Labeling Schemes for Weighted Dynamic Trees (Extended Abstract)

Amos Korman * and David Peleg *

The Weizmann Institute of Science,
Rehovot, 76100 Israel
{pandit,peleg}@wisdom.weizmann.ac.il

Abstract. This paper studies β-approximate distance labeling schemes, which are composed of a *marker* algorithm for labeling the vertices of a graph with short labels, coupled with a *decoder* algorithm allowing one to compute a β-approximation of the distance between any two vertices directly from their labels (without using any additional information). As most applications for informative labeling schemes in general, and distance labeling schemes in particular, concern large and dynamically changing networks, it is of interest to focus on *distributed dynamic* labeling schemes. The paper considers the problem on dynamic weighted trees and cycles where the vertices of the tree (or the cycle) are fixed but the (positive integral) weights of the edges may change. The two models considered are the *fully dynamic* model, where from time to time some edge changes its weight by a fixed quanta, and the *increasing dynamic* model in which edge weights can only grow. The paper presents distributed β-approximate distance labeling schemes for the two models, for $\beta > 1$, and establishes upper and lower bounds on the required label size and the communication complexity involved in updating the labels following a weight change.

1 Introduction

In order for a network representation method to be effective in the context of a large and distributed communication network, it must allow users to efficiently retrieve useful information about the network. Recently, a number of studies focused on a *localized* network representation method based on assigning a (hopefully short) *label* to each vertex, allowing one to infer information about any two vertices *directly* from their labels, without using *any* additional information sources. Labeling schemes have been developed for a variety of information types, including vertex adjacency [6, 5, 13], distance [19, 17, 12, 11, 9, 14, 21, 7], tree routing [8, 22], flow and connectivity [16], tree ancestry [1, 3, 4, 15], and various other tree functions, such as center, least common ancestor, separation level, and Steiner weight of a given subset of vertices [20]. See [10] for a survey.

By now, the basic properties of localized labeling schemes for *static* (fixed topology) networks are reasonably well-understood. However, when considering applications such as distributed systems and communication networks, the typical setting is dynamic, namely, the network topology undergoes repeated changes. Therefore, for a representation scheme to be useful in practice, it should be capable of reflecting up-to-date information in a dynamic setting, which may require occasional updates to the labels. Moreover, the algorithm for generating and revising the labels must be *distributed*, in contrast with the sequential and centralized label assignment algorithms described in the above cited papers.

The study of distributed labeling schemes for the dynamic setting was initiated in [18], which concentrates on the setting of an unweighted tree where at each step a leaf can be added to or removed from the tree. The labeling scheme presented therein for distances in this "leaf-dynamic"

* Supported in part by a grant from the Israel Science Foundation.

tree model has amortized message complexity $O(\log^2 n)$ per operation, where n is the size of the tree when the operation takes place. The protocol maintains $O(\log^2 n)$ bit labels, when n is the current tree size. This label size is known to be optimal even in the static scenario. A second result of [18] introduces a more general labeling scheme for the leaf-dynamic tree model, based on extending an existing *static* tree labeling scheme to a dynamic setting. The approach fits a number of natural tree functions, such as distance, separation level and flow. The main resulting scheme incurs an overhead of $O(\log n)$ multiplicative factor in both the label size and amortized message complexity in the case of dynamically growing trees (with no deletions). If an upper bound on n is known in advance, this method can yield a different tradeoff, with $O(\log^2 n/\log\log n)$ multiplicative overhead on the label size but only an $O(\log n/\log\log n)$ overhead on the amortized message complexity. In the non-restricted leaf-dynamic tree model (where both additions and deletions are allowed) the scheme incurs also an increased *additive* overhead in amortized communication, of $O(\log^2 n)$ messages per operation.

One key limitation of the setting studied in [18] is that the links are assumed to be unweighted. In reality, network distances are often based on link weights, and operator-initiated or traffic-dictated changes in these weights affect the resulting distances and subsequently the derived routing and circuit establishing decisions. In fact, whereas physical topology changes are relatively rare and are usually viewed as a disruption in the normal operation of the network, link weight changes are significantly more frequent and may be considered as part of the normal network operation. Subsequently, while it may be conceivable to approach physical topology changes by an offline label reorganization algorithm, this is an unreasonable approach when it comes to link weight changes, and a distributed update mechanism is desirable.

The current paper makes a step towards overcoming this limitation by investigating distance labeling schemes in dynamic settings involving changing link weights. The first model studied is the *fully dynamic model*. This model considers an underlying topology network with positive integer edge weights, where the vertices and edges of the network are fixed but at each time an edge weight can increase or decrease by a fixed quanta (which for notational convenience is set to be 1), as long as the weight remains positive. (Our algorithms and bounds apply also for larger weight changes, as clearly, a weight change of $\Delta > 1$ can be handled, albeit naively, by simulating it as Δ individual weight changes of 1.)

The second model considered is the *increasing dynamic model* which is the fully dynamic model restricted to events where an edge weight can only increase by one at each step.

The underlying network topologies considered for the first model are trees and cycles. The second model considers only trees.

As shown in Sect. 2, any *exact* distance labeling scheme for either of the models cannot avoid a linear message complexity per operation in some worst-case scenarios. We therefore weaken the demands from the labeling scheme, and only require it to maintain a β-approximation of the distances (for $\beta > 1$) rather than exact distances. Such a scheme is refered to as a *β-approximate distance labeling scheme*.

Our main results are as follows. Throughout the paper, denote by n the number of vertices in the network, $G = (V, E)$. For a tree network, we present a β-approximate distance labeling scheme in each of the two models described above. Let W be the maximum weight assigned to an edge in the tree. Then, in both schemes, the maximum size of a label given to any vertex is bounded by $O(\log^2 n + \log n \log W)$ which as shown in [12] is optimal for (exact) distance labeling schemes even in the static scenario.

For an edge e, let $B(e, d)$ be the number of vertices at distance d or less from an endpoint of the edge e and let $\Lambda = \max\{B(e, d)/d \mid d \geq 1, e \in E\}$. Denote by m the number of edge changes occurring in the tree. We show that for $\beta > 1$ bounded away from 1, the message and bit complexities of the protocol for the fully dynamic model are $O(m\Lambda \log^2 n)$ and $O(m\Lambda \log^2 n \cdot$

$\log \log n$) respectively. The message and bit complexities for the increasing dynamic model are $O(m \log^2 n + n \log n \log m)$ and $O(m \log^2 n \cdot \log \log n + n \log n \log m \cdot \log \log n)$ respectively.

For the fully dynamic model, if the underlying network topology is a path we describe a different β-approximate distance labeling scheme yielding a different tradeoff between the size of the labels and the communication complexity. The scheme uses maximum label size $O(\log n \log m)$ and its message complexity is $O(m \log^2 n)$.

Similarly, if the underlying topology is a cycle then we get two schemes with the same asymptotic complexities as for the path.

2 Preliminaries

Our network model is restricted to either tree or cycle topologies. We assume that the vertices of the network are fixed and that the edges of the network are assigned positive integer weights. The network is assumed to dynamically change via weight changes of the edges. For two vertices u and v in some graph, denote by $d^\omega(u, v)$ the weighted distance between u and v. In the *fully dynamic* (tree or cycle) model the following events may occur:

1. An edge (u, v) increases its weight by one.
2. An edge (u, v) with weight at least 2 decreases its weight by one.

Subsequent to an event on an edge $e = (u, v)$, its end points u and v are informed of this event. In the *increasing dynamic* model the only event that may occur is that an edge (u, v) increases its weight by one and subsequently u and v are informed of this event.

For $\beta \geq 1$, a *static β-approximate distance labeling scheme* $\pi = \langle \mathcal{M}(\beta), \mathcal{D} \rangle$ for a family of graphs $\mathcal{F}$ is composed of the following components:

1. A *marker* algorithm $\mathcal{M}(\beta)$ that given a graph in $\mathcal{F}$, assigns labels to its vertices.
2. A polynomial time *decoder* algorithm $\mathcal{D}$ that given the labels $Label(u)$ and $Label(v)$ of two vertices u and v in some graph $G \in \mathcal{F}$, outputs a *distance estimate* $\tilde{d^\omega}(u, v)$ satisfying $d^\omega(u, v)/\beta \leq d^\omega(u, v) \leq \beta \cdot \tilde{d^\omega}(u, v)$.

A *static distance labeling scheme* is a static 1-approximate distance labeling scheme. For examples of static distance labeling schemes see [18, 12, 19].

In this paper we are interested in distributed networks where each vertex in the graph represents a processor. This does not affect the definition of the decoder algorithm of the labeling scheme, since it is performed locally, but the marker algorithm must be implemented as a *distributed marker protocol*.

The approximate labeling schemes for the fully dynamic and the increasing dynamic models involve a marker protocol $\mathcal{M}$ which is activated after every change in the network topology. The protocol $\mathcal{M}$ maintains the labels of all vertices in the underlying graph so that the corresponding decoder algorithm will work correctly. We assume that the topological changes occur sequentially and are sufficiently spaced so that the protocol has enough time to complete its operation in response to a given topological change before the occurrence of the next change. We refer to scenario where m weight changes have occurred as an m-change scenario.

For a dynamic β-approximate labeling scheme $\pi = \langle \mathcal{M}(\beta), \mathcal{D} \rangle$, for either one of the models, we are interested in the following complexity measures.

- *Label Size*, $\mathcal{LS}(\mathcal{M}(\beta), m)$: the maximum size of a label assigned by $\mathcal{M}(\beta)$ to a vertex in the worst case n-vertex underlying graph and the worst vase m-change scenario. (the graph classes considered are trees and cycles).

– *Message Complexity*, $\mathcal{MC}(\mathcal{M}(\beta), m)$: the maximum number of messages sent by $\mathcal{M}(\beta)$ in the worst case n-vertex underlying graph and the worst vase m-change scenario.
– *Bit Complexity*, $\mathcal{BC}(\mathcal{M}(\beta), m)$: the maximum number of bits sent by $\mathcal{M}(\beta)$ in the worst case n-vertex underlying graph and the worst vase m-change scenario.

Next we establish some lower bounds for the message complexity. Throughout, we omit some proofs due to lack of space. See the full paper for detailed proofs.

Lemma 1. *Any* exact *distance labeling scheme $\pi = (\mathcal{M}, \mathcal{D})$ for the class of trees or for the class of cycles in either of the above dynamic models incurs a message complexity of $\mathcal{MC}(\mathcal{M}, m) = \Omega(mn)$.*

Next we establish a lower bound on the message complexity of β-approximate distance labeling schemes in the fully dynamic model. Let T be a rooted tree and let $\pi(\beta)$ be a β-approximate distance labeling scheme in the fully dynamic model on some graph family containing T. Let $B(e,d)$ be the number of vertices at distance d or less from an endpoint of the edge e. Depicting the tree with the root at the top, let $B_{down}(e,d)$ be the number of vertices at distance d or less below an endpoint of the edge e. Let $B_{up}(e,d) = B(e,d) - B_{down}(e,d)$. Let $\tilde{B}(e,d) = \min\{B_{up}(e,d), B_{down}(e,d)\}$ and $\tilde{\Lambda} = \max\{\frac{\tilde{B}(e,d)}{d} \mid d \geq 1, e \in E\}$. $\tilde{\Lambda}$ is an attempt at capturing the graph-theoretic parameter governing the complexity of the problem. We use the parameter $\tilde{\Lambda}$ in our lower bound and a slightly different parameter Λ in our upper bounds.

Lemma 2. *For constant β, $\mathcal{MC}(\pi(\beta), m) = \Omega(m\tilde{\Lambda})$.*

3 Dynamic labeling schemes for distance

3.1 Estimating the distance to the root

A key ingredient in our schemes concerns a mechanism for allowing each vertex v to estimate its distance from the root of the tree at any given moment. Throughout, we denote the path from a node v to the root by $\mathcal{P}_v$. We introduce two *root-distance* protocols in which each node v keeps a β-approximate of $d^\omega(v)$, v's weighted distance to the root. The complexity bounds of these protocols are expressed in terms of the following quantities. Let $B(e,d)$ be the number of vertices at distance d or less from an endpoint of the edge e. Let $\Lambda = \max\left\{\frac{B(e,d)}{d} \mid d \geq 1, e \in E\right\}$ and set $\alpha = \frac{\beta}{\beta-1}$ and $\gamma = \frac{\sqrt{\beta}}{\sqrt{\beta}-1}$. The first root-distance protocol, $\mathcal{R}_{dyn}(\beta)$, is applied to the fully dynamic model and satisfies $\mathcal{MC}(\mathcal{R}_{dyn}(\beta), m) = O(m\alpha\Lambda\log^2 n)$. The second root-distance protocol, $\mathcal{R}_{inc}(\beta)$, is applied to the increasing dynamic model and satisfies $\mathcal{MC}(\mathcal{R}_{inc}(\beta), m) = O(m\gamma\log^2 n + n\log_\beta m \log n)$.

The root-distance protocol $\mathcal{R}_{dyn}(\beta)$ for the fully dynamic model Inspired by [2], protocol $\mathcal{R}_{dyn}(\beta)$ is designed so that in the fully dynamic model each node v has a β-approximate to $d^\omega(v)$, v's weighted distance to the root. The message complexity of the protocol on m weight changes is $\mathcal{MC}(\mathcal{R}_{dyn}(\beta), m) = O(m\alpha\Lambda\log^2 n)$.

Each node v maintains two bins, a "local" bin $b_l(v)$ and a "global" bin $b_g(v)$, storing a varying number of tokens throughout the execution. Let $H(v)$ denote the height of v in the tree, namely, its unweighted (hop) distance from the root. The bins of each non-root node v at height $H(v)$, are assigned a *level*, defined as $Level(b_g(v)) = \max\{i \mid 2^i \text{ divides } H(v)\}$ and $Level(b_l(v)) = -1$.

Note that the level of the bin determines whether it is of type b_l or b_g. Therefore, in the following discussion, we omit the subscripts g and l unless it might cause confusion. For each bin b at node

v, on any path from v to a leaf, the closest bin b' such that $Level(b') = Level(b) + 1$ is set to be a *supervisor* of b. If for some path (from v to a leaf) there is no such bin then the leaf is set to be a supervisor of b. Note that the supervisors of a local bin are either the global bin of the same node or the global bins of its children. This defines a bin hierarchy.

1. The depth of the bin hierarchy is at most $\log n + 1$.
2. If $Level(b(v)) = l$ then any path from v to a node that holds one of $b(v)'s$ supervisors has at most $O(2^l)$ nodes.
3. On any path of length p, the number of level l bins that are supervisors is at most $1 + \frac{p}{2^{l-1}}$.

Let $\sigma = 2^{\lfloor \log \frac{1}{\alpha(\log n+1)} \rfloor}$. The number of tokens stored at each bin b at a given time is denoted $\tau(b)$. The tokens can be of either positive or negative type so by saying $\tau(b) = -x$ where x is a positive integer, we mean that b holds x negative tokens. The *capacity* of each bin depends on its level. Specifically, a bin b on $Level(b) = l$ may store $-C(l) \leq \tau(b) \leq C(l)$ tokens, where $C(l) = \max\{\sigma \cdot 2^l, 1\}$. Intuitively a level l bin can be thought of as storing at most $C(l)$ "positive" tokens and at most $C(l)$ "negative" ones. Positive and negative tokens cancel each other out, so at any given moment a bin is either empty or it stores tokens of at most one type. In fact, it will follow from the algorithm description that at any given moment a nonempty bin is half-full, namely, it stores either $C(l)/2$ positive tokens or $C(l)/2$ negative ones, or formally, $\tau(b) \in \{-C(l)/2, 0, C(l)/2\}$.

The protocol $\mathcal{R}_{dyn}(\beta)$.

1. Initially all bins are empty.
2. Each time a node learns that the edge to its parent has increased (respectively, decreased) its weight by one, it adds a +1 (resp., -1) token to fill its local bin.
3. Whenever a bin b with $\tau(b) = x$ (positive or negative) tokens gets a signal to add y tokens (where again, y is either positive or negative) it will now be considered having $\tau(b) = x + y$ tokens.
4. Whenever a bin b on level l at a node v gets filled with tokens (i.e., it either has $C(l)$ positive tokens or $C(l)$ negative tokens), v immediately empties the bin and broadcasts a signal to all its supervisor bins to add $C(l)$ (positive or negative) tokens to their bins. This signal message can consist of just l along with the appropriate sign.

In addition, each node v monitors the signals passing through it and estimates $d^\omega(v)$, v's weighted distance to the root, in the following way. Each node v keeps a counter $\tilde{d}(v)$, initially set to $v's$ distance to the root in the original tree. When a signal of level l with positive (resp. negative) sign reaches v or passes through it, v adds (resp. subtracts) $C(l)$ to $\tilde{d}(v)$.

For a node v, consider the path $\mathcal{P}_v$ from the root to v. Define $\phi(\mathcal{P}_v)$, the *amount of wasted tokens* on $\mathcal{P}_v$, as the sum of tokens left in the non-empty bins on $\mathcal{P}_v$ counted with their appropriate signs (i.e., with positive and negative tokens canceling each other out). Define $\mu(\mathcal{P}_v)$, the *number of wasted tokens* on $\mathcal{P}_v$, as the total number of (either positive or negative) tokens left in the non-empty bins on $\mathcal{P}_v$. More formally, $\phi(\mathcal{P}_v) = \sum_{b \text{ on } \mathcal{P}_v} \tau(b)$ and $\mu(\mathcal{P}_v) = \sum_{b \text{ on } \mathcal{P}_v} |\tau(b)|$.

Lemma 3. *At any given moment, $\phi(\mathcal{P}_v) = d^\omega(v) - \tilde{d}(v)$.*

Lemma 4. *For each node v, $\tilde{d}(v)/\beta \leq d^\omega(v) \leq \beta \cdot \tilde{d}(v)$*

Proof: Initially all bins are empty. If the capacity of a bin equals 1, the bin always remains empty since it serves only as a relay between the node it supervises and its supervisors (i.e., when a token is added to the bin it gets full and immediately gets empty while sending a message to its supervisor bins). The only bins that might not be empty are global bins $b_g(v)$ that are supervisors with capacity larger than 1. A bin which is not empty is necessarily half-full.

Fix v, denote the length of the path $\mathcal{P}_v$ by p. On $\mathcal{P}_v$, for each level l, there are at most $\frac{p}{2^{l-1}}$ bins at level l that are supervisors. Even if all of them are half-full we get that the total number of wasted tokens in bins of level l is at most $\frac{p}{2^{l-1}} \cdot \frac{1}{2} \cdot \frac{2^l}{\alpha(\log n+1)} = \frac{p}{\alpha(\log n+1)}$. Therefore, the number of wasted tokens on $\mathcal{P}_v$ (on all levels) satisfies $\mu(\mathcal{P}_v) \leq \frac{p}{\alpha}$. The proof follows since $|d^\omega(v) - \tilde{d}(v)| = |\phi(\mathcal{P}_v)| \leq \mu(\mathcal{P}_v) \leq \frac{p}{\alpha}$ and since both $\tilde{d}(v)$ and $d^\omega(v)$ are always at least p. ▮

Lemma 5. *1.* $\mathcal{MC}(\mathcal{R}_{dyn}(\beta), m) = O(m\alpha\Lambda \log^2 n)$
2. $\mathcal{BC}(\mathcal{R}_{dyn}(\beta), m) = O(m\alpha\Lambda \log^2 n \log\log n)$

Proof: We say that a bin b' *affects* bin b if by filling b' sufficiently many times, b gets full at least once. Define the *local fillers* of b to be the local bins that affect b. The local fillers of $b(v)$ lie on $\mathcal{P}_v$, the path from the root to v. Moreover, if b is of level l, then all the local fillers of b are at distance $O(2^l)$ from b. Define the *lowest local filler* of b, denoted $LLF(b)$, to be the local filler of b at the lowest height. For fixed l, we now use the LLF function to define a partition of the l-level bins. Let $b_{low} = LLF(b)$ for some level l bin b. Define the *local tree* of b, denoted $LT(b)$, as the smallest subtree of T rooted at b_{low} that contains any level l bin b' such that $b_{low} = LLF(b')$. The following properties are easy to show.

1. All the local fillers of b belong to $LT(b)$.
2. If b' is a level l bin such that $LT(b) \cap LT(b') \neq \emptyset$ then $LT(b') = LT(b)$.
3. The depth of $LT(b)$ is $O(2^l)$.

We now bound the communication caused by the level l bins. Enumerate the local trees $T_1, T_2, \ldots$ of the level l bins. Let e_i be an edge adjacent to the root of T_i. Let m_i be the number of weight changes occurring in T_i. Then none of the level l bins that belong to T_i got filled more than $\frac{m_i}{C(l)}$ times. Each time such a bin gets filled it sends a message to all its supervisors. These supervisors are at distance at most $O(2^l)$ below it, which is also at distance $O(2^l)$ from e_i. Hence even if all the level l bins in T_i get filled exactly $\frac{m_i}{C(l)}$ times, the message complexity incurred by them is bounded by $B(e_i, O(2^l)) \cdot \frac{m_i}{C(l)} = B(e_i, O(2^l)) \cdot \frac{\alpha(\log n+1)}{2^l} \cdot m_i = O(\alpha\Lambda m_i \log n)$. Therefore, the message complexity caused by level l bins during weight changes is bounded by $O(\alpha\Lambda m \log n)$ and the first part of the lemma follows as there are at most $\log n + 1$ levels. The second part of the lemma follows from the first part and from the fact that all messages are of $O(\log\log n)$ bits. ▮

The root-distance protocol, $\mathcal{R}_{inc}(\beta)$, for the increasing dynamic model As $\mathcal{R}_{dyn}(\beta)$, protocol $\mathcal{R}_{inc}(\beta)$ is designed so that each node v keeps a β-approximate to $d^\omega(v)$. However, the $\mathcal{R}_{inc}(\beta)$ protocol is designed to work in the increasing dynamic model where at each step an edge weight can only increase by one and indeed achieves better performance. We show that $\mathcal{MC}(\mathcal{R}_{inc}(\beta), m) = O(m\gamma \log^2 n + n \log n \log_\beta m)$.

For a node v, let $T(v)$ be the subtree rooted at v and let t_v be the number of vertices in $T(v)$. We say a child u of v is a *heavy child* of v if $t_u \geq t_w$ for any child w of v. For each non-leaf node v we choose a heavy-child u (breaking ties arbitrarily) and mark the edge connecting u and v. The non-heavy children of v are referred to as v's *light children*. The trees hanging down from v's light children are referred to as *v's light-subtrees*. The marked edges induce a decomposition of the graph into a collection $\mathcal{S}$ of edge-disjoint paths in the following stages. At the first stage, starting at the root, take into $\mathcal{S}$ the longest path that starts at the root and is composed of only marked edges. At the i'th stage, take into $\mathcal{S}$ all the longest paths that start with an unmarked edge emanating from some node on a path taken at the $i-1$'st stage and continue over marked edges.

We say a non-root node v *belongs* to a path P if the edge from v's parent to v belongs to P. Therefore each non-root node v belongs to exactly one path in $\mathcal{S}$; we denote this path by $P(v)$. We

denote by $P'(v)$ the subpath of $P(v)$ truncated at v (namely, $P'(v)$ doesn't include any descendent of v). For each path P we denote by $|P|$ the number of edges in P and by $|P|^w$ we denote the weighted length of P.

The decomposition $\mathcal{S}$ has the following property. Each path $\mathcal{P}_v$ from the root to v is decomposed into $k \leq \log n$ edge-disjoint paths $P_1 ... P_{k-1}$ (where each P_i is prefix of a path in S and $P_k = P'(v)$). Moreover, all edges in $\mathcal{P}_v$ are marked except maybe the first edges in $P_1, \ldots, P_k$. Denote the number of unmarked edges along $\mathcal{P}_v$ by $\eta(v)$.

The protocol $\mathcal{R}_{inc}(\beta)$ Each node v simultaneously invokes two protocols. The first, $\mathcal{R}_1$, is the protocol $\mathcal{R}_{dyn}(\sqrt{\beta})$ restricted to the path $P(v)$. The second protocol, $\mathcal{R}_2$, monitors the behavior of $\tilde{d}(v)$ where $\tilde{d}(v)$ is the approximated weighted distance from v to the root of $P(v)$ maintained by $\mathcal{R}_1$. Each time $\tilde{d}(v)$ increases by a multiplicative factor of $\sqrt{\beta}$, v broadcasts a signal to all the vertices in its light subtrees, containing the number of unmarked edges on the path $\mathcal{P}_v$ from the root to v.

Each node v monitors the $\mathcal{R}_1$ and $\mathcal{R}_2$ signals passing through it and estimates $d^\omega(v)$ in the following way. Decompose the path from the root to v into $P_1, \ldots, P_k$ as before. The node v keeps counters $d_1, \ldots, d_k$ for approximating $|P_1|^w, \ldots, |P_k|^w$ respectively, as follows. For each $1 \leq i \leq k-1$, denote by u_i the bottom node of P_i.

1. Initially $d_i = |P_i|$ for each i.
2. The counter d_k is maintained by $\mathcal{R}_1$.
3. Each time v gets an $\mathcal{R}_2$ signal from u_i, v raises d_i by a multiplicative factor of $\sqrt{\beta}$.
4. At all times, $\tilde{d}(v) = \sum d_i$.

Lemma 6. *At all times, $\tilde{d}(v) \leq d^\omega(v) \leq \beta \cdot \tilde{d}(v)$ and therefore $\mathcal{R}_{inc}$ guarantees that each node v maintains a β-approximation to $d^\omega(v)$.*

Lemma 7. *1. $\mathcal{MC}(\mathcal{R}_{inc}(\beta), m) = O(m\gamma \log^2 n + n \log n \log_\beta m)$*
2. $\mathcal{BC}(\mathcal{R}_{inc}(\beta), m) = O(m\gamma \log^2 n \cdot \log\log n + n \log n \log_\beta m \cdot \log\log n)$

3.2 Dynamic labeling schemes for distance in trees

throughout this subsection, the underlying topology network is restricted to trees. Given the fully dynamic or the increasing dynamic model, we show how to use a root-distance protocol (specifically, $\mathcal{R}_{dyn}(\beta)$ for the former or $\mathcal{R}_{inc}(\beta)$ for the latter) to give a dynamic labeling scheme for distances.

Definitions Given a tree T, a *separator* is a vertex v whose removal breaks T into disconnected subtrees of at most $n/2$ vertices each. It is a well known fact that every n-vertex tree T has a separator. As described in [19] one can recursively partition the tree by separators. For convenience, whenever a subtree T' on some level of this recursive partition is split into subtrees $T_1, \ldots, T_q$ by removing a separator node v from T', we formally define each of these subtrees T_i to include v as its root. In the resulting recursive partitioning, each vertex v belongs to a unique subtree $T_l(v)$ on each level l of the hierarchy, up to the level in which v itself is selected as the separator. For a vertex v and a level l of this recursive partitioning, denote by $r_l(v)$ the root of $T_l(v)$, which as explained above is the level l separator that defined $T_l(v)$.

The Marker Protocol $\mathcal{M}(\beta)$ The following discusion applies to both models with $\mathcal{R}$, their appropriate root-distance protocol ($\mathcal{R}_{dyn}(\beta)$ for the fully dynamic model and $\mathcal{R}_{inc}(\beta)$ for the increasing model). Simultaneously, for each level l, the marker protocol $\mathcal{M}(\beta)$ invokes $\mathcal{R}$ separately on each level l subtree. These subtrees are all disjoint and the root of such a tree is the appropriate level l separator. Thus, for each level l, vertex v keeps a β-approximation $\tilde{d}_l(v)$ for $d^\omega(v, r_l(v))$. Fix a vertex v and let $l(v)$ be the level on which v was selected as a separator. The label of v will consist of $l(v)$ pairs, $Label(v) = (\Psi_1(v), \ldots, \Psi_{l(v)}(v))$, where the first field of the pair $\Psi_j(v)$ gives the index of $r_l(v)$ and the second field gives $\tilde{d}_l(v)$.

Lemma 8. *Let W be the maximum weight given to an edge. Then $\mathcal{LS}(\mathcal{M}(\beta), m) = O(\log^2 n + \log n \log W)$.*

As shown in [12], the above label size is optimal for (exact) distance labeling schemes even in the static model.

Lemma 9. *1. For the fully dynamic model, $\mathcal{MC}(\mathcal{M}(\beta), m) = O(m\alpha\Lambda \log^3 n)$ and $\mathcal{BC}(\mathcal{M}(\beta), m) = O(m\alpha\Lambda \log^3 n \cdot \log\log n)$*

2. For the increasing dynamic model, $\mathcal{MC}(\mathcal{M}(\beta), m) = O(m\gamma \log^3 n + n \log^2 n \log_\beta m)$ and $\mathcal{BC}(\mathcal{M}(\beta), m) = O((m\gamma \log^3 n + n \log^2 n \log_\beta m) \cdot \log\log n)$.

The Decoder algorithm $\mathcal{D}(\beta)$ Algorithm $\mathcal{D}(\beta)$ gets as input the labels $Label(u)$ and $Label(v)$ of two vertices u and v.$\mathcal{D}(\beta)$ scans the pairs of $Label(u)$ and $Label(v)$ from right to left looking for the first pairs Ψ_u and Ψ_v with common first field, r. Denote by $\tilde{d}(u, r)$ and $\tilde{d}(v, r)$ the second fields of Ψ_u and Ψ_v respectively. Return $\mathcal{D}(\beta)(Label(u), Label(v)) = \tilde{d}(u, r) + \tilde{d}(v, r)$.

Lemma 10. *The decoder $\mathcal{D}$ yields a β-approximation to the weighted distance.*

3.3 A different labeling scheme $\mathcal{L}_{path-dyn}(\beta)$ for paths

Consider the fully dynamic model restricted to paths. We show a β-approximation labeling scheme for this model using label size $O(\log n \log m)$ and only $O(m\alpha \log^2 n)$ message complexity. Consider the root-distance protocol $\mathcal{R}_{dyn}(\beta)$ for the fully dynamic model on the path. In this model each node v monitors the messages passing through it and uses them to estimate its distance to the root. This is applied to all of v's separators and thus causes an overhead of $\log n$ on the message and bit complexities of $\mathcal{M}_{dyn}(\beta)$. The improved labeling scheme, $L_{path-dyn}(\beta)$, uses protocol $\mathcal{M}_{path-dyn}(\beta)$ which is very similar to protocol $\mathcal{R}_{dyn}(\beta)$, however, it is applied only on the path (and not separately for all the separators). Note that in the protocol $\mathcal{M}_{dyn}(\beta)$ each supervisor bin b supervisors at most two bins b_0 and b_1. Assume without loss of generality that b_0 is higher than b_1. Apart from the way the vertices monitor the messages, the only difference between $\mathcal{M}_{path-dyn}(\beta)$ and $\mathcal{R}_{dyn}(\beta)$ is that the messages include an additional bit indicating whether the message was originated at b_0 or b_1. Since the message and bit complexities of $\mathcal{L}path - dyn(\beta)$ is the same as the complexities of $\mathcal{R}_{dyn}(\beta)$ we get the following.

Lemma 11. $\mathcal{MC}(\mathcal{L}_{path-dyn}(\beta), m) = O(m\alpha \log^2 n)$ *and* $\mathcal{BC}(\mathcal{L}_{path-dyn}(\beta), m) = O(m\alpha \log^2 n \cdot \log\log n)$

The label structure Each node v keeps counters $c_i^0(v)$ and $c_i^1(v)$, where $c_i^0(v)$ (respectively, $c_i^1(v)$) is the number of messages passing through v whose destiny is a level i bin and whose origin bin was b_0 (resp., b_1). Let $h(v)$ be the height of v in T and let r be the number of levels.

Let $Label(v) = (h(v), c_1^0(v), c_1^1(v) \ldots, c_r^0(v), c_r^1(v))$. Since $r = O(\log n)$ and each c_i^k is of size at most $\log m$ we get the following.

Lemma 12. $\mathcal{LS}(\mathcal{L}_{path-dyn}(\beta), m) = O(\log n \log m)$.

The decoder - sketch: Intuitively, the decoder uses $Label(v)$ and $Label(u)$ to estimate the weighted distance between u and v by looking at the tokens coming in and out of u and v and checking which of them was originated at the interval (u, v). The amount of tokens originated in that interval and not known to the decoder is small and we therefore get a good estimation to the distance. A detailed description of the decoder is deferred to the full paper.

4 Dynamic labeling schemes for the fully dynamic model on cycles

We modify an approximate labeling scheme for the class of paths (namely, $\mathcal{L}_{path-dyn}$ or $\mathcal{L}_{dyn}$) to get a β-approximate labeling scheme for the fully dynamic model with the underlying topology restricted to cycles using the same asymptotic complexities as for paths. We give an intuitive review of the modified scheme, a detailed description will appear in the full paper.

Two vertices are pivots if they divide the weighted cycle into two arcs so that up to some small constant factor, the shortest way between two vertices on the same arc is on that arc (and not by going through the other arc). Our marker protocol constantly holds two vertices as pivots. The pivots simultaneously invoke the desired path-scheme for the clockwise paths from each pivot to itself. The label of a vertex is a concatenation of its labels in these two schemes. The decoder protocol uses the decoder protocol of the two schemes (on the appropriate labels) and outputs the smaller value.

We therefore get the following lemma for constant β.

Lemma 13. *By using $\mathcal{L}_{dyn}$ (resp., $\mathcal{L}_{path-dyn}$), we get a β-approximate distance labeling scheme for the fully dynamic model on cycles with the same asymptotic complexities of $\mathcal{L}_{dyn}$ (resp., $\mathcal{L}_{path-dyn}$) for paths.*

References

1. S. Abiteboul, H. Kaplan and T. Milo. Compact labeling schemes for ancestor queries. In *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2001.
2. Y. Afek, B. Awerbuch, S.A. Plotkin and M. Saks. Local management of a global resource in a communication. *J. of the ACM*, pages 1–19, 1989.
3. S. Alstrup, C. Gavoille, H. Kaplan and T. Rauhe. Identifying nearest common ancestors in a distributed environment. IT-C Technical Report 2001-6, The IT University, Copenhagen, Denmark, Aug. 2001.
4. S. Alstrup and T. Rauhe. Improved Labeling Scheme for Ancestor Queries. In *Proc. 19th ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2002.
5. M.A. Breuer and J. Folkman. An unexpected result on coding the vertices of a graph. *J. of Mathematical Analysis and Applications*, 20:583–600, 1967.
6. M.A. Breuer. Coding the vertexes of a graph. *IEEE Trans. on Information Theory*, IT-12:148–153, 1966.
7. E. Cohen, E. Halperin, H. Kaplan and U. Zwick. Reachability and Distance Queries via 2-hop Labels. In *Proc. 19th ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2002.
8. P. Fraigniaud and C. Gavoille. Routing in trees. In *Proc. 28th Int. Colloq. on Automata, Languages & Prog.*, LNCS 2076, pages 757–772, July 2001.
9. C. Gavoille and C. Paul. Split decomposition and distance labelling: an optimal scheme for distance hereditary graphs. In *Proc. European Conf. on Combinatorics, Graph Theory and Applications*, Sept. 2001.
10. C. Gavoille and D. Peleg. Compact and Localized Distributed Data Structures. Research Report RR-1261-01, LaBRI, Univ. of Bordeaux, France, Aug. 2001.
11. C. Gavoille, M. Katz, N.A. Katz, C. Paul and D. Peleg. Approximate Distance Labeling Schemes. In *9th European Symp. on Algorithms*, Aug. 2001, Aarhus, Denmark, SV-LNCS 2161, 476–488.
12. C. Gavoille, D. Peleg, S. Pérennes and R. Raz. Distance labeling in graphs. In *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms*, pages 210–219, Jan. 2001.

13. S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 334–343, May 1988.
14. H. Kaplan and T. Milo. Short and simple labels for small distances and other functions. In *Workshop on Algorithms and Data Structures*, Aug. 2001.
15. H. Kaplan, T. Milo and R. Shabo. A Comparison of Labeling Schemes for Ancestor Queries. In *Proc. 19th ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2002.
16. M. Katz, N.A. Katz, A. Korman and D. Peleg. Labeling schemes for flow and connectivity. In *Proc. 19th ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2002.
17. M. Katz, N.A. Katz, and D. Peleg. Distance labeling schemes for well-separated graph classes. In *Proc. 17th Symp. on Theoretical Aspects of Computer Science*, pages 516–528, February 2000.
18. A. Korman, D. Peleg, and Y. Rodeh. Labeling schemes for dynamic tree networks. In *Proc. 19th STACS Symp. on Theoretical Aspects of Computer Science*, March. 2002.
19. D. Peleg. Proximity-preserving labeling schemes and their applications. In *Proc. 25th Int. Workshop on Graph-Theoretic Concepts in Computer Science*, pages 30–41, June 1999.
20. D. Peleg. Informative labeling schemes for graphs. In *Proc. 25th Symp. on Mathematical Foundations of Computer Science*, volume LNCS-1893, pages 579–588. Springer-Verlag, Aug. 2000.
21. M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. In *Proc. 42nd IEEE Symp. on Foundations of Computer Science*, Oct. 2001.
22. M. Thorup and U. Zwick. Compact routing schemes. In *Proc. 13th ACM Symp. on Parallel Algorithms and Architecture*, pages 1–10, Hersonissos, Crete, Greece, July 2001.