Informative Labeling Schemes for the Least Common Ancestor Problem

Saverio Caminiti[1]

Joint work with:
Irene Finocchi[1] and Rossella Petreschi[1]

[1] Computer Science Department, *Sapienza* University of Rome
{caminiti,finocchi,petreschi}@di.uniroma1.it

Abstract. We address the problem of labeling the nodes of a tree such that one can determine the identifier of the least common ancestor of any two nodes by looking only at their labels. This problem has application in routing and in distributed computing in peer-to-peer networks. A labeling scheme using $\Theta(\log^2 n)$-bit labels has been presented by Peleg. By engineering this scheme and a new one due to the authors, we obtain a variety of data structures with the same asymptotic performances. We conduct a thorough experimental evaluation of all these data structures. Our results clearly show which variants achieve the best performances in terms of space usage, construction time, and query time.

Effective representations of large, geographically dispersed communication networks should allow the users to efficiently retrieve information about the network in a distributed and localized way. Labeling schemes provide an answer to this problem by assigning labels to the network nodes in such a way that queries can be computed alone from the labels of the involved nodes, without any extra information source. The primary goal of a labeling scheme is to minimize the maximum label length, while keeping queries fast. Adjacency labeling schemes were first introduced by Breuer and Folkman in [5, 6], and further studied in [11]. The interest in informative labeling schemes, however, was revived only more recently, after Peleg showed the feasibility of the design of efficient labeling schemes capturing distance information [15]. Since then, upper and lower bounds for labeling schemes have been proved on a variety of graph families (including weighted trees, bounded arboricity graphs, intersection-based and c-decomposable graphs) and for a large variety of queries, including distance [2, 8, 10], tree ancestry [1, 3], flow and connectivity [13]. In spite of a large body of theoretical works, to the best of our knowledge only few experimental investigations of the efficiency of informative labeling schemes have been addressed in the literature [8, 12].

In our work [7] we focus on labeling schemes for answering least common ancestor queries in trees. Labeling schemes for least common ancestors are mainly useful in routing messages on tree networks: the ability to compute the identifier of the least common ancestor of any two nodes u and v turns out to be useful when a message has to be sent from u to v in the network, because the message

has to go through $lca(u, v)$. Other applications are related to query processing in XML search engines and distributed computing in peer-to-peer networks (see, e.g., [3, 4, 12]). In [16], Peleg has proved that for the class of n-node trees there exists a labeling scheme for least common ancestors using $\Theta(\log^2 n)$-bit labels, which is also shown to be asymptotically optimal.

We finally remark that, when node levels are known, it is trivial to compute the distance between any two nodes given their *lca*. Therefore, all the data structures considered in this work can be easily exploited to answer distance queries.

1 Labeling Schemes

All the tree labeling schemes that we studied follow the same basic approach: the tree is decomposed into a set of node disjoint paths, that we will call *solid paths*, and information related to the highest node in each path, called *head* of the path, is suitably encoded into the node labels. In the following we will consider different path decomposition approaches, then we will describe two possible ways of designing node labels. Different combinations of these two ingredients yield different labeling schemes: one of them coincides with the tree labeling scheme for least common ancestors originally proposed by Peleg in [16].

Path Decompositions. Let T be a tree with n nodes rooted at a given node r. For any node u, we denote its parent and its level in T by $p(u)$ and $\ell(u)$, respectively. We assume that the root has level 0. We also denote by T_u the subtree of T rooted at u and by $|T_u|$ the number of its nodes. In all the decompositions, for any solid path π, we call *head* the of π node with smallest level in π.

Decomposition by Large Child. This decomposition hinges upon the distinction between small and large nodes: a nonroot node v with parent u is called *small* if $|T_v| \leq |T_u|/2$, i.e., if its subtree contains at most half the number of nodes contained in its parents' subtree. Otherwise, v is called *large*. It is not difficult to see that any node has at most one large child: we will consider the edge to that large child, if any, as a solid edge. Solid edges induce a decomposition of the tree into solid paths (some path may consist of a single node).

Decomposition by Maximum Child. This is a minor variant of the previous decomposition, using a relaxed definition of large nodes: a nonroot node v with parent u is considered a *maximum child* of u if $|T_v| = \max_{w:(u,w)\in T} |T_w|$. If two or more children of u satisfy this condition, ties are broken arbitrarily. The edge to the maximum child is considered as a solid edge.

Decomposition by Rank. In this decomposition, an edge (u, v) is solid if and only if $\lceil \log |T_u| \rceil = \lceil \log |T_v| \rceil$. It is not difficult to prove that for any node u there exists at most one child v such that (u, v) is solid (see, e.g., [9, 14]). This implies that solid edges univocally partition the tree into disjoint paths.

Label Structure. Now we present two different ways of constructing node labels. When combined with any of the path decompositions, both schemes yield labels of size $O(\log^2 n)$. Due to the lack of space, we omit the descriptions of query algorithms.

Peleg scheme. The first scheme is based on a depth-first numbering of the tree T: as a preprocessing step, each node v is assigned an interval $Int(v) = [DFS(v); DFS(w)]$, where w is the last descendent of v visited by the depth-first tour and $DFS(x)$ denotes the depth-first number of node x. The label of each node v of the tree is defined as $label(v) = < Int(v), list(v) >$; where $list(v)$ contains information related to all the heads $(t_1, t_2, \ldots, t_h)$ of solid paths from the root of T to v: for each head t_i, $list(v)$ contains a quadruple $(t_i, \ell(t_i), p(t_i), succ_v(t_i))$, where $succ_v(t_i)$ is the unique child of t_i on the path to node v. We remark that this is slightly different (and optimized) with respect to the scheme originally proposed in [16].

CFP scheme. This scheme avoids the use of depth-first numbers as well as of successors. The label of each node v of the tree is now defined as $label(v) = < isHead(v), list(v) >$. The Boolean value $isHead(v)$ discriminates whether v is the head of its solid path or not. As in the previous scheme, $list(v)$ contains information related to all the heads $(t_1, t_2, \ldots, t_h)$ of solid paths from the root of T to v. In this case, the information for each head is less demanding and $list(v)$ consists just of a sequence of triples: $list(v) = [\,(t_1, \ell(t_1), p(t_1)), \ldots, (t_h, \ell(t_h), p(t_h)), (v, \ell(v), p(v))\,]$; where t_1 always coincides with the root of T. The sentinel triple $(v, \ell(v), p(v))$ is not necessary when v is head of its solid path, since in this case $t_h = v$.

2 Experimental Results

We implemented six labeling schemes, obtained by combining the three path decompositions and the two label structures. Each scheme comes in two variants: word aligned and bit aligned.

We tested them on instances consisting of both synthesized (randomly generated trees[1]) and of real test sets (spanning trees of real networks have been obtained from data provided by the CAIDA project). In our experiments we averaged each data point on 1000 different instances. When computing running times of query operations, we averaged the time on (at least) 10^6 random queries.

Main objectives that we considered to evaluate the data structures include space usage, construction time, and query time. Other structural measures have been used to study the effect of the different path decompositions on the labeling schemes: among them, we considered the number of paths in which the tree is decomposed, the average and maximum length of paths, and the variance of path lengths.

The main findings of our experiments can be summarized as follows.

- Among different path decompositions, those that generate the smallest number of paths (with the largest average path length) appear to be preferable in order to minimize the total size of the data structure.
- CFP scheme achieves the best performances in terms of space usage and construction time.

[1] We tested on both random trees generated with uniform distribution and random balanced trees generated using a variety of balancing factors.

– Peleg scheme, used with Maximum Child path decomposition, exhibits the fastest query times.
– All the data structures are very fast in practice. Although node labels have size $O(\log^2 n)$, only a small fraction of the labels is considered when answering random queries: typically, no more than a constant number of words per query is read in all our experiments. However, query times slightly increase with the instance size due to cache effects.
– Data structures implemented with bit alignment save 20% up to 40% of the space, but increase the query times approximately by a factor 1.3 on our data sets. The space saving reduces as the instance size gets larger.

3 Future Directions

The experimental analysis of the query times has raised an interesting theoretical question that deserves further investigation. The experiments suggest that, even if the instance size increases, the average number of list elements scanned during (random) queries remains almost constant: thus, using labels of size $\Theta(\log n)$, it appears that one can answer the majority of queries in constant time. Moreover, as far as labels are designed and the query algorithms work, if a query on two nodes u and v fails, then u and v must belong to a subtree of small size and must be rather close to each other. It may be therefore interesting to devise an alternative approach for answering these failing queries, to be combined with the use of node labels for "long-distance" queries: the alternative approach, for instance, might exploit a few extra, localized information sources.

References

1. S. Abiteboul, S. Alstrup, H. Kaplan, T. Milo, and T. Rauhe. Compact labeling schemes for ancestor queries. In *SIAM J. on Comp.*, 35(6), 1295–1309, 2006.
2. S. Alstrup, P. Bille, and T. Rauhe. Labeling schemes for small distances in trees. *SIAM J. on Disc. Math.*, 19(2), 448–462, 2005.
3. S. Alstrup, C. Gavoille, H. Kaplan and T. Rauhe. Nearest common ancestors: a survey and a new distributed algorithm. In *Proc. of SPAA'02*, 258–264, 2002.
4. N. Bonichon, C. Gavoille, and A. Labourel. Short labels by traversal and jumping. In *Proc. of SIROCCO'06*, 143–156, 2006.
5. M.A. Breuer. Coding the vertexes of a graph. *IEEE Trans. on Inf. Th.*, 12, 148–153, 1966.
6. M.A. Breuer and J. Folkman. An unexpected result on coding the vertices of a graph. *J. of Math. Analysis and App.*, 20, 583–600, 1967.
7. S. Caminiti, I. Finocchi, and R. Petreschi. Engineering Tree Labeling Schemes: a Case Study on Least Common Ancestors. In *Proc. of ESA'08*, 234–245, 2008.
8. E. Cohen, E. Halperin, H. Kaplan and U. Zwick. Reachability and Distance Queries via 2-hop Labels. In *Proc. of SODA'02*, 937–946, 2002.
9. R. Cole and R. Hariharan. Dynamic LCA Queries on Trees. *SIAM J. on Comp.*, 34(4), 894–923, 2005.
10. C. Gavoille, D. Peleg, S. Perennes and R. Raz. Distance labeling in graphs. In *Proc. of SODA'01*, 210–219, 2001.
11. S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. In *Proc. of STOC'88*, 334–343, 1988.
12. H. Kaplan, T. Milo and R. Shabo. A Comparison of Labeling Schemes for Ancestor Queries. In *Proc. of SODA'02*, 954–963, 2002.
13. M. Katz, N.A. Katz, A. Korman and D. Peleg. Labeling schemes for flow and connectivity. *SIAM J. on Comp.*, 34(1), 23–40, 2004.
14. T. Kopelowitz and M. Lewenstein. Dynamic weighted ancestors. In *Proc. of SODA'07*, 565–574, 2007.
15. D. Peleg. Proximity-preserving labeling schemes and their applications. In *Proc. of WG'99*, 30–41, 1999.
16. D. Peleg. Informative labeling schemes for graphs. *Th. Comp. Sci.*, 340, 577–593, 2005. Preliminary version in *Proc. of MFCS'00*, LNCS 1893, 579–588, 2000.