Distance Labeling Schemes for Trees

Stephen Alstrup[*1], Inge Li Gørtz[†2], Esben Bistrup Halvorsen[3], and Ely Porat[4]

1   University of Copenhagen, Copenhagen, Denmark
     s.alstrup@di.ku.dk
2   Technical University of Copenhagen, Copenhagen, Denmark
     inge@dtu.dk
3   University of Copenhagen, Copenhagen, Denmark
     esben@bistruphalvorsen.dk
4   Bar-Ilan University, Bar Ilan, Israel
     porately@cs.biu.ac.il

―――― **Abstract** ――――

We consider distance labeling schemes for trees: given a tree with n nodes, label the nodes with binary strings such that, given the labels of any two nodes, one can determine, by looking only at the labels, the distance in the tree between the two nodes.

A lower bound by Gavoille et al. [Gavoille et al., J. Alg., 2004] and an upper bound by Peleg [Peleg, J. Graph Theory, 2000] establish that labels must use $\Theta(\log^2 n)$ bits[1]. Gavoille et al. [Gavoille et al., ESA, 2001] show that for very small approximate stretch, labels use $\Theta(\log n \log \log n)$ bits. Several other papers investigate various variants such as, for example, small distances in trees [Alstrup et al., SODA, 2003].

We improve the known upper and lower bounds of exact distance labeling by showing that $\frac{1}{4} \log^2 n$ bits are needed and that $\frac{1}{2} \log^2 n$ bits are sufficient. We also give $(1+\varepsilon)$-stretch labeling schemes using $\Theta(\log n)$ bits for constant $\varepsilon > 0$. $(1+\varepsilon)$-stretch labeling schemes with polylogarithmic label size have previously been established for doubling dimension graphs by Talwar [Talwar, STOC, 2004].

In addition, we present matching upper and lower bounds for distance labeling for caterpillars, showing that labels must have size $2 \log n - \Theta(\log \log n)$. For simple paths with k nodes and edge weights in $[1, n]$, we show that labels must have size $(k-1)/k \log n + \Theta(\log k)$.

1998 ACM Subject Classification E.1 Distributed data structures, E.1 Graphs and networks, G.2.2 Graph Theory

Keywords and phrases Distributed computing, Distance labeling, Graph theory, Routing, Trees

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.132

1   Introduction

A *distance labeling scheme* for a given family of graphs assigns *labels* to the nodes of each graph in the family such that, given the labels of two nodes in the graph and no other information, it is possible to determine the shortest distance between the two nodes. The labels are assumed to be composed of bits, and the goal is to make the worst-case label size as

―――――――――――――

1 Throughout this paper we use log for $\log_2$.

■ **Table 1** Results presented in this paper. $\varepsilon > 0$ is a constant.

Problem	Lower bound	Upper bound
Exact, general trees	$\frac{1}{4}\log^2 n$	$\frac{1}{2}\log^2 n$
$(1 + \varepsilon)$-stretch, general trees	$\Theta(\log n)$	
Caterpillars	$2\log n - \Theta(\log\log n)$	
Weighted paths, k nodes, weights in $[1, n]$	$\frac{k-1}{k}\log n + \Theta(\log k)$	

small as possible. Labeling schemes are also called *implicit representation of graphs* [64, 71]. The problem of finding implicit representations with small labels for specific families of graphs was introduced in the 1960s [16, 17], and efficient labeling schemes were introduced in [44, 57]. Distance labeling for general graphs has been considered since the 1970/80s [40, 72], and later for various restricted classes of graphs and/or approximate distances, often tightly related to distance oracle and routing problems, see e.g. [7]. This paper focuses on distance labels for the well studied case of trees.

Exact distances.   In [61] Peleg presented an $O(\log^2 n)$ bits distance labeling scheme for general unweighted trees. In [39] Gavoille et al. proved that distance labels for unweighted binary trees require $\frac{1}{8}\log^2 n - O(\log n)$ bits and presented a scheme with $1/(\log 3 - 1)\log n \approx 1.7\log n$ bits. This paper presents a scheme of size $\frac{1}{2}\log^2 n + O(\log n)$ and further reduces the gap by showing that $\frac{1}{4}\log^2 n - O(\log n)$ bits are needed. Our upper bound is a somewhat straightforward application of a labeling scheme for nearest common ancestors [8, 9].

Approximate distances.   Let $\text{dist}_T(x, y)$ denote the shortest distance between nodes x, y in a tree T. An r-additive approximation scheme returns a value $\text{dist}'_T(x, y)$, where $\text{dist}_T(x, y) \leq \text{dist}'_T(x, y) \leq \text{dist}_T(x, y) + r$. An s-stretched approximation scheme returns a value $\text{dist}'_T(x, y)$, where $\text{dist}_T(x, y) \leq \text{dist}'_T(x, y) \leq \text{dist}_T(x, y) \cdot s$. For trees of height h Gavoille et al. [32, theorem 4] gave a 1-additive $O(\log n \log h)$ bit labeling scheme. However, using an extra bit in the label for the node depth modulo 2, it is easy to see that any 1-additive scheme can be made exact. Gavoille et al. [32] also gave upper and lower bounds of $\Theta(\log\log n \log n)$ bits for $(1 + 1/\log n)$-stretched distance. This paper presents a scheme of size $\Theta(\log n)$ for $(1 + \varepsilon)$-stretch for constant $\varepsilon > 0$. Labeling schemes for $(1 + \varepsilon)$-stretch with polylogarithmic size label have previously been given for graphs of doubling dimension [65] and planar graphs [67].

Distances in caterpillars and paths.   Labeling schemes for caterpillars have been studied for various queries, e.g., adjacency [15]. Here we present upper and lower bounds showing that distance labeling caterpillars requires $2\log n - \Theta(\log\log n)$ bits. The upper bound is constructed by reduction to the case of *weighted paths* with $k > 1$ nodes and positive integer edge weights in $[1, n]$, for which we give upper and lower bounds showing that labels must have size $\frac{k-1}{k}\log n + \Theta(\log k)$.

1.1   Related work

Distances in trees with small height.   It is known that, for unweighted trees with bounded height h, labels must have size $\Theta(\log n \log h)$. The upper bound follows from [32, Theorem 2]

and the lower bound from [39, Section 3][2]. In [45] distance labeling for various restricted class of trees, including trees with bounded height, is considered, and in [66] another distance labeling scheme for unweighted trees using $O(\log n \log h)$ bits is given.

Small distances in trees. Distances in a tree between nodes at distance at most k can be computed with labels of size $\log n + O(k\sqrt{\log n})$ [46]. In [5] it is shown that size $\log n + \Theta(\log \log n)$ are needed for labeling schemes supporting both parent and sibling queries. More generally, [5] shows that, using labels of size $\log n + O(\log \log n)$, the distance between two nodes can be determined if it is at most k for some constant k, which is optimal for $k > 1$. In [33, 34] further improvements are given for small distances in trees. For $k = 1$, corresponding to adjacency testing, there is a sequence of papers that improve the second order term, recently ending with [6] which establishes that $\log n + \Theta(1)$ bits are sufficient.

Various other cases for trees. Distance labeling schemes for various other cases have been considered, e.g., for weighted trees [32, 39, 61], dynamic trees [52], and a labeling scheme variation with extra free lookup [50, 51].

Exact and approximate distances in graphs. Distance labeling schemes for general graphs [7, 39, 40, 64, 70, 72] and various restricted graphs exist, e.g., for bounded tree-width, planar and bounded degree [39], distance-hereditary [36], bounded clique-width [22], some non-positively curved plane [19], interval [37] and permutation graphs [14]. Approximate distance labeling schemes, both additive and stretched, are also well studied; see e.g., [18, 26, 32, 35, 39, 41, 42, 53, 61, 69]. An overview of distance labeling schemes can be found in [7].

1.2 Second order terms for labeling schemes are theoretically well studied

Chung's solution in [20] gives labels of size $\log n + O(\log \log n)$ for adjacency labeling in trees, which was improved to $\log n + O(\log^* n)$ in FOCS'02 [13] and in [15, 20, 29, 30, 47] to $\log n + \Theta(1)$ for various special cases. Finally it was improved to $\log n + \Theta(1)$ for general trees in FOCS'15 [6].

A recent STOC'15 paper [11] improves label size for adjacency in general graphs from $n/2 + O(\log n)$ [44, 56] to $n/2 + O(1)$ almost matching an $(n-1)/2$ lower bound [44, 56].

Likewise, the second order term for ancestor relationship is improved in a sequence of STOC/SODA papers [2, 4, 12, 30, 31] (and [1, 47]) to $\Theta(\log \log n)$, giving labels of size $\log n + \Theta(\log \log n)$.

Somewhat related, *succinct data structures* (see, e.g., [24, 27, 28, 58, 59]) focus on the space used in addition to the information theoretic lower bound, which is often a lower order term with respect to the overall space used.

1.3 Labeling schemes in various settings and applications

Using labeling schemes to compute locally and distributed, it is possible to avoid costly access to large global tables. Such properties are used, e.g., in XML search engines [2], network routing and distributed algorithms [23, 25, 68, 69], dynamic and parallel settings [21, 52], graph representations [44], and other applications [48, 49, 60, 61, 62].

---

[2] We thank Gavoille for pointing this out.

Trees have always been a subject of special interest in the routing literature (see e.g. [3, 10, 55, 73]). As pointed out by Leighton [54], many networks are simply trees, and, moreover, routing in general graphs may be done using only the edges of a spanning tree of the graph. In [68] a $O(\log n)$ labeling routing scheme for tree is given. This e.g. combined with one of the results in this paper makes it possible using labels of size $O(\log n)$ not only to route packets in trees, but also estimate the distance to its destination.

Various computability requirements are sometimes imposed on labeling schemes [2, 44, 48]. This paper assumes the RAM model.

2   Preliminaries

Trees. Given nodes u, v in a rooted tree T, u is an *ancestor* of v and v is a *descendant* of u, if u is on the unique path from v to the root. For a node u of T, denote by T_u be the subtree of T consisting of all the descendants of u (including itself). The *depth* of u is the number of edges on the unique simple path from u to the root of T. The nearest common ancestor (NCA) of two nodes is the unique common ancestor with largest depth. Let $T[u, v]$ denote the nodes on the simple path from u to v in T. The variants $T(u, v]$ and $T[u, v)$ denote the same path without the first and last node, respectively. The distance between u and v is the number $\text{dist}(u, v) = |T(u, v)|$. We set $\text{distroot}(v) = \text{dist}(v, r)$, where r is the root of T. A *caterpillar* is a tree whose non-leaf nodes form a path, called the *spine*.

Heavy-light decomposition (From [63]). Let T be a rooted tree. The nodes of T are classified as either *heavy* or *light* as follows. The root r of T is light. For each non-leaf node v, pick one child w where $|T_w|$ is maximal among the children of v and classify it as heavy; classify the other children of v as light. The *apex* of a node v is the nearest light ancestor of v. By removing the edges between light nodes and their parents, T is divided into a collection of *heavy paths*. Any given node v has at most $\log n$ light ancestors (see [63]), so the path from the root to v goes through at most $\log n$ heavy paths.

Bit strings. A bit string s is a member of the set $\{0, 1\}^*$. We denote the length of a bit string s by $|s|$, the ith bit of s by s_i, and the concatenation of two bit strings s, s' by $s \circ s'$. We say that s_1 is the most significant bit of s and $s_{|s|}$ is the least significant bit.

Labeling schemes. An *distance labeling scheme* for trees of size n consists of an *encoder* e and a *decoder* d. Given a tree T, the encoder computes a mapping $e_T : V(T) \to \{0, 1\}^*$ assigning a *label* to each node $u \in V(T)$. The decoder is a mapping $d : \{0, 1\}^* \times \{0, 1\}^* \to \mathbb{Z}^+$, where $\mathbb{Z}^+$ denotes the positive integers, such that, given any tree T and any pair of nodes $u, v \in V(T)$, $d(e(u), e(v)) = \text{dist}(u, v)$. Note that the decoder does not know T. The *size* of a labeling scheme is defined as the maximum label size $|e_T(u)|$ over all trees T and all nodes $u \in V(T)$. If, for all trees T, the mapping e_T is injective we say that the labeling scheme assigns *unique* labels.

3   Distances on weighted paths

In this section we study the case of paths with k nodes and integral edge weights in $[1, n]$. The solution to this problem will later be used to establish the upper bound for caterpillars.

3.1 Upper Bound

▶ **Theorem 1.** *There exist a distance labeling scheme for paths with k nodes and positive integral edge weights in $[1, n]$ with labels of size $\frac{k-1}{k} \log n + O(\log k)$.*

Proof. We begin by considering the family of paths with k nodes, integral edge weights and diameter $< n$. We shall prove that there exists a distance labeling scheme for this family with labels of size $\frac{k-1}{k} \log n + \log k + \mathrm{O}(\log \log k)$.

So consider such a path, and root it in one of its end nodes, denoted v_0. Denote the nodes on the path $v_0, \ldots, v_{k-1}$ in order. Let $d_i = \mathrm{distroot}(v_i)$ and note that, by assumption, $d_i < n$ for all i. We will let the label for v_i store the number $d_i + x$ for some $x < n$ that allows us to represent $d_i + x$ compactly. Since we use the same x for all nodes, we can easily compute the distance between any pair of nodes v_i, v_j as $|(d_i + x) - (d_j + x)|$.

Since we choose $x < n$, the largest number stored in a label will be $d_k + x < 2n$, which can be represented with *exactly* $L = \lceil \log(2n) \rceil$ bits. Divide those L bits to $k + 1$ segments, whereof k have $\ell = \lfloor L/k \rfloor$ bits and the last segment contains the remaining bits. The first segment, segment 0, will contain the ℓ least significant bits, segment 1 the following ℓ bits and so on. We will choose x such that the representation of $d_i + x$ has 0s in all the bits in the i'th segment. If we manage to do so, we will be able to encode each $d_i + x$ with $L - \ell + \lceil \log k \rceil$ bits. Indeed, we can use exactly $\lceil \log k \rceil$ bits to represent i, and the next $L - \ell$ bits to represent $d_i + x$ where we skip the i'th segment. Prefixing with a string in the form $0^{\lceil \log \lceil \log k \rceil \rceil} 1$, we get a string from which we can determine the number of bits needed to write $\lceil \log k \rceil$ and therefrom the numbers i and $d_i + x$. We use this string as the label for v_i. The label length is $L - \ell + \lceil \log k \rceil + \lceil \log \lceil \log k \rceil \rceil + 1 = \frac{k-1}{k} \log n + \log k + O(\log \log k)$.

It remains to show that there exist a number $x < n$ as described. In the following we shall, as in the above, represent numbers $< 2n$ with L bits that are divided into $k + 1$ segments whereof the first k have size ℓ. For $i < k$ and $y < 2n$, let $a(i, y)$ be a function which returns a number z with the following properties:

(i) In z, all bits outside segment i are 0.

(ii) $z + y$ has only 0s in segment i.

This function is constructed as follows. If y only has 0s in segment i, let $a(i, y) = 0$. Otherwise take the representation of y, zero out all bits outside segment i, reverse the bits in segment i and add v to the resulting number, where v has a 1 in the least significant bit of segment i and 0s in all other positions.

Note that from (i) it follows that adding z to any number will not change bits in less significant positions than segment i. We can now scan through the nodes $v_0, \ldots v_{k-1}$, increasing x by adding bits to x in more and more significant positions (in non-overlapping segments), as follows:

- Set $x = 0$.

- For $i = 1 \ldots, k - 1$, set $x = x + a(i, x + d_i)$.

After iteration i we have that $x + d_i$ in segment i only has 0s, and in the following iterations, 1s are only added to x in more significant bit positions, meaning that $d_i + x$ continues to have only 0s in segment i. Since the segments are non-overlapping, we end up with $x < n$.

For the more general family of paths with k nodes and edge weights in $[1, n]$, we simply note that the diameter of any path in this family is at most kn. Using the above result thus immediately gives us a labeling scheme with labels of size $\frac{k-1}{k} \log n + O(\log k)$.   ◀

3.2   Lower bound

▶ **Theorem 2.** *Labeling scheme for distances on weighted paths with k nodes and edge weights $[1, n]$ require $\frac{k-1}{k} \log n + \Omega(\log k)$ bits.*

Proof. Let $\mathcal{F}$ denote the family of paths with k nodes and integral edge weights in $[1, n]$. We can construct all the members of $\mathcal{F}$ by selecting $(k-1)$ different edge weights in the range $[1, n]$, skipping the paths which have already been constructed by the reverse sequence of edge weights. With this construction we will at most skip half of the paths, and hence $|\mathcal{F}| \geq \frac{1}{2} n^{k-1}$. Let the worst-case label size of an optimal distance labeling scheme for such paths have length L. The number of different labels with length at most L is $N = 2^{L+1} - 1$. We can uniquely represent each of the paths in $\mathcal{F}$ with the collection of their labels, and hence $|\mathcal{F}| \leq \binom{N}{k}$. Thus, we have found that $\frac{1}{2} n^{k-1} \leq \binom{N}{k}$. Since $\binom{N}{k} \leq (Ne/k)^k$, it follows that $\frac{k-1}{k} \log n \leq \log N - \log k + O(1)$ and hence that $L \geq \frac{k-1}{k} \log n + \log k - O(1)$ as desired. ◀

Combining Theorem 2 with Theorem 1 we see that distance labels for paths of k nodes with integral weights in $[1, n]$ must have length $\frac{k-1}{k} \log n + \Theta(\log k)$.

4   Distances in caterpillars

4.1   Upper bound

▶ **Theorem 3.** *There exist a distance labeling scheme for caterpillars with worst-case label size $2 \log n - \log \log n + O(\log \log \log n)$.*

Proof. We will start by giving a simple $2 \log n$ bits scheme and then improve it. The simple solution assigns two numbers to each node. The nodes on the spine save distroot and the number 0. The nodes not on the spine save their parent's distroot and a number that is unique among their siblings. The second number is required to distinguish siblings, and hence determine if the distance between two nodes is 0 or 2. The worst-case label size for this solution is $2 \log n + O(1)$.

To improve the solution, we split up the nodes on the spine into two groups: (1) nodes with $> \frac{n}{k}$ leaves and (2) nodes with $\leq \frac{n}{k}$ leaves, for some parameter k to be chosen later. We add the root to the first group no matter what. Note that the first group can contain at most k nodes.

As before, all nodes save two numbers: distroot and the number 0 for spine nodes or a number to distinguish siblings. The idea is to reduce label size with $\log k$ bits by using fewer bits for the first number for nodes in the first group and for the second number for nodes in the second group.

The nodes in the first group form a path with at most k nodes and edge weights in $[1, n]$ (where each weight corresponds to the distance between the nodes in the original graph). The algorithm from Theorem 1 will add a number x, which is less than the diameter, which again is less than n, to the numbers representing the root distances of the nodes. Using this technique, we can, as seen in the proof of Theorem 1, encode the (modified) distroots of the nodes in the first group with only $\frac{k-1}{k} \log n + \log k + O(\log \log k)$ bits. This gives labels of size $\frac{2k-1}{k} \log n + \log k + O(\log \log k)$ for non-spine nodes whose parents are in the first group.

We will also add x to the distroots of nodes in the second group, but since $x < n$ this will not change the label size by more than a single bit. For non-spine nodes whose parents are in the second group, we need at most $\log n - \log k + O(1)$ bits for the second number, giving a total label size of $2 \log n - \log k + O(1)$.

Finally, since the two numbers that form a label now have different lengths, we need an additional $O(\log \log k)$ bits to determine when one number ends and the next begins. Indeed, it wil be possible to split up labels into their components if we know the number of bits used to write $\lceil \log k \rceil$, and we represent this number with $O(\log \log k)$ bits.

Setting $k = \frac{\log n}{2 \log \log n}$, we now see that our worst-case label size is the maximum of

$$2 \log n - \log k + O(\log \log k) = 2 \log n - \log \log n + O(\log \log \log n)$$

and

$$\frac{2k-1}{k} \log n + \log k + O(\log \log k) = 2 \log n - 2 \log \log n + \log \log n + O(\log \log \log n)$$
$$= 2 \log n - \log \log n + O(\log \log \log n).$$

This proves the theorem.   ◀

It is worth noting that the ability to distinguish nodes plays a significant part in the label size. Indeed, if the two nodes given as input to the decoder could always be assumed to be distinct, then a distance labeling scheme for caterpillars with worst-case label size $\log n + O(1)$ would be possible.

4.2 Lower bound

We present a technique that counts tuples of labels that are known to be distinct and compares the result to the number of tuples one can obtain with labels of size L. The technique may have applications to distance labeling for other families of graphs.

▶ **Theorem 4.** *For any $n \geq 4$, any distance labeling scheme for the family of caterpillars with at most n nodes has a worst-case label size of at least $2\lfloor \log n \rfloor - \lfloor \log \lfloor \log n \rfloor \rfloor - 4$.*

Proof. Set $k = \lfloor \log n \rfloor$ and $m = 2^k$. Let $(i_1, \ldots, i_k)$ be a sequence of k numbers from the set $\{1, \ldots, m/2\}$ with the only requirement being that $i_1 = 1$. Now consider, for each such sequence, the caterpillar whose main path has length $m/2$ and where, for $t = 1, \ldots, k$, the node in position i_t has $\lfloor m/2k \rfloor$ leaf children (not on the main path). We shall refer to these children as the *t'th group*. Note that two disjoint groups of children may be children of the same node if $i_t = i_s$ for some s, t. Each of these caterpillars has $m/2 + k\lfloor m/2k \rfloor \leq m \leq n$ nodes.

Suppose that σ is a distance labeling scheme for the family of caterpillars, and consider one of the caterpillars defined above. Given distinct nodes u, v not on the main path, their distance will be $\mathrm{dist}(u, v) = |i_s - i_t| + 2$, where i_s and i_t are the positions on the main path of the parents of u and v, respectively. In particular, if $s = 1$, so that $i_s = 1$, then $\mathrm{dist}(u, v) = i_t + 1$. Thus, if σ has been used to label the nodes of the caterpillar, the number i_t for a child in the t'th group can be uniquely determined from its label together with the label of any of the children from the first group. It follows that any k-tuple of labels $(l_1, \ldots, l_k)$ where l_t is a label of a child in the t'th group uniquely determines the sequence $(i_1, \ldots, i_k)$. In particular, k-tuples of labels from distinct caterpillars must be distinct. Of course, k-tuples of labels from the same caterpillar must also be distinct, since labels are unique in a distance labeling scheme.

Now, there are $(m/2)^{k-1}$ choices for the sequence $(i_1, \ldots, i_k)$, and hence there are $(m/2)^{k-1}$ different caterpillars in this form. For each of these, there are $\lfloor m/2k \rfloor^k$ different choices of k-tuples of labels. Altogether, we therefore have $(m/2)^{k-1}\lfloor m/2k \rfloor^k$ distinct k-tuples of labels. If the worst-case label size of σ is L, then we can create at most $(2^{L+1} - 1)^k$

distinct k-tuples of labels, so we must have $(m/2)^{k-1}\lfloor m/2k\rfloor^{k} \leq (2^{L+1} - 1)^{k}$. From this it follows that

$$
\begin{aligned}
L &\geq \lfloor \frac{k-1}{k}(\log m - 1) + \log\lfloor m/2k\rfloor\rfloor \\
&\geq \lfloor \frac{(k-1)^2}{k} + k - \log k\rfloor - 2 \\
&\geq 2k - \lfloor \log k\rfloor - 4 \\
&= 2\lfloor \log n\rfloor - \lfloor\log\lfloor\log n\rfloor\rfloor - 4.
\end{aligned}
$$
◀

5 Exact distances in trees

5.1 Upper bound

Let u, v be nodes in a tree T and let w be their nearest common ancestor. We then have

$$
\text{dist}(u, v) = \text{distroot}(u) - \text{distroot}(v) + 2\,\text{dist}(w, v). \tag{1}
$$

If $w = u$ so that u is an ancestor of v, then the above equation is just a difference of distroots, which can be stored for each node with $\log n$ bits. The same observation clearly holds if $w = v$.

Assume now that $w \notin \{u, v\}$ so that u and v are not ancestors of each other. Consider the heavy-light decomposition [63] described in the preliminaries. At least one of the nodes u and v must have an ancestor which is a light child of w. Assume that it is v. Now, v has at most $\log n$ light ancestors. Saving the distance to all of them together with distroot gives us sufficient information to compute the distance between u and v using equation (1). This is the idea behind Theorem 6 below.

By examining the NCA labeling scheme from [8, 9], we see that it can easily be extended as follows.

▶ **Lemma 5** ([8, 9]). *There exists an NCA labeling scheme of size $O(\log n)$. For any two nodes u, v the scheme returns the label of $w = \text{nca}(u, v)$ as well as:*
- *which of u and v (if any) have a light ancestor that is a child of w; and*
- *the number of light nodes on the path from the root to w and from w to u and v, respectively.*

▶ **Theorem 6.** *There exists a distance labeling scheme for trees with worst-case label size $\frac{1}{2}\log^2 n + O(\log n)$.*

Proof. We use $O(\log n)$ bits for the extended NCA labeling in Lemma 5 and for distroot. Using (1) it now only remains to efficiently represent, for each node, the distance to all its light ancestors.

We consider the light ancestors of a node v encountered on the path from the root to v. The distance from v to the root is at most $n - 1$ and can therefore be encoded with *exactly* $\lceil\log n\rceil$ bits (by adding leading zeros if needed). By construction of the heavy-light decomposition, the next light node on the path to v will be the root of a subtree of size at most $n/2$, meaning that the distance from v to that ancestor is at most $n/2 - 1$ and can be encoded with *exactly* $\lceil\log n\rceil - 1$ bits. Continuing this way, we encode the i'th light ancestor on the path from the root to v with exactly $\lceil\log n\rceil - i$ bits. When we run out of light ancestors, we concatenate all the encoded distances, resulting in a string of length at most

$$
\lceil\log n\rceil + (\lceil\log n\rceil - 1) + \cdots + 2 + 1 = \frac{1}{2}\lceil\log n\rceil^2 + \frac{1}{2}\lceil\log n\rceil.
$$
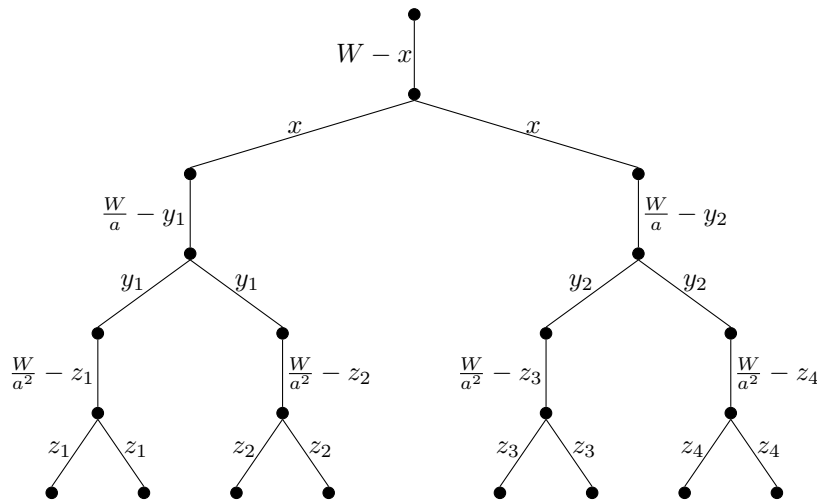
■ **Figure 1** An (h, W, a)-tree, where $h = 3$. We require that $x < W$, $y_1, y_2 < W/a$ and $z_1, \ldots, z_4 < W/a^2$.

We can use $O(\log n)$ extra bits to encode n and to separate all sublabels from each other. The decoder can now determine $\lceil \log n \rceil$ and split up the entries in the list of distances. When applying formula (1), it can then determine the distance between v and w by adding together the relevant distances in the list of light ancestors, using the fact from Lemma 5 that it knows the number of light ancestors from v to w. ◀

5.2 Lower bound

In the case of general trees, Gavoille et al [39] establish a lower bound of $\frac{1}{8} \log^2 n - O(\log n)$ using an ingenious technique where they apply a distance labeling scheme to a special class of trees called (h, M)-trees[3]. The following uses a generalization of (h, M)-trees to improve their ideas and leads to a lower bound of $\frac{1}{4} \log^2 n - O(\log n)$.

(h, W, a)-trees. We begin with some definitions. For integers $h, W \geq 0$ and a number $a \geq 1$ such that W/a^i is integral for all $i = 0, \ldots, h$, an (h, W, a)-*tree* is a rooted binary tree T with edge weights in $[0, W]$ that is constructed recursively as follows. For $h = 0$, T is just a single node. For $h = 1$, T is a claw (i.e. a star with three edges) with edge weights $x, x, W - x$ for some $0 \leq x < W$ rooted at the leaf node of the edge with weight $W - x$. For $h > 1$, T consists of an $(1, W, a)$-tree whose two leaves are the roots of two $(h - 1, W/a, a)$-trees T_0, T_1. We shall denote an (h, W, a)-tree constructed in this way by $T = \langle T_0, T_1, x \rangle$ An example for $h = 3$ can be seen in Figure 1. Note that the case $a = 1$ simply corresponds to the (h, W)-trees defined in [39].

It is easy to see that an (h, W, a)-tree has 2^h leaves and $3 \cdot 2^h - 2$ nodes. Further, it is straightforward to see that, if u, v are leaves in an (h, W, a)-tree $T = \langle T_0, T_1, x \rangle$, then

$$\mathrm{dist}_T(u, v) = \begin{cases} 2W \frac{a^{-1} - a^{-h}}{1 - a^{-1}} + 2x, & \text{if } u \in T_0 \text{ and } v \in T_1, \text{ or vice versa,} \\ \mathrm{dist}_{T_i}(u, v), & \text{if } u, v \in T_i \text{ for some } i = 0, 1. \end{cases} \tag{2}$$

---

[3] Note that their exposition has some minor errors as pointed out (and corrected) in [43].

Leaf distance labeling schemes. In the following we shall consider *leaf distance labeling schemes* for the family of (h, W, a)-trees: that is, distance labeling schemes where only the leaves in a tree need to be labeled, and where only leaf labels can be given as input to the decoder. Since an ordinary distance labeling scheme obviously can be used only for leaves, any lower bound on worst-case label sizes for a leaf distance labeling scheme is also a lower bound for an ordinary distance labeling scheme. We denote by $g(h, W, a)$ the smallest number of labels needed by an optimal leaf distance labeling scheme to label all (h, W, a)-trees.

▶ **Lemma 7.** *For all $h \geq 1$ and $W \geq 2$, $g(h, W, a)^2 \geq W g(h - 1, W^2/a^2, a^2)$.*

Proof. Fix an optimal leaf distance labeling scheme σ which produces exactly $g(h, W, a)$ distinct labels for the family of (h, W, a)-trees. For leaves u and v in an (h, W, a)-tree, denote by $l(u)$ and $l(v)$, respectively, the labels assigned by σ. For $x = 0, \ldots, W - 1$, let $S(x)$ be the set consisting of pairs of labels $(l(u), l(v))$ for all leaves $u \in T_0$ and $v \in T_1$ in all (h, W, a)-trees $T = \langle T_0, T_1, x \rangle$.

The sets $S(x)$ and $S(x')$ are disjoint for $x \neq x'$, since every pair of labels in $S(x)$ uniquely determines x due to (2). Letting $S = \bigcup_{x=0}^{W-1} S(x)$, we therefore have $|S| = \sum_{x=0}^{W-1} |S(x)|$. Since S contains pairs of labels produced by σ from leaves in (h, W, a)-trees , we clearly also have $|S| \leq g(h, W, a)^2$, and hence it only remains to prove that $|S| \geq W g(h - 1, W^2/a^2, a^2)$, which we shall do by showing that $|S(x)| \geq g(h - 1, W^2/a^2, a^2)$ for all x.

The goal for the rest of the proof is therefore to create a leaf distance labeling scheme for $(h-1, W^2/a^2, a^2)$-trees using only labels from the set $S(x)$ for some fixed x. So let x be given and consider an $(h - 1, W^2/a^2, a^2)$-tree T'. Let $V = W/a$. From T' we shall construct an $(h-1, V, a)$-tree $\phi_i(T')$ for $i = 0, 1$ such that every leaf node v in T' corresponds to nodes $\phi_i(v)$ in $\phi_i(T')$ for $i = 0, 1$. The trees $\phi_i(T')$ are defined as follows. If $h = 1$, so that T' consists of a single node, then $\phi_i(T') = T'$ for $i = 0, 1$. If $h > 1$, then T' is in the form $T' = \langle T_0', T_1', y \rangle$ for some $0 \leq y < V^2$. We can write y in the form $y = y_0 + y_1 V$ for uniquely determined y_0, y_1 with $0 \leq y_0, y_1 < V$. For $i = 0, 1$, we recursively define $\phi_i(T') = \langle \phi_i(T_0'), \phi_i(T_1'), y_i \rangle$. Thus, $\phi_i(T')$ is an $(h - 1, V, a)$-tree that is similar to T' but where we replace the top edge weight y by edge weights y_i and, recursively, do the same for all $(h - 2, V^2/a^2, a^2)$-subtrees. Note also that the corresponding edge weight $V^2 - y$ in T' automatically is replaced by the edge weight $V - y_i$ in $\phi_i(T')$ in order for $\phi_i(T')$ to be an $(h - 1, V, a)$-tree.

Denote by $\phi_i(v)$ the leaf in $\phi_i(T')$ corresponding to the leaf v in T'.

Consider now the (h, W, a)-tree $T = \langle \phi_0(T'), \phi_1(T'), x \rangle$. Every leaf v in T' corresponds to the leaves $\phi_0(v), \phi_1(v)$ in T where $\phi_i(v) \in \phi_i(T')$ for $i = 0, 1$. Using formula (2) for the distances in T', it is straightforward to see that

$$\text{dist}_{T'}(u, v) = \big(\text{dist}_{\phi_0(T')}(\phi_0(u), \phi_0(v)) \bmod (2V)\big) + V \text{dist}_{\phi_1(T')}(\phi_1(u), \phi_1(v)). \qquad (3)$$

We can now apply the leaf distance labeling scheme σ to T and obtain a label for each leaf node in T. In particular, the pair of leaves $(\phi_0(v), \phi_1(v))$ corresponding to a node v in T' will receive a pair of labels. We use this pair to label v in T', whereby we have obtained a labeling of the leaves in T' with labels from $S(x)$. Using the formula in (3) we can construct a decoder that can compute the distance between two nodes in T' using these labels alone, and hence we have obtained a leaf distance labeling scheme for $(h - 1, V^2, a^2)$-trees using only labels from $S(x)$ as desired. ◀

▶ **Lemma 8.** *For all $h \geq 1$ and $W \geq 2$, $g(h, W, a) \geq \frac{W^{h/2}}{a^{h(h-1)/4}}$.*

Proof. The proof is by induction on h. For $h = 1$ we note that an $(0, W, a)$-tree has only one node, so that $g(0, W^2/a^2, a^2) = 1$. Lemma 7 therefore yields $g(1, W, a)^2 \geq W$ from which

it follows that $g(1, W, a) \geq \sqrt{W}$. The lemma therefore holds for $h = 1$. Now let $h > 1$ and assume that the lemma holds for $h - 1$. Lemma 7 and the induction hypothesis now yield

$$g(h, W, a)^2 \ \geq \ W g(h-1, W^2/a^2, a^2) \ \geq \ W \frac{(W^2/a^2)^{(h-1)/2}}{a^{2(h-1)(h-2)/4}} \ = \ \frac{W^h}{a^{h(h-1)/2}}$$

from which it follows that $g(h, W, a) \geq \frac{W^{h/2}}{a^{h(h-1)/4}}$ as desired.   ◀

The previous lemma implies that any (leaf and hence also ordinary) distance labeling scheme for (h, W, a)-trees must have labels with worst-case length at least $\frac{h}{2}(\log W - \frac{h-1}{2} \log a) = \frac{1}{2} h \log W - \frac{1}{4} h^2 \log a + \frac{1}{4} h \log a$. Since the number of nodes in such a tree is $n = 3 \cdot 2^h - 2$, it follows that $h = \log(n+2) - \log 3$, and hence that $\log n - 2 \leq h \leq \log n$ for sufficiently large n. From this we see that the worst case label length is at least

$$\frac{1}{2} \log n \log W - \frac{1}{4} \log n (\log n - 1) \log a - \log W - \frac{1}{2} \log a.$$

In the case where $a = 1$, we retrieve the bound of $\frac{1}{2} \log n \log W - \log W$ obtained in [38]. It seems that larger values of a only makes the above result weaker, but the the real strength of the above becomes apparent when we switch to the unweighted version of (h, W, a)-trees, in which we replace weighted edges by paths of similar lengths. Note that a distance labeling scheme for the family of unweighted (h, W, a)-trees can be used as a distance labeling scheme for the weighted (h, W, a)-trees, and hence any lower bound in the weighted version automatcially becomes a lower bound in the unweighted version.

The number of nodes n in an *unweighted* (h, W, a)-tree is upper bounded by

$$n \leq 2W + 2 \cdot 2W/a + 2^2 \cdot 2W/a^2 + \cdots + 2^{h-1} \cdot 2W/a^{h-1} + 1$$

In the case $a = 2$, we get $n \leq 2Wh + 1$.

▶ **Theorem 9.** *Any distance labeling scheme for unweighted $(h, W, 2)$-trees, and hence also for general trees, has a worst-case label size of at least $\frac{1}{4} \log^2 n - O(\log n)$.*

Proof. Choose the largest integer h with $2 \cdot 2^h h + 1 \leq n$, and note that we must have $h \geq \log n - O(\log \log n)$. Set $W = 2^h$ and consider the family of $(h, W, 2)$-trees, which is a subfamily of the family of trees with n nodes. From Lemma 8 it therefore follows that the worst-case label length is

$$\frac{1}{2} h \log W - \frac{1}{4} h^2 + \frac{1}{4} h = \frac{1}{4} h^2 + \frac{1}{4} h = \frac{1}{4} \log^2 n + \frac{1}{4} \log n - O(\log \log n).$$ ◀

6   Approximate distances in trees

In this section we present a $(1 + \varepsilon)$-stretch distance labeling schemes with labels of size $O(\log n)$.

▶ **Theorem 10.** *For constant $\varepsilon > 0$, $(1 + \varepsilon)$ stretch distance labeling schemes for trees use $\Theta(\log n)$ bits.*

Proof. As in the case of exact distances, we will create labels of size $O(\log n)$ bits that contain the extended NCA labels from Lemma 5 as well as distroot. We will also be using the formula in (1). However we can not afford to store exact distance to each apex ancestor. Even directly storing an 2-approximate distance to each apex ancestor would require $\log n \log \log n$ bits. Instead we show how to compactly represent all the $(1 + \varepsilon)$-approximate distances to

light ancestors for a node using a total of $O(\log n)$ bits, and show how to use this to obtain a $(1 + 2\varepsilon)$-approximation.

Let $w = \text{nca}(u, v)$ and assume $w \notin \{u, v\}$, since otherwise we can compute the exact distance using only distroot. Suppose we know a $(1 + \varepsilon)$-approximation α of $\text{dist}(w, v)$ for some $\varepsilon \geq 0$. That is,

$$\text{dist}(w, v) \leq \alpha \leq (1 + \varepsilon)\,\text{dist}(w, v). \tag{4}$$

Define $\tilde{d} = \text{distroot}(u) - \text{distroot}(v) + 2\alpha$. First we show that $\tilde{d}$ is a $(1 + 2\varepsilon)$-approximation of $\text{dist}(u, v)$. Next we show how to represent all the $(1 + \varepsilon)$-approximate distances to light ancestors for a node using a total of $O(\log n)$ bits. Together with formula (1), these two facts prove that we can compute $(1 + 2\varepsilon)$-stretch distances between any pair of nodes with labels of size $O(\log n)$. To prove the theorem, we can then simple replace ε by $\frac{1}{2}\varepsilon$.

To see that $\tilde{d}$ is a $(1 + 2\varepsilon)$-approximation of $\text{dist}(u, v)$, first note that

$$\tilde{d} = \text{distroot}(u) - \text{distroot}(v) + 2\alpha \geq \text{distroot}(u) - \text{distroot}(v) + 2\,\text{dist}(w, v) = \text{dist}(u, v).$$

For the other inequality, note that

$$
\begin{aligned}
\tilde{d} &= \text{distroot}(u) - \text{distroot}(v) + 2\alpha \\
&\leq \text{distroot}(u) - \text{distroot}(v) + 2(1 + \epsilon)\,\text{dist}(w, v) \\
&= \text{distroot}(u) - (\text{distroot}(v) - \text{dist}(w, v)) + (1 + 2\epsilon)\,\text{dist}(w, v) \\
&= \text{distroot}(u) - \text{distroot}(w) + (1 + 2\epsilon)\,\text{dist}(w, v) \\
&= \text{dist}(u, w) + (1 + 2\epsilon)\,\text{dist}(w, v) \\
&\leq (1 + 2\epsilon)\,(\text{dist}(u, w) + \text{dist}(w, v)) \\
&= (1 + 2\epsilon)\,\text{dist}(u, v)\,.
\end{aligned}
$$

It now only remains to show that we can compactly store all the approximate distances α to light ancestors using $O(\log n)$ bits space.

We use a heavy light path decomposition of the tree. For each node v we can save a 2 approximate distance to all its k proper light ancestors as follows. Let S be a binary string initially with k zeros. Before each 0 we now insert 1s such that, if we have j 1s in total from the beginning of S and to the i'th 0, then the distance to the ith light ancestor a of v satisfies that $2^{j-1} \leq \text{dist}(v, a) \leq 2^j$. This is the same as traversing the tree bottom-up from v and, for each light node encountered on the way, adding a 0 and each time the distance doubles adding a 1. The number of 0s equal the number of light nodes which is at most $\log n$, and the number of 1s is also limited by $\log n$ since n is the maximum distance in the tree. In total the length of S is at most $2 \log n$.

Using the $O(\log n)$ bits label from Lemma 5 we can tell if one node is an ancestor of another, and if not which one has a light ancestor a that is a child of their nearest common ancestor w. In addition, we can determine the total number i of light ancestors up to a. This means that we can compute j, and hence the 2-approximation $j - 1$, as the number of 1's in S until the i'th 0.

We have now obtained a 2-approximation with labels of size $O(\log n)$. We can improve this to a $(1 + \varepsilon)$-approximation by setting a 1 in S each time the distance increases with $1 + \varepsilon$ rather than 2. This will increase the label size with a constant factor $\frac{1}{\log(1+\varepsilon)}$.

This proves that there is a $(1 + \varepsilon)$-stretch distance labeling scheme with $O(\log n)$ bit label length. To complete the proof of the theorem, we note that, given any $(1 + \varepsilon)$-stretch distance scheme, we can always distinguish nodes (since identical nodes have distance 0), which means that we always need at least n different labels, and hence labels of size at least $\log n$ bits. ◀

References

1 S. Abiteboul, S. Alstrup, H. Kaplan, T. Milo, and T. Rauhe. Compact labeling scheme for ancestor queries. *SIAM J. Comput.*, 35(6):1295–1309, 2006. `doi:10.1137/S0097539703437211`.

2 S. Abiteboul, H. Kaplan, and T. Milo. Compact labeling schemes for ancestor queries. In *Proc. of the 12th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 547–556, 2001.

3 N. Alon, F .R. K. Chung, and R. L. Graham. Routing permutations on graphs via matchings. In *Proc. of the 25th Annual ACM Symp. on the Theory of Computing (STOC)*, pages 583–591, 1993.

4 S. Alstrup, P. Bille, and T. Rauhe. Labeling schemes for small distances in trees. In *Proc. of the 14th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 689–698, 2003. URL: `http://dl.acm.org/citation.cfm?id=644108.644220`.

5 S. Alstrup, P. Bille, and T. Rauhe. Labeling schemes for small distances in trees. *SIAM J. Discrete Math.*, 19(2):448–462, 2005. See also SODA'03. `doi:10.1137/S0895480103433409`.

6 S. Alstrup, S. Dahlgaard, and M. B. T. Knudsen. Optimal induced universal graphs and labeling schemes for trees. In *Proc. 56th Annual Symp. on Foundations of Computer Science (FOCS)*, 2015.

7 S. Alstrup, C. Gavoile, E. B. Halvorsen, and H. Petersen. Simpler, faster and shorter labels for distances in graphs. In *Proc. of the 27th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2016.

8 S. Alstrup, C. Gavoille, H. Kaplan, and T. Rauhe. Nearest common ancestors: A survey and a new algorithm for a distributed environment. *Theory of Computing Systems*, 37(3):441–456, 2004. `doi:10.1007/s00224-004-1155-5`.

9 S. Alstrup, E. B. Halvorsen, and K. G. Larsen. Near-optimal labeling schemes for nearest common ancestors. In *Proc. of the 25th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 972–982, 2014. `doi:10.1137/1.9781611973402.72`.

10 S. Alstrup, J. Holm, K. de Lichtenberg, and M. Thorup. Direct routing on trees. In *Proc. of the 9th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 342–349, 1998.

11 S. Alstrup, H. Kaplan, M. Thorup, and U. Zwick. Adjacency labeling schemes and induced-universal graphs. In *Proc. of the 47th Annual ACM Symp. on Theory of Computing (STOC)*, 2015.

12 S. Alstrup and T. Rauhe. Improved labeling schemes for ancestor queries. In *Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2002.

13 S. Alstrup and T. Rauhe. Small induced-universal graphs and compact implicit graph representations. In *Proc. 43rd Annual Symp. on Foundations of Computer Science (FOCS)*, pages 53–62, 2002.

14 F. Bazzaro and C. Gavoille. Localized and compact data-structure for comparability graphs. *Discrete Mathematics*, 309(11):3465–3484, 2009. `doi:10.1016/j.disc.2007.12.091`.

15 N. Bonichon, C. Gavoille, and A. Labourel. Short labels by traversal and jumping. In *Structural Information and Communication Complexity*, pages 143–156. Springer, 2006. Include proof for binary trees and caterpillars.

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

17 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.

18 V. D. Chepoi, F. F. Dragan, B. Estellon, M. Habib, and Y. Vaxès. Diameters, centers, and approximating trees of delta-hyperbolic geodesic spaces and graphs. In *24st Annual ACM*

Symp. on Computational Geometry (SoCG), pages 59–68, 2008. `doi:10.1145/1377676.1377687`.

19   V. D. Chepoi, F. F. Dragan, and Y. Vaxès. Distance and routing labeling schemes for non-positively curved plane graphs. *J. of Algorithms*, 61(2):60–88, 2006. `doi:10.1016/j.jalgor.2004.07.011`.

20   F. R. K. Chung. Universal graphs and induced-universal graphs. *J. of Graph Theory*, 14(4):443–454, 1990.

21   E. Cohen, H. Kaplan, and T. Milo. Labeling dynamic XML trees. *SIAM J. Comput.*, 39(5):2048–2074, 2010. `doi:10.1137/070687633`.

22   B. Courcelle and R. Vanicat. Query efficient implementation of graphs of bounded clique-width. *Discrete Applied Mathematics*, 131:129–150, 2003. `doi:10.1016/S0166-218X(02)00421-3`.

23   L. J. Cowen. Compact routing with minimum stretch. *J. of Algorithms*, 38:170–183, 2001. See also SODA'91.

24   Y. Dodis, M. Pătraşcu, and M. Thorup. Changing base without losing space. In *Proc. of the 42nd Annual ACM Symp. on Theory of Computing (STOC)*, pages 593–602, 2010.

25   T. Eilam, C. Gavoille, and D. Peleg. Compact routing schemes with low stretch factor. *J. of Algorithms*, 46(2):97–114, 2003. `doi:10.1016/S0196-6774(03)00002-6`.

26   M. Elkin, A. Filtser, and O. Neiman. Prioritized metric structures and embedding. In *Proc. of the 47th Annual ACM Symp. on Theory of Computing (STOC)*, pages 489–498, 2015.

27   A. Farzan and J. I. Munro. Succinct encoding of arbitrary graphs. *Theoretical Computer Science*, 513:38–52, 2013. `doi:10.1016/j.tcs.2013.09.031`.

28   A. Farzan and J. I. Munro. A uniform paradigm to succinctly encode various families of trees. *Algorithmica*, 68(1):16–40, 2014. `doi:10.1007/s00453-012-9664-0`.

29   P. Fraigniaud and A. Korman. On randomized representations of graphs using short labels. In *Proc. of the 21st Annual Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 131–137, 2009. `doi:10.1145/1583991.1584031`.

30   P. Fraigniaud and A. Korman. Compact ancestry labeling schemes for XML trees. In *Proc. of the 21st annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 458–466, 2010.

31   P. Fraigniaud and A. Korman. An optimal ancestry scheme and small universal posets. In *Proc. of the 42nd Annual ACM Symp. on Theory of Computing (STOC)*, pages 611–620, 2010. `doi:10.1145/1806689.1806773`.

32   C. Gavoille, M. Katz, N. Katz, C. Paul, and D. Peleg. Approximate distance labeling schemes. In *Proc. of the 9th Annual European Symp. on Algorithms (ESA)*, pages 476–488, 2001.

33   C. Gavoille and A. Labourel. Distributed relationship schemes for trees. In *18th International Symp. on Algorithms and Computation (ISAAC)*, pages 728–738, 2007. `doi:10.1007/978-3-540-77120-3_63`.

34   C. Gavoille and A. Labourel. On local representation of distances in trees. In *Proc. of the 26th Annual ACM Symp. on Principles of Distributed Computing (PODC)*, pages 352–353, 2007. `doi:10.1145/1281100.1281169`.

35   C. Gavoille and O. Ly. Distance labeling in hyperbolic graphs. In *16th Annual International Symp. on Algorithms and Computation (ISAAC)*, pages 1071–1079, 2005. `doi:10.1007/11602613_106`.

36   C. Gavoille and C. Paul. Distance labeling scheme and split decomposition. *Discrete Mathematics*, 273(1-3):115–130, 2003.

37   C. Gavoille and C. Paul. Optimal distance labeling for interval graphs and related graphs families. *SIAM J. Discrete Math.*, 22(3):1239–1258, 2008. `doi:10.1137/050635006`.

38 C. Gavoille, D. Peleg, S. Pérennes, and R. Raz. Distance labeling in graphs. In *Proc. of the 12th Annual ACM-SIAM Symp. on Discrete algorithms (SODA)*, pages 210–219, 2001. URL: `http://dl.acm.org/citation.cfm?id=365411.365447`.

39 C. Gavoille, D. Peleg, S. Pérennes, and R. Raz. Distance labeling in graphs. *J. of Algorithms*, 53(1):85–112, 2004. See also SODA'01. `doi:10.1016/j.jalgor.2004.05.002`.

40 R. L. Graham and H. O. Pollak. On embedding graphs in squashed cubes. In *Lecture Notes in Mathematics*, volume 303. Springer-Verlag, 1972.

41 A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 534–543, 2003. `doi:10.1109/SFCS.2003.1238226`.

42 A. Gupta, A. Kumar, and R. Rastogi. Traveling with a pez dispenser (or, routing issues in mpls). *SIAM J. on Computing*, 34(2):453–474, 2005. See also FOCS'01.

43 E. B. Halvorsen. Labeling schemes for trees – overview and new results. Master's thesis, University of Copenhagen, 2013. Available at `http://esben.bistruphalvorsen.dk`.

44 S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. *SIAM J. Disc. Math.*, pages 596–603, 1992. See also STOC'88.

45 M. Kao, X. Li, and W. Wang. Average case analysis for tree labelling schemes. *Theor. Comput. Sci.*, 378(3):271–291, 2007. `doi:10.1016/j.tcs.2007.02.066`.

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

47 H. Kaplan, T. Milo, and R. Shabo. A comparison of labeling schemes for ancestor queries. In *Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2002.

48 M. Katz, N. A. Katz, A. Korman, and D. Peleg. Labeling schemes for flow and connectivity. *SIAM J. Comput.*, 34(1):23–40, 2004. See also SODA'02. `doi:10.1137/S0097539703433912`.

49 A. Korman. Labeling schemes for vertex connectivity. *ACM Trans. Algorithms*, 6(2):39:1–39:10, 2010. `doi:10.1145/1721837.1721855`.

50 A. Korman and S. Kutten. Labeling schemes with queries. *CoRR*, abs/cs/0609163, 2006. URL: `http://arxiv.org/abs/cs/0609163`.

51 A. Korman and S. Kutten. Labeling schemes with queries. In *SIROCCO*, pages 109–123, 2007.

52 A. Korman and D. Peleg. Labeling schemes for weighted dynamic trees. *Inf. Comput.*, 205(12):1721–1740, 2007. `doi:10.1016/j.ic.2007.08.004`.

53 R. Krauthgamer and J. R. Lee. Algorithms on negatively curved spaces. In *47th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 119–132, 2006. `doi:10.1109/FOCS.2006.9`.

54 F. T. Leighton. Methods for message routing in parallel machines. In *Proc. of the 24 Annual ACM Symp. on the Theory of Computing (STOC)*, pages 77–96, 1992.

55 M. Mitzenmacher. Constant time per edge is optimal on rooted tree networks. In *Proc. of the 8th Annual ACM Symp. on parallel algorithms and Architectures (SPAA'96)*, pages 162–169, 1996.

56 J. W. Moon. On minimal *n*-universal graphs. *Proc. of the Glasgow Mathematical Association*, 7(1):32–33, 1965.

57 J. H. Müller. *Local structure in graph classes*. PhD thesis, Georgia Institute of Technology, 1988.

58 J. I. Munro, R. Raman, V. Raman, and S. Srinivasa Rao. Succinct representations of permutations and functions. *Theor. Comput. Sci.*, 438:74–88, 2012. `doi:10.1016/j.tcs.2012.03.005`.

59 M. Pătraşcu. Succincter. In *Proc. 49th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 305–313, 2008.

60 D. Peleg. Informative labeling schemes for graphs. In *Proc. 25th Symp. on Mathematical Foundations of Computer Science*, pages 579–588, 2000.

61 D. Peleg. Proximity-preserving labeling schemes. *J. Graph Theory*, 33(3):167–176, 2000.

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

63 D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *J. of Computer and System Sciences*, 26(3):362–391, 1983. `doi:10.1016/0022-0000(83)90006-5`.

64 J. P. Spinrad. *Efficient Graph Representations*, volume 19 of *Fields Institute Monographs*. AMS, 2003.

65 K. Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *Proc. of the 36th Annual ACM Symp. on Theory of Computing (STOC)*, pages 281–290, 2004. `doi:10.1145/1007352.1007399`.

66 M. Tang, J. Yang, and G. Zhang. A compact distance labeling scheme for trees of small depths. In *International Conference on Scalable Computing and Communications / Eighth International Conference on Embedded Computing, ScalCom-EmbeddedCom*, pages 455–458, 2009. `doi:10.1109/EmbeddedCom-ScalCom.2009.87`.

67 M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004. See also FOCS'01. `doi:10.1145/1039488.1039493`.

68 M. Thorup and U. Zwick. Compact routing schemes. In *Proc. of the 13th Annual ACM Symp. on Parallel Algorithms and Architectures*, SPAA'01, pages 1–10, 2001. `doi:10.1145/378580.378581`.

69 M. Thorup and U. Zwick. Approximate distance oracles. *J. of the ACM*, 52(1):1–24, 2005. See also STOC'01.

70 O. Weimann and D. Peleg. A note on exact distance labeling. *Inf. Process. Lett.*, 111(14):671–673, 2011. `doi:10.1016/j.ipl.2011.04.006`.

71 Wikipedia. Implicit graph – wikipedia, the free encyclopedia, 2013. [Online; accessed 15-February-2014]. URL: `http://en.wikipedia.org/w/index.php?title=Implicit_graph&oldid=585232203`.

72 P. M. Winkler. Proof of the squashed cube conjecture. *Combinatorica*, 3(1):135–139, 1983. `doi:10.1007/BF02579350`.

73 L. Zhang. Optimal bounds for matching routing on trees. In *Proc. of the 8th Annual ACM-SIAM Symposium on discrete algorithms (SODA)*, pages 445–453, 1997.