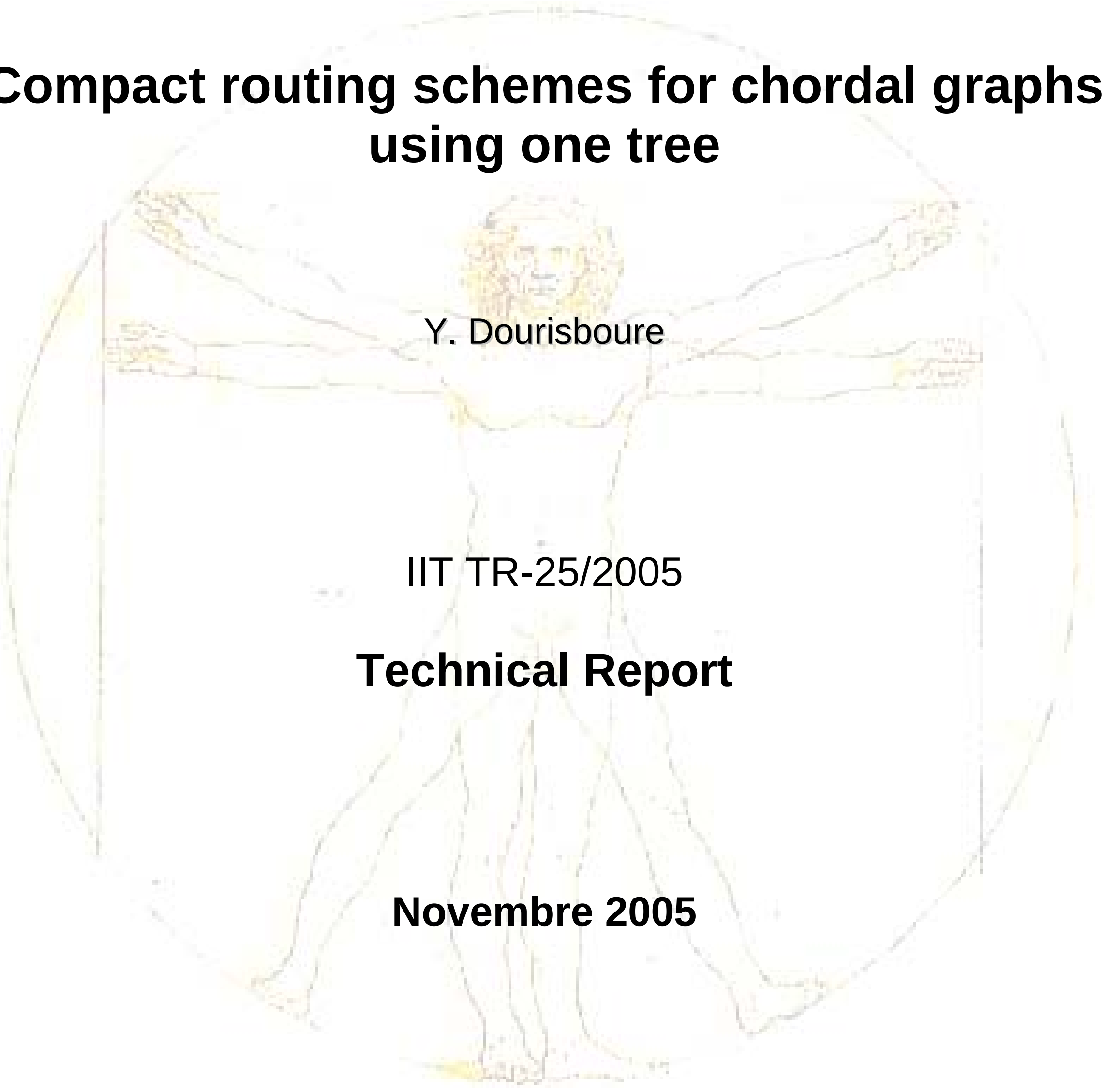
Consiglio Nazionale delle Ricerche



Compact routing schemes for chordal graphs using one tree

Y. Dourisboure

IIT TR-25/2005

Technical Report

Novembre 2005

Istituto di Informatica e Telematica

Compact routing schemes for chordal graphs using one tree

Yon Dourisboure[*]
yon.dourisboure@iit.cnr.it
IIT - CNR, Italy

29th November 2005

Abstract

In this paper we present a two new compact routing schemes designed for chordal graphs, i.e. graphs containing no induced cycle of length greater than 3. The main idea of these two routing schemes is constructing one spanning tree and adding some shortcuts.

The first one, insures that the length of the route between all pairs of vertices never exceed their distance plus 2 (deviation at most 2), and uses addresses and local tables of $O(\log^2 n)$ bits per node. The previous best scheme in date that guarantees the same deviation has a memory requirement of $O(\log^3 n/\log\log n)$ bits per vertex, and the previous one with the same memory requirement, guarantees only a deviation at most 4.

The second routing scheme we proposed, is designed for chordal graphs with maximum clique bounded by k. It guarantees a deviation at most 1 with addresses of $O(\log n)$ bits and local tables of $O(k \log n)$ bits per node. The best routing scheme in date has the same memory requirement, but guarantees only a deviation at most 2.

Keywords: 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 efficient routes and have relatively low *memory requirements*. It was shown in a series of papers (see, e.g., [18, 1, 2, 3, 4, 20]) that there is a trade-off between the memory requirements of a routing scheme and the worst-case stretch factor it guarantees.

[*]Work Partially supported by the Research and Training Network COMBSTRU (HPRN-CT-2002-00278)

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. Trees and cliques are chordal graphs. Note that from an information theory point of view, there is no way to give a compact representation of the underlying topology of such graphs, unlike other regular topologies (as hypercubes, grid, Cayley graphs, etc.). Indeed, there are at least $2^{n^2/4-o(n^2)}$ non-isomorphic chordal graphs with n nodes by considering for instance *split-graphs*, namely a complete graphs of $\lceil n/2 \rceil$ nodes and with $\lfloor n/2 \rfloor$ extra nodes whose neighborhood is randomly selected into the clique

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 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 [8], is derived from an interval routing scheme for *k-trees*, proposed in [16]. If we accept a small deviation, it is shown in [8], that every chordal graph with maximum clique k admits a routing scheme of deviation 2 with addresses of $O(\log n)$ bits and local routing tables of $O(k \log n)$ bits per node.

The first scheme independant of the maximum clique is due to Peleg and Upfal [18], they have constructed a routing scheme of stretch 3 using $O(n \log^2 n)$ bits in total[1] for local routing tables and addresses of $O(\log^2 n)$ bits. Then, in [10] and [12], they constructed routing schemes of deviation 2 for chordal graphs with addresses and local routing tables of $O(\log^3 n / \log \log n)$ bits per node. The last routing scheme in date insures a deviation at most 4 with addresses and local routing tables of $O(\log^2 n)$ bits [9].

In this paper, we propose two routing schemes based on routing along one spanning tree. But, has shown in [15], there exist chordal graphs without spanning tree that garantees a constant stretch or deviation, so our routing schemes route along one tree plus some shortcuts. The first routing scheme controls shortucuts with the method introduced in [9], and then garantees a deviation 2 with addresses and local routing tables of $O(\log^2 n)$ bits per node. The second one, designed for chordal graphs with maximum clique bounded by k, controls shortucuts with the method proposed in [8], and garantees a deviation 1, with addresses of $O(\log n)$ bits and local routing tales of $O(k \log n)$ bits.

2 Basic notions and notations

All graphs occurring in this paper are connected, finite, undirected and unweighed. 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: $d_G(u, X) = \min_{v \in X} d_G(u, v)$. Moreover, 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, $d_G(u, r) = d_T(u, r)$ 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 and a descendant of itself. The *nearest common ancestor* between two vertices u, v in a tree T is denoted by $\mathrm{nca}_T(u, v)$.

[1]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.

2.1 Routing scheme

As already 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 port(e), taken from $[1, \deg(u)]$.
- associates with every node u of the graph two labels: its *address* denoted by address(u), and a *local routing table* denoted by table(u).
- constructs an *initalisation procedure* that is executed by any initial sender u. Given address(u) and 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 table(w) and h_{uv}, this function returns the port number on which w has to send the message. Observe that in our routing schemes, once computed by the initial sender, headers never change. Thus the two routing schemes we present are necessarily loop-free.

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

- memory requirement: The number of bits needed for address(u), table(u) and h_{uv}, in the worst case.
- time requirement: the time[2] 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.

2.2 Routing in trees

The routing scheme of this paper needs route in trees. To implement compact routing in trees we use the following result proposed in [13].

Lemma 1 (cf. [13]) *Every n-node tree T admits a shortest path compact routing scheme, constructible in polynomial time, without initalisation procedure, with a routing function computable in constant time, and such that addresses, local tables and headers are of $O(\log n)$ bits.*

Roughly speeking their scheme assigns to the address of every node u, the list of port numbers used to go from the root to u. Then the routing function is simply: if address(u) is a prefix of address(v) then return the first port number of address(v) that is not in address(u) else return the port number to reach the parent of u. Obviously it is not so simple because to obtain such number of bits for addresses they made many optimisations, like for example, storing in addresses only port numbers of "heavy edges" (see [13] for more details).

In the following 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 ftree(route$_T(u)$, route$_T(v)$) returns the port number of the first edge of the path in T from u, to v.

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

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 [19].

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 $u \in V(G)$, the set of bags containing u induces a subtree of T.*

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 [7].

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 Figure 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 induces a maximal clique of G.*

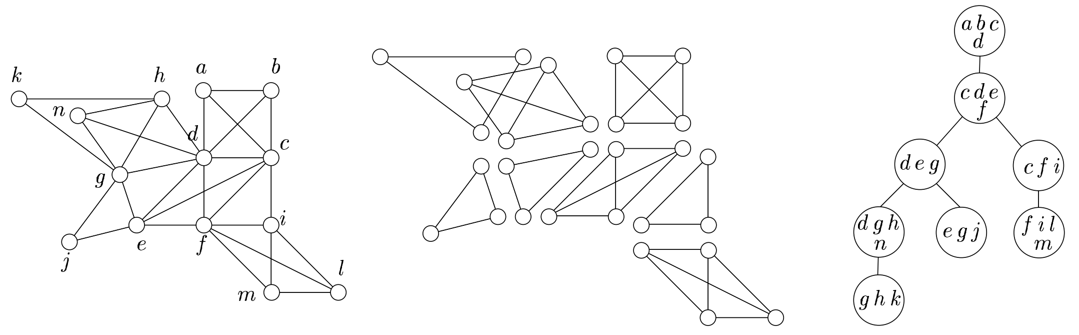


Figure 1: From left-to-right: a chordal graph G, its set of maximal cliques, and a clique-tree of G.

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 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[3] $\log |V(T)|$.

[3]All the logs are in base two.

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 $O(\log^2 n)$ bits, and headers of $O(\log n)$ bits. Moreover, this scheme is polynomial-time constructible, the initialisation procedure and the routing function are both performed in constant time.*

The main idea of our routing scheme is to merge the two ideas of [8] and [9]. That is to say, as done in [8], the message follows a shortest path spanning tree S of G rooted at a vertex that belongs to the root of a clique-tree T of G. This shortest path spanning tree, carefully chosen, given two vertices u, v, insures a shortest path in G form u and from v to a vertex that belongs to $\mathrm{nca}_T(\mathrm{B}(u), \mathrm{B}(v))$. Since routing along a single tree can not guarantees a constant deviation for chordal graphs [15], once in $\mathrm{nca}_T(\mathrm{B}(u), \mathrm{B}(v))$, the message has to use some shortcuts. Unfortunately, since the depth of T can be $\Omega(n)$, covering efficiently each ancestor of $\mathrm{B}(u)$, can produce labels of $\Omega(n)$ bits. Nevertheless, in [9] they showed, thanks to the notion of hierarchical-tree, that it is possible to split this covering, one half in $\mathrm{nca}_T(\mathrm{B}(u), \mathrm{B}(v))$, and one half in $\mathrm{nca}_H(\mathrm{B}(u), \mathrm{B}(v))$, where H is a hierarchical-tree of T. In this way, since the depth of H is at most $\log n$, this insures that the covering needs only labels of $O(\log^2 n)$ bits.

3.1 Preliminaries

Our routing scheme is strongly based on clique-tree of graphs. Nevertheless, in order to simplify the description of our routing scheme and the proofs of this paper, we prefer using *extended clique-tree*, that is a tree-decomposition satisfying the following proposition (see an example in Figure 2).

Proposition 1 *Let G be a chordal graph, there exists a polynomial time constructible tree-decomposition T of G, with n bags, such that every bag of T induces a clique of G and for all vertices u, v of G, $\mathrm{B}(u) = \mathrm{B}(v) \Rightarrow u = v$.*

Proof. Let G be a chordal graph and T be a clique-tree of G. If G has one vertex then the proposition is trivialy true. So assume that G has at least two vertices. Since T is a clique-tree, for all deferents bags X, Y of T, $X \not\subset Y$ so $|V(T)| \leqslant n-1$ (the root of T contains at least two vertices). Moreover, for every bag X of T, let us define the set $\mathrm{B}^{-1}(X)$ by $\mathrm{B}^{-1}(X) = \{u \in V(G) \mid \mathrm{B}(u) = X\}$. It is easy to see that $|V(T)| \leqslant n - \sum_{X \in V(T)}(|\mathrm{B}^{-1}(X)| - 1)$.

Then, let T' be the tree constructed by Algorithm extend defined hereafter. Clerly, each bag introduced by Algorithm extend induces a clique of G. Moreover, after each introducing, T' remains a tree-decomposition of G, and $\sum_{X \in V(T')}(|\mathrm{B}^{-1}(X)| - 1)$ is reduced by 1 and $|V(T')|$ is increased by 1. So, after introducing $\sum_{X \in V(T)}(|\mathrm{B}^{-1}(X)| - 1)$ new bags, the algorithm will stop. That completes the proof.

□

```
Algorithm extend
Input: A clique-tree $T$ of a graph $G$
Result: An extended clique-tree $T'$ of $G$
begin
    $T' \leftarrow T$;
    $r \leftarrow$ any vertex that belongs to the root of $T$;
    Insert $\{r\}$ in $T'$ as the root;
    while Exist a bag $X$ of $T'$ and two vertices $u, v$ of $X$ such that $u \neq v$ and $u, v \in \mathrm{B}^{-1}(X)$
    do
        $Y \leftarrow X \setminus \{v\}$;
        Insert $Y$ between $X$ and its parent in $T'$;
    end
end
```

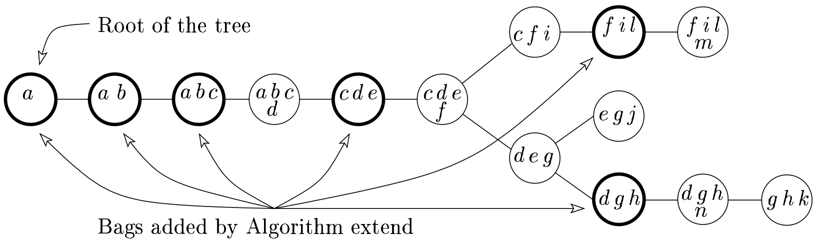


Figure 2: An extended clique-tree of the clique-tree depicted in Figure 1.

From now, T denotes an extended clique-tree of G, and H denotes a hierarchical-tree of T. Morevover,S denotes the spanning tree of G defined as follows:

- the root of S is r, the first vertex chosen by Algorithm extend ($\mathrm{B}(r)$ is the root of T);
- For every $u \neq r$, the parent of u in S, $p(u)$, is set to the vertex of $\mathrm{B}(u)$ whose bag is of minimum depth in T.

Proposition 2 *Let u be a vertex of G and X be an ancestor of* $\mathrm{B}(u)$ *in T, then $d_G(u, X) = d_S(u, X)$. As corollary we have that S is a shortest path spanning tree of G rooted at r, and we have that $d_G(\mathrm{B}(u), X) = d_S(u, X) - 1$.*

Proof.

Let u be a vertex of G and X be a ancestor in T of $\mathrm{B}(u)$. Then, let $v \in X$ such that the path $P = x_0, x_1, \ldots, x_p$ (with $u = x_0$ and $v = x_p$) is a shortest path in G from u to X. By construction of S, $\mathrm{B}(x_1)$ is a descendant in T of $\mathrm{B}(p(u))$, where $p(u)$ si the parent of u in S. If $\mathrm{B}(p(u))$ is an ancestor of X in T then by definition of T, $p(u) \in X$ and the result holds. So assume that $\mathrm{B}(p(u))$ is a descendant of X in T. Then $\mathrm{B}(p(u))$ separates x_1 and v. Thus, there exists $i > 1$ such that $x_i \in \mathrm{B}(p(u))$. Since $\mathrm{B}(p(u))$ is a clique, $\langle p(u), x_i \rangle$ is an edge of G. We can conclude that $d_G(p(u), v) = d_G(u, v) - 1$, completing the proof. □

3.2 Description of labels

For every vertex u of G we have:

- The local routing table of u in G, table(u), is set to $\langle \mathrm{id}(u), \mathrm{route}_S(u) \rangle$ (defined hereafter);
- The address of u, address(u), is set to $\langle \mathrm{id}(u), \mathrm{route}_S(u), \mathrm{path}(u), \mathit{help}(u) \rangle$, where:
    - id(u) is the identifier of u (a unique integer in $\{1, \ldots, n\}$).

- $\text{route}_S(u)$, is the binary label allowing to route in S (see Section 2).
- $\text{path}(u)$ is a binary label allowing to determine, given $\text{path}(u)$ and $\text{path}(v)$, the depth in H of $\text{nca}_H(\text{B}(u), \text{B}(v))$.
- $\text{help}(u)$ is a table with $1 + \text{depth}_H(\text{B}(u))$ entries. Let X be an ancestor of $\text{B}(u)$ in H ($X = \text{B}(u)$ is possible), and let r_X be the vertex of X such that $\text{B}(r_X) = X$ (see Figure 3). Then, $\text{help}(u)[\text{depth}_H(X)] = \langle \text{route}_S(r),\ \text{id}(u'),\ p_1,\ p_2 \rangle$, defined as follows:

  Let $Z = \text{nca}_T(\text{B}(u), X)$, and w be the nearest ancestor or r_X that belongs to Z.
  - u' is the nearest ancestor of u in S that belongs to Z.
  - If $Z = X$ (case for v in Figure 3), then r is set to the parent of w (observe that $w = r_X$, so $r = p(r_X)$), else (case for u in Figure 3) r is set to w. Observe that in the later case, we have necessarily $Z = Y$, where $Y = \text{nca}_T(\text{B}(u), \text{B}(v))$, and so $w = r'_X$, the nearest ancestor in S of r_X that belongs to Y. Then, $\text{route}_S(r)$ is the routing label in S of r.
  - Then, p_1 is the port number to reach r from u', p_2 is the port number to reach u' from r.

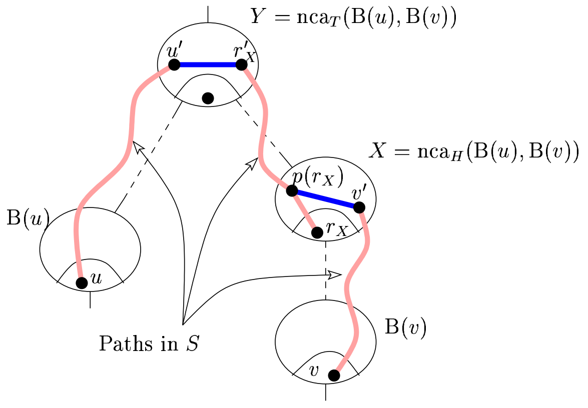

$Y = \text{nca}_T(\text{B}(u), \text{B}(v))$
u'
r'_X
$X = \text{nca}_H(\text{B}(u), \text{B}(v))$
$p(r_X)$
v'
r_X
$\text{B}(u)$
u
Paths in S
$\text{B}(v)$
v

Figure 3: The main situation in T.

3.3 The routing algorithm

Let u, v be two vertices of G, u the sender and v the receiver. Procedure $\text{init}(u, v)$ is in charge of initialising the header attached to the message sent by u. This header, denoted by h_{uv}, is $\langle \text{route}_S(v), \text{route}_S(r), id(u'), p_1, p_2 \rangle$ as described below.

```
**Algorithm** init
**Input**: Two addresses: $\text{address}(u)$ and $\text{address}(v)$
**Result**: $h_{uv}$, the header of a message to $v$
**begin**
    $h_X \leftarrow \text{depth}_H(\text{nca}_H(\text{B}(u), \text{B}(v)))$;
    $id(u') \leftarrow \text{help}(u)[h_X].id(u')$;
    $p_1 \leftarrow \text{help}(u)[h_X].p_1$;
    $p_2 \leftarrow \text{help}(v)[h_X].p_2$;
    $\text{route}_S(r) \leftarrow \text{help}(v)[h_X].\text{route}_S(r)$;
**end**
```

Consider any node w of G that receives a message with a header h_{uv}, the header computed from $\text{init}(u, v)$ (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 $\text{send}(w, h_{uv})$ 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 from $w$
**begin**
    **if** $w$ *is an ancestor or a descendant of* $v$ *in* $S$ **then**
        return $(\text{ftree}(\text{route}_S(w), \text{route}_S(v)))$;
    **if** $w$ *is* $r$ **then**
        return $p_2$;
    **if** $w$ *is an ancestor or a descendant of* $r$ *in* $S$ **then**
        return $(\text{ftree}(\text{route}_S(w), \text{route}_S(r)))$;
    **if** $\text{id}(w) = \text{id}(u')$ **then**
        return $p_1$;
    return (the port number between $w$ and its parent in $S$);
**end**
```

3.4 Correctness and performances of the routing algorithm

In this section we give some lemmas that proved Theorem 1.

Lemma 3 *Let u, v be two vertices of G, then $\rho(u,v) \leqslant d_G(u,v) + 2$, where $\rho(u,v)$ denotes the length of the route from u to v induced by algorithms* init *and* send.

Proof. Let u, v be two vertices of G and let $X = \text{nca}_H(\text{B}(u), \text{B}(v))$. By Property 2, X separates $\text{B}(u)$ and $\text{B}(v)$ in T, so by Property 1, X separates u and v in G. Thus $d_G(u,v) \leqslant d_G(u,X) + d_G(X,v)$.

Let $Y = \text{nca}_T(\text{B}(u), \text{B}(v))$, clearly Y is an ancestor of X (or is equal to X) in T. Thus either Y separates u and X, or separates v and X.

Assume that Y separates u and X, and that $Y \neq X$ (case depicted in Figure 3). Observe that we have $d_G(u,v) \geqslant d_G(u,Y) + d_G(Y,X) + d_G(X,v)$. Let u', v' and r'_X be respectively the nearest ancestors of u, v and r_X in T that belong respectively to Y, Y and X. Then, by Proposition 2 we have $d_G(u,Y) = d_S(u,u')$, $d_G(X,Y) = d_S(r'_X, p(r_X))$ and $d_S(v,v') = d_G(v,X)$. Thus $d_G(u,v) \geqslant d_S(u,u') + d_S(r'_X, p(r_X)) + d_S(v,v')$. Moreover, since we have supposed that Y separates $\text{B}(u)$ and X in T and $Y \neq X$, we have necessarily $Z = Y \neq X$, where $Z = \text{nca}_T(\text{B}(u), X)$. Thus u' and r'_X are the vertices of $\text{help}(u)[\mathit{depth}_H(X)]$. Similarly we can show that v' and $p(r_X)$ are the vertices of $\text{help}(v)[\mathit{depth}_H(X)]$. So the route from u to v is well defined and $\rho(u,v) \leqslant d_G(u,v) + 2$.

The proof of the case where Y separates v and X, and where $Y \neq X$, is similar replacing u by v and v by u.

Assume now that $Y = X$. In this latter case, it is easy to see that for both u, v we have $Z = X$, so both have in their address the information to reach $p(r_X)$. So here again the route from u to v is well defined and $\rho(u,v) \leqslant d_G(u,v) + 2$. □

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

Proof. Let u be any vertex of G.

As seen in Section 2, since port numbers can be chosen in $[1, \deg(u)]$ during the preprocessing treatment, $\mathrm{route}_S(u)$ is a binary label of $O(\log n)$ bits.

To implement $\mathrm{path}(u)$ we use the data structure presented in [14] that produces labels of $O(\log n)$ bits, and a decodable function running in constant time.

Then, for every ancestor of $\mathrm{B}(u)$ in H, $\mathrm{help}(u)[h_X]$ contains one routing label and three integers (one identifer and two port numbers). Since the depth of H is at most $\log n$, $\mathrm{help}(u)$ contains $O(\log^2 n)$ bits.

In the function init, we only make some constant number of copies of pointers. Thus init runs in constant time.

In the function send, each one of the three firsts tests can be done in constant time using the routing function ftree with the three labels $\mathrm{route}(w), \mathrm{route}(r)$ and $\mathrm{route}(v)$. Then, knowing if $\mathrm{id}(w) = \mathrm{id}(u')$ is clearly also done in constante time. □

4 Routing scheme of deviation 1 for chordal graphs with bounded maximum clique

In this section we will prove the following theorem:

Theorem 2 *For every chordal graph of maximum clique k, there exists a routing scheme of deviation 1 using addresses and headers of $O(\log n)$ bits, and local tables of $O(k \log n)$ bits. Moreover, this scheme is polynomial-time constructible, is without initialisation procedure, and has a routing function performed in $O(k)$ time.*

In this section, G denotes a chordal graph of maximum clique k. Then, as in the previous section, T denotes an extended clique-tree of G and S denotes the shortest path spanning tree of G as defined in Section 3.1.

To send a message from a vertex u to a vertex v, the main idea is also to follow S until reaching $\mathrm{nca}_T(\mathrm{B}(u), \mathrm{B}(v))$. Since, the maximum clique of G is bounded by k, the covering of $\mathrm{nca}_T(\mathrm{B}(u), \mathrm{B}(v))$ can be done without the notion of hierarchical-tree. Indeed, it contains at most k vertices, so we show that tables of $O(k \log n)$ bits are enough. Moreover, by doing this, we guarantee a deviation at most 1.

4.1 Description of labels

For every vertex u of G we have:

- The local routing table of u in G, $\mathrm{table}(u)$, is set to $\mathrm{table}(u) = \{(\mathrm{address}(v), \mathrm{port}(u, v)) \mid v \in \mathrm{B}(u)\}$ (defined hereafter);
- The address of u, $\mathrm{address}(u)$, is set to $\langle \mathrm{route}_S(u), \mathrm{path}(u) \rangle$, where:
  - $\mathrm{route}_S(u)$, is the binary label allowing to route in S (see Section 2).
  - $\mathrm{path}(u)$ is a binary label allowing to determine, given $\mathrm{path}(u)$ and $\mathrm{path}(v)$, the depth in T of $\mathrm{nca}_H(\mathrm{B}(u), \mathrm{B}(v))$.

4.2 The routing algorithm

Consider any vertices u, v of G, and assume that u has to follows a message to v. Procedure $\mathrm{send2}(u, v)$ is charge, thanks to the local routing table of u and the address of v, to compute the output port number of the edge on which the message to v has to be sent from u

```
**Algorithm** send2
**Input**: The local routing table of a vertex $u$, the address of the destination $v$
**Result**: The port number of an outgoing edge from $u$
begin
    if $u$ is an ancestor of $v$ in $S$ then
        return (ftree(route$_S(u)$, route$_S(v)$));
    if there exists a vertex $w \in \mathrm{B}(u)$ such that $w$ is an ancestor of $v$ in $S$ then
        return port$(u, w')$; (where $w'$ is the one of maximum depth in $S$ satisfying the condition)
    if there exists a vertex $w \in \mathrm{B}(u)$ such that $\mathrm{B}(w)$ is an ancestor in $T$ of $\mathrm{B}(v)$ then
        return port$(u, w')$; (where $w'$ is the one of maximum depth in $T$ satisfying the condition)
    return (the port number between $w$ and its parent in $S$);
end
```

4.3 Correctness and performances of the routing algorithm

In this section we give some lemmas that proved Theorem 2.

Lemma 5 *Let u, v be two vertices of G, then $\rho(u,v) \leqslant d_G(u,v) + 1$, where $\rho(u,v)$ denotes the length of the route from u to v induced by Algorithm* send2*.*

Proof. Let u, v be two vertices of G, u the sender and v the destination, and let $Y = \mathrm{nca}_T(\mathrm{B}(u), \mathrm{B}(v))$.

Assume that $Y = \mathrm{B}(u)$. Since T is a tree-decomposition of G, there exists at least one ancestor in S of v that belongs to $\mathrm{B}(u)$. Thus Function send2 insures that u send the message to the one, w, of maximum depth in S. Once in w, clearly then the message follows S. By Proposition 2, we know that $d_G(v,Y) = d_S(v,Y) = d_S(v,w)$, thus $\rho(u,v) \leqslant d_G(u,v) + 1$.

Thus, assume now that $\mathrm{B}(u)$ is a strict descendant of Y, and let u' (resp. v') be the nearest ancestor in S of u (resp. v) such that its parent in S $p(u')$ (Resp.$p(v')$) belongs to Y (see Figure 4.(a) for an example). Since T is a tree-decomposition of G we know that $d_G(u,v) \leqslant d_G(u,Y) + d_G(v,Y)$, moreover, Proposition 2 states that $d_G(u,Y) = d_S(u,p(u'))$ and $d_G(v,Y) = d_S(v,p(v'))$. Clearly Function send2 insures that the message follows S until u'. Then, let w be the vertex chosen by Algorithm send2 in u', w is either chosen by the second test, or by the third test. Let us study these two different cases:

1. w is the nearest ancestor in S of v that belongs to $\mathrm{B}(u')$. Then, by construction of S we know that $\mathrm{B}(w)$ is an ancestor in T of $\mathrm{B}(v)$. Since T is a tree-decomposition and since $\mathrm{B}(u')$ is not an ancestor in T of $\mathrm{B}(v)$, we know that $\mathrm{B}(w)$ is an ancestor in T of Y and moreover $w \in Y$. Since Y is a clique, if $w \neq p(v')$ then w is the parent the parent in S of $p(v')$. Observe that once in w, the routing scheme insures that the route follows S until v, so we can conclude $\rho(u,v) = d_S(u,u') + 1 + d_S(w,v) \leqslant d_S(u,Y) + d_S(Y,v) + 1 \leqslant d_G(u,v) + 1$.

2. w is the vertex of $\mathrm{B}(u')$ whose bag is of maximum depth in T and is also an ancestor in T of $\mathrm{B}(v)$. As we have done in the case where $Y = \mathrm{B}(u)$, we can prove that $\rho(w,v) \leqslant 1 + d_G(v, \mathrm{B}(w))$. Morevover, by Proposition 2 we already know that $\rho(u,w) = d_G(u,Y)$. Thus $\rho(u,v) \leqslant d_G(u,Y) + d_G(v,\mathrm{B}(w)) + 1$. Then, observe that $\mathrm{B}(w)$ and $\mathrm{B}(p(v'))$ are both ancestors of Y, and since T is a tree-decomposition there are both in Y. So either $p(v') \in \mathrm{B}(w)$ (Figure 4.(b)), or $w \in \mathrm{B}(p(v'))$ (Figure 4.(c)). Let us study these two last cases:

    - If $p(v') \in \mathrm{B}(w)$, then $d_G(v, \mathrm{B}(w)) \leqslant d_S(v,v')$. Moreover, by Proposition 2 we know that $d_S(v,v') \leqslant d_G(v,Y)$. Thus $d_G(v,\mathrm{B}(w)) \leqslant d_G(v,Y)$, and so $\rho(u,v) \leqslant d_G(u,Y) + d_G(v,Y) + 1 \leqslant d_G(u,v) + 1$.

- If $w \in \mathrm{B}(p(v'))$, then by construction of S, the parent in S of $p(v')$ belongs to $\mathrm{B}(w)$. We obtain that $d_G(v, \mathrm{B}(w)) = d_G(v, Y) + 1$ and so $\rho(u, v) \leqslant d_G(u, Y) + d_G(v, Y) + 2$. Let us prove that in this case $d_G(u, v) = d_G(u, Y) + d_G(v, Y) + 1$. Let P be a shortest path in G from u to v, and assume that $d_G(u, v) = d_G(u, Y) + d_G(v, Y)$, i.e., there exists one, and only one, vertex z among P that belongs to Y. Let z_1 (resp. z_2) be the predecessor (resp. the successor) of z in P. Then, as we have done in the proof of Proposition 2, we can prove that $\mathrm{B}(z_1)$ is a descendant in T of $\mathrm{B}(u')$, in particular, $z \in \mathrm{B}(u')$. By choice of w, we have that $\mathrm{B}(z)$ is an ancestor in T of $\mathrm{B}(w)$, in particular $z \in \mathrm{B}(w)$. Similarily we can prove that $\mathrm{B}(z_2)$ is a descendant in T of $\mathrm{B}(v')$, so by construction of S, $\mathrm{B}(p(v'))$ is an ancestor in T of $\mathrm{B}(z)$. Thus we obtain that $\mathrm{B}(p(v'))$ is an ancestor in T of $\mathrm{B}(w)$ and so $p(v') \in \mathrm{B}(w)$, a contradiction. So $d_G(u, v) = d_G(u, Y) + d_G(v, Y) + 1$ and thus $\rho(u, v) \leqslant d_G(u, v) + 1$.

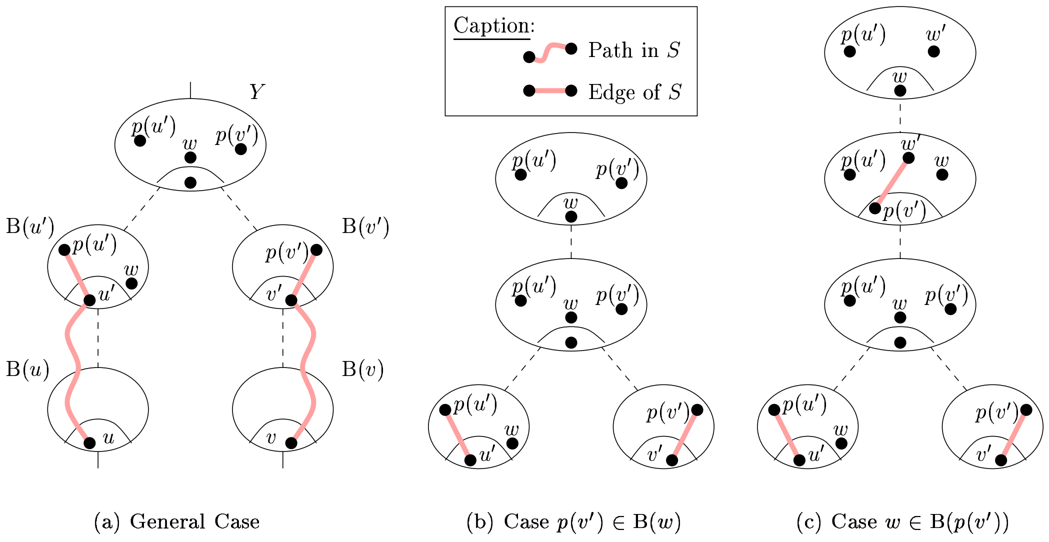


Figure 4: Deviation 1 in chordal graphs with bounded maximum clique.

□

Lemma 6 *For every vertex u,* address(u) *can be implemented with a binary string of $O(\log n)$ bits, and* table(u) *with a binary string of $O(k \log n)$ bits, such that* send2(u, w) *runs in $O(k)$ times.*

Proof. Let u be any vertex of G.

As seen in Lemma 4, address(u) can be implemented with a binary string of $O(\log n)$ bits. Since table(u) contains exactly at most k addresses and k port numbers, it can be implemented with a binary string of $O(k \log n)$ bits.

In Function send2, the first test is done in constant time using ftree. The second test needs k calls to ftree, so it needs $O(k)$ times. The last test need $O(\log k)$ time since vertices of table(u) can be shorted in advance by depth in T. □

5 Conclusion

In this paper we present two efficients routing schemes for chordal graphs, based on one shortest path spanning tree. The first scheme guarantees a deviation at most 2 with addresses and local tables of $O(\log^2 n)$ bits, and with headers of $O(\log n)$ bits. The second routing scheme, designed for chordal graphs with maximum clique bounded by k, has a deviation at most 1 with addresses of $O(\log n)$ bits and tables of $O(k \log n)$ bits.

Our routing schemes use a shortest spanning tree of G and some shortcuts. This technique seems to be optimal because as proved in [15], there exist chordal graphs without spanning tree that garantees a constant stretch or deviation. It should be interesting to evaluate the optimality of our methods for storing shortcuts: 1. Is it possible to obtain a deviation 1 with labels of polylog in n bits ? 2. Is it possible to obtain a deviation 2 with labels of less than $O(\log^2 n)$ bits ?

An other way of research should be *graph spanners*, i.e., sparse subgraphs of graph that are good approximation for distances. Indeed the main idea of the routing scheme of deviation 4 presented in [9] is derived from two results in graph spanners proposed in [6] and [11]. They proved that any chordal graph G has a subgraph H with $O(n)$ edges in which the deviation is at most 4, i.e., for all u, v, $d_H(u,v) \leqslant d_G(u,v) + 4$. Since our first routing scheme is strongly based on the idea of [9], it should be possible to prove that our routing scheme is derived from a spanner with $O(n)$ edges and deviation at most 2. This result would be optimal because it is proved in [17] that a subgraph guaranteeing a deviation 1 for chordal graphs has $\Omega(n\sqrt{n})$ edges.

References

[1] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg, *Compact distributed data structures for adaptive routing*, in 21st 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 31th 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] V. D. Chepoi, F. F. Dragan, and C. Yan, *Additive spanner for k-chordal graphs*, in Algorithms and Complexity: 5th Italian Conference (CIAC 2003), vol. 2653 of Lecture Notes in Computer Science, Springer-Verlag Heidelberg, May 2003, pp. 96–107.

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

[8] 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.

[9] ———, *Compact routing schemes for tree-length δ graphs and for k-chordal graphs*, in 18-th International Symposium on DIStributed Computing (DISC 2004), vol. 3274 of Lecture Notes in Computer Science, Springer, Oct. 2004, pp. 365–378.

[10] 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.

[11] ——, *Sparse additive spanners for bounded tree-length graphs*, in 11th Colloquium on Structural Information and Communication Complexity (SIROCCO), vol. 3104 of Lecture Notes in Computer Science, Springer, June 2004, pp. 123–137.

[12] F. F. Dragan, C. Yan, and I. Lomonosov, *Collective tree spanners of graphs*, in 9th 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.

[13] P. Fraigniaud and C. Gavoille, *Routing in trees*, in 28th 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.

[14] C. Gavoille, M. Katz, N. A. Katz, C. Paul, and D. Peleg, *Approximate distance labeling schemes*, in 9th 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.

[15] 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.

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

[17] D. Peleg and A. A. Schäffer, *Graph spanners*, Journal of Graph Theory, 13 (1989), pp. 99–116.

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

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

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