called *bags*, and the size of a bag is the number of vertices assigned to it. The *width* of a tree-decomposition is the maximum size over its bags minus 1, and the *treewidth* of a graph is the smallest width over its tree-decompositions. By using dynamic programming based on a tree-decomposition, many NP-hard problems have been shown to be linear time solvable for graphs of bounded treewidth [25]. In particular, there are linear-time algorithms to compute an optimal tree-decomposition of a graph with bounded treewidth [12,14]. However, from the practical point of view, this approach has several drawbacks. First, all above-mentioned algorithms are linear in the size of the graph but (at least) exponential in the treewidth. Moreover, due to the high clustering coefficient of large-scale networks, their treewidth is expected to be large [39]. Hence, to face these problems, it is important to focus on the structure of the bags of the tree-decomposition, instead of trying to minimize their size. For instance, several works study the diameter of the bags [27,37]. In this work, we consider tree-decompositions in which each bag admits a particular small dominating set. Such decompositions turn out to be applicable to a large family of graphs (including k-chordal graphs).

1.1 Our results

Our results on tree decomposition are inspired by a study of the so called *cops and robber games* (Winkler and Nowakowski [41], Quilliot [43]). The aim of such a game is to capture a robber moving in a graph, using as few cops as possible. This problem has been intensively studied in the literature, allowing for a better understanding of the structure of graphs [17].

Outline of the paper. We start by presenting our results for the cops and robber problem in Section 2. Next, using these results, in Section 3 we provide a new type of efficiently computable tree-decomposition which we call *good tree decomposition*. Our tree decomposition turns out to be applicable to many real-world graph classes (including k-chordal graphs), and has several algorithmic applications. Finally, we focus on the applications of this decomposition to the *compact routing problem*, a research area in which tree decompositions have already proved useful [26]. The objective of compact routing is to provide a scheme for finding a path from a sender vertex to a known destination, taking routing decisions for the packet at every step using only very limited information stored at each vertex. In Section 4, we show how to use our tree decomposition to minimize the additive stretch of the routing scheme (i.e., the difference between the length of a route computed by the scheme and that of a shortest path connecting the same pair of vertices) in graphs admitting a k-good tree-decomposition for any given integer $k \geq 3$ (including k-chordal graphs), assuming logarithmic size of packet headers and routing tables stored at each vertex.

The necessary terminology concerning cops and robber games, tree decompositions, and compact routing, is introduced in the corresponding sections.

Main contributions. Our main contribution is the design of a polynomial-time algorithm that, given a n-vertex graph G and an integer $k \geq 3$, either returns an induced cycle of length at least $k+1$ in G or computes a tree-decomposition of G with each bag having a dominating path of order (number of vertices on the path) at most $k-1$. More precisely, each bag of our tree-decomposition contains a chordless path with at most $k-1$ vertices, such that any vertex in the bag is either on the path or adjacent to some vertex of the path. In the case when G admits such a decomposition, this ensures that G has treewidth at most $(k-1)(\Delta-1)+2$ (where Δ is the maximum degree), tree-length at most k and Gromov hyperbolicity at most $\lfloor \frac{3}{2}k \rfloor$. In particular, this shows that the treewidth of any k-chordal graph is upper-bounded by $O(k \cdot \Delta)$, improving the exponential bound of [16]. The

proposed algorithm is mainly derived from our proof of the fact that $k-1$ cops are sufficient to capture a robber in k-chordal graphs (generalizing some results in [7,24]).

Our tree-decomposition may be used efficiently for solving problems using dynamic programming in graphs of small chordality and small maximum degree. In particular, we present a compact routing scheme that uses our tree-decomposition and that achieves an additive stretch $\leq 2k(\lceil \log \Delta \rceil + \frac{5}{2}) - 2\lceil \log \Delta \rceil - 4$ with routing tables, addresses and message headers of $O(k \cdot \log \Delta + \log n)$ bits. An earlier approach of Dourisboure achieved stretch $k+1$, but with routing tables of size $O(\log^2 n)$.

1.2 Related Work

Chordality and hyperbolicity. Chordality and hyperbolicity are both parameters measuring "tree-likeness" of a graph. Some papers consider relations between them [11,50]. In particular, the hyperbolicity of a k-chordal graph is at most k, i.e. the hyperbolicity of a graph is at most its chordality. But the gap, i.e. the difference between the two parameters, may be arbitrary large (take a $3 \times n$-grid). The seminal definition of hyperbolicity is the following. A graph G is δ-hyperbolic provided that for any vertices $x,y,u,v \in V(G)$, the two larger of the three sums $d(u,v)+d(x,y), d(u,x)+d(v,y)$ and $d(u,y)+d(v,x)$ differ by at most 2δ [30]. With this definition, it is proved that any graph with tree-length at most k has hyperbolicity at most k [22]. This definition is equivalent to that of Gromov hyperbolicity (mentioned at the beginning of the introduction), which we use in this paper, up to a constant ratio [8]. No algorithm better than the $O(n^4)$-brute force algorithm (testing all 4-tuples in G) is known to compute hyperbolicity of n-vertex graphs. The problem of deciding whether the chordality of a graph G is at most k is NP-complete if k is as part of the input. Indeed, if G' is obtained by subdividing all the edges in G once, then there is an induced cycle of length $2|V(G)|$ in G' if and only if G has a Hamilton cycle. It is coNP-hard to decide whether an n-vertex graph G is k-chordal for $k = \Theta(n)$ [48].

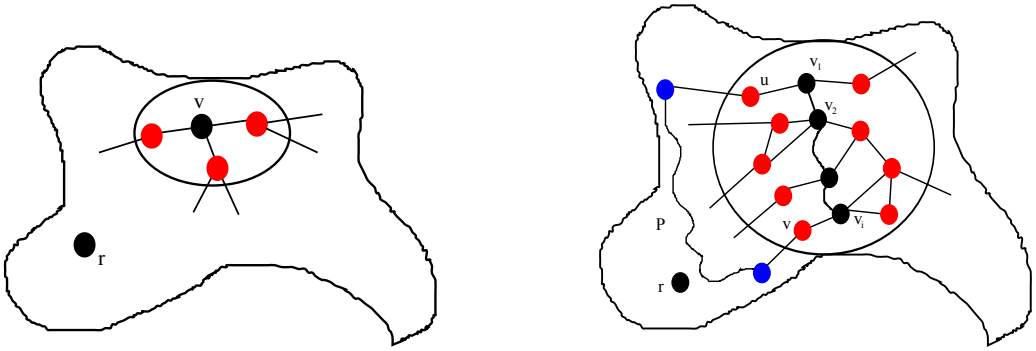
There are several problems related to chordality are considered. Finding the longest induced path is $W[2]$-complete [21]. In [33], the problem of deciding whether there is an induced cycle passing through k given vertices is studied. This problem is NP-Complete in planar graphs when k is part of the input and in general graphs even for $k=2$. However, this problem is Fixed Parameter Tractable (FPT) in planar graphs, i.e., there is an algorithm to solve this problem in time $O(f(k)p(n))$ where f is an arbitrary function of k and p is a polynomial in the size n of the graph. Finding an induced cycle of size exactly k in d-degenerate graph (every induced subgraph has a vertex of degree at most d) is FPT if k and d are fixed parameters [18]. Note that, any planar graph is 5-degenerate.

Treewidth. It is NP-complete to decide whether the treewidth of a graph G is at most k [10]. For (4-)chordal graphs, cographs [15], circular arc graphs [47], chordal bipartite graphs [32] and etc., the treewidth problem is polynomially solvable. Bodlaender and Thilikos proved that the treewidth of a k-chordal graph for ($k \geq 4$) with maximum degree Δ is at most $\Delta(\Delta-1)^{k-3}$ which implies that treewidth is polynomially computable in the class of graphs with chordality and maximum degree bounded by constants [16]. They also proved that the treewidth problem is NP-complete for graphs with small maximum degree [16].

Compact routing. In a name-independent routing scheme, the designer of the scheme is not allowed to label the vertices in the way he wants, that is, each vertex in the network has a predefined fixed label. Abraham et al. provided a universal name-independent routing scheme with stretch linear in k and $n^{1/k}polylog(n)$ space in [5]. There are weighted trees for which every name-independent routing scheme with space less than $n^{1/k}$ requires stretch at least $2k+1$ and average stretch at least $k/4$ [4]. Subsequently, the interest of the scientific community was turned toward specific properties of graphs. Several routing schemes have been proposed for particular graph classes: e.g., trees [28], bounded growth [6], bounded hyperbolic graph [23], bounded doubling dimension [2,34], excluding a fixed graph as a minor [3,1], etc. The best compact routing scheme in k-chordal graphs (independent from the maximum degree) is due to Dourisboure and achieves a stretch of $k+1$ using routing tables of size $O(\log^2 n)$ bits [26]. A routing scheme achieving stretch $k-1$ with a distributed algorithm for computing routing tables of size $O(\Delta \log n)$ bits has been proposed in [40].

1.3 Notations

Throughout the paper, denote G as a simple connected undirected graph with vertex set V and edge set E. Let $n = |V|$ be the *order* of G and $m = |E|$ is the *size* of G. For any subgraph H of G, denoted as $H \subseteq G$, we use $V(H)$



(a) All the cops are placed on v. The robber is on r.

(b) The induced subgraph $(v_1,\ldots,v_i,v,P,u)$ is a chordless cycle $\geq i+2$. The robber is on r.

Fig. 1: illustration for the proof of Theorem 1

and $E(H)$ to denote the vertex and edge set of H, respectively. The set of vertices adjacent to $v \in V$ in G is denoted $N_G(v)$ and called *open neighborhood* of v. Let $N_G[v] = N_G(v) \cup \{v\}$ be the *closed neighborhood* of v. We extend this notation for a vertex set $U \subset V$ to write $N_G[U] = \cup_{u \in U} N_G[u]$ and $N_G(U) = N_G[U] \setminus U$. Let $d_G(v) = |N_G(v)|$ be the *degree* of v and Δ denote the maximum degree among the vertices of G. If the context is clear for graph G, then we use $N(v)$ instead of $N_G(v)$ and similarly for $N[v]$, $N(U)$ and $N[U]$. The graph obtained from G by *removing an edge* $\{x,y\}$ is denoted $G \setminus \{x,y\}$; the result of *removing a vertex* v and all adjacent edges is denoted $G \setminus \{v\}$. Like above, we extend this to denote removing sets of vertices or edges. For $U \subset V$, the subgraph of G *induced* by U is denoted as $G[U]$. It can be obtained as the result of removing from G the vertices in $V \setminus U$, denoted by $G \setminus (V \setminus U)$. Given two paths $P = (p_1,\ldots,p_k)$ and $Q = (q_1,\ldots,q_r)$, we denote their concatenation by (P,Q) the path induced by $V(P) \cup V(Q)$; to make descriptions more concise, we omit the detail of reversing P or Q if necessary.

2 A detour through Cops and Robber games

In this section, we study the cops and robber games introduced by Winkler and Nowakowski [41], independently defined by Quilliot [43]. Given a graph G, a player starts by placing $k \geq 1$ cops on some vertices of G, then a visible robber is placed on one vertex of G. Alternately, the cop-player may move each cop along one edge, and then the robber can move to an adjacent vertex. The robber is captured if, at some step, a cop occupies the same vertex.

Aigner and Fromme introduced the notion of *cop-number* of a graph G, i.e., the fewest number of cops $cn(G)$, such that there exists a strategy for the cop-player that assures to capture the robber whatever he does [7]. A long standing conjecture due to Meyniel states that $cn(G) = O(\sqrt{n})$ for any n-vertex graph G [29]. To tackle this question, many researchers have focused on particular graph classes and provided many nice structural results (see the recent book [17]). For any n-vertex graph G, $cn(G) = O(\frac{n}{2^{(1-o(1))\sqrt{\log n}}})$ [38,46], $cn(G) \leq 3$ in any planar graph G [7], $cn(G) \leq 3 + \frac{3}{2}g$ in any graph G with genus at most g [45], $cn(G) = O(m)$ in any graph G excluding a m-edge graph as a minor [9], etc. Bounded hyperbolicity graphs have been considered in [19]. The cop number of graphs with minimum degree d and smallest induced cycle (girth) at least $8t-3$ is known to be $\Omega(d^t)$ [29]. Strangely, little is known related to the largest induced cycle (chordality): in [7], it is shown that $cn(G) \leq 3$ for any 2-connected 5-chordal graph G. In this section, we consider the class of k-chordal graphs.

Theorem 1 *Let $k \geq 3$. For any k-chordal connected graph G, $cn(G) \leq k-1$, and there exists a strategy where all $k-1$ cops always occupy a chordless path except for the move that captures the robber.*

Proof Let $v \in V$ be any vertex and place all cops at it (see in Fig. 1(a)). Then, the robber chooses a vertex. Now, at some step, assume that the cops are occupying $\{v_1,\cdots,v_i\}$ which induce a chordless path, $i \leq k-1$, and it is the turn of the cops (initially $i = 1$). Let $N = \bigcup_{1 \leq j \leq i} N[v_j]$, if the robber occupies a vertex in N, it is captured during the next move. Else, let $R \neq \emptyset$ be the connected component of $G \setminus N$ occupied by the robber. Finally, let S be the set of vertices in N that have some neighbor in R. Clearly, since R is non-empty, so is S.

Now, there are two cases to be considered.

– If $N(v_1) \cap S \subseteq \bigcup_{1<j\leq i} N[v_j]$. This case may happen only if $i > 1$. Then,"remove" the cop(s) occupying v_1. That is, the cops occupying v_1 go to v_2. Symmetrically, if $N(v_i) \cap S \subseteq \bigcup_{1\leq j<i} N[v_j]$, then the cops occupying v_i go to v_{i-1}. Then, the cops occupy a shorter chordless path while the robber is still restricted to R.
– Otherwise, there is $u \in (N(v_1) \cap S) \setminus (\bigcup_{1<j\leq i} N[v_j])$ and $v \in (N(v_i) \cap S) \setminus (\bigcup_{1\leq j<i} N[v_j])$. First, we show that this case may happen only if $i < k-1$. Indeed, let P be a shortest path between such u and v with all internal vertices in R (possibly, P is reduced to an edge). Such a path exists by definition of S. Then $(v_1, \cdots, v_i, v, P, u)$ is a chordless cycle of length at least $i+2$ (See in Fig. 1(b)). Since G is k-chordal, this implies that $i+2 \leq k$. Then one cop goes to $v_{i+1} := v$ while all the vertices in $\{v_1, \cdots, v_i\}$ remain occupied. Since $v \in S$, it has some neighbor in R, and then, the robber is restricted to occupy R', the connected component of $G \setminus (N \cup N[v])$ which is strictly contained in R.

Therefore, proceeding as described above strictly reduces the area of the robber (i.e., R) after $< k$ steps, R decreases progressively and the robber is eventually captured. □

Note that previous Theorem somehow extends the model in [24], where the authors consider the game when two cops always remaining at distance at most 2 from each other must capture a robber. It is possible to improve the previous result in the case of 4-chordal graphs, i.e. $k = 4$. In the following theorem, we prove that $cn(G) \leq 2$ for any 4-chordal connected graph G.

Theorem 2 *For any* 4*-chordal connected graph* G*,* $cn(G) \leq 2$ *and there always exists a winning strategy for the cops such that they are always at distance at most one from each except for the move that captures the robber.*

Proof Initially, place the cops on any two adjacent vertices. At some step of the strategy, let us assume that the cops are on two adjacent vertices a and b (or $a = b$) and it is the turn of the cops. If the robber stands at some vertex in $N = N[a] \cup N[b]$, then it is captured during the next move. Hence, let R be the connected component of $G \setminus N$ where the robber stands. Let $S \subseteq N$ be the set of the vertices adjacent to a or b and at least one vertex of R, i.e., S is an inclusion-minimal separator between $\{a,b\}$ and R.

We will prove that there is $z \in \{a,b\}$ and a vertex c in $S \cap N(z)$, such that, $S \subset N[z] \cup N[c] = N'$. Since $c \in S$, $N(c) \cap V(R) \neq \emptyset$. Hence, if the cops move from a,b to c,z, which can be done in one step, then the robber is constrained to occupy a vertex of R' where R' is the connected component of $G \setminus N'$ which is strictly contained in R. Note that, R' is a proper subgraph of R. Iterating such moves, the robber will eventually be captured.

It remains to prove the existence of $z \in \{a,b\}$ and $c \in S \cap N(z)$, such that, $S \subseteq N[z] \cup N[c]$.

– If there is $z \in \{a,b\}$ such that $S \subseteq N[z]$, then any vertex in $N(z) \cap S$ satisfies the requirements.
– Else, let $c \in S \setminus N(b)$ (such a vertex exists because otherwise we would be in the previous case). Clearly, $S \cap N(a) \subseteq N[a] \cup N[c]$. Now, let $x \in S \setminus N(a)$. By definition of S, there is a path P from x to c with internal vertices in R. Moreover, all internal vertices of P are at distance at least two from a and b; also c is not adjacent to b and x is not adjacent to a. Hence, considering the cycle a,b,x,P,c, there must be an edge between x and c because G is 4-chordal. So $S \setminus N(a) \subset N(c)$. Therefore, $S = (S \cap N(a)) \cup (S \setminus N(a)) \subseteq N[a] \cup N[c]$.

The bound provided by this theorem is tight because of the cycle with 4 vertices. □

Theorem 1 relies on chordless paths P in G such that $N[V(P)]$ is a separator of G, i.e., there exist vertices a and b of G such that all paths between a and b intersect $N[V(P)]$. In the next section, we show how to adapt this to compute particular tree-decompositions.

3 Structured Tree-decomposition

In this section, we present our main contribution, that is, an algorithm that, given a n-vertex graph G and an integer $k \geq 3$, either returns an induced cycle of length at least $k+1$ in G or computes a tree-decomposition of G. First, we need some definitions.

A *tree-decomposition* of a graph $G = (V,E)$ is a pair $(\{X_i | i \in I\}, T = (I,M))$, where T is a tree and $\{X_i | i \in I\}$ is a family of subsets, called bags, of vertices of G such that (1) $V = \bigcup_{i \in I} X_i$; (2) $\forall \{u,v\} \in E$ there is $i \in I$ such that $u,v \in X_i$; and (3) $\forall v \in V$, $\{i \in I | v \in X_i\}$ induces a (connected) subtree of T. The *width* of a tree-decomposition is the size of its largest bag minus 1 and its ℓ*-width* is the largest distance between two vertices of a bag of a tree-decomposition. The *treewidth* [44] denoted by $tw(G)$ (resp., *tree-length* [27] denoted by $tl(G)$) of a graph G is the minimum width (resp., ℓ-width), over all possible tree-decompositions of G.

Let $k \geq 2$. Let us define a k-super-caterpillar as a graph that has a dominating set, called *backbone*, which induces a chordless path of order at most $k-1$. That is, any vertex of a k-super-caterpillar either belongs to the backbone or is adjacent to a vertex of the backbone. A tree-decomposition is said to be *k-good* if each of its bags induces a k-super-caterpillar. Clearly, the width of a k-good tree decomposition is at most $O(k\Delta)$ and its ℓ-width is at most k.

Theorem 3 *Given an m-edge-graph G and an integer $k \geq 3$, there is a $O(m^2)$-algorithm which:*

- *either returns an induced cycle of length at least $k+1$;*
- *or returns a k-good tree-decomposition of G.*

Proof The proof is by induction on $|V(G)| = n$. We prove that either we find an induced cycle larger than k, or for any chordless path $P = (v_1, \ldots, v_i)$ with $i \leq k-1$, there is a k-good tree-decomposition for G with one bag containing $N_G[V(P)]$. Note that the later case does not mean that a large induced cycle does not exist. Obviously, it is true if $|V(G)| = 1$. Now we assume that it is true for any graph G with n' vertexs, $1 \leq n' < n$, and we show it is true for n-vertex graphs.

Let G be a connected n-vertex graph, $n > 1$. Let $P = (v_1, \ldots, v_i)$ be any chordless path with $i \leq k-1$ and let $N = N_G[V(P)]$ and $G' = G \setminus N$. There are three cases to be considered:

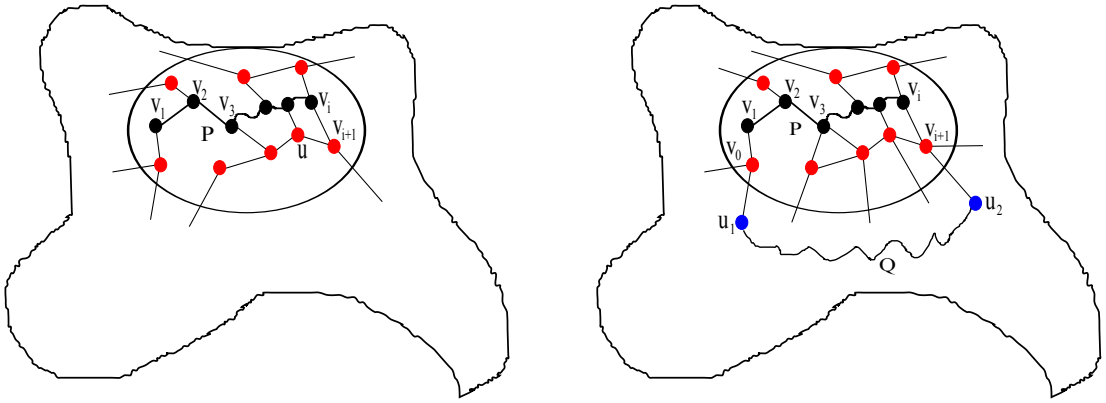
Case 1. Let $G' = \emptyset$. In this case, we have $V(G) = N$. The desired tree-decomposition consists of one node, corresponding to the bag N.

Case 2. Let G' be disconnected. Let $C_1, \ldots, C_r$, $r \geq 2$, be the connected components of G' For any $j \leq r$, let G_j be the graph induced by $C_j \cup N$. Note that any induced cycle in G_j, for any $j \leq r$, is an induced cycle in G. By the induction hypothesis, either there is an induced cycle $\mathscr{C}$ larger than k in G_j, then $\mathscr{C}$ is also an induced cycle larger than k in G, or our algorithm computes a k-good tree-decomposition TD_j of G_j with one bag X_j containing N. To obtain the k-good tree-decomposition of G, we combine the TD_j's, for $j \leq r$, by adding a bag $X = N$ adjacent to all the bags X_j for $j = 1, \ldots, r$. It is easy to see that this tree-decomposition satisfies our requirements.

Case 3. Let G' be connected. We consider the order of the path $P = (v_1, \ldots, v_i)$. In the following proof, first we prove that if the order of path P, $i = k-1$, then we can find either an induced cycle larger than k or the required tree-decomposition for G. Subsequently, we prove it is also true for path with order $i < k-1$ by backward induction on i. More precisely, if $i < k-1$, either we find directly the desired cycle or tree-decomposition, or we show that there exists a vertex v_{i+1} such that $P \cup \{v_{i+1}\}$ induces a chordless path P' of order $i+1$. By backward induction on i we can find either an induced cycle larger than k or a k-good tree-decomposition of G with one bag containing $N_G[V(P')] \supseteq N_G[V(P)]$.

(a) If $i = k-1$, then we consider the following two cases.
   - Assume first that there is $u \in N_G(V(P)) \cup \{v_1, v_i\}$ (in particular, $u \notin P \setminus \{v_1, v_i\}$) such that $N_G(u) \subseteq N_G[V(P) \setminus \{u\}]$ (See in Fig. 2(a)). Let $\tilde{G} = G \setminus \{u\}$. Then $\tilde{G}$ is a graph with $n' = n-1$ vertices. By the induction hypothesis on $n' < n$, the algorithm either finds an induced cycle larger than k in $\tilde{G}$, then it is also the one in G; Otherwise our algorithm computes a k-good tree-decomposition $\widetilde{TD}$ of $\tilde{G}$ with one bag $\tilde{X}$ containing $N_{\tilde{G}}[V(P) \setminus \{u\}]$. To obtain the required tree-decomposition of G, we just add vertex u into the bag $\tilde{X}$. The tree-decomposition is still k-good.
   - Otherwise, there exist two distinct vertices $v_0 \in N_G(v_1) \setminus N_G(V(P) \setminus v_1)$ and $v_{i+1} \in N_G(v_i) \setminus N_G(V(P) \setminus v_i)$ and there are vertices $u_1, u_2 \in V(G')$ (possibly $u_1 = u_2$) such that $\{v_0, u_1\} \in E(G)$ and $\{v_{i+1}, u_2\} \in E(G)$ (See in Fig. 2(b)). If $\{v_0, v_{i+1}\} \in E(G)$, (P, v_0, v_{i+1}) is an induced cycle with $k+1$ vertices. Otherwise, let Q be a shortest path between u_1 and u_2 in G' (Q exists since G' is connected). So $(P, v_{i+1}, u_2, Q, u_1, v_0)$ is an induced cycle with at least $k+1$ vertices in G.

(b) If $i < k-1$, we proceed by backward induction on i. Namely, assume that, for any chordless path Q with $i+1$ vertices, our algorithm either finds an induced cycle larger than k in G or computes a k-good tree-decomposition of G with one bag containing $N[V(Q)]$. Note that the initialization of the induction holds for $i = k-1$ as described in case (a). We show it still holds for a chordless path with i vertices. We consider the following two cases.
   - Either there is $u \in N_G(V(P)) \cup \{v_1, v_i\}$ (in particular, $u \notin P \setminus \{v_1, v_i\}$) such that $N_G(u) \subseteq N_G[V(P) \setminus \{u\}]$. That is, we are in the same case as the first item of (a). We proceed as above and the result holds by induction on n.
   - Or there is $w \in (N_G(v_1) \cup N_G(v_i)) \setminus V(P)$ such that (P, w) is chordless (i.e., the vertex w is a neighbor of v_1 or v_i but not both and $w \notin N_G(V(P) \setminus \{v_1, v_i\})$). Therefore, we apply the induction hypothesis



(a) The vertex $u \in N_G(V(P))$ and all the neighbors of u are the neighbors of P.

(b) The induced subgraph $(P, v_{i+1}, u_2, Q, u_1, v_0)$ is an induced cycle with at least $i+2 \geq k+1$ vertices in G.

Fig. 2: illustration for the proof of Theorem 3

(on i) on $P' = (P, w)$. By the assumption on i, either our algorithm returns an induced cycle larger than k or it computes a k-good tree-decomposition of G with one bag containing $N_G[V(P')] \supseteq N_G[V(P)]$.

Now we describe the algorithm and study its complexity. Let G be an m-edge n-vertex graph with maximum degree Δ. Roughly speaking, the algorithm proceeds by steps. At each step, one vertex is considered and the step takes $O(m)$ time. We prove that at each step (but the initial one), at least one edge will be *considered* and that each edge is considered at most once (but one vertex may be considered several times). This implies a time-complexity of $O(m^2)$ for the algorithm.

The algorithm starts from an arbitrary vertex $v \in V(G)$ and computes the connected components $C_1, \cdots, C_j$ of $G \setminus N[v]$ ($j \geq 1$) in time $O(m)$ [31]. We start with the k-good tree-decomposition for the induced graph of $N[v]$ in G that consists of one bag $B = N[v]$ adjacent to, for any $i \leq j$, each bag $B_i = \{v\} \cup \{w \in N(v) \ : \ N(w) \cap C_i \neq \emptyset\}$. This takes time $O(m)$.

Now, at some step of the strategy, assume that we have built a k-good tree-decomposition $(T, \mathscr{X})$ of a connected subgraph G_0 of G. Let $C_1, \cdots, C_j$ ($j \geq 1$) be the connected components of $G \setminus G_0$, and, for any $i \leq j$, let S_i be the set of the vertices of G_0 that are adjacent to some vertex of C_i. Assume finally that, for any $i \leq j$, there is a leaf bag $B_i \supset S_i$ of $(T, \mathscr{X})$ where $P_i = B_i \setminus S_i$ is a chordless path dominating B_i and has minimum number of vertices.

For any $e \in E(G)$, we say that $e = \{x, y\}$ is *alive* if there is $i \leq j$ such that $x \in S_i \cup C_i$ and $y \in C_i$. Note that, if an edge is alive, such an integer i is unique. An edge that is not alive is said *dead*. Note also that, after the initial step, all edges in the bag B are dead and other edges are alive.

The next step consists of the following. Choose any $i \leq j$ and let w be any vertex of S_i such that $Q = P_i \cup \{w\}$ is a chordless path. (Such w exists because P_i is the dominating path with the minimum order. Suppose $P_i = \{v_1, \ldots, v_l\}$. If $N_G(v_1) \setminus V(P_i) = \emptyset$, then the chordless path $P_i \setminus v_1$ dominates B_i and has less vertices than P_i. So $N_G(v_1) \setminus V(P_i) \neq \emptyset$. If any $w \in N_G(v_1) \setminus V(P_i)$ is a neighbor of some vertices in P_i, then the chordless path $P_i \setminus v_1$ dominates B_i and has less vertices than P_i.) Note that by definition of S_i, there is at least one edge from w to C_i and that such an edge is alive before this step. We add the bag $B' = Q \cup B_i \cup (N(w) \cap C_i)$ adjacent to B_i. If Q is larger than k, by the above proof, the algorithm finds a large cycle. Otherwise, the connected components $C'_1, \cdots, C'_r$ of $C_i \cup B_i \setminus B'$ are computed in time $O(m)$. Let S'_h, $h \leq r$, be the subset of the vertices of S_i that are adjacent to some vertex in C'_h, and let Q_h be the smallest subpath of Q dominating S'_h. Computing the sets $S'_1, \cdots, S'_r$ only requires a time $O(m)$ since we have only to check the edges in B'. For any $h \leq r$, add a bag $B'_h = Q_h \cup S'_h$ adjacent to B'.

One can check that this algorithm follows the above proof and that it eventually computes the desired tree-decomposition or returns a large cycle.

To conclude, we can check that the set of edges alive after one step is contained in the set of edges alive before this step, and that, at each step at least one edge (the one(s) from w to C_i) becomes dead. Therefore, at each step, the number of alive edges strictly decreases and the algorithm terminates when there are no more. Since each step takes time $O(m)$ and there are at most m steps, the result follows. □

The following two theorems discuss some properties of the graphs with k-good tree decompositions.

Theorem 4 *Let G be a graph that admits a k-good tree-decomposition. Let Δ be the maximum degree of G. Then $tw(G) \leq (k-1)(\Delta-1)+2$ and $tl(G) \leq k$.*

Proof It directly follows the fact that, in a k-good tree-decomposition, each bag has a dominating path with $< k$ vertices. □

Recall that a graph G has Gromov hyperbolicity $\leq \delta$ if, for any $u,v,w \in V(G)$ and any shortest paths P_{uv}, P_{vw}, P_{uw} between these three vertices, any vertex in P_{uv} is at distance at most δ from $P_{vw} \cup P_{uw}$. In the next theorem, we prove that the Gromov hyperbolicity of the graph admitting a k-good tree-decomposition is at most $\lfloor \frac{3}{2}k \rfloor$.

Notice that the result given in [22] refers to the seminal hyperbolicity and does not imply our result for Gromov hyperbolicity.

Theorem 5 *Any graph G that admits a k-good tree-decomposition has Gromov hyperbolicity at most $\lfloor \frac{3}{2}k \rfloor$.*

Proof Let $G=(V,E)$ be a graph that admits a k-good tree-decomposition $(\{X_i | i \in I\}, T=(I,M))$. Let T be rooted at bag X_0, $0 \in I$. For any $u,v \in V$, let us denote the distance between u and v in G by $d(u,v)$. By definition of a k-good decomposition, for any $i \in I$ and for any $u,v \in X_i$, we have $d(u,v) \leq k$.

Let $x,y,z \in V$ and let P_1,P_2,P_3 be any three shortest paths in G between x and y, y and z, x and z respectively. Let $u \in P_1$. To prove the Theorem, we show that there is $v \in P_2 \cup P_3$ such that $d(u,v) \leq \lfloor \frac{3}{2}k \rfloor$.

First, let us assume that there is $i \in I$ such that $u \in X_i$ and there is $v \in (P_2 \cup P_3) \cap X_i \neq \emptyset$. In that case, $d(u,v) \leq k$ and the result holds.

Otherwise, let T_u be the subtree of T induced by $\{i \in I \ : \ u \in X_i\}$. Similarly, let T_x be the subtree of T induced by $\{i \in I \ : \ x \in X_i\}$ and T_y be the subtree of T induced by $\{i \in I \ : \ y \in X_i\}$. Let P be the path in T between T_x and T_y. Note that P may be empty if $V(T_x) \cap V(T_y) \neq \emptyset$. Let $j \in V(T_x) \cup V(T_y) \cup P$ that is closest to T_u in T. If $j \in V(T_u)$, then X_j is a separator between x and y or $x \in X_j$ or $y \in X_j$. If $x \in X_j$ or $y \in X_j$, then we are in the first case above; otherwise we have X_j is a separator between x and y. Then $z \in X_j$ or z cannot be in both the component of $G \setminus X_j$ containing x and of the one containing y. So one of the paths P_2 or P_3 should pass trough X_j and we are in the the first case again.

Assume that $j \notin V(T_u)$, then we have that either X_j is a separator between x and u or $x \in X_j$, and that either X_j is a separator between y and u or $y \in X_j$. Let P_{xu} and P_{uy} be the subpaths of P_1 from x to u and from u to y respectively. By remark above, there exist vertices $w \in P_{xu} \cap X_j$ and $t \in P_{uy} \cap X_j$. Possibly, $w=t$. Then $d(w,u)+d(u,t)=d(w,t)$ because P_1 is a shortest path, therefore, $d(w,u)+d(u,t)=d(w,t) \leq k$. So there is $\ell \in X_j$ with $d(u,\ell) \leq \lfloor \frac{k}{2} \rfloor$.

Finally, let us show that there is $h \in (P_2 \cup P_3) \cap X_j$. If $x \in X_j$ or $y \in X_j$ or $z \in X_j$, it is obvious. Otherwise, z cannot be in both the component of $G \setminus X_j$ containing x and of the one containing y, because X_j separates x and y in G. Therefore one of the paths P_2 or P_3 should pass trough X_j.

To conclude, $d(u,h) \leq d(u,\ell)+d(\ell,h) \leq \lfloor \frac{k}{2} \rfloor + k \leq \lfloor \frac{3}{2}k \rfloor$. □

From the above theorems, it is easy to get the following corollaries.

Corollary 1 *Any k-chordal graph G with maximum degree Δ has treewidth at most $(k-1)(\Delta-1)+2$, tree-length at most k and Gromov hyperbolicity at most $\lfloor \frac{3}{2}k \rfloor$.*

Proof By definition of k-chordal graph and Theorem 3, any k-chordal graph admits a k-good tree-decomposition. The result follows from Theorems 4 and 5. □

Corollary 2 *There is an algorithm that, given an m-edge graph G and $k \geq 3$, states that either G has chordality at least $k+1$ or G has Gromov hyperbolicity at most $\lfloor \frac{3}{2}k \rfloor$, in time $O(m^2)$.*

4 Application of k-good tree-decompositions for routing

In this section, we propose a compact routing scheme for any n-vertex graph G that admits a k-good tree-decomposition (including k-chordal graphs). Recall that Δ denotes the maximum degree of G and that the degree of any $v \in V(G)$ is denoted as $d_G(v)$.

4.1 Model and performance of the routing scheme

We propose a *labelled* routing scheme which means that we are allowed to give one identifier, $name(v)$, of $O(\log n)$ bits to any vertex v of G. Moreover, following [28], we consider the *designer-port* model, which allows us to choose the permutation of ports (assign a label of $\log d_G(v)$ bits to any edge incident to v in $V(G)$). Finally, to any vertex $v \in V(G)$, we assign a routing table, denoted by $Table(v)$, where local information of $O(k \cdot \log \Delta + \log n)$ bits is stored. Any message has a *header* that contains the address $name(t)$ of the destination t, three modifiable integers $pos \in \{-1,1,2,\cdots,k-1\}, cnt, cnt' \in \{-1,0,\cdots,\Delta+1\}$, one bit $start$ and some memory, called $path$, of size $O(k \cdot \log \Delta)$ bits. The two items $start$ and $path$ change only once.

Following our routing scheme, a vertex v that receives a message uses its header, $name(v)$, $Table(v)$ and the port-numbers of the edges incident to v to compute its new header and to choose the edge $e = \{v,u\}$ over which it relays the message. Then, the vertex u knows that the message arrived from v. The length of the path followed by a message from a source $s \in V(G)$ to a destination $t \in V(G)$, using the routing scheme, is denoted by $|P(s,t)|$, and the *stretch* of the scheme is $\max_{s,t \in V(G)}(|P(s,t)| - d(s,t))$ where $d(s,t)$ is the distance between s and t in G.

To design our routing scheme, we combine the compact routing scheme in trees of [28] together with the k-good tree-decomposition. Roughly, the scheme consists of following the paths in a BFS-tree F of G according to the scheme in [28], and using one bag of the tree-decomposition as a short-cut between two branches of F. Intuitively, if the source s and the destination t are "far apart", then there is a bag X of the tree-decomposition that separates s and t in G. The message follows the path in F to the root of F until it reaches X, then an exhaustive search is done in X until the message finds an ancestor y of t, and finally it follows the path from y to t in F using the scheme of [28]. The remaining part of this Section is devoted to the proof of the next Theorem that summarizes the performances of our routing scheme.

Theorem 6 *For any n-vertex m-edge graph G with maximum degree Δ and with a k-good tree-decomposition, there is a labelled routing scheme $\mathscr{R}$ with the following properties. The scheme $\mathscr{R}$ uses addresses of size $O(\log n)$ bits, port-numbers of size $O(\log \Delta)$ bits and routing tables of size $O(k \cdot \log \Delta + \log n)$ bits. The routing tables, addresses and port-numbers can be computed in time $O(m^2)$. Except the address of the destination (not modifiable), the header of a message contains $O(k \cdot \log \Delta)$ modifiable bits. The header and next hop are computed in time $O(1)$ at each step of the routing. Finally, the additive stretch is $\leq 2k(\lceil \log \Delta \rceil + \frac{5}{2}) - 2\lceil \log \Delta \rceil - 4$.*

4.2 Data structures

4.2.1 Routing in trees [28].

Since we use the shortest path routing scheme proposed in [28] for trees, we start by recalling some of the data structures that this scheme uses. Let F be a tree rooted in $r \in V(F)$. For any $v \in V(F)$, let F_v be the subtree of F rooted in v and let $w_F(v) = |V(F_v)|$ be the *weight* of v. Consider a Depth-First-Search (DFS) traversal of F, starting from r, and guided by the weight of the vertices, i.e., at each vertex, the DFS visits first the largest subtree, then the second largest subtree, and so on. For any $v \in V(F)$, let $Id_F(v) \in \{1,\cdots,n\}$ be the preordering rank of v in the DFS.

Lemma 1 *For any $u,v \in V(F)$, $v \in V(F_u)$ if and only if $Id_F(u) \leq Id_F(v) \leq Id_F(u) + w_F(u) - 1$.*

For any $v \in V(F)$ and any e incident to v, the edge e receives a *port-number* $p_F(e,v)$ at v as follows. Set $p_F(e,v) = 0$ if $v \neq r$ and e leads to the parent of v in F, i.e., the edge e is the first edge on the path from v to r. Otherwise, let $(u_1,\cdots,u_d)$ be the children of v (where $d = d_F(v)$ if $v = r$ and $d = d_F(v) - 1$ otherwise) ordered by their weight, i.e., such that $w_F(u_1) \geq \cdots \geq w_F(u_d)$. Then, let $p_F(\{u_i,v\},v) = i$, for any $i \leq d$. Finally, each vertex $v \in V(F)$ is assigned a routing table $RT_F(v)$ and an address $\ell_F(v)$ of size $O(\log n)$ bits allowing a shortest path routing in trees (see details in [28]).

4.2.2 Our data structures.

Let G be a graph with the k-good tree-decomposition $(T = (I,M), \{X_i | i \in I\})$. Let $r \in V(G)$. Let F be a Breadth-First-Search(BFS) tree of G rooted at r. Let T be rooted in $b \in I$ such that $r \in X_b$.

We use (some of) the data structures of [28] for both trees F and T. More precisely, for any $v \in V(G)$, let $Id_F(v), w_F(v), \ell_F(v)$ and $RT_F(v)$ be defined as above for the BFS-tree F. Moreover, we add $d_F(v)$ to store the degree of v in the tree F . Set $p_{e,v} = p_F(e,v)$ for edges that belong to F defined as above, the ports $> d_F(v)$ will be

assigned to edges that do not belong to F. Knowing $d_F(v)$, the ports that correspond to edges in F can be easily distinguished from ports assigned to edges in $G \setminus E(F) \equiv \overline{F}$.

For any $v \in V(G)$, let $(u_1, \cdots, u_d) = N_{\overline{F}}(v)$ be the neighborhood of v in $\overline{F}$ ordered such that $Id_F(u_1) < \cdots < Id_F(u_d)$. We assign $p_{e_i,v} = d_F(v) + i$, where $e_i = \{v, u_i\}$, for each u_i in this order. This ordering will allow to decide whether one of the vertices in $N_{\overline{F}}(v)$ is an ancestor of a given vertex t in time $O(\log \Delta)$ by binary search.

For any $i \in I$, let $Id_T(i)$ and $w_T(i)$ be defined for the tree T as above. For any $v \in V(G)$, let $B_v \in I$ be the bag of T containing v which is closest to the root b of T. To simplify the notations, we set $Id_T(v) = Id_T(B_v)$ and $w_T(v) = w_T(B_v)$. These structures will be used to decide "where" we are in the tree-decomposition when the message reaches $v \in V(G)$.

Finally, for any $i \in I$, let $P_i = (v_1, \cdots, v_\ell)$ be the backbone of B_i with $\ell \leq k-1$ (recall that we consider a k-good tree decomposition). Let $(e_1, \cdots, e_{\ell-1})$ be the set of edges of P_i in order. Set $Backbone_i = (p_{e_1,v_1}, p_{e_1,v_2}, p_{e_2,v_2}, \cdots, p_{e_{\ell-1},v_\ell})$. For any $v \in V(G)$ such that $Id_T(v) = i \in I$, if $v = v_j \in P_i$, then $back(v) = (\emptyset, j)$ and if $v \notin P_i$, let $back(v) = (p_{e_v}, j)$ where $e = \{v, v_j\}$ and v_j ($j \leq \ell$) is the neighbor of v in P_i with j minimum. This information will be used to cross a bag (using its backbone) of the tree-decomposition.

Now, for every $v \in V(G)$, we define the address $name(v) = \langle \ell_F(v), Id_T(v) \rangle$. Note that, in particular, $\ell_F(v)$ contains $Id_F(v)$. We also define the routing table of v as $Table(v) = \langle RT_F(v), d_F(v), w_T(v), Backbone(v), back(v) \rangle$, where $Backbone(v) = Backbone_i$ for $i = B_v$, i.e. the backbone of the bag containing v and closest to the root of T.

Next table summarizes all these data structures.

	notation	description
$name(v)$	$\ell_F(v)$	the address of v in tree F [28]
	$Id_T(v)$	the identifier of the highest bag B_v containing v in T
$Table(v)$	$RT_F(v)$	the routing table used of v for routing in F [28]
	$d_F(v)$	the degree of v in F
	$w_T(v)$	the weight of the subtree of T rooted in B_v
	$Backbone(v)$	information to navigate in the backbone of B_v
	$back(v)$	information to reach the backbone of B_v from v

Clearly, $name(v)$ has size $O(\log n)$ bits and $Table(v)$ has size $O(k \cdot \log \Delta + \log n)$ bits. Moreover, any edge e incident to v receives a port-number $p_{e,v}$ of size $O(\log \Delta)$ bits.

4.3 Routing algorithm in k-good tree-decomposable graphs

Let us consider a message that must be sent to some destination $t \in V(G)$. Initially, the header of the message contains $name(t)$, the three counters $pos, cnt, cnt' = -1$, the bit $start = 0$ and the memory $path = \emptyset$, which stores the backbone of the bag containing an ancestor (in F) of the destination vertex of the message. Let $v \in V(G)$ be the current vertex where the message stands. First, using $Id_F(t)$ in $name(t)$, $Id_F(v)$ in $name(v)$ and $w_F(v)$ in $RT_F(v) \in Table(v)$, it is possible by using Lemma 1 to decide in constant time if v is an ancestor of t in F. Similarly, using $Id_T(t)$ in $name(t)$, $Id_T(v)$ in $name(v)$ and $w_T(v)$ in $Table(v)$, it is possible to decide if the highest bag B_v containing v is an ancestor of B_t in T. There are several cases to be considered.

- If v is an ancestor of t in F, then using the protocol of [28] the message is passed to the child w of v that is an ancestor of t in F towards t. Recursively, the message arrives at t following a shortest path in G, since F is a BFS-tree.
- Else, if $path = \emptyset$, then
  - if neither B_v is an ancestor of B_t in T nor $B_t = B_v$, then the message follows the edge leading to the parent of v in F, i.e., the edge with port-number $p_{e,v} = 0$. Note that the message will eventually reach a vertex w that either is an ancestor of t in F or B_w is an ancestor of B_t in T, since the message follows a shortest path to the root r of F and B_r is the ancestor of any bag in T.
  - Else, an ancestor of t belongs to B_v since either $B_v = B_t$, or B_v is an ancestor of B_t. (This is because T is a tree-decomposition, B_v has to contain a vertex on the shortest path from t to r in F.) Now the goal is to explore the bag B_v using its backbone $P = (v_1, \cdots, v_\ell)$ ($\ell < k$), until the message finds an ancestor of t in F.

    In this case we put the message on the backbone, and then explore the backbone using $Backbone(v)$ copied in $path$ in the header of the message. Using $back(v) = (p, j) \in Table(v)$, pos is set to j. If $p = \emptyset$ then the message is already on the backbone. Otherwise, the message is sent over the port p. Recall that by the definition of $back(v)$, port p leads to $v_j \in P$. The idea is to explore the neighborhoods of vertices on the backbone, starting from v_1. Note that in what follows $path \neq \emptyset$ and $pos \neq -1$.

- Else, if $start = 0$ (This is the case initially), then the message is at $v = v_j \in P$ and pos indicates the value of j. Moreover, in the field $path$ of the header, there are the port-numbers allowing to follow P. If $pos > 1$ then $pos = j-1$ is set and the message follows the corresponding port-number $p_{e_{j-1},v_j} \in Backbone(v_j)$ to reach v_{j-1}. Otherwise, $start$ is set to 1, $cnt = d_F(v_1)$ and $cnt' = d_G(v_1)+1$.
- Else, if $start = 1$, then the exploration of a bag containing an ancestor of t (or t itself) has begun. The key point is that any ancestor w of t in F satisfies that $Id_F(w) \leq Id_F(t) \leq Id_F(w) + w_F(w) - 1$ by Lemma 1. Using this property, for each vertex v_j of the backbone $P = (v_1, \cdots, v_\ell)$, the message visits v_j first. If v_j is an ancestor of t or $v_j = t$ then we are in the first case; otherwise the message is sent to the parent of v_j in F. If v_j's parent is an ancestor of t (or t itself) then we are in the first case; otherwise we explore $N_{\overline{F}}(v_j)$ by binary search. Notice that the other neighbors of v_j are its descendants in F, so if t has an ancestor among them, then v_j also is an ancestor of t.
  - If $cnt = cnt' - 1$, the neighborhood of the current vertex $v = v_j$, where $j = pos$, has already been explored and no ancestor of t has been found. In that case, using $path$, the message goes to v_{j+1} the next vertex in the backbone. Then pos is set to $j+1$.
  - Otherwise, let $pn = \lfloor \frac{cnt'+cnt}{2} \rfloor$. The message takes port-number pn from v towards vertex w. If w is an ancestor of t, we go to the first case of the algorithm. Otherwise, the message goes back to $v = v_j$. This is possible since the vertex w knows the port over which the message arrives. Moreover, if $Id_F(t) > Id_F(w) + w_F(w) - 1$, then cnt is set to pn and cnt' is set to pn otherwise.

The fact that the message eventually reaches its destination follows from the above description. Moreover, the computation of the next hop and the modification of the header clearly take time $O(1)$.

4.4 Performance of our routing scheme

In this subsection, we give an upper bound on the stretch of the routing scheme described in previous section.

Lemma 2 *Our routing scheme has stretch* $\leq 2k(\lceil \log \Delta \rceil + \frac{5}{2}) - 2\lceil \log \Delta \rceil - 4$.

Proof Let s be the source and t be the destination. Recall the main idea of the algorithm: We route along the path from s to r in tree F until we arrive a vertex x, whose bag B_x is an ancestor of t's bag B_t in tree T. Then applying binary search algorithm, we search in the bag B_x for a vertex y, which is an ancestor of t in tree F. In the end, we route from y to t in tree F.

Because F is a BFS tree and x is an ancestor of s in F, the length of the path followed by the message from s to x is $d(s,x)$, the distance between s and x in G. Similarly, because y is an ancestor of t in F, the length of the path followed by the message from y to t is $d(y,t)$. Let $track(x,y)$ be the length of the path followed by the message in B_x from x to y. Therefore, the length of the path followed by the message from s to t is $d(s,x) + track(x,y) + d(y,d)$.

From the binary search algorithm, for any vertex of the backbone, the message visits at most $\lceil \log \Delta \rceil$ neighbors and this causes a path of length $2\lceil \log \Delta \rceil$. There are at most $k-1$ vertices on the backbone of the bag B_x. The worst case occurs when x is the neighbor of the last vertex of the backbone v_l, for $l \leq k-1$, then the message goes to the first vertex of the backbone, v_1, while y is a neighbor of v_l. After arriving at x, the message goes to v_1, i.e., visits $l \leq k-1$ vertices, then it visits $\lceil \log \Delta \rceil$ neighbors of each of the $l \leq k-1$ vertices of the backbone and y is the last vertex visited. Therefore, $track(u,a) \leq 2k(\lceil \log \Delta \rceil + 1) - 2\lceil \log \Delta \rceil - 4$. Then it is sufficient to prove $d(s,x) + d(y,t) \leq d(s,t) + 3k$.

If B_s is an ancestor of B_t, then $x = s$ and $d(s,x) = 0$. Moreover, if $B_t = B_x$, $d(y,t) = 0$. Otherwise, let B be the nearest common ancestor of B_s and B_t in the tree-decomposition T. Let Q be a shortest path between s and t. Because the set of vertices in B separates s from t in G, let x' be the first vertex of Q in B and let y' the last vertex of Q in B. Let $Q = Q_1 \cup Q_2 \cup Q_3$ where Q_1 is the subpath of Q from s to x', Q_2 is the subpath of Q from x' to y' and Q_3 is the subpath of Q from y' to t. Note that because each bag has diameter at most k, $d(x',y') \leq k$.

We first show that $x \in B$. If $B_x = B$, it is trivially the case. Let P_x be the path followed from s to x. Since B_x is an ancestor of B, B separates s from x. Therefore, $P_x \cap B \neq \emptyset$. Let h be the first vertex of P_x in B. Since $h \in B$, the highest bag containing h is an common ancestor of B_s and B_t. Therefore, when arriving at h, the message must explore B_h. Hence, we have $h = x \in B$.

Finally, since $x \in B$, $d(x,x') \leq k$. Moreover, $y \in B_x$ therefore $d(y,x) \leq k$. Thus, $d(y,y') \leq d(y,x) + d(x,x') + |Q_2| \leq 2k + |Q_2|$. Finally, $d(s,x) \leq d(s,x') + d(x',x) \leq k + |Q_1|$ and $d(y,t) \leq d(y,y') + d(y',t) \leq 2k + |Q_2| + |Q_3|$. Therefore, $d(s,x) + d(y,t) \leq |Q_1| + |Q_2| + |Q_3| + 3k \leq |Q| + 3k = d(s,t) + 3k$. □

5 Conclusion and Further Work

Inspired by the study of cops and robber games on k-chordal graphs, we get a polynomial-time algorithm that, given a graph G and $k \geq 3$, either returns an induced cycle larger than k in G, or computes a k-good tree decomposition of G. A graph with a k-good tree decomposition is proved to have bounded ($O(k)$) tree-length and hyperbolicity; also its treewidth is bounded by $O(k-1)(\Delta-1)+2$, where Δ is the maximum degree of the graph. Furthermore, a k-good tree decomposition is used to design a compact routing scheme with routing tables, addresses and headers of size $O(k \log \Delta + \log n)$ bits and achieving an additive stretch of $O(k \log \Delta)$. It would be interesting to reduce the $O(k \cdot \log \Delta)$ stretch due to the dichotomic search phase of our routing scheme.

Any k-chordal graph admits a k-good tree decomposition, so it has treewidth at most $O(k-1)(\Delta-1)+2$. A clique of size $\Delta+1$ is a (3-)chordal graph with treewidth Δ. Then the bound $O(k-1)(\Delta-1)+2$ is tight up to a constant ratio 2. For $k > 3$, it would be interesting to find a better bound or to prove the tightness.

A natural problem is to find the minimum k for a given graph G such that G has a k-good tree decomposition. The complexity of this problem is still open even for $k = 2$. It would also be interesting to use the k-good tree decomposition to solve other combinatorial problems, e.g. the (connected) dominating set problem. Another interesting topic concerns the computation of tree-decompositions not trying to minimize the sizes of the bags but imposing some specific algorithmically useful structure to the bags.

References

1. Abraham, I., Gavoille, C.: Object location using path separators. In: Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing, PODC '06, pp. 188–197. ACM (2006)
2. Abraham, I., Gavoille, C., Goldberg, A., Malkhi, D.: Routing in networks with low doubling dimension. In: Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on, pp. 75–75 (2006)
3. Abraham, I., Gavoille, C., Malkhi, D.: Compact routing for graphs excluding a fixed minor. In: Distributed Computing, *Lecture Notes in Computer Science*, vol. 3724, pp. 442–456. Springer Berlin Heidelberg (2005)
4. Abraham, I., Gavoille, C., Malkhi, D.: On space-stretch trade-offs: Lower bounds. In: Proceedings of the Eighteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '06, pp. 207–216. ACM (2006)
5. Abraham, I., Gavoille, C., Malkhi, D., Nisan, N., Thorup, M.: Compact name-independent routing with minimum stretch. ACM Trans. Algorithms **4**(3), 37:1–37:12 (2008)
6. Abraham, I., Malkhi, D.: Name independent routing for growth bounded networks. In: Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '05, pp. 49–55. ACM (2005)
7. Aigner, M., Fromme, M.: A game of cops and robbers. Discrete Applied Mathematics **8**(1), 1–12 (1984)
8. Aksoy, A.G., Jin, S.: The apple doesn't fall far from the (metric) tree: The equivalence of definitions (2013). URL http://arxiv.org/abs/1306.6092
9. Andreae, T.: On a pursuit game played on graphs for which a minor is excluded. Journal of Combinatorial Theory, Series B **41**(1), 37–47 (1986)
10. Arnborg, S., Corneil, D., Proskurowski, A.: Complexity of finding embeddings in a k-tree. SIAM Journal on Algebraic Discrete Methods **8**(2), 277–284 (1987)
11. Bandelt, H., Chepoi, V.: 1-hyperbolic graphs. SIAM Journal on Discrete Mathematics **16**(2), 323–334 (2003)
12. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. **25**(6), 1305–1317 (1996)
13. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theoretical Computer Science **209**(1-2), 1–45 (1998)
14. Bodlaender, H.L., Kloks, T.: Efficient and constructive algorithms for the pathwidth and treewidth of graphs. Journal of Algorithms **21**(2), 358–402 (1996)
15. Bodlaender, H.L., Mhring, R.: The pathwidth and treewidth of cographs. SIAM Journal on Discrete Mathematics **6**(2), 181–188 (1993)
16. Bodlaender, H.L., Thilikos, D.M.: Treewidth for graphs with small chordality. Discrete Applied Mathematics **79**(1-3), 45–61 (1997)
17. Bonato, A., Nowakovski, R.: The game of Cops and Robber on Graphs. American Math. Soc. (2011)
18. Cai, L., Chan, S.M., Chan, S.O.: Random separation: a new method for solving fixed-cardinality optimization problems. In: Proceedings 2nd International Workshop on Parameterized and Exact Computation, IWPEC 2006, pp. 239–250. Springer-Verlag (2006)
19. Chalopin, J., Chepoi, V., Nisse, N., Vaxs, Y.: Cop and robber games when the robber can hide and ride. SIAM Journal on Discrete Mathematics **25**(1), 333–359 (2011)
20. Chen, W., Sommer, C., Teng, S.H., Wang, Y.: Compact routing in power-law graphs. In: Distributed Computing, *Lecture Notes in Computer Science*, vol. 5805, pp. 379–391. Springer Berlin Heidelberg (2009)
21. Chen, Y., Flum, J.: On parameterized path and chordless path problems. In: Computational Complexity, 2007. CCC '07. Twenty-Second Annual IEEE Conference on, pp. 250–263 (2007)
22. Chepoi, V., Dragan, F., Estellon, B., Habib, M., Vaxès, Y.: Diameters, centers, and approximating trees of delta-hyperbolicgeodesic spaces and graphs. In: Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry, SCG '08, pp. 59–68. ACM (2008)
23. Chepoi, V., Dragan, F., Estellon, B., Habib, M., Vaxs, Y., Xiang, Y.: Additive spanners and distance and routing labeling schemes for hyperbolic graphs. Algorithmica **62**(3-4), 713–732 (2012)
24. Clarke, N.E., Nowakowski, R.J.: Tandem-win graphs. Discrete Mathematics **299**(1-3), 56–64 (2005)
25. Courcelle, B., Mosbah, M.: Monadic second-order evaluations on tree-decomposable graphs. Theoretical Computer Science **109**(1-2), 49–82 (1993)
26. Dourisboure, Y.: Compact routing schemes for generalised chordal graphs. J. of Graph Alg. and App **9**(2), 277–297 (2005)
27. Dourisboure, Y., Gavoille, C.: Tree-decompositions with bags of small diameter. Discrete Mathematics **307**(16), 2008–2029 (2007)
28. Fraigniaud, P., Gavoille, C.: Routing in trees. In: Automata, Languages and Programming, *Lecture Notes in Computer Science*, vol. 2076, pp. 757–772. Springer Berlin Heidelberg (2001)

29. Frankl, P.: Cops and robbers in graphs with large girth and cayley graphs. Discrete Applied Mathematics **17**(3), 301–305 (1987)
30. Gromov, M.: Hyperbolic groups. Essays in Group Theory **8**, 75–263 (1987)
31. Hopcroft, J., Tarjan, R.: Algorithm 447: Efficient algorithms for graph manipulation. Commun. ACM **16**(6), 372–378 (1973)
32. Kloks, T., Kratsch, D.: Treewidth of chordal bipartite graphs. Journal of Algorithms **19**(2), 266–281 (1995)
33. Kobayashi, Y., Kawarabayashi, K.i.: Algorithms for finding an induced cycle in planar graphs and bounded genus graphs. In: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '09, pp. 1146–1155 (2009)
34. Konjevod, G., Richa, A.W., Xia, D.: Optimal-stretch name-independent compact routing in doubling metrics. In: Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing, PODC '06, pp. 198–207. ACM (2006)
35. Kosowski, A., Li, B., Nisse, N., Suchan, K.: k-chordal graphs: From cops and robber to compact routing via treewidth. In: Automata, Languages, and Programming, *Lecture Notes in Computer Science*, vol. 7392, pp. 610–622. Springer Berlin Heidelberg (2012)
36. Krioukov, D., Papadopoulos, F., Boguñá, M., Vahdat, A.: Greedy forwarding in scale-free networks embedded in hyperbolic metric spaces. SIGMETRICS Performance Evaluation Review **37**(2), 15–17 (2009)
37. Lokshtanov, D.: On the complexity of computing treelength. Discrete Appl. Math. **158**(7), 820–827 (2010)
38. Lu, L., Peng, X.: On meyniel's conjecture of the cop number. Journal of Graph Theory **71**(2), 192–205 (2012)
39. de Montgolfier, F., Soto, M., Viennot, L.: Treewidth and hyperbolicity of the internet. In: Network Computing and Applications (NCA), 2011 10th IEEE International Symposium on, pp. 25–32 (2011)
40. Nisse, N., Rapaport, I., Suchan, K.: Distributed computing of efficient routing schemes in generalized chordal graphs. Theoretical Computer Science **444**(0), 17–27 (2012)
41. Nowakowski, R., Winkler, P.: Vertex-to-vertex pursuit in a graph. Discrete Mathematics **43**(2-3), 235–239 (1983)
42. Opsahl, T., Panzarasa, P.: Clustering in weighted networks. Social Networks **31**(2), 155–163 (2009)
43. Quilliot, A.: A short note about pursuit games played on a graph with a given genus. Journal of Combinatorial Theory, Series B **38**(1), 89–92 (1985)
44. Robertson, N., Seymour, P.: Graph minors. iii. planar tree-width. Journal of Combinatorial Theory, Series B **36**(1), 49–64 (1984)
45. Schröder, B.S.W.: The copnumber of a graph is bounded by $\lfloor \frac{3}{2} genus(G) \rfloor + 3$. In: Categorical Perspectives, Trends in Mathematics, pp. 243–263. Birkhuser Boston (2001)
46. Scott, A., Sudakov, B.: A bound for the cops and robbers problem. SIAM Journal on Discrete Mathematics **25**(3), 1438–1442 (2011)
47. Sundaram, R., Singh, K., Rangan, C.: Treewidth of circular-arc graphs. SIAM Journal on Discrete Mathematics **7**(4), 647–655 (1994)
48. Uehara, R.: Tractable and intractable problems on generalized chordal graphs. Tech. Rep. COMP98-83, IEICE (1999)
49. Watts, D.J., Strogatz, S.: Collective dynamics of 'small-world' networks. Nature **393**(6684), 440–442 (1998)
50. Wu, Y., Zhang, C.: Hyperbolicity and chordality of a graph. Electronic Journal of Combinatorics **18**(1) (2011)