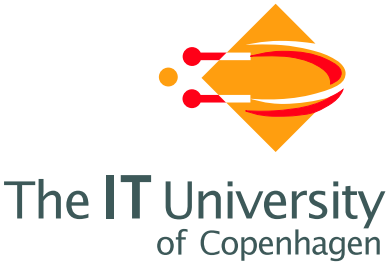

Lower Bounds for Labeling Schemes Supporting Ancestor, Sibling, and Connectivity Queries

Stephen Alstrup
Theis Rauhe

IT University Technical Report Series **TR-2001-10**

ISSN 1600–6100 **December 2001**

ISSN 1600–6100

ISBN 87-7949-013-1

Copies may be obtained by contacting:

IT University of Copenhagen
Glentevej 67
DK-2400 Copenhagen NV
Denmark

Telephone: **+45 38 16 88 88**
Telefax: **+45 38 16 88 99**
Web **`www.it-c.dk`**

Lower Bounds for Labeling Schemes Supporting Ancestor, Sibling, and Connectivity Queries

Stephen Alstrup* Theis Rauhe*

Abstract

Given a tree, a labeling scheme assigns a label, which is a binary string, to each node. Given the labels of any two nodes u and v, it can be determined, if a certain property holds for u and v, alone from these labels.

For trees of size n, we show that any labeling scheme for testing ancestor relation uses labels of size $\log n + \Omega(\log\log n)$ to the nodes. If the labels has to distinguish the nodes, i.e., the label of a node is unique, we testing for sibling requires labels of size $\log n + \Theta(\log\log d)$ for trees with degree d, and connectivity queries for forests requires labels of size $\log n + \Theta(\log\log n)$.

1 Introduction

A labeling scheme for ancestor queries for a rooted tree, assigns labels to each node v of a given tree T, where each label $l(v) \in \{0,1\}^*$ is a binary string. For any two nodes u, v, the labels $l(u), l(v)$ suffices to determine whether u is ancestor to v. We present a lower bound for such labeling schemes, showing that for a tree of size n, the label size must be $\log n + \Omega(\log\log n)$[1], if the scheme supports ancestor queries.

For the sibling query we give a labeling scheme which assigns labels of size $\lceil \log n \rceil$ bits. This labeling scheme will not assign unique labels to the nodes in a tree. If uniqueness is required as in [9], we give upper and lower bounds, showing that such labeling scheme use label of size $\log n + \Theta(\log\log d)$, for trees with degree d.

Furthermore, we give a lower bound showing that a labeling scheme supporting both sibling and parent queries needs to use labels of size $\log n + \Omega(\log\log n)$.

In general a labeling scheme for a family $\mathcal{F}$ of graphs with n nodes, supporting a certain general property between pair of nodes (for instance v is ancestor of w), consist of two mappings EC and DC. The encoder EC maps graphs from $\mathcal{F}$ into a labeling of the nodes of the graph, i.e., for $G \in \mathcal{F}$, $EC_G = EC(G)$ maps the nodes from G into labels. The decoder DC maps pair of labels into $\{0,1\}$ such that for all G, $DC(EC_G(v), EC_G(w)) = 1$ iff $v, w \in G$ satisfy the supported property for graph G. Notice that DC is independent of the graph from which the pair of labels is taken from. ln different papers [13, 1, 9] there are additional requirements, such as different time constrains on the function DC. The lower bounds presented in this paper are independent on such time constraints. As mentioned, a constraint we sometime use for labeling scheme is whether they use unique labels, i.e., it uses unique labels if for all graphs $G \in \mathcal{F}$, the encoder ensure that EC_G is a one-to-one mapping of the nodes in G.

We show that a labeling scheme for forests supporting connectivity queries between pair of nodes require labels of length $\lceil \log n \rceil$ if uniqueness of the labels is not required [13]. However, if uniqueness is required, as in [9], we give upper and lower bounds, showing that such schemes use label of size $\log n + \Theta(\log\log n)$.

*IT University of Copenhagen, Glentevej 67, DK-2400 Copenhagen NV, Denmark. Email: {stephen, theis}@it-c.dk

[1] log is the logarithm with base 2 throughout the paper

Labeling schemes have several nice properties. Since query computation is based solely on local label information, such schemes can avoid costly access to external memory that store global information about the tree. Furthermore, the locally distributed information is also required in some routing schemes and for efficient XML search engines, see e.g. [16, 1].

The lower bounds presented in this paper are for the Boolean relations between two nodes, e.g. to detect if two nodes are connected or not, where the output can be represented by a single bit saying "yes" or "no". To the best of our knowledge previous lower bounds for tree labeling schemes have been for more involved functions returning information such as the distance between two nodes as in [7].

For the ancestor relation Tsakalidis [17] shows that if we assign the preorder and postorder number to each node in a tree, a node v is an ancestor to a node w iff $preorder(v) \leq preorder(w) \leq postorder(v)$. This leads to a simple labeling scheme with labels of size $2\lceil \log n \rceil$ for the ancestor relation. Similarly, simple $2\lceil \log n \rceil$ labeling schemes are presented in [9] and [15]. Recently, several papers [1, 16, 3, 10] have made progress, bounding the label size to $\log n + \mathrm{O}(\sqrt{\log n})$ bits per label. The same bound have been achieved for labeling schemes supporting both sibling and parent queries [10]. An experimental comparison of different labeling schemes for testing ancestor relationship on real XML data can be found in [11]

Labeling schemes for sibling queries are studied in [10], nearest common ancestor labeling in [2]. Labeling schemes for routing can be found in [16], distance labeling in [7, 12, 14], and vertex-connectivity and adjacency labeling [10, 4, 5, 9, 13, 15]. In [6] contain an extensive survey of labeling schemes and their applications.

Outline of the paper

In section 3 we present a technique to show lower bounds for labeling schemes. In section 4 to section 7 we use this technique to give lower bounds for labeling schemes supporting e.g. ancestor, connectivity and sibling queries. Finally, in Section 8 we give matching upper bounds to some of the lower bounds.

2 Preliminaries

Let T be an undirected rooted tree. For a node $v \in T$, v is ancestor to all the nodes in the subtree of T rooted in v, and $V(T)$ is the number of nodes in T. Two nodes are siblings if they have the same parent. A node which not have a child is a leaf, and otherwise an internal node. Two nodes in a forest are connected if and only if there is a path between the two nodes.

The upper and lower bound given in this paper shows that for sibling and connectivity queries the number of bits needed to represent labels depends on if unique labels are required.

A test for whether v is an ancestor to w, we denote as an ancestor test, whereas a test for either v is ancestor to w or vice versa is called a weak ancestor test. A lower bound for weak ancestor tests is clearly also a lower bound for ancestor tests. The lower bound presented in this paper is for weak ancestor test.

The lower bounds presented in this paper for ancestor test use the following technique. First we give a family of trees $\mathcal{F}_{\mathcal{A}}$, where each tree consist of cn nodes, for a constant c. A labeling scheme can use the same label for different nodes in the family. However we show that any labeling schemes for weak ancestor queries need to use $\Omega(n \log n)$ different labels for $\mathcal{F}_{\mathcal{A}}$. If m different labels are necessary, then the label size must be at least $\log m$ bits. Since $\log(dn \log n) = \log n + \Omega(\log \log n)$, for any constant d, we establish the lower bound. A similar construction are used for the other lower bounds

3 Lower Bound Technique

Let $\mathcal{S}$ be a set of elements, and $l : \mathcal{S} \to \mathcal{T}$. We will assume $|S| = nk$, where k is an integer $\leq \log n$ and n is a power of two. Next we describe a partition P of S. P partition S to k *boxes*, each of n elements. The elements in the i'th box, $1 \leq i \leq \log n$, denoted B_i are partition into $n/2^i$ *groups*, each of 2^i elements.

Lemma 1 *If there exist a partition P of $\mathcal{S}$, where $|\mathcal{S}| = nk$, such that for l, the following two properties hold:*

1. *For two different elements $s_1, s_2 \in \mathcal{S}$, if s_1 and s_2 belongs to the same box, then $l(s_1) \neq l(s_2)$.*
2. *For elements $s_1, s_2, s_3, s_4 \in \mathcal{S}$, if s_1 and s_2 belongs to two different groups in the same box, $l(s_1) = l(s_3)$ and $l(s_2) = l(s_4)$, then s_3 and s_4 belongs to two different groups.*

then $|\mathcal{T}| = \Omega(nk)$ is a necessary size of the domain $\mathcal{T}$.

Proof. We will say the function l associate labels to the elements from $\mathcal{S}$. The elements associated the same label we denote as *neighbours*. In the following we give a strategy to choose a subset S' of elements from $\mathcal{S}$, guaranteeing that $\forall s_1, s_2 \in S'$, where $s_1 \neq s_2$, s_1 and s_2 will not be neighbours. We denote a strategy, with such a guarantee, for a *safe* strategy. The number of labels needed by l to $\mathcal{S}$, will be at least the size of S', since $|\mathcal{T}| \geq |S'|$, when choosing S' by a safe strategy. An element chosen to belong to S', we say is a *marked* element. Hence, no two elements which have the same label will be marked.

If one or more elements from a group are marked, we say the group also are marked. For a box B we let $M(B)$ denote the number of marked groups belonging to the box.

We first mark elements from the box B_k, and next for B_i in order of decreasing i.

All elements in B_k will be marked. From the first property of Lemma 1 there is no neighbours in the same box, and the marking is therefore safe.

When marking elements from the remaining boxes B_i, $i < k$, we keep the invariant that $M(B_i) \leq n/2^{i+1}$. Hence, we will at most mark elements from halve of the groups belonging to B_i.

Let $F(i)$ be the set of groups belonging to the boxes B_j, $j \geq i$, and let $M(F(i))$ be the number of marked groups belonging to $F(i)$. Since, we keep the invariant that $M(B_i) \leq n/2^{i+1}$, for $i < k$, it follows for $i \leq k$ that $M(F(i)) \leq n/2^k + \sum_{j=i}^{j=k-1} n/2^{i+1} = n/2^i$.

Next, we describe how to mark elements from B_i, after marking element from B_j, $j > i$. If a group in B_i includes an element, with a marked neighbour in B_j, $j > i$, we denote the group as *closed*. If a group not is closed, we denote it as *open*.

Let $s_1, s_2 \in B_i$ belongs to two different groups. If s_1 have a marked neighbour s_3, and s_2 have a marked neighbour s_4, then by the second property of Lemma 1, s_3 and s_4 must belongs to two different marked groups from $F(i+1)$. Hence, for each closed group in B_i, we can associate a marked group from $F(i+1)$, which will not be associated to any other groups in B_i. Since the number of groups in B_i is $n/2^i$, and we keep the invariant that $M(F(i+1)) \leq n/2^{i+1}$, at least $n/2^{i+1}$ of the groups in B_i will be open. Since the elements from the open groups not have a marked neighbour, and by the first property of Lemma 1, none of them are neighbours, it is safe marking all elements from $n/2^{i+1}$ groups of B_i. Doing this we keep the invariant, at most marking elements from halve of the groups in B_i, $i < k$.

Summarizing, we mark all elements in B_k, and halve of the elements from the remaining $k-1$ boxes, in total we mark $\Omega(nk)$ elements. ∎

In the following sections we will define different families of graphs, for which the nodes from these graphs can be partition, such that labeling, have to obey the properties given in Lemma 1.

4 Lower Bound for Ancestor Queries

To show a lower bound for ancestor labeling l, we give a family $\mathcal{F}_{\mathcal{A}}$ of $\log n$ trees $\{T_1, T_2, \cdots T_{\log n}\}$, each of size $2n+1$. We show that for a subset $\mathcal{S}$ of the nodes from $\mathcal{F}_{\mathcal{A}}$, where $|\mathcal{S}| = n \log n$, there is a partition P of $\mathcal{S}$, such that any l, must obey the two properties in Lemma 1. This implies, that at least $\Omega(n \log n)$ labels are needed, and will conclude our proof.

The tree T_i in $\mathcal{F}_{\mathcal{A}}$ consist of a root node with $n/2^i$ children. Each child v is the root of a path $p(v)$ of length 2^i. Furthermore each node on these paths have a child which is a leaf not belonging to the path.

We have $|V(T_i)| = 2(n/2^i)2^i + 1 = 2n+1$. We let $\mathcal{S}$ be the subset of nodes from $\mathcal{F}_{\mathcal{A}}$, which belongs to a path $p(v)$, for v being a child to one of the root nodes in the family. Hence, $|\mathcal{S}| = n \log n$. Box B_i is the subset of nodes from $\mathcal{S}$, which belongs to the tree T_i. The nodes from box B_i are partition into groups, such that two nodes from the same group, belongs to the same path $p(v)$, for a child v to the root in T_i. Next we show that the two properties from Lemma 1 must be fulfilled for any labeling l for this partition.

Consider the first property. Let $s_1, s_2 \in B_i$, $s_1 \neq s_2$. If s_1 is an ancestor to s_2, s_2 cannot be an ancestor to s_1. Assume with out loss of generality that s_1 not is an ancestor to s_2, and let c be the leaf in T_i which are child to s_2. Since, s_1 not is ancestor to s_2, s_1 cannot be ancestor to c. Therefore, should $DC(l(s_1), l(c))$ and $DC(l(s_2), l(c))$ give two different answers, which only can be the case if $l(s_1)$ and $l(s_2)$ are different.

Next we consider the second property. Let $s_1, s_2, s_3, s_4 \in \mathcal{S}$, where s_1 and s_2 belongs to two different groups in the same box. This, implies that there is no ancestor relation between s_1 and s_2. If $l(s_1) = l(s_3)$ and $l(s_2) = l(s_4)$ there can therefore not be an ancestor relation between s_3 and s_4, and therefore s_3 and s_4 must belongs to different groups.

Theorem 1 *A labeling scheme to answer weak ancestor queries for trees with n nodes needs to use labels of size* $\log n + \Omega(\log \log n)$.

5 Lower Bound for Connectivity Queries

In this section we consider the needed size of labels, to answer connectivity queries, if the labels assigned to the nodes in a tree should be unique. Let $\mathcal{F}_{\mathcal{C}}$ be the family of $\log n$ forest F_i, $1 \leq i \leq \log n$, where F_i consist of $2^{\log n - i}$ paths of length 2^i. We let $\mathcal{S}$ be the nodes from $\mathcal{F}_{\mathcal{C}}$.

The nodes in box B_i we let be the nodes from the forest F_i. The nodes from box B_i are partition into groups, such that two nodes from F_i which belongs to the same group are connected in F_i. We have $|V(T_i)| = n$. We assume the labels assigned to a forest F_i should be unique, hence the first property from Lemma 1 is trivial obtained. Let $s_1, s_2, s_3, s_4 \in \mathcal{S}$. If s_1 and s_2 belongs to two different groups from the same box B_i, s_1 and s_2 are not connected in F_i. If s_3 and s_4 are in the same group, s_3 and s_4 are connected in some forest, and $DC(l(s_1), l(s_2))$ should therefore be different from $DC(l(s_3), l(s_4))$, which not can be the case if $l(s_1) = l(s_3)$ and $l(s_2) = l(s_4)$.

Theorem 2 *A labeling scheme, which assign unique labels to nodes in a forest, to answer connectivity queries, needs to use labels of size* $\log n + \Omega(\log \log n)$.

6 Lower Bounds for Sibling Queries

In this section we consider the needed size of labels, to answer sibling queries, if the labels assigned to the nodes in a tree should be unique. We consider a forest of trees $\mathcal{F}_{\mathcal{S}}(k)$ of k trees T_i, $1 \leq i \leq k \leq \log n$. Let $B(j)$ be a complete balanced binary rooted tree with 2^j leafs and $2^{j+1} - 1$ nodes. The tree T_i consist of a tree $B = B(\log n - i)$, where each leaf from B in T_i have 2^i children. These children are the set $\mathcal{S}$. The

box B_i consist of the subset of nodes from $\mathcal{S}$ which comes from T_i. The nodes in box B_i are partition into groups such that two nodes which belongs to the same group are siblings. We assume that the labels assign to a tree should be unique, hence the first property of Lemma 1 is trivial obtained. Let $s_1, s_2, s_3, s_4 \in \mathcal{S}$. Since s_1 and s_2 not belongs to the same group, s_1 and s_2 are not siblings. If s_3 and s_4 belongs to the same group, s_3 and s_4 are sibling. Therefore should $DC(l(s_1), l(s_2))$ be different from $DC(l(s_3), l(s_4))$, which not can be the case if $l(s_1) = l(s_3)$ and $l(s_2) = l(s_4)$. The maximum degree d of a tree in $\mathcal{F}_{\mathcal{S}}(k)$ is 2^k, and $|\mathcal{S}| = nk$, giving:

Theorem 3 *A labeling scheme to answer sibling queries, which assigns unique labels to the nodes in a tree, needs labels of size* $\log n + \Omega(\log \log d)$ *for trees with degree* d.

7 Lower Bounds for Combined Sibling and Parent Queries

In this section we consider the needed size of labels, to answer both parent and sibling queries. Let $\mathcal{F}_{\mathcal{SP}}$ be the forest $\mathcal{F}_{\mathcal{S}}(\log n)$ to which we have added a child to each leaf in the forest $\mathcal{F}_{\mathcal{S}}(\log n)$. We let $\mathcal{S}$ be the same subset of nodes as in the previous section. Above we made the assumption that the labels used in a tree should be unique, now it follows as a constraint for the nodes in $\mathcal{S}$. Let s_1, s_2 belong to the same box, $s_1 \neq s_2$, and let c be the child to s_1. Since s_2 not is a parent to s_1, s_1 and s_2 must be assigned different labels.

Theorem 4 *A labeling scheme to answer sibling and parent queries, to trees with* n *nodes, needs labels of size* $\log n + \Omega(\log \log n)$.

8 Upper Bounds for Sibling and Connectivity Queries

First we consider sibling queries. If two nodes in the same tree can be given the same label, we can label the nodes with label of length $\lceil \log n \rceil$ as follows; partition the nodes into groups such that two nodes are siblings if and only if they belong to the same group, next merge all groups which consist of a single node to one special group. This construction gives $g < n$ groups, which are numbered $1, 2 \cdots g$. Nodes in the same group are given the same label, namely the number of the group. The special group is given number 1. Now two nodes are siblings if and only if they have the same label different from 1.

Theorem 5 *A labeling scheme to answer sibling queries only needs label of size* $\lceil \log n \rceil$.

Next we show how to assign unique labels, matching the lower bound from the previous section for trees with maximum degree d. We group the nodes as above. We assign to each node v two numbers: A group number, $g(v)$ to answer sibling queries as above and an individual number $i(v)$ to make its label unique. Two nodes in the same group will be given the same group number. Assume we have g groups $g_1, g_2 \cdots g_g$. Let $Size(g_i)$ be the number of nodes in g_i. Using Huffman code [8] we give each node in group g_i, a group number of length $\log n - \log Size(g_i) + O(1)$. The individual numbers given to the nodes in group g_i are simply $1, 2, \cdots Size(g_i)$, of length $\log Size(g_i) + O(1)$. In total we use $\log n + O(1)$ bits to the group and individual numbers, however coding these to numbers as one label, we also need to be able to separate these two numbers given the label of a node. For this purpose we will use the first $O(\log \log d)$ bits of the label to code the length of the individual number as follows. The individual number in a tree with maximum degree d is at most d, and can be represented with at most $q = \log d + O(1)$ bits. To represent the length of the individual number we need $O(\log q) = O(\log \log d)$ bits. Now, we also need to represent the length of the code representing the length of the individual number, but this can be done simply by using an unary code of length $O(\log \log d)$.

Theorem 6 *A labeling scheme to answer sibling queries, that assigns unique labels to the nodes in a tree, only needs labels of size* $\log n + O(\log \log d)$ *for trees with degree* d*.*

Finally, using the same observations, grouping connected nodes, we have :

Theorem 7 *A labeling scheme to answer connectivity queries in a forest, that assigns unique labels to the nodes in the forest, only needs labels of size* $\log n + O(\log \log n)$.

References

[1] S. Abiteboul, H. Kaplan, and T. Milo. Compact labeling schemes for ancestor queries. In *Proceedings of the twelfth annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 547–556, 2001.

[2] S. Alstrup, C. Gavoille, H. Kaplan, and T. Rauhe. Finding nearest common ancestors in a distributed enviroment. Technical Report 2001-6, IT-University of Copenhagen, 2001.

[3] S. Alstrup and T. Rauhe. Improved labeling schemes for ancestor queries. In *Prooceeding of the thirteen annual ACM-SIAM Symposium on discrete Algorithms (SODA)*, 2002.

[4] M. A. Breuer. Coding vertexes of a graph. *IEEE Trans. on Information Theory*, IT–12:148–153, 1966.

[5] M. A. Breuer and J. Folkman. An unexpected result on coding vertices of a graph. *J. of Mathemathical analysis and applications*, 20:583–600, 1967.

[6] C. Gavoille and D. Peleg. Compact and localized distributed data structures. Technical Report RR-1261-01, Laboratoire Bordelais de Recherce en Informatique, 2001.

[7] C. Gavoille, D. Peleg, S. Perennes, and R. Raz. Distance labeling in graphs. In *12th Symp. On Discrete algorithms*, 2001.

[8] D. A. Huffman. A methode for construction of minimum-redundancy codes. *Proceedings of the IRE*, 1952.

[9] S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. *SIAM J. DISC. MATH.*, 1992. Preliminary version appeared in STOC'88.

[10] H. Kaplan and T. Milo. Short and simple labels for small distances and other functions. In *7nd Work. on Algo. and Data Struc.*, LNCS, 2001.

[11] H. Kaplan, T. Milo, and R. Shabo. A comparison of labeling schemes for ancestor queries. In *Proceedings of the thirteen annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2002.

[12] M. Katz, N. Katz, and D. Peleg. Distance labeling schemes for well-seperated graph classes. In *STACS'00*, volume 1170 of *LNCS*. Springer Verlag, 2000.

[13] M. Katz, N. A. Katz, A. Korman, and D. Peleg. Labeling schemes for flow and connectivity. In *Proceedings of the thirteen annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2002.

[14] D. Peleg. Proximity-preserving labeling schemes and their applications. In *Graph-Theoretic concepts in computer science, 25th international workshop WG'99*, volume 1665 of *LNCS*, pages 30–41. Springer Verlag, 1999.

[15] N. Santoro and R. Khatib. Labeling and implicit routing in networks. *The computer J.*, 28:5–8, 1985.

[16] M. Thorup and U. Zwick. Compact routing schemes. In *ACM Symposium on Parallel Algorithms and Architectures*, volume 13, 2001.

[17] A. K. Tsakalidis. Maintaining order in a generalized linked list. *Acta Informatica*, 21(1):101–112, 1984.