Compact routing scheme with low deviation for chordal graphs

Yon Dourisboure* Cyril Gavoille†

2nd December 2005

Abstract

In this paper[1], we present a compact routing scheme designed for chordal graphs, i.e. graphs containing no induced cycle of length greater than 3. This routing scheme, based on the notion of tree-decomposition introduced in [17], uses addresses and local tables of $O(\log^3 n/\log\log n)$ bits per vertex and insures that the length of the route between all pairs of vertices never exceed their distance plus 2.

In contrast, we prove that any shortest path routing scheme on chordal graphs requires $\Omega(n\log n)$-bit addresses or local routing tables.

Keywords: Graph algorithms, distributed computing, compact routing, chordal graphs, tree-decomposition, hierarchical-tree.

1 Introduction

Delivering messages between pairs of processors is a basic activity of any distributed communication network. This task is performed using a routing scheme, which is a mechanism for routing messages in the network. The routing mechanism can be invoked at any origin node and be required to deliver a message to any destination node.

It is naturally desirable to route messages along paths that are as short as possible. Routing scheme design is a well-studied subject. The efficiency of a routing scheme is measured in terms of its *deviation* or in terms of its *stretch*. The deviation of a routing scheme is d if it guarantees that the length of the route between all pairs of vertices never exceeds their distance plus d. Similarly, the stretch is s if lengths of routes are bounded by distances multiplied by s. A straightforward approach to achieving the goal of guaranteeing optimal routes is to store a *complete routing table* in each node u in the network, specifying for each destination v the first edge (or an identifier of that edge, indicating the output port) along some shortest path from u to v. However, this approach may be too expensive for large systems since it requires $O(n\log\deg(u))$ memory bits for a node u of degree $\deg(u)$ in an n-node network. Thus, an important problem in large-scale communication networks is the design of routing schemes that produce short routes and have relatively low *memory requirements*. It was shown in a series of papers (see, e.g., [1, 2, 3, 4, 16, 18]) that there is a trade-off between the memory requirements of a routing scheme and the worst-case stretch factor it guarantees.

*IIT - CNR ; yon.dourisboure@iit.cnr.it
†LaBRI, Bordeaux 1 University; gavoille@labri.fr
[1]A preliminary version can be found in [9]

The routing problem can be presented as requiring to assign two kinds of labels to every node of a graph. The first is the *address* of the node, whereas the second label is a data structure called the *local routing table*. The labels are assigned in such a way that at every source node u and given the address of any destination node v, one can decide the output port of an edge outgoing from u that leads to v. The decision must be taken locally at u, based solely on the two labels of u and with the address label of v. In order to allow each intermediate node to proceed similarly, a *header* is attached to the message to v. This header consists either of the destination label, or of a new label created by the current node.

In this paper we investigate *chordal graphs*, namely the class of graphs containing no induced cycles of length greater than 3. For example, trees and cliques are chordal graphs. If we insist on shortest path (i.e., optimal stretch $s = 1$ or deviation $d = 0$), no strategy better than complete routing tables is known for chordal graphs. Nevertheless, every chordal graph with maximum clique k has a shortest path routing scheme with local tables of $O((2^k + \deg(u))\log n)$ bits for every vertex u and using addresses $\in [1, n]$. This result, obtained in [7], is derived from an interval routing scheme for k-trees, proposed in [15]. For such chordal graphs, in [7] they constructed a routing scheme of deviation 2 with $O(\log n)$-bit addresses and local routing tables of $O(k \log n)$ bits per node. For general chordal graphs, Peleg and Upfal in [16] have constructed a routing scheme of stretch 3 using $O(n \log^2 n)$ bits in total for local routing tables[2] and addresses of $O(\log^2 n)$ bits.

In this paper we consider *loop-free* routing schemes, i.e., a message cannot use several times a same node. Such routing schemes have to follow shortest paths in trees. We consider also that the port numbers of every vertex are fixed by an adversary. Thus any routing strategy of finite deviation d on chordal graphs requires addresses and local routing tables of $\Omega(\log^2 n/\log\log n)$ bits in the worst-case since this memory bound is necessary and sufficient to shortest path routing scheme on trees [12].

In this paper we present a compact routing scheme designed for chordal graphs. Polynomially constructible, it uses $O(\log^3 n/\log\log n)$-bit addresses and routing tables while guaranting a deviation of 2. Headers are of $O(\log^2 n/\log\log n)$ bits, and each routing decision takes a constant time. This scheme is based on the notion of *tree-decomposition* of Robertson and Seymour [17], on the notion of *hiearchical-tree*, and on the tree-routing of [11, 18]. However, our solution does not reduce to routing along a single tree, since it is known that there are chordal graphs without any constant stretched spanning tree [14].

Since the publication of the preliminary version of this work [9], two new routing schemes for chordal graphs have been presented. Inspired by the construction of [9], Dragan et al. showed in [10] that every chordal graph admits a $O(\log n)$ *collective-tree spanners* of deviation 2, i.e., admits collection of $O(\log n)$ spanning trees such that for any two vertices, their distance in at least one tree of the collection does not exceed their distance in the graph plus 2. Then, superposing $O(\log n)$ tree-routing schemes of [11], they obtain a scheme with $O(\log^3 n/\log\log n)$-bit addresses. However, identifying the right tree increases the routing decision time to $\Omega(\log n)$. A second result has been proposed in [8]. Thanks to a new technique they reduce the label length to $O(\log^2 n)$, however raising up the deviation to 4. Moreover, this result is based on the fact that port numbers can be permuted in advance by the designer of the scheme (a.k.a. the *designer port model*). The scheme can be converted in the fixed port model, but the label length complexity comes back to $O(\log^3 n/\log\log n)$. Finally, within this label length complexity, the scheme we present here is the only one guaranting constant time routing decisions and a deviation of 2.

We also show that poly-log memory routing schemes on chordal graphs do not exist if we

[2] The average memory requirement is $O(\log^2 n)$ bits per node, but for some nodes, the local routing table is of $\Theta(n)$ bits. Moreover the route can use several times a same node.

insist on shortest paths. Namely, we prove that, in the fixed port model, routing in some chordal graphs along shortest paths requires $\Omega(n \log n)$ bits per vertex, which is an optimal bound.

2 Basic notions and notations

All graphs occurring in this paper are connected, finite, undirected and unweighted. Let $G = (V, E)$ be any graph, let u be any vertex of G and let X, Y be two subsets of V, then the distance in G between u and X, denoted $d_G(u, X)$ is: $d_G(u, X) = \min_{v \in X} d_G(u, v)$. The distance in G between X and Y is: $d_G(X, Y) = \min_{u \in X} d_G(u, Y)$.

A *shortest path spanning tree* T of a graph G is a rooted tree such that $V(T) = V(G)$, $E(T) \subseteq E(G)$ and such that for every vertex u, $d_T(u, r) = d_G(u, r)$, where r is the root of T.

In the following, we will use the standard notions of *parent, children, ancestor, descendant* and *depth* in trees. For simplicity we assume that a node is an ancestor of itself. The *nearest common ancestor* between two vertices u, v in a tree T is denoted by $\mathrm{nca}_T(u, v)$.

2.1 Routing scheme

As describe informally in introduction, a routing scheme is a preprocessing treatment that:

- associates with every outgoing edges e, of every vertex u, an integer, called *port number* and denoted by $\mathrm{port}(e)$, taken from $\{1, \ldots, \deg(u)\}$. Nevertheless, results presented in this paper hold for the fixed port model [11], i.e., this operation is done by an adversary, so cannot be optimized by the designer of the routing scheme.
- associates with every node u of the graph two labels: its *address* denoted by $\mathrm{address}(u)$, and a *local routing table* denoted by $\mathrm{table}(u)$.
- constructs an *initalisation procedure* that is executed by any initial sender u. Given $\mathrm{address}(u)$ and $\mathrm{address}(v)$, this function computes the header h_{uv}, a message of source u and destination v has to have.
- constructs a *routing function* that is executed by any vertex w ($w = u$ possible) that receives a message. Given $\mathrm{table}(w)$ and h_{uv}, this function returns the port number on which w has to send the message. Observe that in our routing scheme, once computed by the initial sender, headers never change, thus it is necessarily loop-free.

The efficiency of a routing scheme is evaluated in terms of:

- memory requirement: The number of bits needed for $\mathrm{address}(u)$, $\mathrm{table}(u)$ and h_{uv}, in the worst case.
- time requirement: the time[3] the initialisation procedure and the routing function need in the worst-case.
- length of routes: the difference between the length of the route and the distance in the graph in the worst-case. In this paper, it is measured in term of *deviation*: the deviation of a routing scheme is d if it guarantees that the length of the route between all pairs of vertices never exceeds their distance plus d.
- preprocessing time: the time the preprocessing treatment needs to construct all labels.

[3] We assume that the standard bitwise operations (like addition, xor, shift, etc.) on $O(\log n)$ bit words run in constant time.

2.2 Routing in trees

Our routing scheme needs to route in trees. To implement compact routing in trees we use the following result proposed in [11] (the lower bound is proved in [12]).

Lemma 1 (cf. [11, 12]) *In the fixed port model, every n-node tree T admits a shortest path compact routing scheme, constructible in polynomial time, without initalisation procedure, with a constant time routing function, and such that addresses, local tables and headers are of $\Theta(\log^2 n/\log\log n)$ bits.*

In the following $\mathrm{route}_T(u)$ denotes the binary labels corresponding to the address and the local table of u for routing in the tree T. Then ftree denotes the routing function such that $\mathrm{ftree}(\mathrm{route}_T(u), \mathrm{route}_T(v))$ returns the port number of the first edge of the path in T from u, to v.

2.3 Tree-decomposition and clique-tree

Our routing scheme is based on the notion of *tree-decomposition* introduced by Robertson and Seymour in their work on graph minors [17].

Definition 1 *A tree-decomposition of a graph G is a tree T whose nodes, called* bags*, are subsets of $V(G)$ such that:*

1. $\bigcup_{X\in V(T)} X = V(G)$;
2. *for all* $\{u,v\} \in E(G)$, *there exists* $X \in V(T)$ *such that* $u,v \in X$; *and*
3. *for all* $X,Y,Z \in V(T)$, *if* Y *is on the path from* X *to* Z *in* T *then* $X \cap Z \subseteq Y$.

The following property states that every bag of a tree-decomposition is a separator in the original graph. A proof of it can be found in [6].

Property 1 *Let T be a tree-decomposition of a graph G, let $\{X,Y\}$ be an edge of T, and let T_1, T_2 be the two trees obtained by removing $\{X,Y\}$ from T. Then for all vertices u,v of G such that $u \in \cup_{X\in V(T_1)}X$ and $v \in \cup_{X\in V(T_2)}X$, every path from u to v uses at least one vertex of $X \cap Y$.*

Chordal graphs admit special tree-decomposition (see Fig. 1 for an example):

Lemma 2 (cf. [5]) *Let G be a chordal graph. Then it is possible, in linear time, to compute a* clique-tree *of G, i.e., a tree-decomposition of G in which each bag induced a maximal clique of G.*

In the following we assume that tree-decompositions are rooted. Let G be any graph and let T be any tree-decompostion of G, for every vertex u of G, the *bag* of u, denoted by $\mathrm{B}(u)$, is the bag of T of minimum depth containing u. Observe that, once T has been fixed, $\mathrm{B}(u)$ is unique for each u by Rule 3 of Definition 1.

2.4 Hierarchical-tree

It is well known that every tree T on n nodes has a node u, called *median*, such that each connected component of $T \setminus \{u\}$ has at most $\frac{1}{2}n$ nodes. A *hierarchical-tree* of T is then a

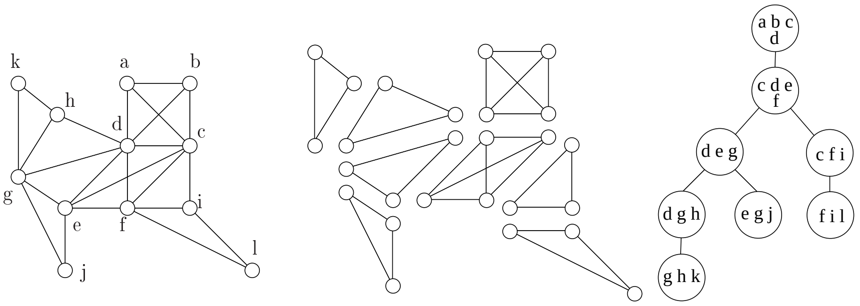

Figure 1: A chordal graph G, its set of maximal cliques, and a clique-tree of G.

rooted tree H defined as follows: the root of H is the median of T, u, and its children are the roots of the hierarchical trees of the connected components of $T \setminus \{u\}$ (the hierarchical-tree of a 1-node tree being the node it-self). Observe that T and H share the same set of nodes, and that the depth of H is at most[4] $\log |V(T)|$.

Property 2 *Let u, v be two nodes of a tree T, let P be the path from u to v in H (any hierarchical-tree of T) and let $w = \mathrm{nca}_H(u, v)$. Then, $w \in P$, and w is an ancestor in H of all the vertices of P.*

Proof. By construction, the subtree induced by w and its descendants in H is a connected component of T, say A. Thus, w, u, v are in A, but u and v are in two different components of $T \setminus \{w\}$. Thus in T, the path P from u to v is wholly contained in A and intersects w. So, $w \in P$ and w is an ancestor of all vertices of P in H. □

3 Routing scheme of deviation 2 for chordal graphs

In this section we will prove the following theorem:

Theorem 1 *For every chordal graph there exists a routing scheme of deviation 2 using tables and addresses of $O(\log^3 n / \log\log n)$ bits, and headers of $O(\log^2 n / \log\log n)$ bits. Moreover, this scheme is polynomial-time constructible, the initialisation procedure and the routing function are both performed in constant time. This scheme works even if port numbers are fixed by an adversary.*

In this section G denotes a chordal graph, T denotes a clique-tree of G, and H denotes a hierarchical-tree of T. Then for every bag X of T, S_X denotes a shortest path spanning tree of G rooted at any vertex r_X of X.

Now, let us outline the routing scheme we present in this section. Each vertex u of G contains in its address the information needed to route in all trees S_X such that X is an ancestor in H of $\mathrm{B}(u)$ ($\mathrm{B}(u)$ included). Since the depth of H is at most $\log n$, they are at most $1+\lfloor \log n \rfloor$ such trees for every vertex. To send a message from a source u to a destination v, we use a route through the tree S_X where $X = \mathrm{nca}_H(\mathrm{B}(u), \mathrm{B}(v))$. Indeed, since X is a

[4] All the logs are in base two.

separator between u and v in G (Properties 1, and 2) we will easily prove that the deviation is at most 2. Morevover, both u, v contain the information about routing in S_X. Unfortunately this is not true for all vertices along the route. Indeed the route can contain some *lost vertices*, i.e., vertices w such that $\mathrm{B}(w)$ is not a descendant of X in H. By construction, lost vertices do not have the routing information about S_X and so a message that arrives in a lost vertex will never reach to the destination. To solve this problem we store in the address of every vertex u, the identifier, the port number to reach the predecessor, and the port-number to reach the successor, of every lost vertex along the path from u to the root of the tree. This surplus of information is negligible since for a given tree and a given destination: we will prove that there is a constant number of lost vertices.

3.1 Description of the labels

For every vertex u of G we have:

- The local routing table of u, $\mathrm{table}(u)$, is set to $\langle \mathrm{id}(u), \mathrm{info}(u) \rangle$ (defined hereafter).
- The address of u, $\mathrm{address}(u)$ is set to $\langle \mathrm{id}(u), \mathrm{path}(u), \mathrm{help}(u), \mathrm{info}(u) \rangle$, where:
  - $\mathrm{id}(u)$ is the identifier of u (a unique integer in $\{1, \ldots, n\}$).
  - $\mathrm{path}(u)$ is a binary label allowing to determine, given $\mathrm{path}(u)$ and $\mathrm{path}(v)$, the depth of the nearest common ancestor between $\mathrm{B}(u)$ and $\mathrm{B}(v)$ in H.
  - $\mathrm{help}(u)$ is a table with $1 + \mathrm{depth}_H(\mathrm{B}(u))$ entries. Let X be an ancestor of $\mathrm{B}(u)$ in H ($X = \mathrm{B}(u)$ is possible) and P be the path in S_X from u to r_X. Then for each lost vertex w of P, $\mathrm{help}(u)[\mathrm{depth}_H(X)]$ contains the identifier, the port numbers to reach the predecessor and the successor of w in P.
  - $\mathrm{info}(u)$ is a table with $1 + \mathrm{depth}_H(\mathrm{B}(u))$ entries. Let X be an ancestor of $\mathrm{B}(u)$ in H, $\mathrm{info}(u)[\mathrm{depth}_H(X)] = \mathrm{route}_{S_X}(u)$, the binary label allowing to route in S_X (see Section 2).

3.2 The routing algorithm

Consider u, v two nodes of G, u the sender and v the receiver. Procedure $\mathrm{init}(u, v)$ is in charge of initialising the header attached to the message sent by u. This header, denoted by h_{uv}, is $\langle h_X, \mathrm{route}, \mathrm{rescue}_1, \mathrm{rescue}_2 \rangle$ computed as follows:

Algorithm init
Input: Two addresses: $\mathrm{address}(u)$ and $\mathrm{address}(v)$
Result: h_{uv}, the header of a message to v
begin
  $h_X \leftarrow \mathrm{depth}_H(\mathrm{nca}_H(\mathrm{part}(u), \mathrm{part}(v)))$;
  $\mathrm{route} \leftarrow \mathrm{info}(v)[h_X]$;
  $\mathrm{rescue}_1 \leftarrow \mathrm{help}(u)[h_X]$;
  $\mathrm{rescue}_2 \leftarrow \mathrm{help}(v)[h_X]$;
end

Consider any node w of G that receives a message with a header h_{uv}, the header computed from $\mathrm{init}(u, v)$ (possibly $w = u$). The output port number of the edge on which the message to v has to be sent from w is computed by the function $\mathrm{send}(w, h_{uv})$ described below.

Algorithm send
Input: The local routing table of a vertex w, a header h_{uv}
Result: The port number of an outgoing edge of w
begin
    if $\mathrm{id}(w) \in \mathrm{rescue}_2$ **then**
        return (the port number associated with w to reach its predecessor);
    if $\mathrm{id}(w) \in \mathrm{rescue}_1$ **then**
        return (the port number associated with w to reach its successor);
    return $(\mathrm{ftree}(\mathrm{info}(w)[h_X], \mathrm{route}))$;
end

3.3 Correctness and performances of the routing algorithm

In this paragraph we give some lemmas that prove Theorem 1.

Lemma 3 *Let u, v be two nodes of G, $\rho(u,v) \leqslant d_G(u,v) + 2$.*

Proof. Let P be the route from u to v induced by the routing scheme. By the routing algorithm described above, P is the path from u to v in S_X, where X is the bag chosen by $\mathrm{init}(u,v)$, i.e., $X = \mathrm{nca}_H(\mathrm{B}(u), \mathrm{B}(v))$. Thus $\rho(u,v) \leqslant d_G(u, r_X) + d_G(r_X, v)$.

By Properties 1, and 2, X is a separator between u and v in G. Thus there exists $w \in X$ such that $d_G(u,v) = d_G(u,w) + d_G(w,v)$. Since X is a clique, $d_G(u, r_X) \leqslant d_G(u,w) + 1$, and $d_G(r_X, v) \leqslant d_G(w,v) + 1$. Thus $\rho(u,v) \leqslant d_G(u,w) + d_G(w,v) + 2 = d_G(u,v) + 2$. □

Lemma 4 *For every vertex u,* $\mathrm{address}(u)$ *and* $\mathrm{table}(u)$ *can be implemented with a binary string of $O(\log^3 n / \log\log n)$ bits, such that* $\mathrm{init}(u,v)$ *runs in constant time. Moreover headers can be implemented with a binary string of $O(\log^2 n / \log\log n)$ bits such that for every vertex w,* $\mathrm{send}(w, h_{uv})$ *runs also in constant time.*

Proof. To implement $\mathrm{path}(u)$ we use the data structure presented in [13] that produces labels of $O(\log n)$ bits, and a query function running in constant time.

To implement the routing in trees S_X we use binary labels of binary of $O(\log^2 n / \log\log n)$ bits (see Section 2). Since the depth of H is at most $\log n$, $\mathrm{info}(u)$ can be implement with $O(\log^3 n / \log\log n)$ bits.

Since paths in trees S_X are shortest paths, Lemma 5 and Property 2 state that every entry of $\mathrm{help}(u)$ contains at most two lost vertices. Since for every lost vertex $\mathrm{help}(u)$ stores three intergers, $\mathrm{help}(u)$ can be implemented with a binary labels of $O(\log^2 n)$ bits.

It follows that addresses can be implemented with $O(\log^3 n / \log\log n)$ bits as claimed. Moreover headers of messages have just routing information of one tree, plus a constant number of integers, so headers are binary labels of $O(\log^2 n / \log\log n)$ bits.

In the function init, we compute once the depth of $\mathrm{nca}_H(\mathrm{B}(u), \mathrm{B}(v))$, then we make some constant number of copies of pointers. Thus init runs in constant time.

Since rescue_1 and rescue_2 contains a constant number of integers and since ftree runs in constant time, it follows that function send runs in constant time. $\square$

Lemma 5 *Let u, v be any two vertices of G, let P be shortest path in G from u to v, and let Q be the path in T from $\mathrm{B}(u)$ to $\mathrm{B}(v)$. Then, there exists at most two vertices $w_1, w_2 \in P$ such that $\mathrm{B}(w_1) \notin Q$ and $\mathrm{B}(w_2) \notin Q$.*

Proof. Let w be any vertex of P such that $\mathrm{B}(w) \notin Q$.

Assume that $\mathrm{B}(w)$ is a descendant in T of some bags of Q, and let Z be the closest one from $\mathrm{B}(w)$. By Property 1, any path from w to u or from w to v has to use a vertex of Z. So there exists a predecessor z, and a successor t, of w in P that belong to Z. Recall that Z is a clique so $\{z, t\}$ is an edge. Morevover since $w \notin Z$, $w \neq z$ and $w \neq t$: a contradiction with P is a shortest path.

So assume now tha $Y = \mathrm{nca}_T(\mathrm{B}(u), \mathrm{B}(v))$ separates u, v in one hand and z in the other hand. There exist a predecessor z, and a successor t, of w in P that both belong to Y. Here again we have that Y is a clique, so either $w = z$ or $w = t$. Lemma 5 directly follows. $\square$

4 A lower bound for shortest path routing

The next lower bound holds for the fixed port model [11]. Recall that in this model, the port number of the vertex u towards vertex v is fixed by an adversary, so cannot be optimized by the designer of the routing scheme. Recall that the port number is a unique integer ranging in $\{1, \ldots, \deg(u)\}$.

Theorem 2 *Let G be any graph with n vertices and minimum degree d. Every shortest path routing scheme on G requires, in the fixed port model, $\Omega(d \log d)$ bits for some local routing tables or some vertex addresses. In particular, there are chordal graphs with n vertices (i.e., some cliques) for which every shortest path routing scheme requires $\Omega(n \log n)$ bits, which is optimal by the use of standard routing tables.*

Proof. Let G be a graph with vertex set $V = \{1, \ldots, n\}$, with minimum degree d, and with a fixed port labeling, i.e., such that the port of every edge from u to v, denoted by $\mathrm{port}(u, v)$, is a unique integer taken from $\{1, \ldots, \deg(u)\}$.

Let $P(G)$ be the sequence of all the port numbers of G ordered according to the total ordering on the vertices, and then, for each vertex, by increasing ordering of its neighbors. More precisely, $\mathrm{port}(u, v)$ appears before $\mathrm{port}(u', v')$ in $P(G)$ if and only if $u < u'$, or $u = u'$ and $v < v'$.

Consider a shortest path routing scheme on G in the fixed port model, i.e., that does not permute the port numbers of G. And, let A_u be the address of u and let R_u be the local routing algorithm of u assigned by the routing scheme. Finally, let $A(G) = (A_u)_{u=1}^n$ and $R(G) = (R_u)_{u=1}^n$ be the sequences of all adresses and local routing algorithms for G.

Because R_u must generate a shortest path from u to any destination, in particular, for every neighbor v of u, the port number returned by $R_u(A_v)$ has to be precisely $\mathrm{port}(u, v)$. Therefore, by testing successively for $u = 1, \ldots, n$, R_u on A_v for all neighbors v of u in increasing order, it follows that:

Claim 1 *The sequence $P(G)$ can be computed given the pair of sequences $(A(G), R(G))$.*

Let us assume now that the addresses and the local routing algorithms assigned by the scheme are on at most k bits. (We want to give a lower bound on k.) So, there are at most $(2^{k+1}-1)^n$ sequences $A(G)$, since there is at most $2^{k+1}-1$ different binary addresses of length $\leqslant k$. For the same reason, there are at most $(2^{k+1}-1)^n$ different sequences for $R(G)$. It follows that there are less than $2^{2(k+1)n}$ pairs $(A(G), R(G))$. On the other hand, the number of port labelings for the underlying graph G, i.e., the number of distinct sequences $P(G)$, is $\prod_{u=1}^{n} \deg(u)!$. Since from Claim 1, each pair $(A(G), R(G))$ determines $P(G)$, we must have (recall that $\deg(u) \geqslant d$ for every u):

$$\begin{aligned} 2^{2(k+1)n} &> \prod_{u=1}^{n} \deg(u)! \ \geqslant \ d!^n \\ 2^{2(k+1)} &> d! \ > \ (d/e)^d \\ 2(k+1) &> d \log(d/e)\ . \end{aligned}$$

In particular, for a complete graph on n vertices, which is chordal, the latter equation rewrites:

$$2(k+1) \ > \ (n-1)\log((n-1)/e)\ .$$

Therefore $k \geqslant \frac{1}{2} n \log n - O(n)$ completing the proof. □

5 Conclusion

In this paper we present a low deviation routing scheme for chordal graphs. It uses addresses and local tables of $O(\log^3 n / \log\log n)$ bits, headers of $O(\log^2 n / \log\log n)$ bits, has a deviation at most 2, and each routing decision takes a constant time. In the same model, we prove that any shortest path routing schemes on chordal graphs requires $\Omega(n \log n)$-bit addresses or local routing tables in the worst-case.

The open problems we leave open for compact routing in chordal graphs are: 1) to determine whether there exists or not routing scheme of deviation 1 with still poly-log labels; and 2) to determine the exact label complexity of deviation 2 schemes, namely whether $o(\log^3 n / \log\log n)$-bit label routing scheme of deviation 2 exists or not.

References

[1] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg, *Compact distributed data structures for adaptive routing*, in 21^{st} Symposium on Theory of Computing (STOC), vol. 2, May 1989, pp. 230–240.

[2] ——, *Improved routing strategies with succinct tables*, Journal of Algorithms, 11 (1990), pp. 307–341.

[3] B. Awerbuch and D. Peleg, *Sparse partitions*, in 31^{th} Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society Press, 1990, pp. 503–513.

[4] ——, *Routing with polynomial communication-space trade-off*, SIAM Journal on Discrete Mathematics, 5 (1992), pp. 151–162.

[5] J. R. S. Blair and B. W. Peyton, *On finding minimum-diameter clique trees*, Nordic Journal of Computing, 1 (1994), pp. 173–201.

[6] R. Diestel, *Graph Theory (second edition)*, vol. 173 of Graduate Texts in Mathematics, Springer, Feb. 2000.

[7] Y. Dourisboure, *An additive stretched routing scheme for chordal graphs*, in 28-th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '02), vol. 2573 of Lecture Notes in Computer Science, Springer-Verlag, June 2002, pp. 150–163.

[8] ——, *Compact routing schemes for generalised chordal graphs*, Journal of Graph Algorithms and Applications, (2005). To appear, preliminary version published in DISC 2004 lncs 3274 pp 365-378.

[9] Y. Dourisboure and C. Gavoille, *Improved compact routing scheme for chordal graphs*, in 16-th International Symposium on DIStributed Computing (DISC 2002), vol. 2508 of Lecture Notes in Computer Science, Springer-Verlag, Oct. 2002, pp. 252–264.

[10] F. F. Dragan, C. Yan, and I. Lomonosov, *Collective tree spanners of graphs*, in 9^{th} Scandinavian Workshop on Algorithm Theory (SWAT), T. Hagerup and J. Katajainen, eds., vol. 3111 of Lecture Notes in Computer Science, Springer, 2004, pp. 64–76.

[11] P. Fraigniaud and C. Gavoille, *Routing in trees*, in 28^{th} International Colloquium on Automata, Languages and Programming (ICALP), F. Orejas, P. G. Spirakis, and J. v. Leeuwen, eds., vol. 2076 of Lecture Notes in Computer Science, Springer, July 2001, pp. 757–772.

[12] ——, *A space lower bound for routing in trees*, in 19^{th} Annual Symposium on Theoretical Aspects of Computer Science (STACS), vol. 2285 of Lecture Notes in Computer Science, Springer, Mar. 2002, pp. 65–75.

[13] C. Gavoille, M. Katz, N. A. Katz, C. Paul, and D. Peleg, *Approximate distance labeling schemes*, in 9^{th} Annual European Symposium on Algorithms (ESA), F. M. auf der Heide, ed., vol. 2161 of Lecture Notes in Computer Science, Springer, Aug. 2001, pp. 476–488.

[14] D. Kratsch, H.-O. Le, H. Müller, E. Prisner, and D. Wagner, *Additive tree spanners*, SIAM Journal on Discrete Mathematics, 17 (2004), pp. 332–340.

[15] L. Narayanan and N. Nishimura, *Interval routing on k-trees*, Journal of Algorithms, 26 (1998), pp. 325–369.

[16] D. Peleg and E. Upfal, *A trade-off between space and efficiency for routing tables*, Journal of the ACM, 36 (1989), pp. 510–530.

[17] N. Robertson and P. D. Seymour, *Graph minors. II. Algorithmic aspects of tree-width*, Journal of Algorithms, 7 (1986), pp. 309–322.

[18] M. Thorup and U. Zwick, *Compact routing schemes*, in 13^{th} Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), Hersonissos, Crete, Greece, July 2001, ACM PRESS, pp. 1–10.