Average Case Analysis for Tree Labelling Schemes

Ming-Yang Kao*[1], Xiang-Yang Li**[2], and WeiZhao Wang[2]

[1] Northwestern University, Evanston, IL, USA,
kao@cs.northwestern.edu
[2] Illinois Institute of Technology, Chicago, IL, USA,
xli@cs.iit.edu, wangwei4@iit.edu

Abstract. We study how to label the vertices of a tree in such a way that we can decide the distance of two vertices in the tree given only their labels. For trees, Gavoille *et al.* [7] proved that for any such distance labelling scheme, the maximum label length is at least $\frac{1}{8}\log^2 n - O(\log n)$ bits. They also gave a separator-based labelling scheme that has the optimal label length $\Theta(\log n \cdot \log(H_n(T)))$, where $H_n(T)$ is the height of the tree. In this paper, we present two new distance labelling schemes that not only achieve the optimal label length $\Theta(\log n \cdot \log(H_n(T)))$, but also have a much smaller expected label length under certain tree distributions. With these new schemes, we also can efficiently find the least common ancestor of any two vertices based on their labels only.

1 Introduction

For commonly used graph representations such as adjacency matrices and lists [15], one cannot determine whether or not two vertices are adjacent in the graph only based on the names of the two vertices. In contrast, Breuer and Folkman [5, 6] proposed to label the vertices in such a way that there exists a polynomial-time algorithm that can determine the adjacency of two vertices given only their labels. Such a labelling scheme is generally known as an *adjacency labelling scheme*. If the length of a label is allowed to be arbitrarily large, then one can encode any desired information. However, for a labelling scheme to be useful, the label length should be relatively short (say, polylogarithmic in the size of the graph) and yet allows one to decode the adjacency efficiently (say, time polynomial in the input label lengths). Breuer and Folkman [5, 6] proposed to use Hamming distances to label general graph. An (m, t)-*labelling scheme* labels each vertex with an m-bit label such that two vertices are adjacent if and only if their labels are at Hamming distance t or less of each other. Breuer and Folkman [6] showed that every n-vertex graph has a $(2n\Delta, 4\Delta - 4)$-labelling scheme, where Δ is the maximum vertex degree in the graph. Kannan *et al.* [14] gave adjacency labelling schemes with $O(\log n)$-bit labels for several families of graphs, including graphs of bounded degrees, graphs of bounded genuses, trees, and various intersection-based graphs such as internal graphs and c-decomposable graphs. Alstrup and Rauhe [4] improved the bound to $k \log n + O(\log^* n)$ for the family $\mathcal{A}_k$ of graphs with arboricity k and n vertices.

* Supported in part by NSF Grant IIS-0121491.
** Supported in part by NSF Grant CCR-0311174.

It is useful and possible to design a more general labelling scheme that also contains the distance information. A *distance labelling scheme* permits one to determine the distance between two vertices efficiently based only on their labels [7, 12]. Peleg [12] gave an $O(\log^2 n)$-bit distance labelling scheme for general trees and c-decomposable graphs. He showed [12] that for a family of n-vertices graphs with $\Omega(\exp(n^{1+\epsilon}))$ non-isomorphic graphs, any distance labelling scheme must use labels with a total length $\Omega(n^{1+\epsilon})$. Gavoille *et al.* [7] studied the bounds for the label length of the distance labelling schemes for several graph families. For general graphs, they gave a tight bound of $\Theta(n)$ bits; for planar graphs, an upper bound of $O(\sqrt{n}\log n)$ and a lower bound of $\Omega(n^{1/3})$; for bounded-degree graphs, a lower bound of $\Omega(\sqrt{n})$; and for trees, a tight bound of $\Theta(\log n \cdot \log(H_n(T)))$, where $H_n(T)$ is the height of the tree. Alstrup and Rauhe [3] built the lower-bounds of length of the label for supporting ancestor, sibling and connectivity. Recently, several distance labelling schemes considering bounded distance and weighted distance have been devised and surveyed by Gavoille and Peleg [10]. Alstrup *et al.* [2] designed a labelling scheme for a rooted tree to compute in constant time the least common ancestor from the labels of any two vertices. The labels assigned are of size $O(\log n)$ bits for a tree of n vertices. Alstrup *et al.* [1] studied labelling schemes for trees, supporting various relationships (ancestor, sibling, and connectivity) between vertices at small distance.

In this paper, we study distance labelling schemes for unweighted trees. For trees, Gavoille *et al.* [7] proved that for any distance labelling scheme, the label length is at least $\frac{1}{8}\log^2 n - O(\log n)$; they also gave a separator-based labelling scheme that has a label length $O(\log^2 n)$. Gavoille [9] improved the label scheme to $O(\log n \cdot \log(H_n(T)))$. Here, we present two new distance labelling schemes- backbone-based scheme and rake-based scheme, that not only achieve the asymptotically optimal label length $O(\log n \cdot \log(H_n(T)))$ but also have a much smaller expected label length under certain tree distributions. With these new schemes, we can also find the least common ancestor of any two vertices based on their labels only. Table 1 summarizes our main results, where k is the maximum vertex degree, $\mathbb{E}(H_n)$ is the expected height of a tree.

2 Preliminaries

Unless explicitly stated otherwise, a tree is always rooted at vertex r. The relative positions of the children are significant. The *size* of a tree T, denoted as $|T|$, is the number of the vertices in T. Given two vertices u and v in a tree T, the unique simple path between u and v in T is denoted as $\mathsf{P}(u, v, T)$, and the number of edges on $\mathsf{P}(u, v, T)$ is the *distance* between u and v, denoted as $d_T(u, v)$. The *level* of a vertex u is $d_T(u, r)$. The *height* of a tree T with n vertices, denoted as $H_n(T)$, is $\max_{u\in T} d_T(u, r)$. A vertex w is an *ancestor* of a vertex u if it is on the path $\mathsf{P}(u, r, T)$; the vertex u is then called a *descendant* of w. A vertex w is the *least common ancestor* of two vertices u, v if w has the largest level among all common ancestors of u and v. For a tree T and a vertex u, let T^u denote the subtree of T formed by u and all its descendants in T.

A *vertex labelling* for a tree T is a function L that assigns an integer $L(u, T)$ to each vertex in the tree T. A *distance calculator* is a function f that computes the distance of two vertices u,v in tree T given only their labels $L(u, T)$ and $L(v, T)$ but

tree labelling schemes		separator-based	backbone-based	rake-based
worst case deterministic analysis		$\Theta(\log n \cdot \log(H_n(T)))$ Theorem 5	$\Theta(\log n \cdot \log(H_n(T)))$ Theorem 2	$\Theta(\log n \cdot \log(H_n(T)))$ Theorem 4
binary search tree Distribution	upper	$O(\log n \cdot \log\log n)$ Theorem 7	$O(\log n \cdot \log\log n)$ Theorem 7	$O(\log n \cdot \log\log\log n)$ Theorem 10
	lower	$\Omega(\log n \cdot \log\log n)$ Theorem 9	$\Omega(\frac{\log n \cdot \log\log n}{\log\log\log n})$ Theorem 8	$\Omega(\log n)$ Lemma 1
uniform tree distribution	upper	$O(\log^2 n)$ Theorem 5	$O(\log^2 n)$ Theorem 2	$O(\log^2 n)$ Theorem 4
	lower	$\Omega(\log^2 n)$ Theorem 13	$\Omega(\frac{\log^2 n}{\log\log n})$ Theorem 12	$\Omega(\frac{\log^2 n}{\log\log n})$ Theorem 11
distributions with $\mathbb{E}(H_n) = O(\log^\epsilon n)$	upper	$O(\log n \cdot \log\log n)$ Theorem 14	$O(\log n \cdot \log\log n)$ Theorem 14	$O(\log n \cdot \log\log n)$ Theorem 14
	lower	$\Omega(\frac{\log n \cdot \log\log n}{\log k})$ Theorem 6	$\Omega(\log n)$ Lemma 1	$\Omega(\log n)$ Lemma 1
distributions with $\mathbb{E}(H_n) = \Omega(n^\epsilon)$	Upper	$O(\log^2 n)$ Theorem 5	$O(\log^2 n)$ Theorem 2	$O(\log^2 n)$ Theorem 4
	lower	$\Omega(\log^2 n)$ Lemma 1	$\Omega(\log n)$ Lemma 1	$\Omega(\log n)$ Lemma 1

Table 1. Summary of the main results of this paper.

not T. A *distance labelling scheme* is a two-component tuple $\mathcal{L} = \langle L, f\rangle$ such that $f(L(u,T), L(v,T)) = d_T(u,v)$ for any pair of vertices $u, v \in T$. The *length* of a labelling scheme $\mathcal{L}$ for a tree T with n vertices, denoted as $\ell_n(\mathcal{L}, T)$, is defined as $\ell_n(\mathcal{L}, T) = \max_{u\in T} |L(u,T)|$, where $|x|$ is the number of bits in the integer x. The *length* $\ell_n(\mathcal{L})$ of a labelling scheme $\mathcal{L}$ is defined as $\ell_n(\mathcal{L}) = \max_T \ell_n(\mathcal{L}, T)$. All logarithmic functions $\ln$ in this paper are in base 2. It is easy to show that

Lemma 1. *For any tree labelling scheme $\mathcal{L}$ and tree distribution, $E(\ell_n(\mathcal{L})) \geq \log n$.*

3 Three Tree Labelling Schemes

In this section, we first present two new tree labelling schemes, namely, *the backbone-based labelling scheme* and *the rake-based labelling scheme*. We then review the separator-based labelling scheme and discuss the worst case performances of these three schemes.

3.1 Backbone-Based Labelling

Given a tree T with root r, a *backbone* $\mathcal{B}(T)$ is a path from the root r to leaf formed recursively as follows. If r has no child, then the backbone is r itself. If r has one child, say h_1, then the backbone is the path of r concatenated by $\mathcal{B}(T^{h_1})$, i.e., $\mathcal{B}(T) = r \oplus \mathcal{B}(T^{h_1})$. If r has more than one child, then the backbone is the path of r concatenated by $\mathcal{B}(T^{h_1})$ where h_1 is the child of r such that $|T^{h_1}|$ is maximum among all r's children, i.e., $\mathcal{B}(T) = r \oplus \mathcal{B}(T^{h_1})$. Here $\mathsf{P}_1 \oplus \mathsf{P}_2$ stands for the concatenatation of two paths.

Algorithm 1: Backbone-Based Vertex Labelling

for each internal vertex v_i **do**
  Assign a unique positive label $\mu(v_i, v_j)$ between 1 and W_i, where W_i is the number of v_i's child, for every vertex v_j that is v_i's child.
for $i = 0$ to $C_B(T) - 1$ **do**
  for each tree T_j in forest $\mathcal{D}^{(i)}(T)$ **do**
    Let v_j be T_j's root and $\mathcal{B}(T_j)$ be its backbone, and v_ℓ be v_j's parent if it exists.
    for every vertex $v_k \in \mathcal{B}(T_j)$ **do**
      Set $L_B(v_k, T) = L_B(v_\ell, T) \circ \langle d_T(v_k, v_j), \mu(v_\ell, v_j)\rangle$ if v_ℓ exists and set $L_B(v_k, T) = \langle d_T(v_k, v_j), 0\rangle$ otherwise. Here, the $\circ$ separates the label into *chunks*.

Algorithm 2: Backbone-Based Distance Decoder

Without loss of generality, we assume $L_B(u, T) = \mathcal{L}_0(u) \circ \cdots \circ \mathcal{L}_a(u)$ and $L_B(v, T) = \mathcal{L}_0(v) \circ \cdots \circ \mathcal{L}_b(v)$ with $a \geq b$. Here, $\mathcal{L}_i(u)$ is the $i + 1$ part of the label $L_B(u, T)$.
Assume $\mathcal{L}_c(u) = \langle x, y\rangle$. For notational simplicity, we let $\mathcal{L}_c(u)[1] = x$ and $\mathcal{L}_c(u)[2] = y$.
Set $dis = 0$ and find the smallest index c such that $\mathcal{L}_c(u) \neq \mathcal{L}_c(v)$ if such c exists.
if c does not exist **then**
  $dis = dis + \mathcal{L}_i(v)[1]$ for $i = b + 1$ to a.
else
  $dis = dis + \mathcal{L}_i(v)[1]$ for $i = c + 1$ to a and $dis = dis + \mathcal{L}_i(u)[1]$ for $i = c + 1$ to b.
  Set $dis = dis + \mathcal{L}_c(u)[1] + \mathcal{L}c(v)[1]$ if $\mathcal{L}_c(u)[2] \neq \mathcal{L}_c(v)[2]$ and set $dis = dis + |\mathcal{L}_c(u)[1] - \mathcal{L}c(v)[1]|$ otherwise.
Output $f_B(L_B(u, T), L_B(v, T)) = dis$.

Fig. 1. The Backbone-Based Distance Labelling Scheme.

Given a forest F, let $B(F) = \bigcup_{T \in F} \mathcal{B}(T)$. Define a *d-backbone* operation as first removing the edges in $\mathcal{B}(F)$ from F and then removing the resulting isolated vertices in F from F to produce a forest $\mathcal{D}(F)$. For simplicity, we denote $\mathcal{D}^{(k)}(F) = \mathcal{D}(\mathcal{D}^{(k-1)}(F))$, i.e., $\mathcal{D}^{(k)}(F)$ is the forest after k d-backbone operations on the original forest F. Let $C_B(T)$ denote the number of d-backbone operations needed to separate a tree T into isolated vertices. We have the following theorem (proof omitted):

Theorem 1. *For a tree T of n vertices, $C_B(T) \leq \log n$.*

Figure 1 presents our backbone-based labelling scheme $\mathcal{L}_B = \langle L_B, f_B\rangle$. Given a vertex u, its label $L_B(u, T)$ is a series of two element tuples separated by the " $\circ$ " symbol. We call each two element tuple a *chunk* of the label. Let $L_B(u, T) = \mathcal{L}_0(u) \circ \ldots \circ \mathcal{L}_i(u) \circ \ldots \mathcal{L}_a(u)$, where $\mathcal{L}_i(u)$ is the ith chunk of the label. Let c be the smallest index such that $\mathcal{L}_c(u) \neq \mathcal{L}_c(v)$ if it exists. Without loss of generality, assume that $\mathcal{L}_c(u) < \mathcal{L}_c(v)$. A key observation is that the vertex with label $\mathcal{L}_0(u) \circ \cdots \circ \mathcal{L}_c(u)$ is the least common ancestor of vertices with label $L_B(u, T)$ and $L_B(v, T)$.

In Algorithm 1, for every vertex v_i, when we assign child-label to v_j who is v_i's child, we assume the label length is $\log W_i$, where W_i is the number of children of v_i. However, given W_i children, when you assign a label ℓ, the label length is $\log \ell$ instead of $\log W_i$. With this observation [9], we can reduce the total tree label length

by applying the following *reshuffle process*. First, we apply Algorithm 1 to obtain a label $L_B(u,T)$ for every vertex u. Initially, we mark all the internal vertices as "unprocessed" and all leaf vertices as "processed". While there is an "unprocessed" vertex , we pick one vertex v such that all of its children are processed. Without loss of generality, we assume that $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ are v's children who are not on the same backbone of v. For any vertex w in tree $T^{v_{i_j}}$, the label of $L_B(w,T)$ should contain $L_B(v,T)$ as a common prefix and the second element of $(a+1)$th chunk is also the same. Assume that $L_B(w,T) = L_B(v,T) \circ \mathcal{L}_{a+1}(w) \circ \mathcal{L}_{a+2}(w) \circ \ldots \circ \mathcal{L}_c(w)$. Define $\kappa(w) = \sum_{i=a+2}^{c} \log(\mathcal{L}_i(w)[2])$, and $\gamma(v_{i_j}) = \max_{w \in T^{v_{i_j}}} \kappa(w)$. We sort the vertices $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ according to the size of their subtrees $T^{v_{i_j}}$ in an ascending order, and let σ be the index of the sorted list, i.e., $|T^{v_{i_{\sigma(j)}}}|$ is the jth largest. Then we reassign $L_{a+1}(v)[2] = j$ to each vertex v if v is in the tree $T^{v_{i_{\sigma(j)}}}$. Observe that this reassign process does not affect the label of the first element of any chunks and the correctness is straightforward. Following Lemma reveals a property of the reshuffle process (proof omitted due to space limit).

Lemma 2. *After the reshuffle process, $\gamma(r) \leq 2\log n$ for $\mathcal{L}_B$, where r is the root.*

Notice that the reshuffle process does not depend the any specific properties of the Backbone-Based Distance Labelling Scheme. Thus, even we change the labelling scheme for the first element, as long as the label contains at most $\log n$ chunks, Lemma 2 still holds. Recall that the label of vertex u is $L_B(u,T) = \mathcal{L}_0(u) \circ \ldots \circ \mathcal{L}_k(u)$, where $\mathcal{L}_i(u)$ is tuple composed of two integers. Since $\sum_{i=1}^{k} \log(\mathcal{L}_i(u)[2]) \leq \gamma(r)$, we have

Theorem 2. *$\ell_n(\mathcal{L}_B, T)$ and the time complexity of decoding is $O(\log n \cdot \log H_n(T))$ for any tree T with n vertices.*

PROOF. From the definition of tree label length, $\ell_n(\mathcal{L}_B) = \max_T \ell_n(\mathcal{L}_B, T) \leq \log(\max\{H_n(T)\} \cdot C_B(T) + \gamma(r) \leq \log n \cdot [\log(\max\{H_n(T)\} + 2]$. □

3.2 Rake-Based Scheme

In this section, we present a new tree labelling scheme based on the tree decomposition scheme by Kao [11]. A *chain* of T is a path in T such that every vertex of the given path has at most one child in T. A *tube* of T is a maximal chain of T. A *root path* of a tree is a tree path whose head is the root of that tree; similarly, a *leaf path* is one ending at a leaf. A *leaf tube* of T is a tube that is also a leaf path. Let $LT(T)$ denote the set of leaf tubes in T. Let $\mathcal{R}(T) = T - LT(T)$, i.e., the subtree of T obtained by deleting from T all its leaf tubes. The operation $\mathcal{R}$ is called the *rake operation*.

A *tube system* of a tree T is a set of tree paths $P_1, \cdots, P_m$ in T such that $T^{h_1}, \cdots, T^{h_m}$ are pairwise disjoint, where h_i is the head of P_i. We can iteratively rake T to obtain tube systems. Every rake operation produces a tube system of T until T is raked to empty. Given a tree T, let $\mathcal{R}^{(i)}(T)$ be the remaining tree after ith rake operation and $C_R(T)$ be the number of rake operations needed to make the tree empty. Similarly, we have

Theorem 3. *For any tree T of n vertices, $C_R(T) \leq \log n$.*

Algorithm 3: Rake-Based Vertex Labelling

1: **for** each internal vertex v_i **do**
2: Assign a unique positive ID $\mu(v_i, v_j)$ for every vertex v_j that is v_i's child, i.e., $\mu(v_i, v_a) \neq \mu(v_i, v_b)$ if v_a and v_b are v_i's children.
3: Let $C_R(T)$ be the number of rake operations needed to make T empty.
4: **for** $i = C_R(T) - 1$ down to 0 **do**
5: **for** each tube S in LT($R^i(T)$) **do**
6: Let h be the head of the tube S, i.e., the vertex with the smallest level in the tube, and let h' be the parent of h in the tree T if such h' exists.
7: **for** each vertex v_j in tube S **do**
8: Set the label of v_j as $L_R(v_j, T) = L_R(v_k, T) \circ \langle d_T(v_j, h'), \mu(h', h)\rangle$ if h' exists and set $L_R(v_j, T) = \langle d_T(v_j, r), 0\rangle$ otherwise.
9: Apply the reshuffle process to modify the second element of the chunks of the label.

Algorithm 4: Rake-Based Distance Decoder

1: For any pair of vertices $u \neq v$, we assume $L_R(u, T) = \mathcal{L}_0(u) \circ \cdots \circ \mathcal{L}_a(u)$ and $L_R(v, T) = \mathcal{L}_0(v) \circ \cdots \circ \mathcal{L}_b(v)$ with $a \geq b$. Assume $\mathcal{L}_c(u) = \langle x, y\rangle$. For notational simplicity, we let $\mathcal{L}_c(u)[1] = x$ and $\mathcal{L}_c(u)[2] = y$.
2: Set $dis = 0$ and find the smallest index c such that $\mathcal{L}_c(u) \neq \mathcal{L}_c(v)$ if such c exists.
3: **if** c does not exist **then**
4: $dis = dis + \sum_{i=b+1}^{a} d_i(v)$.
5: **else**
6: Set $dis = dis + \sum_{i=c+1}^{a} d_i(v) + \sum_{i=c+1}^{b} d_i(u)$.
7: Set $dis = dis + \mathcal{L}_c(u)[1] + \mathcal{L}c(v)[1]$ if $\mathcal{L}_c(u)[2] \neq \mathcal{L}_c(v)[2]$ and $dis = dis + |\mathcal{L}_c(u)[1] - \mathcal{L}c(v)[1]|$ otherwise.
8: Output $f_R(L_R(u, T), L_R(v, T)) = dis$.

Fig. 2. The Rake-Based Distance Labelling Scheme.

Based on the rake operation, we define a labelling scheme $\mathcal{L}_R = (L_R, f_R)$ as follows. For the rake-based labelling scheme defined in Algorithm 3 and Algorithm 4, similar to the backbone scheme, by assuming that $d_c(u) < d_c(v)$, a key observation about vertex u, v's least common ancestor is that the vertex with label $\mathcal{L}_0(u) \circ \cdots \circ \mathcal{L}_c(u)$ is the least common ancestor of vertices with label $L_R(u, T)$ and $L_R(v, T)$.

From Lemma 2 and Theorem 3, $\ell_n(\mathcal{L}_R) = \max_T \ell_n(\mathcal{L}_R, T) \leq \log(H_n(T)) \cdot C_R(T) + \gamma(r) \leq \log n \cdot (\log(H_n(T)) + 2)$. We thus have

Theorem 4. *The length of $\ell_n(\mathcal{L}_R, T)$ is $O(\log n \cdot \log H_n(T))$ and the time complexity of decoding is $O(\log n \cdot \log H_n(T))$ for any tree T with n vertices.*

3.3 Separator-Based Labelling

In this section, we review a tree labelling scheme first proposed by Peleg [12] and then improved by Gavoille [9]. The key idea is to find a *separator*, i.e., a vertex here, of a

tree such that the removal of the separator breaks the tree into several subtrees each with at most half of the vertices in the original tree. Iteratively remove separators of the remaining trees until all vertices are disconnected. For more details of the separator-based labelling scheme please refer to [12] and [9].

Again, a key observation here is that the vertex with label $\mathcal{L}_0(u) \circ \cdots \circ \mathcal{L}_c(u)$ is the least common ancestor of vertices with label $L_S(u,T)$ and $L_S(v,T)$. Regarding the length of the separator-based labelling scheme, we have the following two theorems (their proofs are omitted here due to space limit).

Theorem 5. *$\ell_n(\mathcal{L}_S, T)$ is $O(\log n \cdot \log(H_n(T)))$ for any tree T with n vertices.*

Theorem 6. *$\ell_n(\mathcal{L}_S, T)$ is $\Omega(\max\{\frac{\log n \cdot \log\log n}{\log k}, \log^2(H_n(T)\})$ for any tree T with n vertices and bounded degree k.*

4 Expected Label Length under Binary Search Tree Distribution

In Section 3, we presented two tree labelling schemes which have the worst case length $\Theta(\log^2 n)$ for any binary tree. We focus on the expected label length under binary search tree distribution in this section and under uniform tree distribution in the next section.

4.1 General Upper Bound

In this subsection, we build a general but not too bad upper bound for the expected length of $\ell_n(\mathcal{L}_R, T)$ and $\ell_n(\mathcal{L}_B, T)$ when the trees are *binary search trees* with usual randomization; that is, the binary search tree is constructed in a standard fashion (n consecutive insertions) from a random permutation of $\{1, 2, \cdots, n\}$, where each permutation is equally likely. It has been proved in [13] that the expected height of a random binary search tree is $\mathbb{E}(H_n) = \alpha \log n - \beta \log\log n + O(1)$, where $\alpha \log\left(\frac{2e}{\alpha}\right) = 1, \alpha \geq 2$ and $\beta = \frac{3}{2\log\frac{\alpha}{2}}$. Numerically, $\alpha = 4.311\cdots$, and $\beta = 1.953\cdots$. With the above fact, we can give an upper bound for the expected length of both backbone-based labelling scheme and rake-based labelling scheme, and this technique can be applied to other tree randomization also. The proof is omitted due space limit.

Theorem 7. *The expected label lengths for both backbone-based scheme, rake-based scheme, and separator based scheme are at most $\log n \cdot \log\log n + \log\alpha \log n$, where α is a constant satisfying the equation $\alpha \log\left(\frac{2e}{\alpha}\right) = 1, \alpha \geq 2$.*

4.2 Lower Bound of the Expected Length for Backbone-Based Scheme and Separator-Based Scheme

Given the upper bound of expected length for backbone-based scheme, we would like to compute the lower bound for $\mathbb{E}(\ell_n(\mathcal{L}_B), T)$ and find the gap between them. Following theorem gives a lower bound for the expected length of a random binary search tree based on backbone-based scheme.

Theorem 8. *The expected label length of a random binary search tree based on the backbone-based scheme is $\Omega(\frac{\log n \cdot \log\log n}{\log\log\log n})$, i.e., $\mathbb{E}(\ell(\mathcal{L}_B), T) = \Omega(\frac{\log n \cdot \log\log n}{\log\log\log n})$.*

The proof of Theorem 8 is omitted here due to space limit. Theorem 8 gives a lower bound that is very close to the upper bound. The gap is only $\log\log\log n$, and we conjecture that the lower bound is $\Omega(\log n \cdot \log\log n)$ which is tight. Similarly, we have a lower bound of the expected length for separator-based Scheme, and it can be obtained directly from Theorem 6 since for a a binary search tree, the degree of the vertex is bounded by 3 and $\ell(\mathcal{L}_B) = \Omega(\log n \cdot \log\log n)$ from Theorem 6.

Theorem 9. *The expected label length of a random binary search tree based on the separator-based scheme is* $\Omega(\log n \cdot \log\log n)$*, i.e.,* $\mathbb{E}(\ell(\mathcal{L}_B), T) = \Omega(\log n \cdot \log\log n)$.

Theorem 9 and Theorem 7 together shows that the expected length for separator-based scheme for random binary search tree is exactly $\Theta(\log n \cdot \log\log n)$.

4.3 Upper Bound of Expected Length for Rake-Based Scheme

In this section, we give a tighter upper bound of the expected tree label length for rake-based scheme. We first present the following theorem (proof omitted here).

Theorem 10. *The expected label length of a random binary search tree for rake-based scheme is* $\log n \cdot \log\log\log n + \log\alpha \cdot \log n + o(1)$*, where* $\alpha = 2^{13}+1$*, i.e.,* $\mathbb{E}(\ell_n(\mathcal{L}_B, T)) = \log n \cdot \log\log\log n + \log\alpha \cdot \log n + o(1)$.

Remember that for a tree with n vertices, we need at least $\log n$ bits to represent the vertices even without the requirement to recover the distance. Thus, from Theorem 10, our rake-based Scheme is almost tight. Our conjecture is that the upper bound could be improved to $O(\log n)$, which matches the lower bound. An interesting result drawn from Theorem 8 and Theorem 10 is that under the binary search tree distribution, usually the rake-based Scheme is better than backbone-based scheme. Recall that for the backbone based scheme, the length of the backbone $\mathcal{B}(T)$ is at least $\log(|T|)$. However, for rake based scheme, every rake operation decreases the height of the tree at least by 1 and most often more than 1. Thus, the last tube of the tree T, as we proved, is $O(\log\log n)$ with high probability, compared with $O(\log n)$ for the backbone. Therefore, it is natural that the rake-based scheme outperforms the backbone-based scheme.

5 Expected Label Length Under Uniform Binary Tree Distribution

In this section, we consider the binary trees with *uniform* distribution; that is every distinct binary tree with n vertices has the same probability. It is well known that there are C_n of enumeration of different binary trees with n vertex, where C_n is *Catalan Number*. Based on this fact, we have the following lower bounds for the backbone-based scheme, rake-based scheme and separator-based scheme.

Theorem 11. *The expected tree label length of backbone-based scheme is* $\Omega(\frac{\log^2 n}{\log\log n})$.

Theorem 12. *The expected tree label length of rake-based scheme is* $\Omega(\frac{\log^2 n}{\log\log n})$.

Theorem 13. *The expected tree label length of separator-based scheme is* $\Theta(\log^2 n)$.

The lower bound of the expected label length of backbone-based, rake-based and separator-based are $\Omega(\frac{\log^2 n}{\log\log n})$, $\Omega(\frac{\log^2 n}{\log\log n})$ and $\Omega(\log^2 n)$ respectively. These lower bounds either are very close to or match the upper bounds $\log^2 n$, and we conjecture that the lower bounds for both the backbone-based and rake-based schemes are also $\Omega(\log^2 n)$, which is asymptotically tight.

6 Expected Label Length under Several Other Tree Distributions

We then discuss the upper and lower bounds in a more general setting. Generally, we have the following results on the expected label length for any tree distribution:

Theorem 14. *Under any tree distribution, we have (1)* $\mathbb{E}(\ell_n(\mathcal{L}_R, T)) \leq \log \mathbb{E}(H_n(T)) \cdot \log n$*; (2)* $\mathbb{E}(\ell_n(\mathcal{L}_B, T)) \leq \log \mathbb{E}(H_n(T)) \cdot \log n$*; (3)* $\mathbb{E}(\ell_n(\mathcal{L}_S, T)) \leq \log \mathbb{E}(H_n(T)) \cdot \log n$.

Theorem 14 reveals an important information about the expected label length: the upper bound of expected label length relates to the expected height of the tree. For the lower bound of the expected label length, we have the following theorem.

Theorem 15. *For any degree bounded tree distribution, if the probability* $\mathbb{P}(H_n(T) \geq \mathbb{E}(H_n(T)) = \alpha$ *where* α *is some constant, then the expected length of separator-based scheme is* $\Omega(\frac{\log(n) \cdot \log(\mathbb{E}(H_n(T))}{\log k})$*, where* k *is the degree bound.*

From the previous two sections, one may observe that for bounded degree tree distribution, the label length depends on the expected tree height and size of the largest subtree. When the expected tree height is $O(n^\epsilon)$ where ϵ is some constant, the label length for the backbone-based, rake-based and separator-based are most likely to be similar, which is close to $O(\log^2 n)$, under most distributions. When the expected tree height is $O(\log^\epsilon n)$, the backbone-based, rake-based and separator-based schemes can achieve a better expected label length, which is $O(\log n \cdot \log\log n)$. We also conjecture that the label length of rake-based scheme can achieve $O(\log n \cdot \log\log\log n)$ or even $O(\log n)$ under certain tree distributions, which is tight.

7 Conclusion

In this paper, we studied how to label the vertices of a tree such that we can decide, given only the labels of two vertices, their distance in the tree. Specifically, we present two new distance labelling schemes that can achieve asymptotic optimal length $O(\log n \cdot \log(H_n(T))$ and have a much smaller expected label length under certain tree distributions. In the meanwhile, we also show how to find the least common ancestor of any two vertices based on their labels only. Rake-based labelling scheme usually achieves a smaller expected label length than backbone-based and separator-based schemes for most tree distributions with average low height. A remaining future work is to close the gaps between the upper bounds and the lower bounds for various tree distributions, and to prove the conjectures listed in our full version [17]. For more details of the proof, please refer [17] also.

References

1. S. ALSTRUP, P. BILLE, AND T. RAUHE, *Labeling schemes for small distances in trees*, In Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, 2003, pp. 689–698.
2. S. ALSTRUP, C. GAVOILLE, H. KAPLAN, AND T. RAUHE, *Nearest common ancestors: A survey and a new distributed algorithm*, in SPAA'02, 2002.
3. S. ALSTRUP AND T. RAUHE, *Lower bounds for labeling schemes supporting, ancestor, sibling, and connectivity queries*, Tech. Report IT-C, nr. 10, IT University of Copenhagen, 2001.
4. ———, *Small induced universal graphs and compact implicit graph representations*, in IEEE FOCS, 2002.
5. M. A. BREUER, *Coding the vertexes of a graph*, in IEEE Transactions on Information Theory, vol. 12, April 1966, pp. 148–153.
6. M. A. BREUER AND J. FOLKMAN, *An unexpected result on coding the vertices of a graph*, in Journal of Mathematical Analysis and Applications, vol. 20, 1967, pp. 583–600.
7. S. P. C. GAVOILLE, D. PELEG AND R. RAZ, *Distance labeling in graphs*, in Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete algorithms, 2001, pp. 210–219.
8. P. FLAJOLET AND A. ODLYZKO, *The average height of binary trees and other simple trees*, in Journal of Computer and System Sciences, vol. 25, 1982, pp. 171–213.
9. C. GAVOILLE, M. KATZ, N. KATZ, C. PAUL, AND D. PELEG, *Approximate distance labeling schemes*, in 9th Annual European Symposium on Algorithms (ESA), vol. 2161 of LNCS, 2001, pp. 476–488.
10. C. GAVOILLE AND D. PELEG, *Compact and localized distributed data structures*, Distrib. Comput., 16 (2003), pp. 111–120.
11. M.-Y. KAO, *Tree contractions and evolutionary trees*, SIAM Journal on Computing, 27 (1998), pp. 1592–1616.
12. D. PELEG, *Proximity-preserving labeling schemes and their applications*, in Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science, Springer-Verlag, 1999, pp. 30–41.
13. B. REED, *The height of a random binary search tree*, Journal of ACM, 50 (2003), pp. 306–332.
14. M. N. S. KANNAN AND S. RUDICH, *Implicit representation of graphs*, in Proceedings of the Twentieth annual ACM symposium on Theory of computing, ACM Press, 1988, pp. 334–343.
15. J. P. SPINRAD, *Efficient Graph Representations*, American Mathematical Society, June 2003.
16. D. B. WEST, *Introduction to Graph Theory*, Prentice Hall, 2nd edition ed., August 2000.
17. MING-YANG KAO, XIANG-YANG LI, AND WEIZHAO WANG, *Average Case Analysis for Tree Labelling Schemes*. Full veresion of the paper is available at `http://www.cs.iit.edu/~xli/publications-select.html`