Compact Forbidden-Set Routing

Bruno Courcelle and Andrew Twigg

[1] LaBRI, Bordeaux 1 University and CNRS
courcell@labri.fr
[2] Computer Laboratory, Cambridge University
andrew.twigg@cl.cam.ac.uk

Abstract. We study labelling schemes for X-constrained path problems. Given a graph (V, E) and $X \subseteq V$, a path is X-constrained if all intermediate vertices avoid X. We study the problem of assigning labels $J(x)$ to vertices so that given $\{J(x) : x \in X\}$ for any $X \subseteq V$, we can route on the shortest X-constrained path between $x, y \in X$. This problem is motivated by Internet routing, where the presence of routing policies means that shortest-path routing is not appropriate. For graphs of tree width k, we give a routing scheme using routing tables of size $O(k^2 \log^2 n)$. We introduce m-clique width, generalizing clique width, to show that graphs of m-clique width k also have a routing scheme using size $O(k^2 \log^2 n)$ tables.

Keywords: Algorithms, labelling schemes, compact routing.

1 Introduction

Given a graph $G = (V, E)$ where each vertex $u \in V$ has a set $S(u) \subseteq V$, a compact *forbidden-set routing scheme* is a compact routing scheme where all routes from u are (approximately) shortest paths in the (possibly disconnected) graph $G \setminus S(u)$. The problem is motivated by Internet routing, where nodes (routers) can independently set routing policies that assign costs to paths, thus making the shortest path not necessarily the most desirable. Shortest-path routing is well-understood, for example Thorup and Zwick [1] have given a compact routing scheme using $\tilde{O}(\sqrt{n})$ size tables, which is almost optimal for stretch-3 paths. On the other hand, very little is known about the complexity of forbidden-set routing. The only known algorithms for policy routing (such as BGP) use Bellman-Ford iteration to construct so-called stable routing trees – for each destination, a tree is rooted at that destination and packets are forwarded along it. Varadhan et al.[2] showed that the policies may conflict, forcing the algorithm to not converge. For general policies, Griffin et al.[3] showed that deciding if it will converge is NP-complete, and Feigenbaum and Karger et al.[4] showed that NP-completeness still holds for forbidden-set policies. This motivates the problem of designing efficient routing schemes that do not suffer from non-convergence, for simple classes of policy such as forbidden-set.

W. Thomas and P. Weil (Eds.): STACS 2007, LNCS 4393, pp. 37–48, 2007.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph and $X \subseteq V$ a set of vertices (the extension to directed graphs is straightforward), and F be a set of edges. An (X, F)-*constrained path* is a path in G that does not use the edges of F and with no intermediate vertex in X (or simply X-constrained if F is empty). We denote by $G[Z]$ the subgraph of G induced by a set of vertices Z. We denote by $G_+[Z]$ the graph consisting of $G[Z]$ and weighted edges where an edge between x and y has weight d iff d is the length of a shortest path in G between x and y of length at least 2 with no intermediate vertex in Z. Between two vertices, one may have one edge without value and another one with value at least 2.

If we know the graph $G_+[Z]$, if $X \subseteq Z$ and every edge of F has its two ends in Z, we can get the length of a shortest (X, F)-constrained path in G between any $x, y \in Z$. The graph $G_+[Z]$ captures the separator structure of G since there is no edge between x, y in $G_+[X \cup \{x, y\}]$ iff X is a separator of x, y in G, thus the problem can be seen as constructing a distributed encoding of the separators of a graph. In all cases we say that we consider a *constrained path problem*.

Our objective is to label each vertex x of G by a label $J(x)$, as short as possible, in such a way that $G_+[Z]$ can be constructed from $\{J(x) : x \in Z\}$. If we can determine the lengths of shortest (X, F)-constrained paths from $\{J(x) : x \in Z\}$, where $X \subseteq Z$ and every edge of F has its two ends in Z, then we call $J(x)$ an (X, F)-*constrained distance labelling*.

The graph problem 'is there an X-constrained path from x to y?' is monadic second-order definable, so the result of Courcelle and Vanicat [5] implies that graphs of bounded clique width have a labelling with labels of $O(\log n)$ bits. However, the constant factor is a tower of exponentials in $cwd(G)$ and is impractical.

Our main result is a labelling scheme with labels of size $O(k^2 \log^2(n))$ where k is a bound on the *m-clique width* (*mcwd*) of the graph, a generalization of clique width that we will introduce. Since graphs with tree width (*twd*) k have *mcwd* at most $k + 3$, and graphs with clique width (*cwd*) k have *mcwd* at most k, the results follow for the case of tree width and clique width. Table 1 in [6] shows that the networks of some important major internet providers are of small tree width, between 10 and 20 and hence our constraint of dealing with graphs of small tree width or clique width is somehow realistic.

The labeling works as follows: given vertices between which we want to determine shortest paths and a set $Z \subseteq V$, we construct from $\{J(x) : x \in Z\}$ the weighted graph $G_+[Z]$. Then we can answer queries about 4-tuples (x, y, X, F) such that $X \cup \{x, y\} \subseteq Z$ and every edge of F has its two ends in Z by using only $G_+[Z]$: in particular the length of a shortest (X, F)-constrained path. The idea is not to repeat for each query the construction of $G_+[Y]$ for some set Y.

Our notation follows Courcelle and Vanicat [5]. For a finite set C of constants, a finite set F of binary function symbols, we let $T(F, C)$ be the set of finite well-formed terms over these two sets (terms will be discussed as labelled trees). The size $|t|$ of a term t is the number of occurrences of symbols from $C \cup F$. Its height $ht(t)$ is 1 for a constant and $1 + \max\{ht(t_1), ht(t_2)\}$ for $t = f(t_1, t_2)$.

Let a be a real number. A term t is said to be *a-balanced* if $ht(t) \leq a \log |t|$ (all logarithms are to base 2). Let t in $T(F_k, C_k)$ and $G = val(t)$, the graph obtained by evaluating t. For a node u in t, $val(t/u)$ is the subgraph represented by evaluating the subterm rooted at u.

3 The Case of Tree Width

Before presenting our main result on m-clique width graphs, we describe a labelling scheme for graphs of tree width k. A graph having tree width k can be expressed as the nondisjoint union of graphs of size $k+1$, arranged as nodes in a tree such that the set of tree nodes containing some graph vertex forms a connected subtree of the tree (often called a *tree decomposition*). We shall work with a different, algebraic representation of graphs.

3.1 Balanced Tree Width Expressions

Every graph of tree width k can be represented by an algebraic expression (term). A *j-source graph* is a graph with at most j distinguished vertices called *sources*, each tagged with a unique label from $\{1, \ldots, j\}$. Courcelle [7][8] shows that a graph has tree width k iff it is isomorphic to $val(t)$ for some term t whose leaves are $(k+1)$-source graphs and where every non-leaf node is labelled with one of the following operations, as illustrated in Figure 1.

- *Parallel composition*: The $(k+1)$-source graph $(G \;//\; H)$ is obtained from the disjoint union of $(k+1)$-source graphs G and H where sources having the same label are fused together into a single vertex.
- *Erasure*: For $a \in \{1, \ldots, k+1\}$, the unary operation $\mathsf{fg}_a(G)$ erases the label a and the corresponding source in G is no longer a source vertex.

As in Courcelle and Vanicat[5], we combine a parallel composition and a sequence of erasure operations to obtain a single binary operation, e.g. $//\ _{\mathsf{fg}_{a,b}}$. The term tree can be constructed given a tree decomposition of the graph – Corollary 2.1.1 of Courcelle [7] shows that given a tree decomposition of width k of a graph, it is possible to construct in linear time a term tree using at most $k+1$ source labels. The nodes of the term tree are the bags of the tree decomposition; hence the height and degree are unchanged.

The following result of Bodlaender shows how to obtain a balanced tree width expression with a small increase in tree width.

Lemma 1 (Bodlaender [9]). *Given a tree decomposition of width k and a graph G with n vertices, one can compute a binary tree decomposition of G of height at most $2\log_{5/4}(2n)$ and width at most $3k+2$ in time $O(n)$.*

3.2 Compact Forbidden-Set Routing for Small Tree Width

Assume we have an a-balanced term tree t for some constant a with $val(t) = G$, assume wlog assume that all sources are eventually erased in t. The vertices of

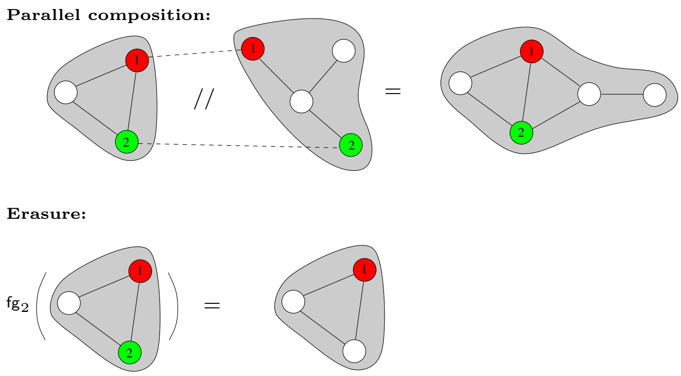

Fig. 1. The parallel composition and erasure operations for constructing graphs of tree width k

G are then in bijection with the erasure operations, so we shall use the same identifier u to refer to both a vertex in G and its unique corresponding erasure operation in t. We now describe a labelling $J(u)$ to compute the length of shortest X-constrained paths.

For a set $Y \subseteq \{1, \ldots, k+1\}$ of source labels and a $(k+1)$-source graph G, we denote by $G \setminus Y$ the induced subgraph of G obtained by removing the source vertices of G whose label is in Y. Every node u in t has a state $Q(u)$ associated with it, which for now assume to be the collection of graphs $\{val(t/u) \setminus Y : Y \subseteq \{1, \ldots, k+1\}\}$. As in Courcelle and Vanicat[5], the label $J(u)$ stores a string describing the *access path* from the root to the node in t representing u (rather than a leaf of t), and the state for every node adjacent to its access path (we assume that every vertex u is adjacent to its own access path). In addition, the label contains the source label of the node u in $val(t/u)$. If u has the source label s_u then the string is of the form

$$J(u) = (s_u, f_1, i_1, Q(s_{3-i_1}(u_1)), \ldots f_h, i_h, Q(s_{3-i_h}(u_h))$$

where h is the height of t, $f_1 \ldots f_h$ are the operations on the path, $i_1 \ldots i_h \in \{1, 2\}$ indicate whether to take the left or right branch and $s_1(u)$ (respectively $s_2(u)$) denote the left (respectively right) child of u in t. The states

$$Q(s_{3-i_1}(u_1))Q(s_{3-i_1}(u_2)) \ldots Q(s_{3-i_1}(u_h))$$

are the states of nodes adjacent to the access path for u. Since each set of at most $O(k)$ erasure operations can be identified with $O(k)$ bits and the term tree has height $O(\log n)$, the access path can be described using $O(k \log n)$ bits (excluding the space to store the states).

We now describe how to use the labelling to find the length of the shortest X-constrained path between u, v. Assume that $u, v \notin X$. For a vertex $x \in G$, we let $Path(x)$ be the path from the corresponding vertex x of t to the root. For a node u of t, let $X(u)$ be the subset of X whose corresponding erasure

Construct-Source-Distance-Graph(G)

Input: a j-source graph G
Output: the source distance graph H on j vertices
Set $w(u, v) = w(v, u) = 1$ if $\{u, v\} \in E(G)$
and ∞ otherwise
while (G contains a non-source node)
    do Let u be any non-source node in G
        for each pair of neighbours x, y of u
            do $w(x, y) = w(y, x)$
                $= \min\{w(x, u) + w(u, y), w(x, y)\}$
        Remove u from G
        Set $w(v, u) = w(u, v) = \infty$ for all v
return $H = G$

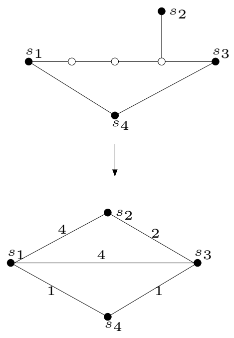

Fig. 2. Constructing source distance graphs by contracting paths of non-source nodes

operations are all ancestors of u in t (i.e. the subset of X represented by sources in $val(t/u)$).

We construct the graph $Rep(t)[X]$ by adding the subgraphs $val(t/w) \backslash X(w)$ for each node $w \in t$ *adjacent* to an access path $Path(x)$ for $x \in X \cup \{u, v\}$ (the states *on* the access path will be reconstructed from these adjacent states). To each vertex $y \in Rep(t)[X]$, associate two pieces of information: a unique identifier $I(y)$ for the corresponding erasure node in t and the source label s_y of y in $val(t/y)$. Then add edges of length zero corresponding to parallel compositions between nodes $x, y \in Rep(t)[X]$ where $I(x) = I(y)$ and $s_x = s_y$. The length of the shortest X-constrained path between u, v in G equals the length of the shortest path in $Rep(t)[X]$ between two vertices x, y where x is a source corresponding to node u in G and y is a source corresponding to v in G.

Now we consider how to efficiently represent the state $Q(u)$. At first it might seem that one needs to store all $2^{O(k)}$ graphs, one for every set of deleted sources. From $val(t/u)$ we construct a compressed graph H called the *source distance graph* with the property that for any set Y of sources and sources x, y, the distance between x, y in $val(t/u) \setminus Y$ equals their distance in $H \setminus Y$. The graph is constructed by contracting paths of non-source vertices in $val(t/u)$, as in Figure 2. Since the edge weights in the source distance graph are in the range $[1, n]$, it can be represented using $O(k^2 \log n)$ bits. This gives labels $J(x)$ of size $O(k^2 \log^2 n)$ bits.

The correctness of the labelling scheme relies on the fact that the connectivity of sources in $G \mathbin{//} H$ is completely determined by their connectivity in G and H: sources u, v are connected in $G \mathbin{//} H$ iff there is a source labelled r in both G, H and paths $u - r$ in G and $r - v$ in H. For routing, we can augment the labelling to compute the next hop on the shortest X-constrained path by associating with each edge (x, y) in the source distance graph $Q(u)$ the next hop (possibly a non-source vertex) on the shortest non-source path represented by the contracted edges from x to y. We can then use this information to construct a *compact* *routing scheme* that routes on shortest X-constrained paths with routing tables of size asymptotically equal to the X-constrained distance labels.

Theorem 1. *Graphs of tree width k have X-constrained distance labels of size $O(k^2 \log^2 n)$ bits, where n is the number of vertices.*

4 The Case of m-Clique Width

We now extend the results of the previous section to clique width graphs. Note that the concept of tree width (*twd*) is weaker than clique width (*cwd*) : any graph with tree width k has clique width at most 3.2^{k-1} [10] but cliques have cwd 2 and unbounded twd. We begin by introducing some tools: balanced terms, the new notion of m-clique width and the main construction. Due to space restrictions, we outline our results and defer some details to a full version.

4.1 Balanced m-Clique Width Expressions

Let L be a finite set of vertex labels. A *multilabelled graph* is a triple $G = (V_G, E_G, \delta_G)$ consisting of a graph (V_G, E_G) and a mapping δ_G associating with each x in V_G the set of its labels, a subset of L. A vertex may have zero, one or several labels.

The following constants will be used: for $A \subseteq L$ we let $\mathbf{A}$ be a constant denoting the graph G with a single vertex u and $\delta_G(u) = A$. We write $\mathbf{A}(u)$ if we need to specify the vertex u. The following binary operations will be used: for $R \subseteq L \times L$, *relabellings* $g, h : L \longrightarrow \mathcal{P}(L)$ ($\mathcal{P}(L)$ is the powerset of L) and for multilabelled graphs G and H we define $K = G \otimes_{R,g,h} H$ iff G and H are disjoint (otherwise we replace H by a disjoint copy) where

$$
\begin{aligned}
V_K &= V_G \cup V_H \\
E_K &= E_G \cup E_H \cup \{\{v,w\} : v \in V_G, w \in V_H, R \cap (\delta_G(v) \times \delta_H(w)) \neq \varnothing\} \\
\delta_K(x) &= (g \circ \delta_G)(x) = \{a : a \in g(b), b \in \delta_G(x)\} \text{ if } x \in V_G \\
\delta_K(x) &= (h \circ \delta_H)(x) \text{ if } x \in V_H
\end{aligned}
$$

As in the operations by Wanke [11] we add edges between two disjoint graphs, that are the 2 arguments of (many) binary operations. This is a difference with clique width [12] using a single binary operation.

Notation and definitions. We let F_L be the set of all binary operations $\otimes_{R,g,h}$ and C_L be the set of constants $\{\mathbf{A} : A \subseteq L\}$. Every term t in $T(F_L, C_L)$ denotes a multilabelled graph $val(t)$ with labels in L, and every multilabelled graph G is the value of such a term for large enough L. We let $mcwd(G)$ be the minimum cardinality of such a set L and call this number the *m-clique width* of G. We now compare *mcwd* with *cwd* and *twd* [5,12].

Proposition 1. *For every unlabelled undirected graph G,*

$$mcwd(G) \leq twd(G) + 3$$
$$mcwd(G) \leq cwd(G) \ \leq \ 2^{mcwd(G)+1} - 1$$

It follows that the same sets of graphs have bounded clique width and bounded m-clique width. Our motivation for introducing m-clique width is that we can prove the following result:

Proposition 2. *There exists a constant a such that, every graph of m-clique width k is the value an a-balanced term in $T(F_L, C_L)$ for some set L of cardinality at most $2k$.*

The proof is deferred to the full version. The above result is very useful since no such result is known for obtaining balanced clique width expressions.

4.2 Adjacency Labelling for m-Clique Width Graphs

For a vertex $x \in G$, let $Path(x)$ be the path $(u_m = x, u_{m-1}, ..., u_0)$ from the corresponding node x of t to the root ($=u_0$). For a term t, let $m = ht(t)$ be its height. We now describe how to construct an adjacency labelling $I(x)$. Let $I(x) = (L_m, e_{m-1}, D_{m-1}, L_{m-1}, e_{m-2}, D_{m-2}, ..., e_0, D_0, L_0)$ where $L_m = A$ if $\mathbf{A} \in C_{[k]}$ is the constant at leaf x in t; L_i is the set of labels of the vertex x in the graph $val(t/u_i)$ for $i = 0, ..., m$. For $i = 0, ..., m-1$, we define $e_i = 1$ if u_{i+1} is the left son of u_i and D_i is the set of labels $\{j'\}$ such that $(j, j') \in R$ for some j in L_{i+1} where $\otimes_{R,g,h}$ occurs at node u_i. Similarly, $e_i = 2$ if u_{i+1} is the right son of u_i and D_i is the set of labels $\{j'\}$ such that $(j', j) \in R$ for some j in L_{i+1} where $\otimes_{R,g,h}$ occurs at node u_i. Each label $I(x)$ has size $O(km)$ and is computable from t in time $O(k^2 ht(t))$, hence at most $O(nk^2 ht(t))$ to compute the entire labelling.

Fact 2. *From the sequences $I(x)$ and $I(y)$ for two distinct vertices x and y, one can determine whether they are linked in G by an edge.*

Proof. From the integers $e_{m-1}, ..., e_0, e'_{m'-1}, ..., e'_0$ in the sequences

$$I(x) = (L_m, e_{m-1}, D_{m-1}, L_{m-1}, e_{m-2}, D_{m-2}, ..., e_0, D_0, L_0)$$
$$I(y) = (L'_{m'}, e'_{m'-1}, D'_{m'-1}, L'_{m'-1}, ..., e'_0, D'_0, L'_0)$$

one can determine the position i in $Path(x)$ and $Path(y)$ of the least common ancestor u_i of x and y. Wlog we assume x below (or equal to) the left son of u_i. Then x and y are adjacent in G iff $D_i \cap {L'}_{i+1} \neq \varnothing$. This is equivalent to $D'_i \cap L_{i+1} \neq \varnothing$. Since the computations of Fact 2 take time $O(ht(t))$ for each pair x, y, we have the following.

Fact 3. *From $\{I(x) : x \in X\}$ for a set $X \subseteq V$, one can determine $G[X]$ in time $O(|X|^2 ht(t))$ (k is fixed).*

We have thus an *implicit representation* in the sense of Kannan et al.[13] for graphs of *mcwd* at most k, using labels of size $O(k \log n)$. □

4.3 Enriching the Adjacency Labelling

We now show how to enrich $I(x)$ to achieve the following.

Proposition 3. *Fix k. For t in $T(F_k, C_k)$ with $G(V, E) = val(t)$ one can build a labelling J such that from $\{J(x) : x \in X\}$ for any $X \subseteq V$, one can determine $G_+[X]$ in polynomial time in $|X|$ and $ht(t)$.*

We shall now show how to do this with labels of size $O(k^2 \log^2(n))$ where k is the m-clique width of G. The basic idea is as follows. From $\{I(x) : x \in X\}$ for any $X \subseteq V$, one can reconstruct $G[X]$. For $G_+[X]$ we need paths going out of X, or at least their lengths. If u is a node of a path $Path(x)$ for some x in X, and w is a son of u not on any path $Path(y)$ for y in X, then we compute the lengths of at most k^2 shortest paths running through the subgraph of G induced on the leaves of t below w, and we insert this matrix of integers at the position corresponding to u in the label $J(x)$.

We shall work with a graph representation of terms in $T(F_k, C_k)$. With a term t in $T(F_k, C_k)$, we associate a graph $Rep(t)$ having directed and undirected edges. The vertices of $Rep(t)$ are the leaves of t and the pairs (u, i) for u a node of t and $i \in [k]$ that labels some vertex x in $val(t/u)$. The undirected edges are $(u_1, i) - (u_2, j)$ whenever u_1, u_2 are respectively the left and right sons of some u labelled by $\otimes_{R,g,h}$ and $(i, j) \in R$. The directed edges are of 3 types :

1. $u \longrightarrow (u, i)$ for u a leaf labelled by $\mathbf{A}$ and $i \in A$.
2. $(u_1, i) \longrightarrow (u, j)$ whenever u_1 is the left son of u, u is labelled by $\otimes_{R,g,h}$ and $j \in g(i)$.
3. $(u_2, i) \longrightarrow (u, j)$ whenever u_2 is the left son of u, u is labelled by $\otimes_{R,g,h}$ and $j \in h(i)$.

As an example, the left half of Figure 3 shows a term t (thick edges) and the graph $Rep(t)$ (fine edges). We use $\longrightarrow^*$ to denote a directed path; $\longleftarrow^*$ denotes the reversal of a directed path.

Fact 4. *For a vertex u of G below or equal to a node w of t, u has label i in $val(t/w)$ iff $u \longrightarrow^* (w, i)$ in $Rep(t)$.*

Fact 5. *For distinct vertices u, v of G : $u - v$ in G iff we have a mixed (directed/undirected) path $u \longrightarrow^* (w, i) - (w', j) \longleftarrow^* v$ in $Rep(t)$ for some w, w', i, j.*

We call such a path an *elementary path* of $Rep(t)$. A *walk* is a path where vertices may be visited several times. A *good walk* in $Rep(t)$ is a walk that is a concatenation of elementary paths. Its *length* is the number of undirected edges it contains (the number of elementary paths).

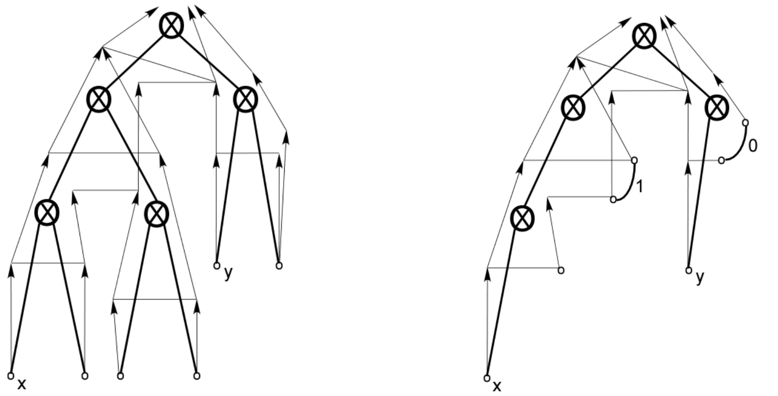

Fig. 3. A term t and the graph *Rep*(t), and the graph *Rep*(t)[$\{x, y\}$] with some valued edges from *Rep*(t)$_+$[X]

Fact 6. *There is a walk $x - z_1 - ... - z_p - y$ in G iff there is in Rep(t) a good walk*

$$W = x \longrightarrow^* - \longleftarrow^* z_1 \longrightarrow^* - \longleftarrow^* ... \longrightarrow^* - \longleftarrow^* z_p \longrightarrow^* - \longleftarrow^* y$$

For a nonleaf vertex u, a *u-walk* in *Rep*(t) is a walk that is formed of consecutive steps of a good walk W and is of the form

$$(u, i) \longleftarrow^* z \longrightarrow^* - \quad \longrightarrow^* \cdots \longrightarrow^* (u, j)$$

where all vertices except the end vertices $(u, i), (u, j)$ are of the forms u, or w or (w, l) for w strictly below u in t. Its *length* is defined as the number of undirected edges.

We let $Min(u, i, j)$ be the smallest length of a u-walk from (u, i) to (u, j), or ∞ if no such u-walk exists. Clearly $Min(u, i, i) = 0$ ((u, i) is a vertex of *Rep*(t), so Fact 4 applies). We let $MIN(u)$ be the $S \times S$ matrix of all such integers $Min(u, i, j)$, where S is the set of labels p such that (u, p) is a vertex of *Rep*(t). It can be stored in space $O(k^2 \log n)$ since n bounds the lengths of shortest u-walks in *Rep*(t).

Fact 7. *If in a good walk we replace a u-walk from (u, i) to (u, j) by another one also from (u, i) to (u, j) we still have a good walk.*

We are now ready to define $J(x)$ for x a vertex of G. We recall that $Path(x)$ is the path $(u_m = x, u_{m-1}, ..., u_0)$ in t from a leaf x to the root u_0, and

$$I(x) = (L_m, e_{m-1}, D_{m-1}, L_{m-1}, e_{m-2}, D_{m-2}, ..., e_0, D_0, L_0).$$

We let then

$$J(x) = (L_m, e_{m-1}, D_{m-1}, L_{m-1}, M_{m-1}, f_{m-1}, e_{m-2}, D_{m-2}, ..., e_0, D_0, L_0, M_0, f_0)$$

where f_i is the binary function symbol (some $\otimes_{R,g,h}$) occurring at node u_i, $M_i = MIN(RightSon(u_i))$ if $e_i = 1$ and $M_i = MIN(LeftSon(u_i))$ if $e_i = 2$ for each $i = 0, ..., m-1$.

Fact 8. *$J(x)$ has size $O(k^2 ht(t)\log(n))$.*

Proof (Proof of Proposition 3). From the set $\{J(x) : x \in X\}$, one can construct the graph $G[X]$ by Fact 3. We let $Rep(t)[X]$ be the subgraph of $Rep(t)$ induced by its vertices that are either elements of X (hence leaves of t), or of the form (w, i) if w is a son of a node u on a path $Path(x)$ for some x in X.

Because a sequence $J(x)$ contains the function symbols f_i and the index sets S of the matrices M_i, we can determine from it the edges of $Rep(t)$, not only between vertices of the form (u, i) for nodes u in $Path(x)$ but also between these vertices and those of the form (w, i) for w that are sons of such nodes u but are not necessarily in $Path(x)$.

It remains to determine the lengths of shortest good walks in $Rep(t)$ in order to get the valued edges of $G_+[X]$. We let $Rep(t)_+[X]$ be the graph $Rep(t)[X]$ augmented with the following integer valued undirected edges: $(u, i) - (u, j)$ valued by $Min(u, i, j)$ whenever this integer (possibly 0) is not ∞.

Example: For the term t in the left half of Figure 3 and $X = \{x, y\}$, the right half of the Figure shows the graph $Rep(t)[X]$ augmented with two valued edges $(u, i) - (u, j)$ for 2 of the 3 nodes u which are not on the paths $Path(x)$ and $Path(y)$ but are sons of nodes on these paths. These 3 nodes yield 5 vertices in the graph $Rep(t)[X]$. Each of these vertices has a loop with value 0 (these loops are not shown). We show the two non-loop edges labelled by 0 and 1.

The shortest good walks in $Rep(t)$ that define the valued edges of $G_+[X]$ are concatenations of edges of $Rep(t)[X]$ (which we have from the $J(x)$'s) and w-walks of minimal lengths for nodes w that are not on the paths $Path(x)$ but are sons of nodes on these paths. We need not actually know these w-walks exactly; we only need the minimal length of one of each type. This information is available from the matrices $MIN(w)$ which we have in the $J(x)$'s. We can thus build the valued graph $Rep(t)_+[X]$, and the desired values are lengths of shortest paths in the graph $Rep(t)_+[X]$ under the alternating edge constraints in Fact 5.

This proves Proposition 3. □

Combining Propositions 2 and 3 gives the following.

Theorem 9. *For a graph G of m-clique width at most k on n vertices, one can assign to vertices labels $J(x)$ of size $O(k^2 \log^2 n)$ such that from $\{J(x) : x \in X\}$ for any set $X \subseteq V$, one can determine the graph $G_+[X]$ in time $O(|X|^3 \log n)$. The graph G must be given along with an mcwd expression of width at most k.*

The problem of determining for a given graph its m-clique width and the corresponding expression is likely to be NP-hard because the corresponding one for clique width is NP-complete [14]. A cubic algorithm that constructs non-optimal clique width expressions given by Oum [15] may be used.

4.4 Compact Forbidden-Set Routing for Small mcwd

We now describe how to use the labelling J to build a compact routing scheme. Recall that the construction of J is based on matrices that give for each node u of a term t the length of a shortest u-walk in $Rep(t)$ from (u, i) to (u, j). Storing the sequence of vertices of the corresponding path in G uses space at most space $n \log n$ instead of $\log n$ for each entry (assuming there are n vertices numbered from 1 to n, so that a path of length p uses space $p \log n$). The corresponding labelling $J'(x)$ uses for each x space $O(k^2 n \log^2 n)$.

We assume that $X \cup \{x, y\} \subseteq Z$ and every edge of F has its two endpoints in Z. For a compact routing scheme, it suffices to be able to construct the path in a distributed manner, by finding the next hop at each node. Here is such a construction, that for such a set $Z \subseteq V$ gives the length of a shortest (X, F)-constrained path from x to y together with z, the first one not in Z on the considered shortest path. For this, we need only store, in addition to the length of a shortest u-walk in $Rep(t)$ from (u, i) to (u, j) (in the matrix $MIN(u)$) its first and last vertices. This uses space $3 \log n$ instead of $\log n$ for each entry. The corresponding labelling $J''(x)$ uses for each x space $O(k^2 \log^2 n)$. This gives the following compact forbidden-set routing scheme.

Theorem 10. *Let each node have a forbidden set of size at most r. Then graphs of m-clique width at most k have a compact forbidden-set routing scheme using routing tables of size $O(rk^2 \log^2 n)$ bits and packet headers of size $O(rk^2 log^2 n)$ bits.*

Proof. Given an mcwd decomposition of $G = (V, E)$ of width at most k and a set $S(u) \subseteq V$ stored at each node u with $|S(u)| \leq r$, the routing table at u is the label J'' as above. To send a packet from u to v on an $S(u)$-constrained path, u writes into the packet header the label $J''(v)$ for the destination and the labels $\{J''(x) : x \in S(u)\}$. Then u forwards the packet to a neighbour w that minimizes the minimizes the distance from w to v, obtained as described above. Since the distances computed are exact distances, the packet always progresses towards the destination and will never loop. Note that if the paths are only approximately shortest, there may be loops – in this case, w adds its label $J''(w)$ to the packet header, setting $S'(u) = S(u) \cup \{w\}$ and we ask for the shortest $S'(u)$-constrained path from w to v. The price we pay here is that the packet headers grow with the length of the path.

In this case however, we may need to compute graphs $G_+[Z]$ for larger and larger sets Z. □

5 Open Problems

A major problem is to get good bounds on planar graphs. Using the $O(\sqrt{n})$ recursive separator structure gives $\tilde{O}(n)$ bits per label, but we believe it is possible to do much better. We would also like to solve other constrained path problems using $G_+[X]$, using the separator structure that we encode.

References

1. Thorup, M., Zwick, U.: Compact routing schemes. In: SPAA '01: Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures, New York, NY, USA, ACM Press (2001) 1–10
2. Varadhan, K., Govindan, R., Estrin, D.: Persistent route oscillations in inter-domain routing. Technical report, USC/ISI (1996)
3. Griffin, T.G., Shepherd, F.B., Wilfong, G.: The stable paths problem and interdomain routing. IEEE/ACM Trans. Netw. **10** (2002) 232–243
4. Feigenbaum, J., Karger, D., Mirrokni, V., Sami, R.: Subjective-cost policy routing. In: Lecture Notes in Computer Science. Volume 3828. (2005) 174–183
5. Courcelle, B., Vanicat, R.: Query efficient implementation of graphs of bounded clique-width. Discrete Applied Mathematics **131** (2003) 129–150
6. Gupta, A., Kumar, A., Thorup, M.: Tree based mpls routing. In: SPAA '03: Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures, New York, NY, USA, ACM Press (2003) 193–199
7. Courcelle, B. In: Graph decompositions. (2006) Chapter of a book in preparation. Available at `www.labri.fr/perso/courcell/Textes/ChapitreDecArbos.pdf`.
8. Arnborg, S., Courcelle, B., Proskurowski, A., Seese, D.: An algebraic theory of graph reduction. J. ACM **40** (1993) 1134–1164
9. Bodlaender, H.L.: NC-algorithms for graphs with small treewidth. In: Proc. 14th Workshop Graph-Theoretic Concepts in Computer Science WG'88, Springer-Verlag, Lecture Notes in Computer Science 344 (1989) 1–10
10. Corneil, D.G., Rotics, U.: On the relationship between clique-width and treewidth. SIAM J. Comput. **34** (2005) 825–847
11. Wanke, E.: k-nlc graphs and polynomial algorithms. Discrete Applied Mathematics **54** (1994) 251–266
12. Courcelle, B., Olariu, S.: Upper bounds to the clique width of graphs. Discrete Appl. Math. **101** (2000) 77–114
13. Kannan, S., Naor, M., Rudich, S.: Implicit representation of graphs. SIAM J. Discret. Math. **5** (1992) 596–603
14. Fellows, M.R., Rosamond, F.A., Rotics, U., Szeider, S.: Clique-width minimization is np-hard. In: STOC 2006: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing, New York, NY, USA, ACM Press (2006)
15. il Oum, S.: Approximating rank-width and clique-width quickly. In Kratsch, D., ed.: WG. Volume 3787 of Lecture Notes in Computer Science., Springer (2005) 49–58