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[1], B. Li[2,3], N. Nisse[2], and K. Suchan[4,5]

[1] CEPAGE, INRIA, LaBRI, Talence, France
[2] MASCOTTE, INRIA, I3S(CNRS/UNS), Sophia Antipolis, France
[3] CAS & AAMS, Beijing, China
[4] FIC, Universidad Adolfo Ibáñez, Santiago, Chile
[5] WMS, AGH - University of Science and Technology, Krakow, Poland

Abstract. *Cops and robber games* 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 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 quadratic 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-node graph admitting such a tree-decomposition, we propose a *compact routing scheme* using routing tables, addresses and headers of size $O(\log n)$ 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.

Keyword: 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. So, 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 *hyperbolicity* of a graph reflects how the metric (distances) of the graph is close to

[⋆] Partially supported by programs Fondap and Basal-CMM, Anillo ACT88 and Fondecyt 11090390 (K.S.), FP7 STREP EULER (N.N.).

the metric of a tree. A graph has hyperbolicity $\leq \delta$ if, for any $u, v, w \in V(G)$ and for 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 $P_{vw} \cup P_{uw}$ [Gro87]. 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 [KPBV09]). Indeed, Internet-type networks have a so-called high clustering coefficient (see e.g. [WS98]), leading to the existence of very few long chordless cycles, whereas their low (logarithmic) diameter implies a small hyperbolicity [dMSV11].

Another way to study tree-likeness of graphs is by *tree-decompositions*. Introduced by Robertson and Seymour, 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 [Bod98]. The nodes of the decomposition tree are called *bags*, and the size of a bag is the number of vertices assigned to it (assigned subtrees intersect the bag). The *width* of a tree-decomposition is the maximum size over its bags, 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 graph with bounded treewidth [CM93]. In particular, there are linear-time algorithms to compute an optimal tree-decomposition of a graph with bounded treewidth [BK96]. 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 [dMSV11]. 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 [DG07]. 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*. 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 [BN11].

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 [Dou05]. The objective of compact routing is to provide a scheme for finding a path from a sender node to a known destination, taking routing decisions for the packet at every step using only very limited information stored at each node. 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 nodes) in graphs admitting with 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 node.

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 $O(m^2)$ algorithm that, given a m-edge 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 $\leq k-1$. That is, 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 in the path or adjacent to some vertex of the path. If G, with maximum degree Δ, admits such a decomposition, then G has treewidth at most $(k-1)(\Delta-1)+2$, tree-length at most k and 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 [BT97]. 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.

Our tree-decomposition may be used efficiently for solving problems using dynamic programming in graphs of small chordality and small maximum degree. In this paper, we focus on different application. We present a compact routing scheme that uses our tree-decomposition and that achieves an additive stretch $\leq 2k(\lceil \log \Delta \rceil + \frac{5}{2}) - 5$ with routing tables, addresses and message headers of $O(\max\{k \cdot \log \Delta, \log n\})$ 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 the tree-likeness of a graph. Some papers consider relations between them [WZ11]. In particular, the hyperbolicity of a k-chordal graph is at most k, but the difference may be arbitrary large (take a $3 \times n$-grid). The seminal definition of Gromov hyperbolicity is the following. A graph G is d-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 $2d$ [Gro87]. This definition is equivalent to the one we use in this paper, using so called *thin triangles*, up to a constant ratio. No algorithm better than the $O(n^4)$-brute force algorithm (testing all 4-tuples in G) is known to compute Gromov hyperbolicity of n-node graphs. The problem of computing the chordality of a graph G is NP-complete since it may be related to computing a longest cycle in the graph obtained from G after subdividing all edges once. Finding the longest induced path is $W[2]$-complete [CF07] and the problem is Fixed Parameter Tractable in planar graphs [KK09].

Treewidth. It is NP-complete to decide whether the treewidth of a graph G is at most k [ACP87]. The treewidth problem is polynomially solvable in chordal graphs, cographs, circular arc graphs, chordal bipartite graphs, etc. [Bod98]. Bodlaender and Thilikos proved that the treewidth of a k-chordal graph 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 [BT97]. They also proved that the treewidth problem is NP-complete for graphs with small maximum degree [BT97].

Compact routing. In [AGM$^+$08], a universal name-independent routing scheme with stretch linear in k and $n^{1/k}polylog(n)$ space is provided. 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 $k/4$ [AGD06]. 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 [FG01], bounded doubling dimension [AGGM06], excluding a fixed graph as a minor [AG06], 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 [Dou05]. 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 [NSR12].

2 A detour through Cops and Robber games

Let us first fix some notations. G denotes a simple connected undirected graph with vertex set V and edge set E. $n = |V|$ is the *order* of G and $m = |E|$ is the *size* of G. $V(H)$ and $E(H)$ denotes the vertex and edge set of H, respectively. The set of vertices adjacent to $v \in V$ is denoted $N(v)$ and called *open neighborhood* of v. $N[v] = N(v) \cup \{v\}$ is the *closed neighborhood* of v. We extend this notation to write $N[U] = \cup\{N[u] \mid u \in U\}$ and $N(U) = N[U] \setminus U$. $d_G(v) = |N_G(v)|$ is the *degree* of v, Δ denotes the maximum degree among the vertices of G. 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 $U \subset V$, $G[U]$ is the subgraph of G *induced* by 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 by (P, Q) the path present in the graph induced by $V(P) \cup V(Q)$ - to make descriptions more concise, we omit the detail of reversing P or Q if necessary.

Let us formally define a cops' strategy to capture a robber. Given a graph G, a player starts by placing $k \geq 1$ cops on some vertices in V, then a visible robber is placed on one vertex v in V. Alternately, the robber can move to a vertex x in $N(v)$, and the cop-player may move each cop along an edge. The robber is captured if, at some step, a cop occupies the same vertex. The *cop-number* of a graph G, denoted by $cn(G)$, is the fewest number of cops required to capture a robber in G (see the recent book [BN11]). Bounds on the cop-number have been provided for various graph classes and provided many nice structural results [BN11]. 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.*

Proof. Let $v \in V$ be any vertex and place all cops at it. 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 $N = \cup_{j \leq i} N[v_j]$, 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, while R is not empty, then so does S.

Now, there are two cases to be considered. If $N(v_1) \cap S \subseteq \cup_{1<j \leq i} N[v_j]$. 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 $N(v_i) \cap S \subseteq \cup_{1 \leq j < i} N[v_j]$, 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.

Hence, there is $u \in (N(v_1) \cap S) \setminus (\cup_{1<j \leq i} N[v_j])$ and $v \in (N(v_i) \cap S) \setminus (\cup_{1 \leq j < i} N[v_j])$. First, we show that this case may happen only if $i < k-1$. Indeed, otherwise, 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 $i+2$. Since G is k-chordal, this implies that $i+2 \leq k$. Then, one cop goes to $v = v_{i+1}$ 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 and then the robber is eventually captured. □

Note that previous Theorem somehow extends the model in [CN05] 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 case $k = 4$. Indeed, for any 4-chordal connected graph G, then $cn(G) \leq 2$. Due to lack of space, the proof is omitted and can be found in [KLNS12].

Theorem 1 relies on chordless paths P in G such that $N[P]$ is a separator of G, i.e., there exist vertices a and b of G such that all paths between a and b intersect $N[P]$. In 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-node 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 interesting structural properties. 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) $V = \cup_{i \in I} X_i$; (2) $\forall \{uv\} \in E$ 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 maximum diameter of the subgraphs induced by the bags. The *treewidth* denoted by $tw(G)$, resp., *tree-length* denoted by $tl(G)$, of a graph G is the minimum width, resp., ℓ-width, over all possible tree-decompositions of G [DG07].

Let $k \geq 2$. A k-caterpillar is 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-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-caterpillar.

Theorem 2. *There is a $O(m^2)$-algorithm that takes a graph G of size m and an integer $k \geq 3$ as inputs and: 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 $N_G[P]$. Obviously, it is true if $|V(G)| = 1$. Now we assume that it is true for any graph G with n' nodes, $1 \leq n' < n$, and we show it is true for n-node graphs.

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

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

Case 2. G' is 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, $j \leq r$, is an induced cycle in G. By the induction hypothesis, either there is an induced cycle $\mathcal{C}$ larger than k in G_j, then $\mathcal{C}$ 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, $j \leq r$, 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. G' is 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 length $i < k - 1$ by reversed 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 $V(P) \cup \{v_{i+1}\}$ induces a chordless path P' of order $i + 1$. By reverse 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, $u \notin V(P) \setminus \{v_1, v_i\}$) such that $N_G(u) \subseteq N_G[V(P) \setminus \{u\}]$. 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 $v_0 \in N_G(v_1)$ and $v_{i+1} \in N_G(v_i)$ 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 $\{v_{i+1}, u_2\} \in E(G)$. 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 reverse 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 $N[Q]$. 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, $u \notin V(P) \setminus \{v_1, v_i\}$) 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., w is a neighbor of v_1 or v_i but not both). Therefore, we apply the induction hypothesis (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 $N_G[V(P')] \supseteq N_G[V(P)]$.

Let us analyze the algorithm and its complexity. Let G be a m-edge n-node graph with maximum degree Δ. The algorithm proceeds in steps of $O(m)$ time, each time considering one vertex. We prove that at each step (but the first one), at least one edge will be *considered* and that all edges are considered at most once. This implies a time-complexity of $O(m^2)$ for the algorithm. Due to lack of space, the proof of the time-complexity is omitted and can be found in [KLNS12]. □

Due to lack of space, the proofs of the following consequences of Theorem 2 are omitted and can be found in [KLNS12].

Theorem 3. *Let G be a graph that admits a k-good tree-decomposition. Then $tw(G) \leq (k-1)(\Delta-1)+2$ where Δ is its maximum degree, $tl(G) \leq k$, and hyperbolicity at most $\lfloor \frac{3}{2}k \rfloor$.*

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 hyperbolicity at most $\lfloor \frac{3}{2}k \rfloor$.*

Corollary 2. *There is an algorithm that, given a m-edge graph G and $k \geq 3$, states that either G has chordality at least $k+1$ or G has 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-node graph G that admits a k-good tree-decomposition (including k-chordal graphs). here here

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 [FG01], we consider the *designer-port* model, which allows us to choose the permutation of ports (assign a label of $\log d_v$ bits to any edge incident to v in $V(G)$). Finally, to any node $v \in V(G)$, 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 $pos \in \{-1, 0, \cdots, k-1\}, cnt_d, cnt'_d \in \{-1, 0, \cdots, \Delta+1\}$, one bit *start* and some memory, called *path*, of size $O(k \cdot \log \Delta)$ bits. *start* and *path* change only once.

Following our routing scheme, a node 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 node 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 [FG01] together with the k-good tree-decomposition. Roughly, the scheme consists in following the paths in a BFS-tree F of G, using the scheme in [FG01], and uses 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 [FG01]. The remaining part of this Section is devoted to the proof of the next Theorem that summarizes the performances of our routing scheme.

Theorem 4. *For any n-node m-edge graph G with maximum degree Δ and with a k-good tree-decomposition, there is a labelled routing scheme $\mathcal{R}$ with the following properties. $\mathcal{R}$ 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 is computed in time $O(1)$ at each step of the routing. Finally, the additive stretch is $\leq 2k(\lceil \log \Delta \rceil + \frac{5}{2}) - 5$.*

4.2 Data structures

Routing in trees [FG01]. Since, we use the shortest path routing scheme proposed in [FG01] for trees, we start by recalling some of the data structures they use. 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. It is important to note that, 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. $p_F(e, v) = 0$ if $v \neq r$ and e leads to the parent of v in F, i.e., e is the first edge on the path from v to r. Otherwise, let $(u_1, \cdots, u_d)$ be the children of v ($d = d_F v$ 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 [FG01]).

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 [FG01] 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 . $p_{e,v} = p_F(e, v)$ for edges that belong to F are defined as above, the ports $> d_F(v)$ will be assigned to edges that do not belong to F. With $d_F(v)$ at hand, the ports that correspond to edges in F can be easily distinguished from ports assigned to edges in $\overline{F}$.

For any $i \in I$, let $Id_T(i)$ and $w_T(i)$ be defined as above for the tree T. For any $v \in V(G)$, let $B_v \in I$ be the bag of T containing v (i.e., $v \in X_{B_v}$) that 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 $v \in V(G)$.

Let $\overline{F} = G \setminus E(F)$. 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 node t in time $O(\log \Delta)$ by binary search.

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 we consider a k-good tree decomposition). Let $(e_1, \cdots, e_{\ell-1})$ be the set of edges of P_i in order. We 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 $Table(v) = \langle RT_F(v), w_T(v), Backbone(v), back(v) \rangle$.

Next table summarizes all these data structures.

	notation	description
$name(v)$	$\ell_F(v)$	the address of v in tree F [FG01]
	$Id_T(v)$	the identifier of the highest bag B_v containing v in T
	$RT_F(v)$	the routing table used of v for routing in F [FG01]
	$d_F(v)$	the degree of v in F
$Table(v)$	$w_T(v)$	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 $pos, cnt_d, cnt'_d = -1$, the bit $start = 0$ and the memory $path = \emptyset$. Let $v \in V(G)$ be the current node 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 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 [FG01] 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 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. Therefore, since 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 of it 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.
      The aim of this case is to 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 already is on the backbone. Otherwise, the message is sent over the port p. The idea is to explore the neighborhoods of vertices on the backbone, starting from v_1. Note that at in what follows $path \neq \emptyset$ and $pos \neq -1$.
- Else, if $start = 0$, 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 to reach v_{j-1}. Otherwise, *start* is set to 1, $cnt_d = d_F(v_1)$ and $cnt'_d = d_G(v_1) + 1$.
- Else, if $start = 1$, then the exploration of a bag containing an ancestor of t has begun. The key point is that any ancestor w of t in F is such that $Id_F(w) \leq Id_F(t) \leq Id_F(w) + w_F(w) - 1$. Using this property, for each vertex v_j of the backbone $P = (v_1, \cdots, v_\ell)$, the message visits v_j and its parent in F, and explores $N_{\overline{F}}(v_j)$ 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 $cnt_d = cnt'_d - 1$, the neighborhood of the current node $v = v_j$, 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. $pos = j + 1$ is set.
    - Otherwise, let $pn = \lfloor \frac{cnt'_d + cnt_d}{2} \rfloor$. 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 node knows the port over which the message arrives. Moreover, if $Id_F(t) > Id_F(w) + w_F(w) - 1$, then cnt_d is set to *pn* and cnt'_d 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 takes time $O(1)$. Due to lack of space, the proof of the stretch, described in next lemma, is omitted and can be found in [KLNS12].

Lemma 1. *Our routing scheme has stretch* $\leq 2k(\lceil \log \Delta \rceil + \frac{5}{2}) - 5$.

5 Conclusion and Further Work

It would be interesting to reduce the $O(k \cdot \log \Delta)$ stretch due to the dichotomic search phase of our routing scheme. Another interesting topic concerns the computation of tree-decompositions not trying to minimize the size of the bag but imposing some specific algorithmically useful structure.

References

[ACP87] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Discrete Methods*, 8:277–284, 1987.

[AG06] I. Abraham and C. Gavoille. Object location using path separators. In *PODC*, pages 188–197. ACM, 2006.

[AGD06] I. Abraham, C. Gavoille, and D.Malkhi. On space-stretch trade-offs: Lower bounds. In *SPAA*, pages 217–224, 2006.

[AGGM06] I. Abraham, C. Gavoille, A.V. Goldberg, and D. Malkhi. Routing in networks with low doubling dimension. In *ICDCS*, page 75, 2006.

[AGM$^+$08] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup. Compact name-independent routing with minimum stretch. *ACM T. Alg.*, 4(3), 2008.

[BK96] H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21(2):358–402, 1996.

[BN11] A. Bonato and R. Nowakovski. *The game of Cops and Robber on Graphs*. American Math. Soc., 2011.

[Bod98] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998.

[BT97] H. L. Bodlaender and D. M. Thilikos. Treewidth for graphs with small chordality. *Disc. Ap. Maths*, 79(1-3):45–61, 1997.

[CF07] Y. Chen and J. Flum. On parameterized path and chordless path problems. In *CCC*, pages 250–263, 2007.

[CM93] B. Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *TCS*, 109:49–82, 1993.

[CN05] N. E. Clarke and R. J. Nowakowski. Tandem-win graphs. *Discrete Mathematics*, 299(1-3):56–64, 2005.

[DG07] Y. Dourisboure and C. Gavoille. Tree-decompositions with bags of small diameter. *Discrete Mathematics*, 307(16):2008–2029, 2007.

[dMSV11] F. de Montgolfier, M. Soto, and L. Viennot. Treewidth and hyperbolicity of the internet. In *NCA*, pages 25–32. IEEE Comp. Soc., 2011.

[Dou05] Y. Dourisboure. Compact routing schemes for generalised chordal graphs. *J. of Graph Alg. and App*, 9(2):277–297, 2005.

[FG01] P. Fraigniaud and C. Gavoille. Routing in trees. In *28th Int. Col. on Aut., Lang. and Prog. (ICALP)*, pages 757–772, 2001.

[Gro87] M. Gromov. Hyperbolic groups. *Essays in Group Theory*, 8:75–263, 1987.

[KK09] Y. Kobayashi and K. Kawarabayashi. Algorithms for finding an induced cycle in planar graphs and bounded genus graphs. In *20th Annual ACM-SIAM Symp. on Discrete Alg. (SODA)*, pages 1146–1155. SIAM, 2009.

[KLNS12] A. Kosowski, B. Li, N. Nisse, and K. Suchan. k-chordal graphs: from cops and robber to compact routing via treewidth, 2012. Report, INRIA-RR7888, http://www-sop.inria.fr/members/Bi.Li/RR-7888.pdf.

[KPBV09] D. V. Krioukov, F. Papadopoulos, M. Boguñá, and A. Vahdat. Greedy forwarding in scale-free networks embedded in hyperbolic metric spaces. *SIGMETRICS Performance Evaluation Review*, 37(2):15–17, 2009.

[NSR12] N. Nisse, K. Suchan, and I. Rapaport. Distributed computing of efficient routing schemes in generalized chordal graphs. *TCS*, 2012. To appear.

[WS98] D. J. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[WZ11] Y. Wu and C. Zhang. Hyperbolicity and chordality of a graph. *Electr. J. Comb.*, 18(1), 2011.