

# Labeling Schemes for Weighted Dynamic Trees

Amos Korman <sup>\*</sup>      David Peleg <sup>†</sup>

## Abstract

A *Distance labeling scheme* is a type of localized network representation in which short *labels* are assigned to the vertices, allowing one to infer the distance between any two vertices *directly* from their labels, without using *any* additional information sources.

As most applications for network representations 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 dynamic weighted trees where the vertices of the trees are fixed but the (positive integral) weights of the edges may change. The two models considered are the *edge-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 approximate distance labeling schemes for the two dynamic models, which are efficient in terms of the required label size and communication complexity involved in updating the labels following the weight changes.

---

<sup>\*</sup>Information Systems Group, Faculty of IE&M, The Technion, Haifa, 32000 Israel. E-mail: [pandit@tx.technion.ac.il](mailto:pandit@tx.technion.ac.il). Supported in part at the Technion by an Aly Kaufman fellowship.

<sup>†</sup>Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot, 76100 Israel. E-mail: [david.peleg@weizmann.ac.il](mailto:david.peleg@weizmann.ac.il). Tel: (+972)-8934-3478. Fax: (+972)-8934-4122. Supported in part by a grant from the Israel Science Foundation.

# 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, 17], distance [27, 22, 16, 15, 13, 19, 29, 9, 24, 18], tree routing [12, 30], flow and connectivity [21], tree ancestry [1, 3, 4, 20], and various other tree functions, such as center, least common ancestor, separation level, and Steiner weight of a given subset of vertices [28]. See [14] 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 highly 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 [25], 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" tree model, has amortized message complexity  $O(\log^2 n)$  per topological event, where  $n$  is the size of the tree when the event takes place. The protocol maintains  $O(\log^2 n)$  bit labels, where  $n$  is the current tree size. This label size is known to be optimal even in the static scenario. A second result of [25] introduces a more general labeling scheme for dynamic trees, 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, routing, separation level as well as the nearest common ancestor relation. In the case of dynamically growing trees, where the only topological event allowed is that a leaf joins the tree, the resulting dynamic scheme incurs an overhead (over the corresponding static scheme) of  $O(\log n)$  multiplicative factor in both the label size and amortized message complexity (the message complexity of a static scheme is the number of messages used to assign the static labels). If an upper bound  $n_f$  on the number of vertices is known in advance, this method can yield a different tradeoff, with  $O(\log n \log n_f / \log \log n)$  multiplicative overhead on the label size but only an

	Growing trees	Growing trees known final $n_f$	Leaf dynamic trees
[25] - Label size	$O(L(\pi) \log n)$	$O(L(\pi) \frac{\log n \log n_f}{\log \log n_f})$	$O(L(\pi) \log n)$
[25] - Amortized message complexity	$O(M(\pi) \log n)$	$O(M(\pi) \frac{\log n}{\log \log n_f})$	$O(M(\pi) \log n$ $+ \log^2 n)$
[23] - Label size	$O(L(\pi) \log_{k(n)} n)$		$O(L(\pi) \log_{k(n)} n)$
[23] - Amortized message complexity	$O(M(\pi) k(n) \log_{k(n)} n)$		$O(M(\pi) k(n) \log_{k(n)} n$ $+ \log^2 n)$

Figure 1: The label sizes and amortized message complexities of previous dynamic labeling schemes on trees extending some static scheme  $\pi$ , where  $L(\pi)$  (respectively  $M(\pi)$ ) is the label size (resp., amortized message complexity) of the given static labeling scheme  $\pi$ , and  $k(n)$  is some function.

$O(\log n / \log \log n_f)$  overhead on the amortized message complexity.

A different method for extending static labeling schemes on trees to the dynamic setting was introduced in [23]. Their method applies for the same class of tree functions as the method of [25]. In the case of dynamically growing trees, [23] showed how to construct for every function  $k(x)$ , a dynamic labeling scheme extending a given static scheme, such that the resulting scheme incurs overheads (over the static scheme) of  $O(\log_{k(n)} n)$  in the label size and  $O(k(n) \log_{k(n)} n)$  in the message complexity.

In the non-restricted leaf-dynamic tree model (where both additions and deletions are allowed) the resulting dynamic schemes of both [23] and [25] incur also an increased *additive* overhead in amortized communication, of  $O(\log^2 n)$  messages per operation. Table 1 summarizes the complexities of the methods in [23] and [25], regarding translating static labeling schemes on trees to a dynamic setting.

One key limitation of the setting studied in [25] and [23] is that the links are assumed to be unweighted. In reality, network distances are often based on link weights. Therefore, 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 *edge-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. The second model considered is the *increasing-dynamic model* which is the edge-dynamic model restricted to events where an edge weight can only increase by one at each step.

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  $\Delta$  individual weight changes of 1.)

As can easily be shown, any *exact* distance labeling scheme for either of the models cannot avoid sending linear number of messages per operation in some worst-case scenarios. We therefore weaken the demands from the labeling scheme, and only require it to maintain a  $\beta$ -approximation of the distances (for  $\beta > 1$ ) rather than exact distances. Such a scheme is referred to as a  *$\beta$ -approximate distance labeling scheme*. Let us note that it can be shown that if one also allows edges to be given zero weight, then any approximated distance labeling scheme cannot avoid sending linear number of messages per operation in some worst-case scenarios. To see this, consider an  $n$ -node path  $P$  and assume that all edge weights are initially zero. Now, repeatedly raise and decrease by 1 the weight of the middle edge in  $P$ . Clearly, after every such weight change,  $\Omega(n)$  vertices must receive a message.

The communication complexity of a  $\beta$ -approximate dynamic distance labeling scheme is measured by the following measures. The *message* (respectively, *bit*) *complexity*, is the number of messages (resp., bits) used by the update protocol, in order to update the vertices following the weight changes. In terms of the amount of information stored at each vertex, we distinguish between the *label*  $L(v)$  used by each node  $v$  to deduce the required information in response to online queries, and the total storage  $Memory(v)$  kept at each node  $v$ , used during the (offline) updates to the labels and the maintenance operations. For certain applications, it is reasonable to assume that the label  $L(v)$  serves as an internal database, whereas the additional storage in  $Memory(v)$  is kept on some external device. Subsequently, the size of  $L(v)$  seems to be a more critical consideration than the total amount of storage needed for information maintenance. The *label size* is the maximum number of bits used at a label  $L(v)$  and the *memory size* is the maximum number of bits used at  $Memory(v)$ .

Throughout the paper, denote by  $n$  the number of vertices in the graph,  $G = (V, E)$ . To describe the bounds for the message complexity, we need the following definition. For a

vertex  $v$  in a graph  $G = \langle V, E \rangle$ , let  $B(v, d)$  be the number of vertices at distance  $d$  or less from  $v$ . Let the *local density* of  $G$  be defined as  $\Lambda(G) = \max\{B(v, d)/2d \mid d \geq 1, v \in V\}$ . This graph theoretic parameter is well studied, especially with relation to the bandwidth  $B(G)$  of the graph  $G$  (see [10, 11, 7, 8, 26]). Specifically, it is easy to show that  $\Lambda(G) \leq B(G)$ . Previous research suggest the conjecture that  $B(G) = O(\Lambda(G) \log n)$ . It is known, however, that  $B(G) = O(\Lambda(G) \cdot \text{Poly log } n)$  ([10]).

**Our contribution:** For a tree network, we present a  $\beta$ -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. In both schemes, the maximum label is bounded by  $O(\log^2 n + \log n \log W)$ . Using the methods of [15], the label size can be reduced further. For example, for  $\beta > 1$  bounded away from one, the label size of our schemes is  $O(\log n \log \log Wn)$ .

We show that for  $\beta > 1$  bounded away from 1, on scenarios with  $m$  weight changes, the total message and bit complexities of the protocol for the edge-dynamic model are  $O(m\Lambda(T) \log^3 n)$  and  $O(m\Lambda(T) \log^3 n \cdot \log \log n)$  respectively, where  $\Lambda(T)$  is the local density of the underlying tree  $T$ . We also show that for any integer value  $1 \leq \nu \leq n$ , there exists a tree  $T$  whose local density satisfies  $\Lambda(T) = \Theta(\nu)$  and that any  $\beta$ -approximate distance labeling scheme in the edge-dynamic model must send  $\Omega(m\Lambda(T))$  messages, in some scenario with  $m$  weight changes. Moreover, we define a variant of the local density  $\tilde{\Lambda}(T)$ , and show that for any underlying tree  $T$ , any  $\beta$ -approximate distance labeling scheme in the edge-dynamic model must send  $\Omega(m\tilde{\Lambda}(T))$  messages, in some scenario with  $m$  weight changes.

In the increasing-dynamic model, for trees with large local density, the communication complexity can be further reduced. Specifically, we show that for this model, on scenarios with  $m$  weight changes, the total message and bit complexities are  $O(m \log^3 n + n \log^2 n \log m)$  and  $O(m \log^3 n \cdot \log \log n + n \log^2 n \log m \cdot \log \log n)$  respectively.

For the edge-dynamic model, if the underlying network topology is a path, we describe a different  $\beta$ -approximate distance labeling scheme yielding a different tradeoff between the size of the labels and the communication complexity. For example, for constant  $\beta > 1$ , the scheme uses memory size  $O(\log n \log W + \log \log n)$  and its total message and bit complexities are only  $O(m \log^2 n)$  and  $O(m \log^2 n \log \log n)$  respectively.

The paper is organized as follows. At a very high level, our approximate distance labels for trees use the separator decomposition technique of [27]. This essentially reduces the problem to dynamic root-distances for  $O(\log n)$  roots. In Section 2 we describe the preliminaries. In Section 3 we describe our root-estimates protocols for the edge-dynamic and for the increasing-dynamic models. These protocols are modifications of the main protocol in [25],

which itself is a modification of the main protocol in [2]. In Section 4 we describe how to use these protocols to obtain dynamic approximate distance labeling schemes for the two dynamic models. In Section 5 we describe our dynamic approximate distance labeling scheme for paths.

## 2 Preliminaries

Our network model is restricted to tree 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 be changed dynamically by weight changes of its edges.

Let  $T$  be an  $n$ -node tree. For two vertices  $u$  and  $v$  in  $T$ , denote by  $d^w(u, v)$  the weighted distance between  $u$  and  $v$ . For every vertex  $v$ , let  $H(v)$  denote the height of  $v$ , namely, its unweighted (hop) distance from the root. For every vertex  $v$ , let  $P_v$  denote the path from the root to  $v$ . For any given path  $P$ , let  $|P|$  denote the number of edges in  $P$ , and let  $|P|^w$  denote the weighted length of  $P$ .

### 2.1 The edge-dynamic and increasing-dynamic models

In the *edge-dynamic* 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.

### 2.2 Labeling schemes and approximation labeling schemes

For  $\beta \geq 1$ , a *static  $\beta$ -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  $\tilde{d}^\omega(u, v)/\beta \leq d^\omega(u, v) \leq \beta \cdot \tilde{d}^\omega(u, v)$ .

An *exact static distance labeling scheme* is a static 1-approximate distance labeling scheme. For examples of static distance labeling schemes see [16, 27, 19, 9]. In particular, see [16, 27, 25, 18] for examples of static distance labeling schemes on trees. Examples for static  $\beta$ -approximate distance labeling schemes can be seen in [15].

In this paper we are interested in distributed dynamic 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 dynamic marker protocol*.

The approximate dynamic labeling schemes involve a marker protocol  $\mathcal{M}$  which initially initializes the network in a preprocessing stage and is then 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.

It is easier to analyze our protocols assuming that the topological changes occur sequentially and are sufficiently spaced so that the update protocol has enough time to complete its operation in response to a given topological change before the occurrence of the next change. However, our schemes can operate also under weaker assumptions. Specifically, after the initial setup, it is allowed for topological changes to occur in rapid succession or even concurrently. Also, we assume that the communication caused by our update protocol is asynchronous. However, correctness is only required after the system becomes quiet for sufficiently long time periods. Specifically, we say that a time  $t$  is *quiet* if all updates (of the relevant update protocol) concerning the previous topological changes, have occurred by time  $t$ . We demand correctness only for quiet times. The above demand is reasonable, as it can easily be shown that for non quiet times, we cannot expect any dynamic  $\beta$ -approximate distance labeling scheme to be correct, for any  $\beta > 1$ .

Let  $\pi = \langle \mathcal{M}(\beta), \mathcal{D} \rangle$  be a dynamic  $\beta$ -approximate labeling scheme (in either the edge-dynamic or the increasing dynamic model) for the family of  $n$ -node underlying trees. Let  $T$  be an  $n$ -node (underlying) tree. We refer to a scenario where  $m$  weight changes have occurred in  $T$  as an  $m$ -change scenario. We are interested in the following complexity measures.

- *Message Complexity*,  $\mathcal{MC}(\mathcal{M}(\beta), T, m)$ : the total number of messages sent by  $\mathcal{M}(\beta)$ , in the worst case  $m$ -change scenario on  $T$ .

- *Bit Complexity*,  $\mathcal{BC}(\mathcal{M}(\beta), T, m)$ : the total number of bits sent by  $\mathcal{M}(\beta)$ , in the worst case  $m$ -change scenario on  $T$ .
- *Label Size*,  $\mathcal{LS}(\mathcal{M}(\beta), T, m)$ : the maximum size of a label assigned by  $\mathcal{M}(\beta)$  to a vertex in the worst case  $m$ -change scenario on  $T$ .
- *Memory Size*,  $\mathcal{MS}(\mathcal{M}(\beta), T, m)$ : the maximum number of bits in  $Memory(v)$  over all vertices  $v \in T$  and all  $m$ -change scenarios.

### 3 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 dynamic tree at any given quiet time. In the next section we show how to use this mechanism to obtain the desired approximate distance labeling schemes.

Let  $T$  be a rooted tree. We introduce two *root-distance* protocols in which each node  $v \in T$  keeps a  $\beta$ -approximate of  $d^\omega(v)$ ,  $v$ 's weighted distance to the root. The first root-distance protocol,  $\mathcal{R}_{dyn}(\beta)$ , is applied to the edge-dynamic model and the second root-distance protocol,  $\mathcal{R}_{inc}(\beta)$ , is applied to the increasing-dynamic model. The complexity bounds of these protocols are expressed in terms of the following quantities. For a vertex  $v$ , let  $B(v, d)$  be the number of vertices at distance  $d$  or less from a  $v$ . Let

$$\begin{aligned} \Lambda(T) &= \max \left\{ \frac{B(v, d)}{2d} \mid d \geq 1, v \in V \right\}, \\ \alpha &= \frac{1}{\beta - 1}, \\ \gamma &= \frac{1}{\sqrt{\beta - 1}}. \end{aligned}$$

The message complexity of  $\mathcal{R}_{dyn}(\beta)$  is  $O(m\alpha\Lambda(T)\log^2 n)$  and the bit complexity of  $\mathcal{R}_{dyn}(\beta)$  is  $O(m\alpha\Lambda(T)\log^2 n \log \log n)$ .

For trees with large local density, our protocol  $\mathcal{R}_{inc}(\beta)$  for the increasing-dynamic model, has better communication complexity. Specifically, the message and bit complexities of  $\mathcal{R}_{inc}(\beta)$  are  $O(m\gamma \log^2 n + n \log_\beta m \log n)$  and  $O(m\gamma \log^2 n \log \log n + n \log_\beta m \log n \log \log n)$ , respectively.

### 3.1 The root-distance protocol $\mathcal{R}_{dyn}(\beta)$ for the edge-dynamic model

Protocol  $\mathcal{R}_{dyn}(\beta)$  is a modification of Protocol WEIGHTWATCH of [25] which is a modification of the main algorithm in [2]. In Protocol WEIGHTWATCH, each node is required to know an estimated to the number of its descendants in a dynamically growing tree, i.e., a dynamic tree allowing only leaf insertions. Let us first give a high level description of Protocol WEIGHTWATCH. Each node  $v$  maintains two bins, a “local” bin  $b_l$  and a “global” bin  $b_g$ , storing a varying number of tokens throughout the execution, where each token represents a leaf joining the tree. Whenever a vertex is added as a child of an existing vertex  $u$ , the node  $u$  adds a token to its local bin indicating this event. When a bin gets filled with tokens, the tokens are disseminated up the tree only to a limited distance and are then stored in an intermediate global bin of larger size. Thus, the information (regarding new vertices joining the tree) propagates up the tree in a gradual manner, yielding message complexity  $O(m \log^2 n)$ . An intermediate node uses the messages passing through it in order to estimate the number of its descendants. The analysis of the approximation is based on bounding the possible overall error made in the way a vertex  $v$  views the number of its descendants. This error corresponds to the number of delayed tokens ‘stuck’ in the various bins of the subtree hanging at  $v$ .

There are two main differences between Protocol WEIGHTWATCH and Protocol  $\mathcal{R}_{dyn}(\beta)$ . The first is that Protocol  $\mathcal{R}_{dyn}(\beta)$  operates in the edge-dynamic model (and not in a growing tree). In the edge-dynamic model, since edge weights can both increase and decrease, Protocol  $\mathcal{R}_{dyn}(\beta)$  uses both ‘positive’ and ‘negative’ tokens, where a ‘positive’ (respectively, ‘negative’) token corresponds to a weight increase (resp., decrease). The second (and more significant) difference is that in Protocol  $\mathcal{R}_{dyn}(\beta)$ , the information (regarding new topological events) propagates down the tree (in contrast to Protocol WEIGHTWATCH where the information propagates up the tree). In particular, when a bin gets filled with tokens, these tokens are sent down the tree and stored in various intermediate global bins. These two differences require only slight modifications to the description of Protocol WEIGHTWATCH. However, different analysis is required, especially for bounding the message complexity (see Lemma 3.4). We therefore give a full description of Protocol  $\mathcal{R}_{dyn}(\beta)$  and its analysis.

#### General structure

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. Intuitively, whenever  $u$  learns that the weight of the edge leading to its parent has changed by 1, a token is created implicitly indicating

this information. This token is stored at  $v$ 's local bin  $b_l(v)$ .

Let  $b$  be some bin at a node  $v$ . In the following discussion, unless it might cause confusion, we do not distinguish between a bin and the node holding that bin. For example, the height of the bin  $b$ , denoted  $H(b)$ , is the height of  $v$ , i.e.,  $H(b) = H(v)$ . Also, the distance between two bins is defined as the distance between the nodes holding them.

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  $Level(b_l(v)) = -1$ .

A bin of level  $l$  is referred to as an  $l$ -level bin. 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  $v$ , and from each path from  $v$  to a leaf, the closest bin  $b'$  such that  $Level(b') = Level(b) + 1$  is 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 any bin  $b$  may have several supervisors. (This should be put in contrast to Protocol WEIGHTWATCH, where each bin has at most one supervisor.) The supervisor relation defines a bin hierarchy. The following observation is similar to the three properties of Protocol WEIGHTWATCH, discussed in [25].

**Observation 3.1**    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  $3 \cdot 2^l$  nodes.*

3. *The number of level  $l$  bins that lie in nodes on  $P_v$  is at most  $\frac{|P_v|}{2^l}$ .*

**Proof:** The proof of the first item is straightforward. Note that the supervisors of a local bin are either the global bin at the same node or the global bins at its children. The reason is that the supervisors of a local bin  $b_l(v)$  are the global bins that belong to descendants of  $v$  (possibly  $v$  itself) of lowest height, such that their least significant bit (LSB) is 1. Therefore, the second item follows in the case where  $b(v)$  is a local bin. Assume now that  $b$  is an  $l$ -level global bin. Note that the binary representation of  $H(b)$  is  $(*, *, \dots, *, 1, 0, 0, \dots, 0)$ , where the  $l$  LSBs are all 0 and the  $l + 1$ 'st LSB is 1. Let  $b'$  be a supervisor of  $b$ . Since the level of  $b'$  is  $l + 1$ , the binary representation of  $H(b')$  is  $(*, *, \dots, *, 1, 0, 0, 0, \dots, 0)$ , where the  $l + 1$ 'st LSBs are all 0 and the  $l + 2$ 'nd LSB is 1. It follows that if the  $l + 2$ 'nd LSB in the binary representation of  $H(b)$  is 0, then the distance between  $b$  and  $b'$ , is  $2^l$ . Otherwise, if the  $l + 2$ 'nd LSB in the binary representation of  $H(b)$  is 1, then the distance between  $b$  and  $b'$  is  $3 \cdot 2^l$ . The second item follows. The third item follows from the fact that the distance between any two level  $l$  bins that belong to vertices in  $P_v$  is  $2^{l+1}$ , and from the fact that the lowest (closest to the root)  $l$ -level bin in  $P_v$ , is at height  $2^l$ . ■

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  $-Cap(l) \leq \tau(b) \leq Cap(l)$  tokens, where  $Cap(l) = \max\{\sigma \cdot 2^l, 1\}$ . Intuitively, a level  $l$  bin can be thought of as storing at most  $Cap(l)$  “positive” tokens and at most  $Cap(l)$  “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.

### 3.1.1 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 an  $l$ -level bin  $b$  at some node  $v$  gets filled with tokens (i.e., it either has  $Cap(l)$  positive tokens or  $Cap(l)$  negative tokens),  $v$  immediately empties the bin and broadcasts a signal to all its supervisor bins to add  $Cap(l)$  (positive or negative) tokens to their bins. The signal is given by sending  $l$  or  $-l$ .

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 weighted 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)  $Cap(l)$  to  $\tilde{d}(v)$ . The counter at  $v$  used to estimate  $d^\omega(v)$  is  $d_{approx}(v) = \max\{p, \tilde{d}(v)\}$ , where  $p = |P_v|$  (i.e,  $p$  is  $v$ 's height).

## 3.2 Analysis of Protocol $\mathcal{R}_{dyn}(\beta)$

### 3.2.1 Approximation

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,

$$\begin{aligned}\phi(\mathcal{P}_v) &= \sum_{b \text{ on } \mathcal{P}_v} \tau(b) , \\ \mu(\mathcal{P}_v) &= \sum_{b \text{ on } \mathcal{P}_v} |\tau(b)| .\end{aligned}$$

**Lemma 3.2** *At any given quiet time,  $\phi(\mathcal{P}_v) = d^\omega(v) - \tilde{d}(v)$ .*

**Proof:** Initially the lemma is trivially correct. Note that for any scenario of weight changes, the final value of each of the parameters in the above equation, after all updates of  $\mathcal{R}_{dyn}(\beta)$  are performed, is not affected by the order and pace in which the messages of  $\mathcal{R}_{dyn}(\beta)$  are sent. We may therefore assume that the weight changes are sufficiently spaced so that Protocol  $\mathcal{R}_{dyn}(\beta)$  has enough time to complete its operation in response to a given topological change, before the occurrence of the next change.

We use induction on the following events.

1. If an edge on  $\mathcal{P}_v$  increases (respectively, decreases) its weight by one and as a result a local bin in a node on  $\mathcal{P}_v$  gets full with 1 (resp., -1) token then both  $\phi(\mathcal{P}_v)$  and  $d^\omega(v)$  increase by 1 (resp., -1). Therefore the equation remains valid.
2. If a message of level  $l$  with positive sign (resp., negative) is sent from bin  $b$  to bin  $b'$  then consider the following.
  - If  $b$  and  $b'$  are both at nodes on  $\mathcal{P}_v$  or  $b$  and  $b'$  are both at nodes not on  $\mathcal{P}_v$  then non of the parameters of the equation change.
  - If  $b$  is at a node on  $\mathcal{P}_v$  and  $b'$  is at a node not on  $\mathcal{P}_v$  then  $v$  is on the path from  $b$  to  $b'$  and the message must pass through  $v$ . Therefore  $\phi(\mathcal{P}_v)$  decreases (resp., increases) by  $Cap(l)$  and  $\tilde{d}(v)$  increases (resp., decreases) by  $Cap(l)$  and the equation remains valid. ■

**Lemma 3.3** *For each node  $v$ , at any quiet time,  $d_{approx}(v)/\beta \leq d^\omega(v) \leq \beta \cdot d_{approx}(v)$*

**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 signal to its supervisor bins). Therefore, the only bins that might not be empty are global bins that are supervisors with capacity larger than 1.

Fix  $v$  and denote the length of the path  $\mathcal{P}_v$  by  $p$  (i.e.,  $p$  is  $v$ 's height). By the third item in Observation 3.1, for each level  $l$ , there are at most  $\frac{p}{2^l}$  bins of level  $l$  on  $\mathcal{P}_v$ . Fix a level  $l$  such that  $Cap(l) > 1$ , i.e.,  $Cap(l) = \sigma \cdot 2^l$ . Even if all of the level  $l$  bins in  $\mathcal{P}_v$  are full, we get that the total number of wasted tokens in these bins is at most  $\frac{p}{2^l} \cdot \frac{2^l}{\alpha(\log n + 1)} \leq \frac{p}{\alpha(\log n + 1)}$ . Therefore, the number of wasted tokens on  $\mathcal{P}_v$  (on all levels) satisfies  $\mu(\mathcal{P}_v) \leq \frac{p}{\alpha}$ . By the previous lemma, at any quiet time,  $|d^\omega(v) - \tilde{d}(v)| = |\phi(\mathcal{P}_v)|$ , and therefore  $|d^\omega(v) - \tilde{d}(v)| = |\phi(\mathcal{P}_v)| \leq \mu(\mathcal{P}_v) \leq \frac{p}{\alpha}$ . In particular,  $d^\omega(v) \leq \tilde{d}(v) + \frac{p}{\alpha}$ . Since  $d_{approx}(v) = \max\{p, \tilde{d}(v)\}$ , we have  $d^\omega(v) \leq d_{approx}(v)(1 + 1/\alpha) = \beta \cdot d_{approx}(v)$ . On the other hand,  $\tilde{d}(v) - d^\omega(v) \leq \frac{p}{\alpha}$ , and since  $d^\omega(v) \geq p$ , we have  $\tilde{d}(v) \leq d^\omega(v)(1 + 1/\alpha) = \beta \cdot d^\omega(v)$ . Since  $p \leq d^\omega(v)$ , we have  $d_{approx}(v) = \max\{p, \tilde{d}(v)\} \leq \beta \cdot d^\omega(v)$ . The lemma follows.  $\blacksquare$

### 3.2.2 Communication complexities

**Lemma 3.4** 1.  $\mathcal{MC}(\mathcal{R}_{dyn}(\beta), T, m) = O(m\alpha\Lambda(T) \log^2 n)$

2.  $\mathcal{BC}(\mathcal{R}_{dyn}(\beta), T, m) = O(m\alpha\Lambda(T) \log^2 n \log \log n)$

**Proof:** We say that a bin  $b'$  *affects* a bin  $b$  if there exists a list of bins  $b_1, b_2, \dots, b_k$  s.t.  $b' = b_1$ ,  $b = b_k$ , and for every  $1 \leq i < k$ ,  $b_{i+1}$  is a supervisor of  $b_i$ . The following claim follows from the definition of the supervisor relation.

**Claim 1:** If a bin  $b'(v')$  affects an  $l$ -level bin  $b(v)$ , then the  $l$ -level bins affected by  $b'(v')$  are precisely the global bins that belong to the descendants of  $v'$  at height  $H(v)$ .

Define the *local fillers* of  $b$  to be the local bins that affect  $b$ . Clearly, the local fillers of a bin  $b(v)$  belong to vertices on  $\mathcal{P}_v$ , the path from the root to  $v$ . Define the *lowest local filler* of  $b$ , denoted  $LLF(b)$ , to be the local filler of  $b$  at the lowest height. Note that  $LLF(b)$  is also the bin of lowest height affecting  $b$ . For fixed  $l$ , we now use the  $LLF$  function to define a partition of the  $l$ -level bins into equivalence classes  $\chi_1, \chi_2, \dots$  as follows. Two  $l$ -level bins  $b'$  and  $b''$  belong to the same equivalence class if and only if they share the same lowest local filler, i.e.,  $LLF(b') = LLF(b'')$ .

**Claim 2:** If a bin  $b'$  affects an  $l$ -level bin  $b_i \in \chi_i$ , then all the  $l$ -level bins that  $b'$  affects belong to  $\chi_i$ .

**Proof of Claim 2:** Assume by way of contradiction that there exist a bin  $b'$  which affects two  $l$ -level bins  $b_i = b_i(v_i)$  and  $b_j = b_j(v_j)$  where  $b_i \in \chi_i$ ,  $b_j \in \chi_j$ , and  $i \neq j$ . We first show that  $LLF(b_i)$  affects  $b_j$ . It follows from Claim 1 that  $H(v_i) = H(v_j)$ . It then follows from the definition of the  $LLF$  function that  $LLF(b_i)$  belongs to an ancestor of  $v'$ , the node holding  $b'$ . Since  $v_j$  is a descendant of  $v'$ ,  $v_j$  is also a descendant of the node holding  $LLF(b_i)$ .

Therefore, by Claim 1,  $LLF(b_i)$  affects  $b_j$ . Similarly,  $LLF(b_j)$  affects  $b_i$ . It follows that  $LLF(b_i) = LLF(b_j)$  and therefore  $i = j$ . Contradiction. ■

Let  $\Upsilon$  be the collection of local bins that affect some  $l$ -level bin. For  $b \in \Upsilon$ , let  $Index(b)$  be the index  $i$  such that the  $l$ -level bins that  $b$  affect belong to  $\chi_i$ . By Claim 2, the  $Index$  function is well defined. We now use the  $Index$  function to define a partition of the local bins in  $\Upsilon$  into equivalence classes  $\Gamma_1, \Gamma_2, \dots$  as follows. Two local bins  $b'$  and  $b''$  in  $\Upsilon$  belong to the same equivalence class  $\Gamma_i$  iff  $Index(b') = Index(b'')$ .

Fix some  $j$  and consider the  $l$ -level bins in  $\chi_j$ . Let  $i(j)$  be the index such that the local bins that affect the  $l$ -level bins in  $\chi_j$  are the bins in  $\Gamma_{i(j)}$ . Note that  $i(\cdot)$  is a one-to-one function. Let  $b$  be the lowest local filler that is common to all bins in  $\chi_j$ . Note that by Claim 1, since all the  $l$ -level bins in  $\chi_j$  are affected by  $b$ , they all belong to nodes at the same height  $h_j$ . The next claim follows from the second item in Observation 3.1.

**Claim 3:** All the bins in  $\chi_j$  are at distance  $O(2^l)$  below  $b$ .

Let us now bound the number of messages sent by the ( $l$ -level) bins in  $\chi_j$  (due to Step 4 in Protocol  $\mathcal{R}_{dyn}(\beta)$ ). For a bin  $b' \in \chi_j$ , let  $E(b')$  be the set of edges on the paths connecting  $b'$  with its  $l + 1$ 'st-level supervisors. Note that if  $b'$  and  $b''$  are two bins in  $\chi_j$ , then  $E(b') \cap E(b'') = \emptyset$ ; the reason is that all the bins in  $\chi_j$  belong to nodes that have the same height  $h_j$ . Moreover, by the second item in Observation 3.1, we have  $\bigcup_{b' \in \chi_j} E(b') \subseteq B(b, O(2^l))$ . See Figure 2.

Note that whenever a bin  $b' \in \chi_j$  gets filled, it sends a signal to all its supervisors, and therefore, the number of messages it sends is precisely  $|E(b')|$ . It follows that if all the bins in  $\chi_j$  get filled precisely once, the number of messages incurred by them is at most  $\bigcup_{b' \in \chi_j} E(b') \subseteq B(b, O(2^l))$ .

Note that the only local fillers that affect the bins in  $\chi_j$  belong to  $\Gamma_{i(j)}$ . Let  $m_j$  be the number of topological events occurring in the vertices holding the (local) bins in  $\Gamma_{i(j)}$ . We therefore obtain that the number of times a bin in  $\chi_j$  gets filled is at most  $\frac{m_j}{Cap(l)}$ . Hence, even if each bin in  $\chi_j$  gets filled exactly  $\frac{m_j}{Cap(l)}$  times, the total number of messages incurred by them is bounded by  $B(b, O(2^l)) \cdot \frac{m_j}{Cap(l)} \leq B(b, O(2^l)) \cdot \frac{\alpha(\log n + 1)}{2^{l-1}} \cdot m_j = O(\alpha\Lambda(T)m_j \log n)$ . (The first inequality follows since  $Cap(l) \geq \sigma \cdot 2^l$  and therefore  $\frac{1}{Cap(l)} \leq \frac{\alpha(\log n + 1)}{2^{l-1}}$ ). It follows that the message complexity caused by all level  $l$  bins during the weight changes is bounded by  $O(\sum_j \alpha\Lambda(T)m_j \log n) = O(\alpha\Lambda(T)m \log n)$  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 use  $O(\log \log n)$  bits. ■

**Corollary 3.5** For  $\beta > 1$  bounded away from 1,  $\mathcal{MC}(\mathcal{R}_{dyn}(\beta), T, m) = O(m\Lambda(T) \log^2 n)$

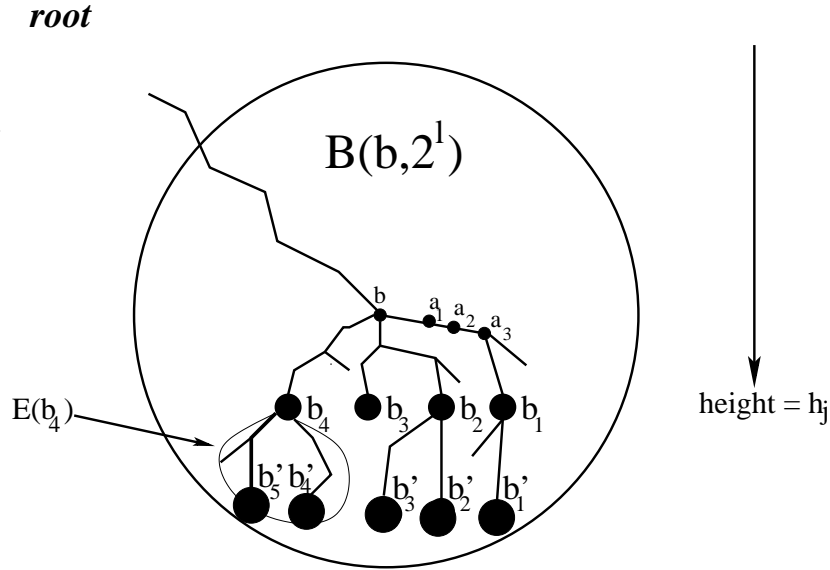


Figure 2: The bins  $b_1, \dots, b_4$  are the  $l$ -level bins in  $\chi_j$  and the bins  $b'_1, \dots, b'_5$  are their supervisors. The bins  $b, a_1, a_2, a_3$  are some of the local bins in  $\Gamma_{i(j)}$ . The local  $b$  is the *LLF* of all the bins in  $\chi_j$ .

and  $\mathcal{BC}(\mathcal{R}_{dyn}(\beta), T, m) = O(m\Lambda(T) \log^2 n \cdot \log \log n)$ .

**Corollary 3.6** *Restricted to simple path topologies,  $\mathcal{MC}(\mathcal{R}_{dyn}(\beta), T, m) = O(m\alpha \log^2 n)$  and  $\mathcal{BC}(\mathcal{R}_{dyn}(\beta), T, m) = O(m\alpha \log^2 n \cdot \log \log n)$ .*

### 3.3 The root-distance protocol, $\mathcal{R}_{inc}(\beta)$ , for the increasing-dynamic model

As Protocol  $\mathcal{R}_{dyn}(\beta)$ , Protocol  $\mathcal{R}_{inc}(\beta)$  is designed so that each node  $v$  keeps a  $\beta$ -approximation to  $d^\omega(v)$ . However, Protocol  $\mathcal{R}_{inc}(\beta)$  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  $\mathcal{MC}(\mathcal{R}_{inc}(\beta), T, m) = O(m\gamma \log^2 n + n \log n \log_\beta m)$ .

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 notions of light and heavy children are hardly new (e.g., see [25, 20, 28]). 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 tree into a collection  $\mathcal{S}$  of edge-disjoint paths in the following stages.

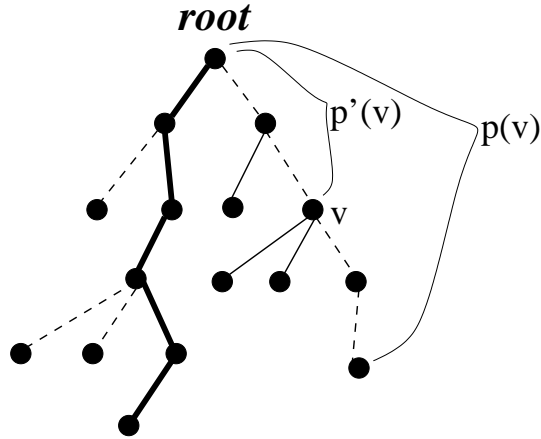


Figure 3: The thick path, the dashed paths and the regular paths are the paths taken to  $\mathcal{S}$  at the first, second and third stages of the decomposition, respectively.

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  $v$ ). See Figure 3 for an illustration of the decomposition into paths.

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  $P_1 \dots P_k$  (where each  $P_i$  is prefix of a path in  $\mathcal{S}$  and  $P_k = P'(v)$ ). Moreover, all edges in  $\mathcal{P}_v$  are marked except maybe the first edges in  $P_1, \dots, P_k$ . Denote the number of unmarked edges along  $\mathcal{P}_v$  by  $\eta(v)$ .

The following claim is used to show that the protocol  $\mathcal{R}_{inc}(\beta)$  has only one-side error.

**Claim 3.7** *When applying  $\mathcal{R}_{dyn}(\beta)$  in the increasing model, at any quiet time, we get  $d_{approx}(v) \leq d^\omega(v)$  for each node  $v$ .*

**Proof:** The claim follows from Lemma 3.2 and from the fact that in the increasing model  $\phi(\mathcal{P}_v) \geq 0$  at any given time. ■

**Protocol  $\mathcal{R}_{inc}(\beta)$**

Each node  $v$  simultaneously participates in 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. This signal is given by sending  $\mu(v)$ , 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  $P_1, \dots, P_k$  as before. (Recall that  $P_k$  is  $P'(v)$ ). The node  $v$  keeps counters  $d_1, \dots, d_k$  for approximating  $|P_1|^\omega, \dots, |P_k|^\omega$  respectively, as follows. For each  $1 \leq i \leq k-1$ , denote by  $u_i$  the bottom node of  $P_i$ , and denote by  $|P_i|_0^\omega$  the initial weighted length of  $P_i$ .

1. Initially  $d_i = |P_i|_0^\omega$  for each  $i$ .
2. The counter  $d_k = \tilde{d}_v$  is maintained by  $\mathcal{R}_1$ .
3. Each time  $v$  gets an  $\mathcal{R}_2$  signal from  $u_i$ , it raises  $d_i$  by a multiplicative factor of  $\sqrt{\beta}$ .
4. At all times, let  $d_{approx}(v) = \sum d_i$ .

**Lemma 3.8** *At all quiet times,  $d_{approx}(v) \leq d^\omega(v) \leq \beta \cdot d_{approx}(v)$  and therefore  $\mathcal{R}_{inc}$  guarantees that each node  $v$  maintains a  $\beta$ -approximation to  $d^\omega(v)$ .*

**Proof:** It suffices to show that at any quiet time,  $d_i \leq |P_i|^\omega \leq \beta \cdot d_i$  for each  $i$ . Initially, the condition is satisfied since initially  $d_i = |P_i|_0^\omega$ . Fix a quiet time  $t$  and fix  $1 \leq i \leq k-1$ . At time  $t$  we have,  $\tilde{d}_{u_i} \leq |P_i|^\omega \leq \sqrt{\beta} \cdot \tilde{d}_{u_i}$ , since by Lemma 3.3,  $|P_i|^\omega \leq \sqrt{\beta} \cdot \tilde{d}_{u_i}$ , and by Claim 3.7,  $\tilde{d}_{u_i} \leq |P_i|^\omega$ . If, at some time before  $t$ , this approximation increases by a multiplicative factor of  $\sqrt{\beta}$ , then by time  $t$ , the node  $v$  will know about it, since  $v$  belongs to  $u_i$ 's light subtrees. Therefore, if by time  $t$ ,  $v$  gets  $q$  signals from  $u_i$  (sent by Protocol  $\mathcal{R}_2$ ), then

$$d_i = |P_i|_0^\omega \cdot \beta^{q/2} \leq \tilde{d}_{u_i} \leq |P_i|^\omega \leq \sqrt{\beta} \cdot \tilde{d}_{u_i} \leq \sqrt{\beta} \cdot |P_i|_0^\omega \cdot \beta^{(q+1)/2} = \beta \cdot d_i.$$

Together with the fact that at any quiet time,  $d_k$  is a  $\sqrt{\beta}$ -approximate to  $|P_k|^\omega$  as guaranteed by  $\mathcal{R}_1$ , the lemma follows.  $\blacksquare$

**Lemma 3.9** 1.  $\mathcal{MC}(\mathcal{R}_{inc}(\beta), T, m) = O(m\gamma \log^2 n + n \log n \log_\beta m)$

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

**Proof:** Note that Protocol  $\mathcal{R}_1$  invokes Protocol  $\mathcal{R}_{dyn}$  with parameter  $\sqrt{\beta}$  on disjoint paths. Therefore, by Lemma 3.4 applied to each one of the paths, we get  $\mathcal{MC}(\mathcal{R}_1, T, m) = O(m\gamma \log^2 n)$ . We now show that  $\mathcal{MC}(\mathcal{R}_2, T, m) = O(n \log n \log_\beta m)$ . For an integer  $0 \leq i \leq \lceil \log n \rceil$ , let  $V_i = \{v \mid \eta(v) = i\}$ . For  $v \in V_i$ , denote by  $T_l(v)$  the subtree of  $T(v)$  that contains precisely  $v$  and its light subtrees. The proofs of the following observations are straightforward.

1. The  $\mathcal{R}_2$  communication incurred by  $v$  flows only on  $T_l(v)$ .
2. The number of times  $v$  broadcasts a message on  $T_l(v)$  is  $O(\log_\beta m)$ . The reason is that  $v$  broadcasts a message on  $T_l(v)$  when  $\tilde{d}_v$  increases by a multiplicative factor of  $\sqrt{\beta}$  and this can only happen  $O(\log_\beta m)$  times in the increasing model.
3. For every  $v$  and  $u$  in  $V_i$ , the trees  $T_l(v)$  and  $T_l(u)$  are disjoint.

The  $\mathcal{R}_2$  message complexity incurred by the nodes of  $V_i$  is therefore bounded by  $O(n \log_\beta m)$  and as there are at most  $\lceil \log n \rceil + 1$  such sets,  $\mathcal{MC}(\mathcal{R}_2, T, m) = O(n \log n \log_\beta m)$  which proves the first part of the lemma. The second part of the lemma follows from the fact that all messages are of  $O(\log \log n)$  bits. ■

## 4 Dynamic labeling schemes for distance in trees

Given the edge-dynamic or the increasing dynamic model, we now show how to use a root-distance protocol (specifically,  $\mathcal{R}_{dyn}(\beta)$  for the edge-dynamic model or  $\mathcal{R}_{inc}(\beta)$  for the increasing dynamic model) to give a dynamic labeling scheme for distances. We use similar ideas to the ones appearing in [15, 27].

Informally, we recursively decompose the given tree into subtrees using separators. In this recursive partitioning, each vertex belongs to  $O(\log n)$  subtrees, one for each level of the recursion. The mechanism for estimating the distance to the root is applied separately to each of the decomposed subtrees. Therefore, each vertex  $v$  participates in  $O(\log n)$  such protocols, each corresponding to a subtree that  $v$  belongs to. This enables  $v$  to maintain estimates to the distances between  $v$  and the roots of these subtrees. These distance estimates are then encoded in  $v$ 's label. It so happens that the distance between any two nodes in the dynamic tree can be retrieved from their corresponding lists of estimates, which are encoded in their labels. Let us note that if such a decomposition can be efficiently maintained in a more general dynamic model (such as one which allows also leaf insertions and leaf deletions), then probably our results could be extended to such a dynamic model.

## 4.1 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 [27], one can recursively partition the tree by separators, as follows. At the first stage, we choose some vertex  $v$  in  $T$  to be the *level-1* separator of  $T$ . By removing  $v$ ,  $T$  breaks into disconnected subtrees  $T_1, T_2, \dots, T_q$ , each of size at most  $n/2$ . Each such subtree is decomposed recursively by choosing some vertex to be a level-2 separator, etc. It is easy to show that the number of levels in the recursion is  $O(\log n)$ . The subtrees chosen at the  $l$ 'th level of the recursion are referred to as *level  $l$  subtrees*. Note that for each  $l$ , the level  $l$  subtrees are edge-disjoint but the entire collection contains overlapping trees. For a vertex  $v$ , let  $l(v)$  denote the level on which  $v$  was selected as a separator. For every vertex  $v$  and every level  $1 \leq l \leq l(v)$ , vertex  $v$  belongs to a unique level  $l$  subtree, denoted  $T_l(v)$ . For convenience, whenever a subtree  $T'$  on some level of this recursive partition is split into subtrees  $T_1, T_2, \dots, T_q$  by removing a separator node  $v$  from  $T'$ , we formally define each of these subtrees  $T_i$  to include  $v$  as its root. For a vertex  $v$  and a level  $l$ , denote by  $s_l(v)$  the root of  $T_l(v)$ , which is, as explained above, the level  $l$  separator that defined  $T_l(v)$ . Define the *separator tree*  $T^{sep}$  to be the tree rooted at the level 1 separator of  $T$ , with the level-2 separators as its children, and generally, with each level  $j + 1$  separator as the child of the level  $j$  separator above it in the decomposition. Define the *level  $j$  separator of  $v$*  to be the ancestor of  $v$  in  $T^{sep}$  at height  $j$ .

## 4.2 The Scheme $\pi(\beta) = \langle \mathcal{M}(\beta), \mathcal{D}(\beta) \rangle$

The following discussion applies to both models with  $\mathcal{R}$ , their appropriate root-distance protocol (specifically,  $\mathcal{R}_{dyn}(\beta)$  for the edge dynamic model and  $\mathcal{R}_{inc}(\beta)$  for the increasing model).

### 4.2.1 The Marker $\mathcal{M}(\beta)$

Initially, during the preprocessing stage, each vertex  $v$  is assigned a unique identity  $id(v)$  which is encoded using  $O(\log n)$  bits. Then, simultaneously for each level  $l$ , the marker protocol  $\mathcal{M}(\beta)$  invokes  $\mathcal{R}$  separately on each level  $l$  subtree. The root of such a tree is the appropriate level  $l$  separator. Thus, for each level  $l$ , each vertex  $v$  keeps a  $\beta$ -approximation  $\tilde{d}(v, l)$  for  $d^\omega(v, s_l(v))$ . Fix a vertex  $v$ . The label of  $v$ , denote  $Label(v)$ , consists of two sublabels, namely, the *separator sublabel*  $Sep(v)$  and the *distance sublabel*  $Dist(v)$ . The

separator sublabel  $Sep(v)$  is assigned at the preprocessing stage and consists of the list of ancestors of  $v$  in  $T^{sep}$ , i.e.,  $Sep(v) = \langle id(s_1(v)), id(s_2(v)) \cdots id(s_{l(v)}(v)) \rangle$ , where  $s_j(v)$  is the level  $j$  separator of  $v$ . The distance sublabel  $Dist(v)$  is the list of  $v$ 's estimated distances to each of these separators, i.e.,  $Dist(v) = \langle \tilde{d}(v, 1), \tilde{d}(v, 2) \cdots \tilde{d}(v, l(v)) \rangle$ .

#### 4.2.2 The Decoder $\mathcal{D}(\beta)$

Algorithm  $\mathcal{D}(\beta)$  gets as input the labels  $Label(u)$  and  $Label(v)$  of two vertices  $u$  and  $v$ . The decoder  $\mathcal{D}(\beta)$  scans the lists of the separator sublabels  $Sep(u)$  and  $Sep(v)$  from left to right looking for the last index  $i$  such that  $s_i(v) = s_i(u)$ . It then returns  $\mathcal{D}(\beta)(Label(u), Label(v)) = \tilde{d}(u, i) + \tilde{d}(v, i)$ .

### 4.3 Correctness and analysis

#### 4.3.1 Correctness

**Lemma 4.1** *The decoder  $\mathcal{D}$  yields a  $\beta$ -approximation to the weighted distance, i.e.,*

1.  $d^\omega(u, v)/\beta \leq \mathcal{D}(\beta)(Label(u), Label(v)) \leq \beta \cdot d^\omega(u, v)$  in the edge-dynamic model,
2.  $\mathcal{D}(\beta)(Label(u), Label(v)) \leq d^\omega(u, v) \leq \beta \cdot \mathcal{D}(\beta)(Label(u), Label(v))$  in the increasing-dynamic model.

**Proof:** The last (from left to right) index  $i$  such that  $s = s_i(v) = s_i(u)$ , is the last separator common to both  $u$  and  $v$ . Therefore, the path connecting  $u$  and  $v$  must pass through  $s$  and therefore  $d^\omega(u, v) = d^\omega(u, s) + d^\omega(v, s)$ . The first part of the lemma follows since by Lemma 3.3,  $d^\omega(u, s)/\beta \leq \tilde{d}(u, s) \leq \beta d^\omega(u, s)$  and  $d^\omega(s, v)/\beta \leq \tilde{d}(s, v) \leq \beta d^\omega(s, v)$  in the edge-dynamic model. The second part of the lemma follows since by Lemma 3.8,  $\tilde{d}(u, s) \leq d^\omega(u, s) \leq \beta \tilde{d}(u, s)$  and  $\tilde{d}(s, v) \leq d^\omega(s, v) \leq \beta \tilde{d}(s, v)$  in the increasing-dynamic model. ■

#### 4.3.2 Communication complexities

**Lemma 4.2** 1. *For the edge-dynamic model,  $\mathcal{MC}(\mathcal{M}(\beta), T, m) = O(m\alpha\Lambda(T) \log^3 n)$  and  $\mathcal{BC}(\mathcal{M}(\beta), T, m) = O(m\alpha\Lambda(T) \log^3 n \cdot \log \log n)$*

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

**Proof:** Given one of the above models, let  $\mathcal{R}$  be the root-distance protocol for that model. Fix  $l$  and let  $T_1, T_2, \dots$  be the level  $l$  subtrees. These subtrees are edge-disjoint and  $\mathcal{R}$  is performed on each of them separately. Therefore, by Lemmas 3.4 and 3.9, the message complexity of  $\mathcal{M}(\beta)$  on all the level  $l$  subtrees together is bounded by  $O(m\alpha\Lambda(T)\log^2 n)$  in the edge dynamic model and by  $O(m\gamma\log^2 n + n\log n\log_\beta m)$  in the increasing-dynamic model. The lemma follows since there are at most  $\log n + 1$  levels and since all messages use  $O(\log \log n)$  bits. ■

### 4.3.3 Memory size

**Lemma 4.3** *Let  $W$  be the maximum weight given to an edge. Then, for the edge-dynamic model,  $\mathcal{MS}(\mathcal{M}_{dyn}(\beta), T, m) = O(\log^2 n + \log n \log W)$ , and for the increasing-dynamic model,  $\mathcal{MS}(\mathcal{M}_{inc}(\beta), T, m) = O(\log^2 n \log W)$*

**Proof:** For each  $v$ , we have  $l(v) = O(\log n)$  and therefore the number of elements in the lists  $Sep(v)$  and  $Dist(v)$  is  $O(\log n)$ . The number of bits in each element in  $Sep(v)$  is bounded by  $O(\log n)$ , and the number of bits in each element in  $Dist(v)$  is bounded by  $O(\log(nW)) = O(\log n + \log W)$ . Therefore, the number of bits in the label given by either  $\mathcal{M}_{dyn}$  (for the edge-dynamic model) or by  $\mathcal{M}_{inc}$  (for the increasing-dynamic model) is  $O(\log^2 n + \log n \log W)$ . Protocol  $\mathcal{R}_{dyn}(\beta)$  uses one counter of size  $O(\log W)$  and Protocol  $\mathcal{R}_{inc}(\beta)$  uses  $O(\log n)$  counters, each of size  $O(\log W)$ . Since, in the corresponding marker protocol, each vertex participates in  $O(\log n)$  root-distance protocols, the lemma follows. ■

### 4.3.4 Label size

As mentioned in the proof of the above lemma, the label size of  $\mathcal{M}(\beta)$  (for either one of our dynamic models) is  $O(\log^2 n + \log n \log W)$ . We now show that this upper bound for the label size can be further reduced. Recall that for each vertex  $v$ , the separator sublabel  $Sep(v)$  contains the list of identifiers of  $v$ 's ancestors in  $T^{sep}$ . In Section 2.2 of [15], they show that one can choose unique identifiers to the vertices so that, for each vertex  $v$ , the size of its separator sublabel  $Sep(v)$  is  $O(\log n)$ . Thus, using their method in the preprocessing stage, we obtain that the size of the separator sublabels is  $O(\log n)$ . In [15] they also show that the elements in the distance sublabels  $Dist(v)$  can be transformed into shorter elements, such that an approximation to the original elements can be extracted from the shorter ones. Let  $\tilde{L}(v)$  denote the separator sublabel  $Sep(v)$  together with the list of such shorter elements. Going along the same lines as in Section 2.2 of [15], it turns out that for any two vertices  $v$  and

$u$ , our decoder, applied on  $\tilde{L}(v)$  and  $\tilde{L}(u)$  (instead of  $Label(v)$  and  $Label(u)$ ), yields a good approximation to  $\mathcal{D}(Label(v), Label(u))$ , which is a  $\beta$ -approximation to  $d^\omega(v, u)$ . Specifically, let us now rephrase some definitions and results from [15], using our terminology.

A pair of integer functions  $\langle \lambda, \varphi \rangle$  is an  $s$ -estimator of  $\{1, 2, \dots, M\}$  if  $\lambda : \{1, 2, \dots, M\} \rightarrow \{1, 2, \dots, 2^\epsilon\}$  (where typically,  $2^\epsilon \ll M$ ),  $\varphi : \{1, 2, \dots, 2^\epsilon\} \rightarrow \{1, 2, \dots\}$ , and for every  $x \in \{1, 2, \dots, M\}$ , we have  $x \leq \varphi(\lambda(x)) \leq s \cdot x$ . Intuitively,  $\lambda$  is a function ‘compacting’  $x$  and  $\varphi$  is a function attempting to ‘reconstruct’  $x$  from  $\lambda(x)$ . The *size* of the estimator is  $\epsilon$ .

**Lemma 4.4** *For every  $k$ , and every  $q \in \{1, 2, \dots, k\}$ , there exists a  $(1 + \frac{1}{2^q})$ -estimator of  $\{1, 2, \dots, 2^k\}$  of size  $\epsilon = q + \lceil \log(k - q + 1) \rceil$ .*

**Lemma 4.5** *Let  $\epsilon$  be the size of an  $s$ -estimator of  $\{1, 2, \dots, nW\}$ . One can transform each label  $Label(v)$  (given by our marker  $\mathcal{M}$  to vertex  $v$ ) into  $\tilde{L}(v)$ , such that the following holds.*

- 1) *For each vertex  $v$ , the number of bits in  $\tilde{L}(v)$  is bounded from above by  $\epsilon \log n + O(\log n)$ ,*
- 2)  *$\mathcal{D}(\tilde{L}(v), \tilde{L}(u))$  is an  $s$ -approximation to  $\mathcal{D}(Label(v), Label(u))$ .*

In particular, for constant  $\beta > 1$ , set  $k = \lceil \log(Wn) \rceil$  and let  $q$  be some constant integer such that  $q \in \{1, 2, \dots, k\}$  and  $1 + \frac{1}{2^q} \leq \sqrt{\beta}$ . We obtain the following.

**Corollary 4.6** *For constant  $\beta > 1$ , one can transform each label  $Label(v)$  into  $\tilde{L}(v)$ , such that the following holds.*

- 1) *For each vertex  $v$ , the number of bits in  $\tilde{L}(v)$  is bounded from above by  $O(\log n \log \log(Wn))$ ,*
- 2)  *$\mathcal{D}(\tilde{L}(v), \tilde{L}(u))$  is an  $\sqrt{\beta}$ -approximation to  $\mathcal{D}(Label(v), Label(u))$ .*

For constant  $\beta > 1$ , let  $\pi'(\beta) = \langle \mathcal{M}'(\beta), \mathcal{D}'(\beta) \rangle$  be the dynamic  $\sqrt{\beta}$ -approximate distance labeling scheme  $\pi(\sqrt{\beta})$ . Note that  $\pi'(\beta)$  has the same asymptotic communication complexities and memory size as  $\pi(\beta)$ .

The following corollary follows, by applying the above mentioned method for compacting the labels given by  $\mathcal{M}'(\beta)$ .

**Corollary 4.7** *For constant  $\beta > 1$ ,  $\mathcal{LS}(\mathcal{M}'(\beta), T, m) = O(\log n \log \log(Wn))$ .*

By combining Lemmas 4.1, 4.2, 4.3, and Corollary 4.7, we obtain the following theorems.

**Theorem 4.8** *For the edge-dynamic model, there exists a dynamic  $\beta$ -approximate distance labeling scheme  $\pi(\beta) = \langle \mathcal{M}, \mathcal{D} \rangle$  with the following complexities.*

- $\mathcal{MC}(\mathcal{M}, T, m) = O(m\alpha\Lambda(T) \log^3 n)$ ,
- $\mathcal{BC}(\mathcal{M}, T, m) = O(m\alpha\Lambda(T) \log^3 n \cdot \log \log n)$ ,
- $\mathcal{LS}(\mathcal{M}, T, m) = \mathcal{MS}(\mathcal{M}, T, m) = O(\log^2 n + \log n \log W)$ ,

- For constant  $\beta > 1$ ,  $\mathcal{LS}(\mathcal{M}, T, m) = O(\log n \log \log(Wn))$ .

**Theorem 4.9** *For the increasing-dynamic model, there exists a dynamic  $\beta$ -approximate distance labeling scheme  $\pi(\beta) = \langle \mathcal{M}, \mathcal{D} \rangle$  with the following complexities.*

- $\mathcal{MC}(\mathcal{M}, T, m) = O(m\gamma \log^3 n + n \log^2 n \log_\beta m)$ ,
- $\mathcal{BC}(\mathcal{M}, T, m) = O((m\gamma \log^3 n + n \log^2 n \log_\beta m) \cdot \log \log n)$ ,
- $\mathcal{LS}(\mathcal{M}, T, m) = O(\log^2 n + \log n \log W)$ ,
- For constant  $\beta > 1$ ,  $\mathcal{LS}(\mathcal{M}, T, m) = O(\log n \log \log(Wn))$ ,
- $\mathcal{MS}(\mathcal{M}, T, m) = O(\log^2 n \log W)$ .

#### 4.4 Lower bounds for message complexity

Next we establish simple lower bounds on the message complexity of  $\beta$ -approximate distance labeling schemes in the edge-dynamic model. Our lower bounds apply also for the weaker dynamic model, which assumes that each topological event may occur only at quiet times.

Let  $T$  be a rooted tree and let  $\pi(\beta) = \langle \mathcal{M}(\beta), \mathcal{D}(\beta) \rangle$  be a  $\beta$ -approximate distance labeling scheme for  $T$  operating in the edge-dynamic model. 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

$$\begin{aligned} \tilde{B}(e, d) &= \min\{B_{up}(e, d), B_{down}(e, d)\}, \\ \tilde{\Lambda} &= \max\left\{\frac{\tilde{B}(e, d)}{d} \mid d \geq 1, e \in E\right\}. \end{aligned}$$

**Lemma 4.10** *For constant  $\beta > 1$ ,  $\mathcal{MC}(\mathcal{M}(\beta), T, m) = \Omega(m\tilde{\Lambda})$ .*

**Proof:** Let  $e$  and  $d$  be the parameters maximizing  $\tilde{\Lambda}$ . Assume that each weight change occurs only at quiet times and consider the following scenario. Initially all the edge-weights of  $T$  are set to 1. At the first stage,  $e$ 's weight,  $\omega(e)$ , is raised to be  $(2d + 1) \cdot \beta^2$ . At the second stage,  $\omega(e)$  is reduced from  $(2d + 1) \cdot \beta^2$  back to 1. These two stages are now executed repeatedly.

We claim that at each stage of each two-stage cycle, at least  $\tilde{B}(e, d)$  messages must be sent by the marker protocol  $\mathcal{M}(\beta)$ . This is because otherwise there must exist a pair of vertices,  $u$  and  $v$ , on different sides of  $e$  and both at distance at most  $d$  from an endpoint of  $e$ , that did not receive any message during that stage. Therefore, their labels have not

changed from the previous stage, contradicting the fact that these labels must maintain a  $\beta$ -approximate to  $d^\omega(u, v)$  at all quiet times. This establishes the lemma. ■

Note that for any integer value  $1 \leq \nu \leq n$ , there exists a tree  $T$  whose local density satisfies  $\Lambda(T) = \Theta(\Lambda(\tilde{T})) = \Theta(\nu)$ . We therefore get the following corollary.

**Corollary 4.11** *For any value  $1 \leq \nu \leq n$ , let  $T$  be a tree with local density  $\Lambda(T) = \Theta(\nu)$ . For any constant  $\beta > 1$ , let  $\pi(\beta) = \langle \mathcal{M}(\beta), \mathcal{D}(\beta) \rangle$  be a  $\beta$ -approximate distance labeling scheme for  $T$  in the edge-dynamic model. We have,  $\mathcal{MC}(\mathcal{M}(\beta), T, m) = \Omega(m\Lambda(T))$ .*

## 5 A different labeling scheme $\pi_{path}(\beta)$ for paths

Consider the edge-dynamic model restricted to paths. We show a compact  $\beta$ -approximation labeling scheme for this model with only  $O(m\alpha \log^2 n)$  message complexity. Let  $P$  be an  $n$ -node path. Consider the root-distance protocol  $\mathcal{R}_{dyn}(\beta)$  for the edge-dynamic model on the  $P$ . In this model each node  $v$  monitors the messages passing through it and uses them to estimate its distance to the root. In  $\mathcal{M}_{dyn}(\beta)$ , this is applied to all of  $v$ 's separators and thus causes an overhead of  $\log n$  on the message and bit complexities of  $\mathcal{R}_{dyn}(\beta)$ . Let  $\beta' = \Theta(\beta)$  be some value to be defined later. In the improved labeling scheme,  $\pi_{path}(\beta) = \langle \mathcal{M}_{path}(\beta), \mathcal{D}_{path}(\beta) \rangle$ , the messages of marker  $\mathcal{M}_{path}(\beta)$  are the same messages of Protocol  $\mathcal{R}_{dyn}(\beta')$  applied on the whole path (and not separately for all the separators). Therefore, by Corollary 3.6, the message complexity of  $\mathcal{M}_{path}(\beta)$  is  $O(m\alpha \log^2 n)$ .

First note that one cannot simply label the vertices with the same labels given by the root-estimate protocol and then, given the labels of  $v$  and  $u$ , output the difference between  $d_{approx}(v)$  and  $d_{approx}(u)$ , the corresponding estimated distances to the root. The reason is that though  $d_{approx}(v)$  and  $d_{approx}(u)$  are good estimates for  $d^\omega(v)$  and  $d^\omega(u)$ , respectively, the difference between them may not be a good estimate to  $d^\omega(v, u)$ .

Recall that in Step 4 of Protocol  $\mathcal{R}_{dyn}(\beta')$ , whenever an  $l$ -level bin  $b$  gets filled with positive (respectively, negative) tokens,  $b$  sends the signal  $l$  (resp.,  $-l$ ) to its supervisors (in the case of a path, only one supervisor). We consider the signal  $l$  as a *positive  $l$ -signal* and the signal  $-l$  as a *negative  $l$ -signal*. In this section, whenever counting either tokens, weight changes or signals, we assume that the counting is made with the appropriate signs, i.e., positive and negative cancel each other out.

Let us first give an informal description of the scheme  $\pi_{path}(\beta)$ . By monitoring the messages of  $\mathcal{M}_{path}(\beta)$ , for each level  $l$ , every vertex  $v$  encodes in its label the number of times it has received an  $l$ -signal. In addition,  $v$ 's label also contains its height and its initial

weighted distance to the root.

Let  $m'$  be the amount of weight changes occurring in the edges of the subpath  $I = [u, v]$ . Given the labels of two vertices  $v$  and  $u$ , the decoder can easily extract the initial weighted distance  $H_0^\omega$  and the initial unweighted distance  $|I|$  between the vertices. Our goal is to estimate the weighted distance between  $v$  and  $u$ , i.e.,  $H_0^\omega + m'$ . Assume without loss of generality that  $u$  is closer to the root than  $v$ . For a vertex  $w$ , let  $C(w)$  denote the amount of tokens passing through  $w$ . It can be shown that at any quiet time,  $C(u) + m' = C(v) + \rho$ , where  $\rho$  is the amount of tokens stuck in bins of the subpath  $I$ . Let us note that the decoder cannot simply output  $H_0^\omega + C(v) - C(u)$ , since this may not be a good approximation to  $H_0^\omega + m'$ . The reason is that  $\rho$  may be very large in comparison to  $H_0^\omega + m'$  (the subpath  $I$  may contain a large bin with many tokens in it, tokens which do not correspond to weight changes occurring in  $I$ ). For a vertex  $w$  and a level  $j$ , define  $C_j(w)$  to be the amount of tokens coming from bins of level at most  $j$  and passing through  $w$ . Clearly,  $C_j(w)$  can be extracted from  $w$ 's label. It turns out that there exists some  $k$  (which roughly equals  $\lceil \log |I| \rceil$ ), such that  $\max\{H_0^\omega + C_k(v) - C_k(u), |I|\}$  is, in fact, a good approximation to  $H_0^\omega + m'$ . We now describe the scheme  $\pi_{path}(\beta) = \langle \mathcal{M}_{path}(\beta), \mathcal{D}_{path}(\beta) \rangle$  more formally.

Let  $I = [u, v]$  denote the subpath connecting  $u$  and  $v$ , and let  $(u, v]$  denote the subpath connecting  $v$  and  $u$ 's child, i.e.,  $(u, v] = [u, v] \setminus \{u\}$ . Let

$$\delta = 4.5 \quad , \quad \beta' = \frac{\beta - 1 + \delta}{\delta} \quad , \quad \text{and} \quad \alpha' = \frac{1}{\beta' - 1} .$$

Note that  $\alpha' = \frac{1}{\beta' - 1} = \frac{\delta}{\beta - 1} = \delta\alpha$ . As mentioned before, the messages of marker  $\mathcal{M}_{path}(\beta)$  are the messages of Protocol  $\mathcal{R}_{dyn}(\beta')$  applied on the whole path (and not separately for all the separators). Therefore, by Lemma 3.4 and by the fact that  $\alpha' = \Theta(\alpha)$ , we obtain the following lemma.

**Lemma 5.1**  $\mathcal{MC}(\mathcal{M}_{path}(\beta), P, m) = O(m\alpha \log^2 n)$  and  $\mathcal{BC}(\mathcal{M}_{path}(\beta), P, m) = O(m\alpha \log^2 n \cdot \log \log n)$

## 5.1 The label structure

We say that a signal *passes through a vertex*  $v$  if  $v$  receives this signal and this signal was originated at a vertex above  $v$  (including, in particular,  $v$  itself) and destined to a vertex strictly below  $v$ . For each vertex  $v$  and level  $l$ ,  $v$  keeps the counter  $c_l(v)$  which is the number of  $l$ -signals passing through  $v$ . (Recall that the counting is done with the appropriate signs i.e., positive and negative  $l$ -signals cancel each other out).

For each vertex  $v$ , let  $H(v)$  denote the height of  $v$  in  $P$  and let  $H_0^\omega(v)$  denote the initial weighted distance between  $v$  and the root. Let  $q$  be the depth of the bin hierarchy. Recall that by the first item in Observation 3.1, we have  $q = O(\log n)$ . The label of a vertex  $v$  is  $Label(v) = (H(v), H_0^\omega(v), c_{-1}(v), c_0(v), c_1(v) \cdots, c_q(v))$ .

## 5.2 The decoder

Given  $Label(v)$  and  $Label(u)$ , the decoder algorithm estimates the weighted distance between  $u$  and  $v$  in the following manner. Without loss of generality assume  $H(v) > H(u)$ , i.e.,  $u$  is closer to the root. Note that the unweighted distance between  $v$  and  $u$  is simply  $|I| = H(v) - H(u)$ . This value, therefore, can easily be extracted from the corresponding labels. Let  $j = \lfloor 1 + \log |I| \rfloor$ . Note that for any level  $l$ , the distance between an  $l$ -level bin  $b$  and the closest  $l'$ -level bin, where  $l' > l$ , is at least  $2^l$ . Therefore, there must exist some  $k \in \{j, j+1\}$  such that, there does not exist a  $(k+1)$ -level bin in  $I$ . Note that given the labels  $Label(v)$  and  $Label(u)$ , the decoder can find such  $k$ , by looking at the heights of  $v$  and  $u$ . Let  $\rho_k$  is the amount of tokens stuck in bins in  $(u, v]$  of level at most  $k$ . Formally, let  $K(I) = \{b \mid b \in (u, v] \text{ and } b \text{ is an } l\text{-level bin, where } l \leq k\}$ , and let  $\rho_k = \sum_{b \in K(I)} \tau(b)$ .

For a vertex  $w$  and a level  $l$ , let  $C_l(w) = \sum_{i=-1}^l c_i(w) \cdot Cap(i)$ . The value  $C_l(w)$  informally denotes the amount of tokens corresponding to messages passing through  $w$  which were originated at a level at most  $l$ . Let  $H_0^\omega = H_0^\omega(v) - H_0^\omega(u)$ , i.e.,  $H_0^\omega$  is the initial weighted distance between  $u$  and  $v$ . Let  $\tilde{d}(v, u) = H_0^\omega + C_k(v) - C_k(u)$ . The decoder outputs  $\mathcal{D}_{path}(Label(v), Label(u)) = \max\{\tilde{d}(v, u), |I|\}$ .

## 5.3 Analysis and correctness

### 5.3.1 Correctness

Let us now show that  $\pi_{path}(\beta) = \langle \mathcal{M}_{path}(\beta), \mathcal{D}_{path}(\beta) \rangle$  is indeed a dynamic  $\beta$ -approximate distance labeling scheme on paths. We begin with the following lemma.

**Lemma 5.2** *For  $k$  as defined above, the equation  $C_k(v) - C_k(u) \leq m' - \rho_k$  holds at all quiet times.*

**Proof:** Initially the lemma is trivially correct. Note that for any scenario of weight changes, the final value of each of the parameters in the above equation, after all updates of  $\mathcal{M}_{path}(\beta)$  are performed, is not affected by the order and pace in which the messages of  $\mathcal{M}_{path}(\beta)$  are sent. We may therefore assume that the weight changes are sufficiently spaced so that the

marker  $\mathcal{M}_{path}(\beta)$  has enough time to complete its operation in response to a given topological change, before the occurrence of the next change.

We use induction on the following events.

1. Assume that the value  $m'$  increases (respectively, decreases) by one because an edge  $(w, z)$  in the subpath  $I = [u, v]$  has increased (resp., decreased) its weight by one. Assume without loss of generality that  $w$  is above  $z$ . Then the local bin at  $z$  gets full with a positive (resp., negative) token and therefore the value of  $m' - \rho_k$  remains the same. The equation remains valid since clearly, the values of both  $C_k(v)$  and  $C_k(u)$  remain the same.
2. Assume that some  $l$ -level bin  $b$  gets full with tokens and a signal from that bin travels from  $b$  to  $b'$ ,  $b'$ 's supervisor. Note that the value of  $m'$  remains the same. Consider the following cases.
  - Assume  $l \geq k + 1$ . Clearly, none of the equation's parameters change. In the following cases we assume  $l \leq k$ .
  - Assume  $b'$  is above  $u$  (including, in particular,  $u$  itself) or  $b$  is strictly below  $v$ . Again this event doesn't influence any of the equation's parameters.
  - Assume both  $b$  and  $b'$  are in  $(u, v]$ . In this case, the signal does not pass through either  $u$  or  $v$ , therefore, the value of both  $C_k(v)$  and  $C_k(u)$  remain the same. Since the distance between  $b$  and its supervisor  $b'$  is at least  $2^l$  we get that  $2^l \leq H(v) - H(u)$  and therefore  $l \leq \log(H(v) - H(u)) < k$ . Thus, the level of the supervisor  $b'$  is at most  $k$ . Therefore, since both  $b$  and  $b'$  are in  $(u, v]$ , the value of  $\rho_k$  doesn't change as well.
  - Assume  $b$  is above  $u$  (including  $u$ ) and  $b'$  is strictly below  $v$  then  $C_k(v)$  and  $C_k(u)$  get even and the value of  $\rho_k$  remains the same.
  - Assume  $b$  is above  $u$  (including  $u$ ) and  $b'$  is in  $(u, v]$ . In this case, by the way  $k$  is defined, the supervisor  $b'$  must be of level at most  $k$  (since there is no  $(k + 1)$ -level bin in  $I$ ). It therefore follows that  $\rho_k$  and  $C_k(u)$  get even and clearly, the value of  $C_k(v)$  remains the same.
  - Assume  $b$  is in  $(u, v]$  and  $b'$  is strictly below  $v$ . In this case,  $\rho_k$  and  $C_k(v)$  get even and clearly, the value of  $C_k(u)$  remains the same. ■

We now show that given the labels of two vertices  $v$  and  $u$ , the output of the decoder yields a  $\beta$ -approximation to  $d^\omega(v, u)$ .

**Lemma 5.3** *At any quiet time, given the labels  $Label(v)$  and  $Label(u)$  (assigned by  $\mathcal{M}_{path}(\beta)$  to the vertices  $v$  and  $u$ ), the decoder  $\mathcal{D} = \mathcal{D}_{path}(\beta)$  satisfies  $d^\omega(v, u)/\beta \leq \mathcal{D}(Label(v), Label(u)) \leq d^\omega(v, u) \cdot \beta$ .*

**Proof:** Fix some quiet time. Assume without loss of generality that  $u$  is closer to the root than  $v$ . Given  $Label(v)$  and  $Label(u)$ , the decoder  $\mathcal{D}(Label(v), Label(u))$  outputs  $\mathcal{D}(Label(v), Label(u)) = \max\{\tilde{d}(v, u), |I|\}$ , where  $\tilde{d}(v, u) = H_0^\omega + C_k(v) - C_k(u)$ .

Let  $m'$  be the number of weight changes occurring in the edges of  $I$ , then  $d^\omega(v, u) = H_0^\omega + m'$ . By the previous lemma,  $|\tilde{d}(v, u) - d^\omega(v, u)| \leq |\rho_k|$ . Let us first bound the value of  $|\rho_k|$ . Note that for every level  $l$ , the distance between any two  $l$  level bin is at least  $2^{l+1}$ . Therefore, the number of  $l$ -level bins in  $I$  is at most  $1 + |I|/2^{l+1}$ . It follows that  $|\rho_k| \leq \sum_{l=-1}^k (1 + |I|/2^l) \cdot Cap(l)$ . Note that  $k \leq \log |I| + 2$ , therefore,  $1 \leq \frac{8|I|}{2^{k+1}}$ . Consequently, for  $l \leq k$ , we have  $1 + |I|/2^{l+1} \leq \frac{9|I|}{2^{l+1}} = \frac{\delta|I|}{2^l}$ . Therefore,

$$|\rho_k| \leq \sum_{l=-1}^k \frac{\delta|I|}{2^l} \cdot Cap(l) \leq \frac{\delta|I|k}{\alpha'(\log n + 1)} = \frac{|I|k}{\alpha(\log n + 1)} \leq \frac{|I|}{\alpha}.$$

Consequently,  $\tilde{d}(v, u) \leq d^\omega(v, u) + \frac{|I|}{\alpha}$ . Therefore, since  $|I| \leq d^\omega(v, u)$ , we obtain  $\tilde{d}(v, u) \leq d^\omega(v, u)(1 + 1/\alpha) = d^\omega(v, u) \cdot \beta$ . Therefore  $\mathcal{D}(Label(v), Label(u)) = \max\{\tilde{d}(v, u), |I|\} \leq \beta \cdot d^\omega(v, u)$ . On the other hand, we have  $d^\omega(v, u) \leq \tilde{d}(v, u) + \frac{|I|}{\alpha} \leq \max\{\tilde{d}(v, u), |I|\} \cdot (1 + \frac{1}{\alpha}) = \beta \cdot \mathcal{D}(Label(v), Label(u))$ . The lemma follows. ■

### 5.3.2 Memory size and label size

Let us now bound the memory size of  $\pi_{path-dym}(\beta)$ . Clearly, all upper bounds for the memory size apply also for the label size.

In this subsection we fix a time  $t$  (not necessarily quiet). All the followings statements hold for time  $t$ . Recall that each vertex  $v$  keeps the values  $H(v)$  and  $H_0^\omega(v)$  as well as  $O(\log n)$  counters of the form  $c_i(v)$ . Note that  $H(v)$  and  $H_0^\omega(v)$  can be encoded using  $O(\log(nW)) = O(\log n + \log W)$  bits. Note also that at time  $t$ , the number of weight changes that have occurred in an edge (counted with the appropriate signs) is  $O(W)$ . Moreover, at time  $t$ , for a fixed edge  $e$  and a counter  $c_i(v)$ , the number of weight changes that have occurred in  $e$ , and correspond to  $i$ -signals passing through  $v$ , is also  $O(W)$ . Therefore, at time  $t$ , the total number of weight changes that correspond to  $i$ -signals passing through  $v$  is  $O(nW)$ . It follows that each counter  $c_i(v)$  can be encoded using  $O(\log nW) = O(\log n + \log W)$  bits. We therefore get the following lemma.

**Lemma 5.4**  $\mathcal{MS}(\mathcal{M}_{path}(\beta), P, m) = O(\log n \log W + \log^2 n)$ .

We now show that each counter  $c_i(v)$  can be encoded using  $O(\log W + \log \log n + \log \alpha)$ , and therefore, for small  $\alpha$ , the above upper bound for the memory size (which also applies for the label size) can be further reduced. In particular, for constant  $\beta > 1$ , we obtain that the memory size is  $O(\log n \log W + \log n \log \log n)$ .

We begin with the following observation which follows from the definition of a supervisor.

**Observation 5.5** *Let  $b'$  be a global bin of level  $l$ . Then  $b'$  is the supervisor of at most two  $(l - 1)$ -level bins.*

**Claim 5.6** *Let  $b'$  be an  $l$ -level global bin and let  $b_1$  and  $b_2$  be the  $(l - 1)$ -level bins who's supervisor is  $b'$  (if  $b'$  supervises only one bin, then  $b_1 = b_2$ ).*

1. *In the case where  $l$  is such that  $Cap(l) > 1$ , if  $b'$  got filled  $\rho$  times then either  $b_1$  or  $b_2$  got filled at least  $\rho$  times.*
2. *In the case where  $l$  is such that  $Cap(l) = 1$ , if  $b'$  got filled  $\rho$  times then either  $b_1$  or  $b_2$  got filled at least  $\rho/2$  times.*

**Proof:** Let  $\rho_1$  (respectively  $\rho_2$ ) be the number of times  $b_1$  (resp.,  $b_2$ ) got filled. Note that if  $Cap(l) > 1$ , then  $Cap(l) = 2 \cdot Cap(l - 1)$ . Therefore, if  $Cap(l) > 1$ , then  $\rho \leq (\rho_1 + \rho_2)/2$ , and the first part of the claim follows. If  $Cap(l) = 1$  then  $Cap(l - 1) = 1$  and  $\rho \leq \rho_1 + \rho_2$ , and the second part of the claim follows. ■

**Corollary 5.7** *Let  $b'$  be a global bin of level  $l$  and let  $\rho$  be the number of times  $b'$  got filled. Then there exists a local bin  $b_l$  such that  $b_l$  got filled  $\Omega(\frac{\rho}{\alpha \log n})$  times.*

**Proof:** Let  $l_1$  be the smallest level such that  $Cap(l) > 1$ . We have  $l_1 = \log(\alpha \log n) + O(1)$ . If  $l \geq l_1$  then by the first part of the previous claim, there exists an  $(l_1 - 1)$ -level bin that got filled  $\rho$  times. We therefore may assume that  $l$  is such that  $Cap(l) = 1$ . The corollary therefore follows from the second part of the previous claim. ■

**Corollary 5.8** *Let  $\rho_{max}$  be the maximum times some bin got filled, then there exists a local bin  $b_l$  such that  $b_l$  got filled  $\Omega(\frac{\rho_{max}}{\alpha \log n})$  times.*

**Lemma 5.9** 1. *For  $\beta > 1$ ,  $\mathcal{MS}(\mathcal{M}_{path}(\beta), P, m) = O(\log n \cdot (\log W + \log \log n + \log \alpha))$ ,*

2. *For constant  $\beta > 1$ ,  $\mathcal{MS}(\mathcal{M}_{path}(\beta), P, m) = O(\log n \log W + \log n \log \log n)$ .*

**Proof:** Let  $\rho_{max}$  be the maximum times some bin got filled. Since the number of times a local bin got filled is at most  $O(W)$ , we have  $\frac{\rho_{max}}{\alpha \log n} = O(W)$  and therefore, by the previous corollary,  $\rho_{max} = O(\alpha W \log n)$ . Consequently, since each vertex  $v$  receives  $l$ -signals

from at most two  $l$  level bins, we obtain that each counter  $c_i(v)$  can be encoded using  $O(\log(\alpha W \log n)) = O(\log W + \log \log n + \log \alpha)$  bits. The first part of the lemma therefore follows since each vertex  $v$  holds  $O(\log n)$  counters. The second part of the lemma follows from the first part. ■

By Lemmas 5.1, 5.3, 5.4 and 5.9, we obtain the following theorem.

**Theorem 5.10**  $\pi_{path}(\beta) = \langle \mathcal{M}_{path}(\beta), \mathcal{D}_{path}(\beta) \rangle$  is a dynamic  $\beta$ -approximate distance labeling scheme on paths with the following complexities.

1.  $\mathcal{MC}(\mathcal{M}_{path}(\beta), P, m) = O(m\alpha \log^2 n)$ ,
2.  $\mathcal{BC}(\mathcal{M}_{path}(\beta), P, m) = O(m\alpha \log^2 n \cdot \log \log n)$ ,
3.  $\mathcal{MS}(\mathcal{M}_{path}(\beta), P, m) = O(\min\{\log n \cdot (\log W + \log \log n + \log \alpha), \log n \log W + \log^2 n\})$ ,
4. For constant  $\beta > 1$ ,  $\mathcal{MS}(\mathcal{M}_{path}(\beta), P, m) = O(\log n \log W + \log n \log \log n)$ .

## 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. ACM* **43**, (1996), 1–19.
- [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] P. Chinn, J. Chavatálová, A. Dewdney, and N. Gibbs. The bandwidth problem for graphs and matrices - survey. *J. of Graph Theory* **6(3)**, (1982), 223–254.
- [8] R. F. Chung and P. D. Seymour. Graphs with small bandwidth and cutwidth. *Discrete Mathematics* **75**, (1989), 113–119.
- [9] 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.
- [10] U. Feige. Approximating the bandwidth via volume respecting embeddings. In *J. of Comput. Syst. Sci.*, **60(3)**, (2000), 510–539.
- [11] U. Feige and K. Talwar. Approximating the bandwidth of caterpillars. In *proc. of Approx 2005*, LNCS 3624 Springer, 62–73, 2005.
- [12] P. Fraigniaud and C. Gavoille. Routing in trees. In *Proc. 28th Int. Colloq. on Automata, Languages & Prog.*, LNCS 2076, pages 757–772, July 2001.
- [13] 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.
- [14] C. Gavoille and D. Peleg. Compact and Localized Distributed Data Structures. Research Report RR-1261-01, LaBRI, Univ. of Bordeaux, France, Aug. 2001.
- [15] 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.

- [16] C. Gavoille, D. Peleg, S. Pérennes and R. Raz. Distance labeling in graphs. *J. of Algorithms*, **53(1)** (2004), 85–112.
- [17] S. Kannan, M. Naor, and S. Rudich. Implicit Representation of Graphs. *SIAM J. on Discrete Math.* **5**, (1992), 596–603.
- [18] M. Kao, X. Li, and W. Wang. Average Case Analysis for Tree Labelling Schemes. In *Proc. 16th Int. Symp. on Algorithms and Computation*, pages 136–145, Dec. 2005.
- [19] H. Kaplan and T. Milo. Short and simple labels for small distances and other functions. In *Workshop on Algorithms and Data Structures*, Aug. 2001.
- [20] 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.
- [21] M. Katz, N.A. Katz, A. Korman and D. Peleg. Labeling schemes for flow and connectivity. *SIAM Journal on Computing* **34** (2004),23–40.
- [22] 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, Feb. 2000.
- [23] A. Korman. General Compact Labeling Schemes for Dynamic Trees. In *Proc. 19th International Symposium on Distributed Computing*, Sep. 2005.
- [24] A. Korman, D. Peleg and Y. Rodeh. Constructing Labeling Schemes through Universal Matrices. In *Proc. 17th Int. Symp. on Algorithms and Computation*, Dec. 2006.
- [25] A. Korman, D. Peleg, and Y. Rodeh. Labeling schemes for dynamic tree networks. *Theory of Computing Systems* **37**, (2004), 49–75.
- [26] R. Krauthgamer, J. Lee, M. Mendel, and A. Naor. Measured descent: A new embedding method for finite metrics. In *Proc. 45th Ann. IEEE Symp. on Foundations of Computer Science*, Oct. 2004.
- [27] 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.
- [28] 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.
- [29] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. of ACM*, **51(6)**, (2004), 993–1024.
- [30] 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.