Brief Announcement: On Local Representation of Distances in Trees

C. Gavoille   A. Labourel

Laboratoire Bordelais de Recherche Informatique (LaBRI)
Université Bordeaux

PODC 2007, 12-15 August
XML is a generic framework for storing any amount of text or any data whose structure can be represented as a tree.
XML search and preprocessing

- We need a data structure that supports efficiently structural queries such as finding all book items of a specified author and a specified price.
- We add to the XML documents a big hash table containing all words and tags (like book or price) of the documents.
- We give to each node of the XML documents a unique label.
- We associate with each entry in the hash table the labels of all nodes containing the corresponding word or tag.
- By giving to the nodes suitable labels that permit ancestor queries, computing structural queries can be done by retrieving labels of relevant nodes and select those having the appropriate ancestor relationships.
**k-relationship scheme**

**Definition**

Two nodes $u$ and $v$ are $(k_1, k_2)$-related if $u$ is at distance $k_1$ of $w = \text{lca}(u, v)$ and $v$ is at distance $k_2$ of $w$.

A $k$-relationship scheme consists in assigning labels to nodes of a tree such that we can decide if two nodes are $(k_1, k_2)$-related ($k_1, k_2 \leq k$) only by examining their labels.
Trivial solution

A trivial solution is to store for each node $v$ identifiers of $v$ and its $k$ closest ancestors.

\begin{center}
\begin{tikzpicture}
    \node (1) at (0,0) {1};
    \node (2) at (-1,-1.5) {2};
    \node (3) at (-2,-3) {3};
    \node (4) at (-3,-4.5) {4};
    \node (5) at (-4,-6) {5};
    \node (6) at (1,-1.5) {6};
    \node (7) at (2,-3) {7};
    \node (8) at (3,-4.5) {8};

    \draw (1) -- (2);
    \draw (2) -- (3);
    \draw (3) -- (4);
    \draw (4) -- (5);
    \draw (1) -- (6);
    \draw (6) -- (7);
    \draw (7) -- (8);

    \node at (2,-4.75) {v stores (7,6,2,1,8)};
    \node at (-2,-4.75) {u stores (5,4,3,2,1)};
\end{tikzpicture}
\end{center}
## Related works

**Theorem (Astrup, Bille, and Rauhe 2003)**

- A \( k \)-relationship scheme can be achieved with \( \log n + O(k^2 \log(k \log n)) \) bits per node.
- \( \log n + \log \log n + \Omega(1) \) bits per node are needed for a \( 1 \)-relationship scheme (siblings).
Our result

**Theorem**

A k-relationship scheme can be achieved with

\[ \log n + O(k \log(k \log(n/k))) \] bits per node.

2nd order term: \( k^2 \log(k \log n) \rightarrow k \log(k \log(n/k)) \)

for \( k = 1 \) : \( 5 \log \log n \rightarrow 2 \log \log n \)

- The principle of our scheme is the same as the trivial solution: store for each node \( v \) identifiers of \( v \) and its \( k \)-closest ancestors.
- The difficulty is to find identifiers such that we can encode the \( k \)-closest ancestors of a node in a very efficient way.
- We achieve this with a special decomposition of the tree using separators.
Concluding remarks and open problems

Our scheme implies that distances in trees can be computed with labels of \( \log n + o(\log n) \) bits if the distance is \( o(\log n / \log \log n) \). One may ask if it is possible to:

- Design a distance labeling scheme for trees with \( \log n + o(\log n) \) bit labels and for larger distances, say for distances up to \( \log n \).
- Design a distance labeling scheme for small distances for bounded treewidth graphs.