Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT

Journal of Algorithms 50 (2004) 168–193

Journal of Algorithms

www.elsevier.com/locate/jalgor

Restructuring ordered binary trees ☆

William Evans * and David Kirkpatrick

Department of Computer Science, University of British Columbia, Vancouver BC, Canada, V6T 1Z4

Received 29 June 2000

Abstract

We consider the problem of restructuring an ordered binary tree T, preserving the in-order sequence of its nodes, so as to reduce its height to some target value h. Such a restructuring necessarily involves the downward displacement of some of the nodes of T. Our results, focusing both on the maximum displacement over all nodes and on the maximum displacement over leaves only, provide (i) an explicit tradeoff between the worst-case displacement and the height restriction (including a family of trees that exhibit the worst-case displacements) and (ii) efficient algorithms to achieve height-restricted restructuring while minimizing the maximum node displacement.

1. Introduction

Suppose we are given an ordered binary tree T, perhaps a binary search tree for some set S of keys (or perhaps an alphabetic prefix code tree for a set S of symbols). Tree T might have been constructed, either through explicit use of known key access frequencies or by some self-adjusting strategy, to minimize the *expected* cost of key accesses (expressed as the expected depth of the keys stored in T). As a consequence, the worst-case cost of key accesses (expressed as the *maximum* depth of a key stored in T), though improbable, may be unacceptably large. The general question that we address is: in the absence of explicit information concerning key access frequencies, to what extent can we improve the worst-case behavior of a given search structure without unduly compromising its expected-case behavior? We focus, in fact, on the more stringent requirement of minimizing the (local) degradation in the access cost for *any* key as a consequence of restricting the (global) worst-case access cost for the *entire* set of keys.

☆ This research was supported in part by the Natural Sciences and Engineering Research Council of Canada.
* Corresponding author.
E-mail addresses: will@cs.ubc.ca (W. Evans), kirk@cs.ubc.ca (D. Kirkpatrick).

0196-6774/$ – see front matter
doi:10.1016/S0196-6774(03)00094-4

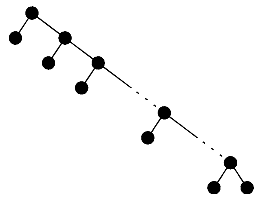


Fig. 1. Tree T in need of restructuring.

In order to clarify the question and (perhaps) strengthen the reader's intuition, the reader is invited to consider the task of restructuring (while preserving the sequence of nodes encountered in an in-order tree traversal) the (arguably most unbalanced) L-leaf tree T in Fig. 1. Obviously, reducing the height of T to $\lceil \lg L \rceil$ (the minimum possible height) would result in a depth increase of $\lceil \lg L \rceil - 1$ for some nodes. On the other hand, T can be restructured to have height $\lceil \lg L \rceil + 1$ without increasing the depth of any node in T by more than one. The fact that this can be done (as the reader, no doubt, has confirmed; otherwise, see Appendix A for a suggestive example) leads to the question: can such restructuring always be (efficiently) done and, more generally, what, if any, tradeoff arises between the global height bound and the depth increase experienced by individual nodes?

The results of this paper are perhaps best appreciated in the context of earlier work on the design and analysis of optimal (and near-optimal) search and coding trees, and their height-restricted variants. Given a sequence of symbols $A_1, \ldots, A_L$ with associated frequencies $a_1, \ldots, a_L$ a *Huffman tree* [5] is an L-leaf tree T, together with an assignment of symbols to the leaves of T, that minimizes the expression $\sum_{i=1}^{L} a_i d_i$, where d_i denotes the depth in T of the leaf to which symbol A_i is assigned. The tree that minimizes $\sum_{i=1}^{L} a_i d_i$ subject to the additional constraint that the symbols $A_1, \ldots, A_L$ are assigned to the leaves of T in left-to-right order is called an *alphabetic Huffman tree* [4]. If symbols (or search keys) are assigned to the internal nodes as well (in an order that respects the in-order listing of T's nodes) then the tree that minimizes $\sum_{i=1}^{L} a_i d_i$, where d_i now denotes the depth of the (possibly internal) node of T to which A_i is assigned, is called an *optimal binary search tree* [8,13]. *Near-optimal binary search trees* for given [12] or fixed but unknown [15] key distributions provide provably good approximations to the minimum expected depth (access cost) with lower construction cost or, as in the case of [15], the ability to adapt to changing access frequencies.

Height-restricted versions of these structures have been extensively studied, particularly with regard to the efficiency of algorithms for their construction [1,6,9–11,14,16]. In particular, it is known that a height-restricted (restricting height to at most h) Huffman tree (respectively, alphabetic Huffman tree, optimal binary search tree) for a set of L symbols can be constructed in $O(Lh)$ (respectively $O(Lh \lg L)$, $O(L^2 h)$) time [10] (respectively [11], [6,16]).

Few papers address the issue of the increase in the access cost imposed by restricting height. E.N. Gilbert, in the paper that introduced the study of length-restricted Huffman codes [2], proves that, for any $h \geqslant 1 + \lg L$, the h height-restricted Huffman tree has expected depth (code-length) at most $H + 1 + L2^{1-h}(\lg L - H)$, where H denotes the entropy of the set of symbol frequencies. Since H provides a lower bound on the expected depth of any (not necessarily height- or order-restricted) Huffman tree (by Shannon's theorem), it follows that the increase in expected depth due to height restriction is at most 1, when h is as small as $\lg L + (\lg\lg L)^{\omega}$, $\omega > 1$ and L is sufficiently large. (In fact it is easy to demonstrate arbitrarily long sequences, with associated frequencies, for which the expected depth of the optimal Huffman tree of height $1 + \lceil \lg L \rceil$ exceeds that of the optimal unconstrained Huffman tree by an amount arbitrarily close to 1.) Milidiú and Laber [14] present an upper bound on the average code length difference between optimal length-restricted and unconstrained Huffman codes. They assume that the symbol frequencies are fixed, known quantities.

By way of contrast, our results, which are set out more completely and formally in the next section, show that *any* L-leaf tree T, with fixed but *unknown* leaf access frequencies, can be restructured into a tree R of height at most $1 + \lceil \lg L \rceil$ such that the expected depth of R exceeds the expected depth of T by at most 2. Much of our effort is devoted to understanding the tradeoff between the height restriction of the restructured trees and the increase in node depth due to restructuring, and to the construction of trees that realize the worst-case depth increase over the full range of possible height restrictions.

2. Strict h-leveling costs

Let $\mathcal{T}_L$ denote the set of rooted, ordered binary trees with L leaves. For $T \in \mathcal{T}_L$, let $v_1^T, v_2^T, \ldots, v_{2L-1}^T$ be the in-order sequence of nodes in T and $\ell_1^T, \ell_2^T, \ldots, \ell_L^T$ the left-to-right sequence of leaves in T. (Of course, $\ell_i^T = v_{2i-1}^T$.) In general, the superscript is omitted when it is clear from context. Let $d_T(v)$ be the depth of node v in T. The depth of the root is 0 and the height of T is just the depth of its deepest leaf.

We define the *h-leveling cost* of $T \in \mathcal{T}_L$ to be

$$\Delta_h^*(T) = \min_{R \in \mathcal{T}_L} \Delta_h^*(T, R) \tag{1}$$

where[1]

$$\Delta_h^*(T, R) = \begin{cases} \infty & \text{if } \mathit{height}(R) > h, \\ \max\limits_{1 \leqslant j \leqslant 2L-1} \left[d_R\left(v_j^R\right) - d_T\left(v_j^T\right) \right] & \text{otherwise.} \end{cases}$$

$\Delta_h^*(T, R)$ measures the maximum "drop," over all nodes in T (the *node-drop*), in going from T to a tree R of height at most h. In other words, $\Delta_h^*(T, R)$ is an upper bound on the additional cost to access an item in R versus T.

We define the *leaf h-leveling cost* of $T \in \mathcal{T}_L$ to be

$$\Delta_h^0(T) = \min_{R \in \mathcal{T}_L} \Delta_h^0(T, R), \tag{2}$$

[1] The multiple definitions of $\Delta_h^*()$ (and similar functions) are intended to minimize notation and highlight the relations between the associated functions. This should present no cause for confusion since the number and type of arguments determine which definition applies.

where

$$\Delta_h^0(T, R) = \begin{cases} \infty & \text{if } height(R) > h, \\ \max\limits_{1 \leqslant j \leqslant L} \left[d_R\left(\ell_j^R\right) - d_T\left(\ell_j^T\right) \right] & \text{otherwise.} \end{cases}$$

$\Delta_h^0(T, R)$ measures the maximum "drop," over all leaves in T (the *leaf-drop*), in going from T to a tree R of height at most h. Clearly, if $h < \lceil \lg L \rceil$ no tree of height at most h can have L leaves, and both $\Delta_h^*(T)$ and $\Delta_h^0(T)$ are infinite.

We seek an algorithm that, given a tree $T \in \mathcal{T}_L$ and target height h, finds a tree R that realizes the minimum in Eq. (1) (or Eq. (2)). In addition, we seek the worst h-leveling and leaf h-leveling cost as a function of the number of leaves in T. Specifically, we seek

$$\Delta_h^*(L) = \max_{T \in \mathcal{T}_L} \Delta_h^*(T) \quad \text{and} \quad \Delta_h^0(L) = \max_{T \in \mathcal{T}_L} \Delta_h^0(T).$$

Explicit tradeoffs that determine both $\Delta_h^*(L)$ and $\Delta_h^0(L)$ are developed in Section 6. Our results are summarized in the following theorems:

Theorem 2.1.

$$\Delta_h^0(L) = \begin{cases} 0 & \text{if } L < h + 2, \\ 1 & \text{if } h + 2 \leqslant L < L_h, \\ 2 & \text{if } L_h \leqslant L \text{ and } \lceil \lg L \rceil < h, \\ \lceil \lg L \rceil - 1 & \text{if } \lceil \lg L \rceil = h, \\ \infty & \text{if } \lceil \lg L \rceil > h, \end{cases}$$

where

$$L_h = \begin{cases} 2^{h-1} + h + 1 & \text{if } 1 \leqslant h \leqslant 4, \\ F_{h+1} + 3F_h - 1 & \text{if } h > 4, \end{cases}$$

and F_k is the kth Fibonacci number ($F_0 = 0$, $F_1 = 1$, and $F_i = F_{i-1} + F_{i-2}$, for $i > 1$).

Among other things, this theorem states that, when $h \geqslant \lceil \lg L \rceil + 1$, restructuring with leaf-drop at most 2 is always possible, and when $h > \log_\phi((L + 1)/(3 + \phi))$, where ϕ is the golden ratio (1.61803...), leaf-drop at most 1 is always possible (and these results are essentially tight). This implies, for example, that an optimal binary search tree (with keys stored at leaves) can be restructured to have worst-case search cost within one of the optimal worst-case search cost ($\lceil \lg L \rceil$) without increasing the average search cost (or, for that matter, the search cost of *any* key) by more than two. Equivalently, an (optimal) Huffman tree for a sequence of symbols can be restructured to give codes of maximum length at most one bit more than the optimal maximum code length and average code length at most two bits more than the optimal average code length. In both cases this can be done without explicit knowledge of the key or symbol frequencies.

Theorem 2.2.

$$\Delta_h^*(L) = \min\left\{k \colon \left\lceil \lg \rho(L, h - k) \right\rceil = k\right\},$$

where $\rho(L, h) = \max\{r \colon \sum_{i=0}^{r-1} \binom{h}{i}(r - i) \leqslant L\}$.

Observation 2.1. *By considering certain ranges of h and using straightforward approximations of the binomial coefficient, we observe*

$$\Delta_h^*(L) \leqslant \begin{cases} 0 & \text{if } L < h+2, \\ \lg k + O(1) & \text{if } L = O(h^k), \\ \lg\lg L & \text{if } \lceil \lg L \rceil < h, \\ \lceil \lg L \rceil - 1 & \text{if } \lceil \lg L \rceil = h, \\ \infty & \text{if } \lceil \lg L \rceil > h. \end{cases}$$

Thus, among other things, Theorem 2.2 implies that, when $h \geqslant \lceil \lg L \rceil + 1$, restructuring with node-drop at most $\lg\lg L$ is always possible, and when $h > \sqrt{L}$ the node-drop is $O(1)$. Hence, even without explicit knowledge of the key frequencies, a given optimal (or near-optimal) binary search tree can be restructured to have worst-case search cost within one of optimal at the expense of an additive increase of at most $\lg\lg L$ in the expected search cost.

3. Near-h-leveling costs

It happens that considering a slightly different cost function results in solutions to our original problems. Define the *near-h-leveling cost* of $T \in \mathcal{T}_L$ to be

$$\Xi_h^*(T) = \min_{R \in \mathcal{T}_L} \Xi_h^*(T, R),$$

where

$$\Xi_h^*(T, R) = \max_{1 \leqslant j \leqslant 2L-1} \left[d_R\left(v_j^R\right) - \min\left\{ d_T\left(v_j^T\right), h \right\} \right].$$

$\Xi_h^*(T)$ differs from $\Delta_h^*(T)$ in that the tree R may have height greater than h. However, each node at depth greater than h in R has cost at least its depth in R minus h. Define the *leaf near-h-leveling cost* of $T \in \mathcal{T}_L$ to be

$$\Xi_h^0(T) = \min_{R \in \mathcal{T}_L} \Xi_h^0(T, R),$$

where

$$\Xi_h^0(T, R) = \max_{1 \leqslant j \leqslant L} \left[d_R\left(\ell_j^R\right) - \min\left\{ d_T\left(\ell_j^T\right), h \right\} \right].$$

We also define

$$\Xi_h^*(L) = \max_{T \in \mathcal{T}_L} \Xi_h^*(T) \quad \text{and} \quad \Xi_h^0(L) = \max_{T \in \mathcal{T}_L} \Xi_h^0(T).$$

4. Relations of h-leveling to near-h-leveling

We can translate bounds on $\Xi_h^*(L)$ into bounds on $\Delta_h^*(L)$ using the following lemma.

Lemma 4.1. *For any pair of trees T and R in $\mathcal{T}_L$, $\Delta_h^*(T,R) = \min\{k\colon\ \Xi_{h-k}^*(T,R) = k\}$ and thus,*

$$\Delta_h^*(L) = \min\{k\colon\ \Xi_{h-k}^*(L) = k\}.$$

Proof. If $\Xi_{h-k}^*(T,R) = k$ then

$$height(R) = \max_{1\leqslant j\leqslant 2L-1} d_R(v_j^R) \leqslant \max_{1\leqslant j\leqslant 2L-1} \left[k + \min\{d_T(v_j^T), h-k\}\right] \leqslant h$$

and

$$\begin{aligned}\Delta_h^*(T,R) &= \max_{1\leqslant j\leqslant 2L-1} \left[d_R(v_j^R) - d_T(v_j^T)\right] \\ &\leqslant \max_{1\leqslant j\leqslant 2L-1} \left[d_R(v_j^R) - \min\{d_T(v_j^T), h-k\}\right] \\ &= k.\end{aligned}$$

If $\Delta_h^*(T,R) = k$ then $height(R) \leqslant h$ and

$$\begin{aligned}\Xi_{h-k}^*(T,R) &= \max_{1\leqslant j\leqslant 2L-1} \left[d_R(v_j^R) - \min\{d_T(v_j^T), h-k\}\right] \\ &= \max_{1\leqslant j\leqslant 2L-1} \max\{d_R(v_j^R) - d_T(v_j^T), d_R(v_j^R) - (h-k)\} \\ &\leqslant \max\{\Delta_h^*(T,R), k\} \\ &= k. \qquad \square\end{aligned}$$

The same proof, restricted to leaf nodes, establishes:

Lemma 4.2. *For any pair of trees T and R in $\mathcal{T}_L$, $\Delta_h^0(T,R) = \min\{k\colon\ \Xi_{h-k}^0(T,R) = k\}$ and thus,*

$$\Delta_h^0(L) = \min\{k\colon\ \Xi_{h-k}^0(L) = k\}.$$

If we have an algorithm Xi*, with inputs T and h, that finds a tree that achieves $\Xi_h^*(T)$ then we can use it to find a tree that achieves $\Delta_h^*(T)$ as follows:

Leveled Tree Algorithm
Input: binary tree $T \in \mathcal{T}_L$, integer height h.
Output: binary tree R with $height(R) \leqslant h$ that minimizes $\Delta_h^*(T,R)$.

1. Find the smallest $0 \leqslant k \leqslant \Delta_h^*(L)$ such that $R = \text{Xi}^*(T, h-k)$ and $\Xi_{h-k}^*(T,R) \leqslant k$.
2. If such k exists then output R.

Correctness. For the sake of contradiction, suppose there exists a tree R' with $height(R') \leqslant h$ such that $k' \equiv \Delta_h^*(T,R') < \Delta_h^*(T,R) \equiv k$. By Lemma 4.1, $\Xi_{h-k'}^*(T,R') \leqslant k'$. This contradicts the minimality of k.

Since $\Xi_h^*(T)$ is monotonically decreasing with increasing h, the smallest k, $0 \leqslant k \leqslant \Delta_h^*(L)$, such that $R = \text{Xi}^*(T, h-k)$ and $\Xi_{h-k}^*(T,R) \leqslant k$ can be found using binary search with $O(\lg(\Delta_h^*(L)))$ invocations of algorithm Xi*.

The preceding algorithm with the superscript "$*$" replaced with "0" performs the analogous task for the leaf-restricted case.

5. The alphabetic minimax tree problem

The near-h-leveling cost of a given tree is related to what is known as the alphabetic minimax cost of an associated weight sequence. If $W = w_1, w_2, \ldots, w_{2L-1}$ is an odd-length sequence of integer weights and $T \in \mathcal{T}_L$ then, for every j, $1 \leqslant j \leqslant 2L-1$, the W-*cost* of node v_j of T is given by

$$c_W(v_j) = \begin{cases} w_j & \text{if } v_j \text{ is a leaf of } T, \\ \max\{w_j, 1 + c_W(v_a), 1 + c_W(v_b)\} & \text{if } v_j \text{ has children } v_a \text{ and } v_b. \end{cases}$$

We define the W-cost of T_j (the subtree of T rooted at node v_j), denoted $c_W(T_j)$, to be $c_W(v_j)$. (So $c_W(T) = c_W(r)$, where r is the root of T.) Since $c_W(T) = \max_{1 \leqslant j \leqslant 2L-1}[d_T(v_j) + w_j]$, $c_W(T)$ can be interpreted as the maximum weighted path length of T, where the path from the root r to v_j is assigned weighted path length $d_T(v_j) + w_j$. The (*full*) *alphabetic minimax cost* of the weight sequence W, denoted $\alpha(W)$, is defined as

$$\alpha(W) = \min_{T \in \mathcal{T}_L} c_W(T).$$

In the event that the even-indexed elements of W (corresponding to internal nodes of T) have values less than or equal to those of all odd-indexed elements (corresponding to the leaves of T), the cost $c_W(T)$ becomes the weighted root-to-leaf path length of T and $\alpha(W)$ gives the *leaf-restricted alphabetic minimax cost* of the subsequence of leaf weights $w_1, w_3, \ldots, w_{2L-1}$.

The *alphabetic minimax problem* takes an odd-length weight sequence W as input and asks for a tree T, an *alphabetic minimax tree* for W, whose W-cost realizes $\alpha(W)$. In fact, we will deal exclusively with *strong* alphabetic minimax trees, in which *every* subtree is an alphabetic minimax tree for its associated weight sequence. The leaf-restricted version of the alphabetic minimax problem has been extensively studied [3,7]; many of the results of this section can be viewed as extensions of earlier leaf-restricted results to the more general problem addressed here.

Observation 5.1 (Optimal substructure). *If T is a* (*strong*) *alphabetic minimax tree for the weight sequence W and if v is any node of T then the tree T', formed by replacing the subtree of T rooted at v by a single leaf, is a* (*strong*) *alphabetic minimax tree for the weight sequence W' formed by replacing the subsequence of weights associated with the nodes in the subtree rooted at v by the weight $c_W(v)$.*

Given a weight sequence $W = w_1, w_2, \ldots, w_{2L-1}$, with $L > 1$, we define the associated *contracted weight sequence* $\widehat{W} = \widehat{w}_0, \widehat{w}_1, \ldots, \widehat{w}_L$ by

$$\widehat{w}_0 = \widehat{w}_L = \infty,$$

and for $1 \leqslant j \leqslant L-1$,

$$\widehat{w}_j = \max\{1 + w_{2j-1}, w_{2j}, 1 + w_{2j+1}\}.$$

It follows from the next lemma that two weight sequences with the same associated contracted weight sequence have the same alphabetic minimax cost.

Lemma 5.1. *Suppose that weight sequences $W = w_1, w_2, \ldots, w_{2L-1}$ and $W' = w'_1, w'_2, \ldots, w'_{2L-1}$ have the same associated contracted weight sequence. Then, for every $T \in \mathcal{T}_L$ and for every internal node v_{2j} of T, $c_W(v_{2j}) = c_{W'}(v_{2j})$.*

Proof. Let $\widehat{W} = \widehat{w}_0, \widehat{w}_1, \ldots, \widehat{w}_L$ be the contracted weight sequence associated with both W and W'. First note that, for every leaf v_{2j-1} of T,

$$c_W(v_{2j-1}) = w_{2j-1} \leqslant \widehat{w}_j - 1 \quad \text{and} \quad c_{W'}(v_{2j-1}) = w'_{2j-1} \leqslant \widehat{w}_j - 1.$$

We prove that $c_W(v_{2j}) = c_{W'}(v_{2j})$, for every internal node v_{2j} of T, by induction on $ht(v_{2j})$, the height of the subtree of T rooted at v_{2j}.

If $ht(v_{2j}) = 1$ then

$$\begin{aligned} c_W(v_{2j}) &= \max\{w_{2j}, 1 + w_{2j-1}, 1 + w_{2j+1}\} = \widehat{w}_j \\ &= \max\bigl\{w'_{2j}, 1 + w'_{2j-1}, 1 + w'_{2j+1}\bigr\} = c_{W'}(v_{2j}). \end{aligned}$$

Suppose that $ht(v_{2j}) = h$ and $c_W(v_k) = c_{W'}(v_k)$ for all internal nodes v_k with $ht(v_k) < h$. Let v_a and v_b denote the left and right children of v_{2j} in T. Then,

$$\begin{aligned} c_W(v_{2j}) &= \max\bigl\{w_{2j}, 1 + c_W(v_a), 1 + c_W(v_b)\bigr\} \\ &= \max\bigl\{w_{2j}, 1 + w_{2j-1}, 1 + w_{2j+1}, 1 + c_W(v_a), 1 + c_W(v_b)\bigr\} \\ &\qquad \text{since } c_W(v_a) \geqslant w_{2j-1} \text{ and } c_W(v_b) \geqslant w_{2j+1} \\ &= \max\bigl\{\widehat{w}_j, 1 + c_W(v_a), 1 + c_W(v_b)\bigr\} \quad \text{by definition of } \widehat{w}_j \\ &= \max\bigl\{\widehat{w}_j, 1 + c_{W'}(v_a), 1 + c_{W'}(v_b)\bigr\} \quad (*) \\ &= \max\bigl\{w'_{2j}, 1 + w'_{2j-1}, 1 + w'_{2j+1}, 1 + c_{W'}(v_a), 1 + c_{W'}(v_b)\bigr\} \\ &\qquad \text{by definition of } \widehat{w}_j \\ &= \max\bigl\{w'_{2j}, 1 + c_{W'}(v_a), 1 + c_{W'}(v_b)\bigr\} \\ &\qquad \text{since } c_{W'}(v_a) \geqslant w'_{2j-1} \text{ and } c_{W'}(v_b) \geqslant w'_{2j+1} \\ &= c_{W'}(v_{2j}). \end{aligned}$$

If v_a and v_b are internal nodes, line $(*)$ follows by the induction hypothesis. If v_a (respectively v_b) is a leaf, line $(*)$ follows since $\widehat{w}_j \geqslant 1 + c_W(v_a)$ and $\widehat{w}_j \geqslant 1 + c_{W'}(v_a)$ (respectively $\widehat{w}_j \geqslant 1 + c_W(v_b)$ and $\widehat{w}_j \geqslant 1 + c_{W'}(v_b)$).

It follows, by induction, that $c_W(v_{2j}) = c_{W'}(v_{2j})$, for every internal node v_{2j} of T. □

If $\widehat{w}_j$ satisfies $\widehat{w}_{j-1} \geqslant \widehat{w}_j < \widehat{w}_{j+1}$, then the triple $(w_{2j-1}, w_{2j}, w_{2j+1})$ is called a *right locally minimum triple* (abbreviated r.l.m. triple) in W. The following two local replacement operations on weight sequences can be used iteratively to reduce an arbitrary odd-length weight sequence W to a weight sequence consisting of the single element $\alpha(W)$.

Contraction: of a right locally minimum triple. A r.l.m. triple $(w_{2j-1}, w_{2j}, w_{2j+1})$ in W is replaced by the single weight $\widehat{w}_j$.
Normalization: of weights. An internal weight w_{2j} is replaced by $w_{2j}+1$ if $w_{2j} < \widehat{w}_j$. A leaf weight w_{2j-1} is replaced by $w_{2j-1}+1$ if $w_{2j-1} < \min\{\widehat{w}_{j-1}, \widehat{w}_j\} - 1$.

Lemma 5.2. *If W' is formed from W by normalization or contraction then $\alpha(W') = \alpha(W)$.*

Proof. First consider the case where weight sequence W' is formed from W by normalization of any subset of its weights. Then, for every j, $1 \leqslant j \leqslant |W|$,

$$\begin{aligned}\widehat{w}'_j &= \max\bigl\{1 + w'_{2j-1}, w'_{2j}, 1 + w'_{2j+1}\bigr\}\\ &\leqslant \max\bigl\{\min\{\widehat{w}_{j-1}, \widehat{w}_j\}, \widehat{w}_j, \min\{\widehat{w}_j, \widehat{w}_{j+1}\}\bigr\} = \widehat{w}_j\end{aligned}$$

and

$$\begin{aligned}\widehat{w}'_j &= \max\bigl\{1 + w'_{2j-1}, w'_{2j}, 1 + w'_{2j+1}\bigr\} \geqslant \max\{1 + w_{2j-1}, w_{2j}, 1 + w_{2j+1}\}\\ &= \widehat{w}_j.\end{aligned}$$

It follows that W' and W have the same associated contracted weight sequence, and hence, by Lemma 5.1, $\alpha(W') = \alpha(W)$.

Next, consider the case where the weight sequence W' is formed from W by contraction of the r.l.m. triple $(w_{2j-1}, w_{2j}, w_{2j+1})$. Suppose that $|W| = 2L - 1$. If T' is any tree in $\mathcal{T}_{L-1}$ then the tree T formed from T' by attaching a pair of leaves to the jth leaf of T' satisfies $c_W(T) = c_{W'}(T')$, and hence, taking T' to be the tree that realizes the minimum $c_{W'}(T')$, it follows that $\alpha(W') \geqslant \alpha(W)$. Conversely, let T be any tree in $\mathcal{T}_L$. It suffices to show that for some tree T' in $\mathcal{T}_{L-1}$, $c_{W'}(T') \leqslant c_W(T)$, and hence $\alpha(W') \leqslant \alpha(W)$.

Suppose first, that nodes v_{2j-1} and v_{2j+1} are siblings in T. Then, $c_W(v_{2j}) = \widehat{w}_j$ and so the tree T' formed from T by removing nodes v_{2j-1} and v_{2j+1} satisfies $c_{W'}(T') = c_W(T)$. Hence, it suffices to show that there exists a tree E in $\mathcal{T}_L$, in which leaves v^E_{2j-1} and v^E_{2j+1} are siblings, that satisfies $c_W(E) \leqslant c_W(T)$. We proceed by induction on L. If $L = 2$ there is nothing to show, so suppose that $L > 2$ and that the hypothesis holds for trees with fewer than L leaves.

Let v_r be the root of T and denote by T^L and T^R the left and right subtrees of v_r. If $v_{2j} \neq v_r$ then nodes v_{2j-1}, v_{2j} and v_{2j+1} lie together in either T^L or T^R. By the induction hypothesis there exist trees E^L and E^R, with nodes $v^E_1, \ldots, v^E_{r-1}$ and $v^E_{r+1}, \ldots, v^E_{2L-1}$, respectively, in which v^E_{2j-1} and v^E_{2j+1} are siblings and $c_{W^L}(E^L) \leqslant c_{W^L}(T^L)$ and $c_{W^R}(E^R) \leqslant c_{W^R}(T^R)$, where $W^L = w_1, \ldots, w_{r-1}$ and $W^R = w_{r+1}, \ldots, w_{2L-1}$. Hence, the tree E with root v_r and subtrees E^L and E^R satisfies

$$\begin{aligned}c_W(E) &= \max\bigl\{w_r, 1 + c_{W^L}\bigl(E^L\bigr), 1 + c_{W^R}\bigl(E^R\bigr)\bigr\}\\ &\leqslant \max\bigl\{w_r, 1 + c_{W^L}\bigl(T^L\bigr), 1 + c_{W^R}\bigl(T^R\bigr)\bigr\} = c_W(T).\end{aligned}$$

So, it remains to consider the situation where $L > 2$ and v_{2j} is the root of T. We consider two cases:

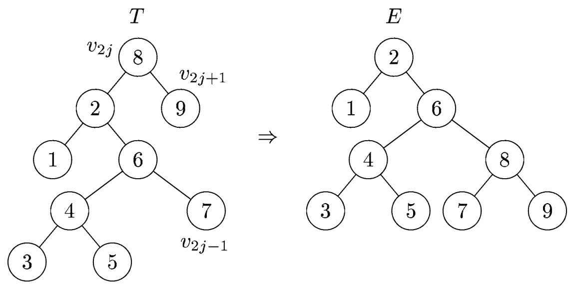


Fig. 2. (Case 1) v_{2j} is root and $j = L - 1$.

Case 1. v_{2j+1} is the right child of v_{2j} in T; equivalently, $j = L - 1$.

In this case, we construct E from T by removing the root (v_{2j}) and its right child (v_{2j+1}) and adding a left and right child to the leaf v_{2j-1} (see Fig. 2). It is straightforward to confirm that $d_E(v_k^E) < d_T(v_k)$, for $k < 2L - 3$, and $d_E(v_{2L-3}^E) = d_E(v_{2L-1}^E) = d_E(v_{2L-2}^E) + 1 = d_T(v_{2L-4}) + 1$. Hence,

$$\max_{1 \leqslant k < 2L-3} \left[d_E\left(v_k^E\right) + w_k \right] < \max_{1 \leqslant k < 2L-3} \left[d_T(v_k) + w_k \right].$$

Furthermore,

$$\begin{aligned}
&\max\left\{ d_E\left(v_{2L-3}^E\right) + w_{2L-3}, d_E\left(v_{2L-2}^E\right) + w_{2L-2}, d_E\left(v_{2L-1}^E\right) + w_{2L-1} \right\} \\
&\quad = d_E\left(v_{2L-2}^E\right) + \max\{w_{2L-2}, 1 + w_{2L-3}, 1 + w_{2L-1}\} \\
&\quad \leqslant d_E\left(v_{2L-2}^E\right) + \max\{w_{2L-4}, 1 + w_{2L-5}, 1 + w_{2L-3}\} \\
&\qquad\quad \text{since } (w_{2L-3}, w_{2L-2}, w_{2L-1}) \text{ is a r.l.m. triple} \\
&\quad = d_T(v_{2L-4}) + \max\{w_{2L-4}, 1 + w_{2L-5}, 1 + w_{2L-3}\} \\
&\quad \leqslant \max\left\{ d_T(v_{2L-5}) + w_{2L-5}, d_T(v_{2L-4}) + w_{2L-4}, d_T(v_{2L-3}) + w_{2L-3} \right\}.
\end{aligned}$$

Hence,

$$c_W(E) = \max_{1 \leqslant k \leqslant 2L-1} \left[d_E\left(v_k^E\right) + w_k \right] \leqslant \max_{1 \leqslant k \leqslant 2L-1} \left[d_T(v_k) + w_k \right] = c_W(T).$$

Case 2. v_{2j+1} is not the right child of v_{2j} in T; equivalently, $j < L - 1$.

In this case, we construct E from T by removing leaf v_{2j-1}, replacing v_{2j-2} (assuming $j > 1$) by its left subtree (if $j = 1$ then the root v_2 is simply removed), and adding a left and right child to leaf v_{2j+1} (see Fig. 3). It is straightforward to confirm that $d_E(v_k^E) \leqslant d_T(v_k)$, for $k \leqslant 2j - 2$ or $k \geqslant 2j + 2$, and $d_E(v_{2j-1}^E) = d_E(v_{2j+1}^E) = d_E(v_{2j}^E) + 1 \leqslant d_T(v_{2j+2}) + 2$. Hence,

$$\max_{\substack{k \leqslant 2j-2 \\ \text{or } k \geqslant 2j+2}} \left[d_E\left(v_k^E\right) + w_k \right] \leqslant \max_{\substack{k \leqslant 2j-2 \\ \text{or } k \geqslant 2j+2}} \left[d_T(v_k) + w_k \right].$$

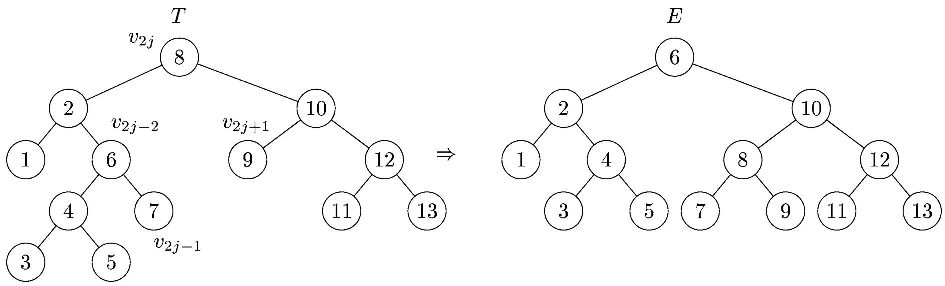


Fig. 3. (Case 2) v_{2j} is root and $j < L - 1$.

Furthermore,

$$\begin{aligned}
&\max\bigl\{d_E\bigl(v^E_{2j-1}\bigr) + w_{2j-1}, d_E\bigl(v^E_{2j}\bigr) + w_{2j}, d_E\bigl(v^E_{2j+1}\bigr) + w_{2j+1}\bigr\} \\
&\quad = d_E\bigl(v^E_{2j}\bigr) + \max\{w_{2j}, 1 + w_{2j-1}, 1 + w_{2j+1}\} \\
&\quad \leqslant d_E\bigl(v^E_{2j}\bigr) + \max\{w_{2j+2}, 1 + w_{2j+1}, 1 + w_{2j+3}\} - 1 \\
&\qquad\quad \text{since } (w_{2j-1}, w_{2j}, w_{2j+1}) \text{ is a r.l.m. triple} \\
&\quad \leqslant d_T(v_{2j+2}) + \max\{w_{2j+2}, 1 + w_{2j+1}, 1 + w_{2j+3}\} \\
&\quad \leqslant \max\bigl\{d_T(v_{2j+1}) + w_{2j+1}, d_T(v_{2j+2}) + w_{2j+2}, d_T(v_{2j+3}) + w_{2j+3}\bigr\}.
\end{aligned}$$

Hence,

$$c_W(E) = \max_{1 \leqslant k \leqslant 2L-1} \bigl[d_E\bigl(v^E_k\bigr) + w_k\bigr] \leqslant \max_{1 \leqslant k \leqslant 2L-1} \bigl[d_T(v_k) + w_k\bigr] = c_W(T). \qquad \square$$

Normalization of weights does not change the length of the weight sequence W, nor does it change the cost $\widehat{w}_j$ associated with any triple $(w_{2j-1}, w_{2j}, w_{2j+1})$; in particular it does not change any r.l.m. triple of W. Normalization serves as a tool for maintaining a structural invariant of the weight sequences that we will encounter in our applications. On the other hand, each application of contraction reduces the length of the weight sequence W by 2. Each such application also corresponds to a join operation in the construction of an alphabetic minimax tree associated with W. In particular, if T' is an alphabetic minimax tree for the weight sequence W' formed from W by contraction of the r.l.m. triple $(w_{2j-1}, w_{2j}, w_{2j+1})$, and the tree T is formed from T' by adding two leaves to the node $v^{T'}_{2j-1}$, then $c_W(T) = c_{W'}(T') = \alpha(W') = \alpha(W)$. Hence, T is an alphabetic minimax tree for W. Thus the construction of an alphabetic minimax tree for W is implicit in any algorithm for reducing the sequence W to a singleton sequence by application of normalization and contraction operations.

Theorem 5.1. *The alphabetic minimax problem has a linear time solution.*

Proof. Successive r.l.m. triples in a given weight sequence can be located and contracted in amortized constant time using a simple stack-driven procedure. This is a direct generalization of previously presented linear-time leaf-restricted alphabetic minimax tree algorithms (cf. [7]). □

The iterative algorithm implicit in the proof of the above theorem uses contraction operations only. It not only produces the alphabetic minimax cost $\alpha(W)$ of a given weight sequence W, but also lends itself to the construction of an upper bound on $\alpha(W)$ as a function of the multiset of weights in W. Specifically, if $W = w_1, \dots, w_{2L-1}$ and $\widehat{W} = \widehat{w}_0, \dots, \widehat{w}_L$ is the associated contracted weight sequence then we can define $\Psi(W) = \sum_{i=1}^{L-1} 2^{\widehat{w}_i}$. If W_0 denotes the initial weight sequence and W_f denotes the (length 3) weight sequence prior to the final contraction, then it is straightforward to confirm that:

(1) $\Psi(W_0) < 4\sum_{w \in W_0} 2^w$,
(2) $\Psi(W_f) = 2^{\alpha(W_0)}$, and
(3) if sequence W' is formed from W by contraction then $\Psi(W') \leqslant \Psi(W)$.

As an immediate consequence we have the following:

Lemma 5.3. *If W_0 is any weight sequence then $\alpha(W_0) < 2 + \lg \sum_{w \in W_0} 2^w$.*

6. Applications to tree restructuring

The motivation for developing results about alphabetic minimax trees is their close connection with near-h-leveling costs of binary trees.

For $T \in \mathcal{T}_L$, we define the *h-leveled weight sequence* associated with T to be the sequence $w_1, \dots, w_{2L-1}$, where

$$w_j = -\min\{d_T(v_j^T), h\}.$$

Similarly, we define the *h-leveled leaf-restricted weight sequence* associated with T to be the sequence $w_1, \dots, w_{2L-1}$, where

$$w_j = \begin{cases} -\min\{d_T(v_j^T), h\} & \text{if } j \text{ is odd,} \\ -h & \text{if } j \text{ is even.} \end{cases}$$

Lemma 6.1. *For $T \in \mathcal{T}_L$, if W is the h-leveled weight sequence associated with T and W' is the h-leveled leaf-restricted weight sequence associated with T, then $\Xi_h^*(T) = \alpha(W)$ and $\Xi_h^0(T) = \alpha(W')$.*

Proof. From the definitions of $\Xi_h^*(T)$, w_j, $c_W(R)$, and $\alpha(W)$:

$$\begin{aligned} \Xi_h^*(T) &= \min_{R \in \mathcal{T}_L} \max_{1 \leqslant j \leqslant 2L-1} \left[d_R(v_j^R) - \min\{d_T(v_j^T), h\}\right] \\ &= \min_{R \in \mathcal{T}_L} \max_{1 \leqslant j \leqslant 2L-1} \left[d_R(v_j^R) + w_j\right] = \min_{R \in \mathcal{T}_L} c_W(R) = \alpha(W). \end{aligned}$$

The leaf-restricted case is similar, since for h-leveled leaf-restricted weight sequences $c_W(R)$ is realized by one of its leaves. □

Lemma 6.1 allows us to re-express our results on alphabetic minimax trees from Section 5 in terms of the near-h-leveling of an arbitrary tree T. Specifically, using Lemma 6.1 together with Theorem 5.1, we obtain

Corollary 6.1. *For $T \in \mathcal{T}_L$, both $\Xi_h^*(T)$ and $\Xi_h^0(T)$ (and their realizations) can be determined in time linear in L. Furthermore, using the Leveled Tree Algorithm, $\Delta_h^*(T)$ (respectively $\Delta_h^0(T)$), together with its realization, can be determined in $O(L\lg(\Delta_h^*(L)))$ (respectively $O(L\lg(\Delta_h^0(L)))$) time.*

Furthermore, using Lemma 6.1 together with Lemma 5.3 we obtain

Corollary 6.2. *If $h \geqslant \lceil \lg L \rceil + 1$ then $\Xi_h^*(L) < 2 + \lg(2 + \lg L)$ and $\Xi_h^0(L) \leqslant 2$.*

Referring to Corollary 6.1, it is natural to ask if $\Delta_h^*(T)$ and $\Delta_h^0(T)$ can also be determined in time linear in L. The algorithm described above determines $\Delta_h^0(T)$ in linear time (since $\Delta_h^0(L)$ is constant) except for $h = \lceil \lg L \rceil$ when a different linear time algorithm exists. For $\Delta_h^*(T)$, the existence of a linear time algorithm is an open problem.

The bounds on $\Xi_h^*(L)$ and $\Xi_h^0(L)$ presented in Corollary 6.2 are not tight. To derive exact values for $\Xi_h^*(L)$ and $\Xi_h^0(L)$ (and hence $\Delta_h^*(L)$ and $\Delta_h^0(L)$) it turns out to be useful to reformulate the iterative alphabetic minimax algorithm of the preceding section. This less efficient but more structured alternative makes explicit use of normalization and more constrained applications of contraction. At its core is a reduction procedure that serves to incrementally reduce a measure of the spread of weights in a given sequence while preserving its alphabetic minimax cost. When this measure is reduced to zero the structure of the resulting sequence, and hence its alphabetic minimax cost, is completely determined.

If $W = w_1, \ldots, w_{2k-1}$ is a weight sequence, denote by $\widetilde{W}$ the set

$$\widetilde{W} = \{w_j + 1 \mid j \text{ odd}\} \cup \{w_j \mid j \text{ even}\}.$$

The *thickness* of W, denoted $\theta(W)$, is defined as $\theta(W) = \max(\widetilde{W}) - \min(\widetilde{W})$.

Reduce

Input: weight sequence W with $|W| = 2k - 1$

Output: weight sequence W' with $\alpha(W') = \alpha(W)$ and either

i) $\theta(W') = \theta(W) - 1$ or

ii) $\theta(W') = \theta(W) = 0$ and $|W'| = 2\lceil k/2 \rceil - 1$

if $|W| = 1$ then output W

$a \leftarrow \min(\widetilde{W})$

while there exists a r.l.m. triple $(w_{2j-1}, w_{2j}, w_{2j+1})$ in W with $\widehat{w}_j = a$

    contract$(w_{2j-1}, w_{2j}, w_{2j+1})$

while there exists a weight w_{2j-1} in W with $w_{2j-1} = a - 1$

    normalize(w_{2j-1})

while there exists a weight w_{2j} in W with $w_{2j} = a$

    (w_{2j})

output W

The correctness of the procedure *reduce* is an immediate consequence of the following three lemmas.

Lemma 6.2. $\alpha(W) = \alpha(reduce(W))$.

Proof. The lemma follows from Lemma 5.2 since *reduce* alters W only by applications of contraction and normalization. □

Lemma 6.3. *If* $\theta(W) > 0$ *then* $\theta(reduce(W)) = \theta(W) - 1$.

Proof. Let $W' = reduce(W)$. The definitions of $\widetilde{W}$ and $\widehat{w}_j$ insure that a triple $(w_{2j-1}, w_{2j}, w_{2j+1})$ contracted in line 4 has $w_{2j-1} = w_{2j+1} = a - 1$ and $w_{2j} = a$. After the contract loop, $\widehat{w}_j > a$ for all j. Thus, during the leaf normalization loop (line 5), any w_{2j-1} with value $a - 1$ will normalize to value a. For the same reason, during the internal normalization loop (line 7), any w_{2j} with value a will normalize to value $a + 1$. Thus, $\min(\widetilde{W}') = \min(\widetilde{W}) + 1$. Since $\theta(W) > 0$, $\max(\widetilde{W}) \geqslant a + 1$ and $\max(\widetilde{W}') = \max(\widetilde{W})$. □

Lemma 6.4. *Let* $|W| = 2k - 1$ *for* $k > 1$. *If* $\theta(W) = 0$ *and* $W' = reduce(W)$ *then* $|W'| = 2\lceil k/2 \rceil - 1$ *and* $\min(\widetilde{W}') = 1 + \min(\widetilde{W})$.

Proof. Let $b = \min(\widetilde{W})$, $a = b - 1$, $t = \lceil k/2 \rceil$, and $r = 2t - k$ (r is either 0 or 1). Since $\theta(W) = 0$, it follows that W has the form

$$a(ba)^{k-1} = a(ba)^{2t-r-1} = a(ba)^{1-r}\big(b(aba)\big)^{t-1}.$$

By the definitions of r.l.m. triple and contraction, after $t - 1$ repetitions of the contract loop (line 3), W has the form

$$a(ba)^{1-r}(bb)^{t-1}.$$

After $1 - r$ more repetitions of the contract loop and r repetitions of the leaf normalization loop (line 5), W has the form

$$b(bb)^{t-1}.$$

After $t - 1$ repetitions of the internal normalization loop (line 7), we are left with a sequence of the form

$$b\big((b+1)b\big)^{t-1}.$$

As a direct consequence of Lemma 6.4, we have

Corollary 6.3. *If* $\theta(W) = 0$ *and* $|W| = 2k - 1$ $(k \geqslant 1)$ *then* $\alpha(W) = \lceil \lg k \rceil + \min(\widetilde{W}) - 1$.

Lemmas 6.2 and 6.3 and Corollary 6.3 combine to motivate the notion of a *reduction sequence* W_p, $p \geqslant 0$, associated with a weight sequence W, defined as

$$W_0 = W \quad \text{and} \quad W_p = reduce(W_{p-1}) \quad \text{for } p > 0.$$

We refer to each application of procedure *reduce* as a *phase*, and the algorithm that takes a weight sequence W as input and produces a reduction sequence terminating in a sequence of length one as the Phased Alphabetic Minimax Algorithm. In fact, Corollary 6.3 shows that the alphabetic minimax cost of W can be directly determined as soon as W_p has thickness zero. An example of the reduction sequence associated with an h-leveled weight sequence is shown in Fig. 9.

If the input weight sequence W happens to be an h-leveled weight sequence associated with a tree $T \in \mathcal{T}_L$ then its special structure can be exploited to derive tight bounds on its alphabetic minimax cost. For example, if $h \geqslant \mathit{height}(T)$ then $w_j = -d_T(v_j^T)$ and it is easy to confirm that the jth element of W_p is just $-d_{T'}(v_j^{T'})$, where T' denotes the tree formed from T by truncating at depth $\mathit{height}(T) - p$ (i.e., removing all deeper nodes). It follows that $\alpha(W) = 0$, and hence $\Xi_h^*(T) = \Xi_h^0(T) = 0$, whenever $h \geqslant \mathit{height}(T)$. Thus, $\Xi_h^*(L) = \Xi_h^0(L) = 0$ when $h \geqslant L - 1$. On the other hand, if $h \leqslant 1 < \mathit{height}(T)$ then $\theta(W) = 1$ and W_1 has the form $(-h)((-h+1)(-h))^{L-1}$. Hence, $\alpha(W) = \lceil \lg L \rceil - h$ and $\Xi_h^*(T) = \Xi_h^0(T) = \lceil \lg L \rceil - h$. Thus, $\Xi_h^*(L) = \Xi_h^0(L) = \lceil \lg L \rceil - h$, when $h \leqslant 1 < \lceil \lg L \rceil$.

We summarize the above discussion in the following:

Observation 6.1.

$$\Xi_h^*(L) = \Xi_h^0(L) = \begin{cases} 0 & \text{if } L < h+2, \\ \lceil \lg L \rceil - h & \text{if } h \leqslant 1 < \lceil \lg L \rceil. \end{cases}$$

It remains to consider h-leveled weight sequences with $1 < h < \mathit{height}(T)$. In this case $\theta(W) = h$ (since one internal node has associated weight 0 and at least one internal node has associated weight h). In the reduction sequence associated with W, the sequence W_p has thickness $h - p$, for $0 \leqslant p \leqslant h$, and the sequence W_h has the form $(-1)(0(-1))^k$, for some $k > 0$.

We are able to construct worst-case families of trees that provide exact bounds for $\Xi_h^0(L)$ and $\Xi_h^*(L)$ over the full range of possible height bounds h. These results are developed in Sections 6.1 and 6.2 for the case of leaf-h-leveling cost and general h-leveling costs, respectively.

6.1. Leaf-restricted weight sequences

Suppose $T \in \mathcal{T}_L$ and let W be the h-leveled leaf-restricted weight sequence for T, where $h > 1$. Let W_p denote the pth sequence in the reduction sequence associated with W. Let U_p denote the subsequence of W_p corresponding to the original leaf weights (i.e., the odd-indexed elements). It is easy to confirm that the remaining (even-indexed) elements of W_p all have value $-h + p$. We analyse the effect of applying the phased reduction of W in terms of the potential function Φ, where

$$\Phi(U_i) = \sum_{u \in U_i} 2^u.$$

Since W_h has the form $(-1)(0(-1))^k$, for some $k > 0$, U_h has the form $(-1)^{k+1}$. It follows that $\Phi(U_h) = |U_h|/2$ and, by Corollary 6.3,

$$\alpha(W) = \lceil \lg |U_h| \rceil - 1 = \lceil \lg \lfloor 2\Phi(U_h) \rfloor \rceil - 1. \tag{3}$$

To determine $\Phi(U_h)$, we determine the potential $\Phi(U_1)$ and the potential increase $\Phi(U_h) - \Phi(U_1)$. Since the first phase in the reduction sequence associated with W affects only the even-indexed elements (changing their value from $-h$ to $-h+1$), $U_1 = U_0$. Hence, by several applications of the Kraft equality, we have

$$\begin{aligned}\Phi(U_1) = \Phi(U_0) &= \sum_{u \in U_0} 2^u = \sum_{k:\, d_T(\ell_k) \leqslant h} 2^{-d_T(\ell_k)} + \sum_{k:\, d_T(\ell_k) > h} 2^{-h} \\ &= \sum_{1 \leqslant k \leqslant L} 2^{-d_T(\ell_k)} + \sum_{k:\, d_T(\ell_k) > h} \left(2^{-h} - 2^{-d_T(\ell_k)}\right) = 1 + \left(L - \lambda\left(\underline{T}_h\right)\right) 2^{-h},\end{aligned}$$

where, for any binary tree R, $\lambda(R)$ denotes the number of leaves of R and $\underline{R}_h$ denotes the tree formed from R by truncation at depth h (removing all deeper nodes).

The potential difference $\Phi(U_h) - \Phi(U_1)$ is bounded by looking at the potential increase associated with each phase. First, note that each application of contraction in phase $p > 1$ (phase p creates W_p) has no impact on the potential (since it replaces two successive elements of value $-h + p - 2$ in U_p by one new element of value $-h + p - 1$). Each application of (leaf) normalization in phase $p > 1$ replaces an element of value $-h + p - 2$ by one of value $-h + p - 1$ thereby raising the potential of W_p by 2^{-h+p-2}. We assign the potential increase associated with each such normalized weight u_i, with $i > 1$, to its predecessor u_{i-1} in U_p (which must have value greater than u_i, since by assumption u_{i-1} does not belong to a r.l.m. triple, and must correspond to a weight in the original sequence U_1). The potential increase associated with weight u_1 is accumulated in a separate boundary total. We observe that an initial weight u_j could be assigned potential increases in phases $h + u_{j+1} + 2, \ldots, h + u_j + 1$, if initial weight u_{j+1} is less than u_j, totalling at most

$$\sum_{t=h+u_{j+1}+2}^{h+u_j+1} 2^{-h+t-2} = 2^{u_j} - 2^{u_{j+1}}.$$

Furthermore, the boundary total could receive increments in phases $h + u_1 + 2, \ldots, h$ totalling at most

$$\sum_{t=h+u_1+2}^{h} 2^{-h+t-2} = 1/2 - 2^{u_1}.$$

Thus, if $U_1 = u_1, \ldots, u_L$, the total potential increase $\Phi(U_h) - \Phi(U_1)$ is at most

$$1/2 - 2^{u_1} + \sum_{1 \leqslant j < L} \left[2^{u_j} \dot{-} 2^{u_{j+1}}\right]. \tag{4}$$

Note that the monus operation, $\dot{-}$, is defined as $x \dot{-} y = \max\{x - y, 0\}$. Since $u_j = -\min\{d_T(\ell_j^T), h\}$, this bound on the total potential increase can be re-expressed as

$$1/2 - 2^{-d_R(\ell_1^R)} + \sum_{1 \leqslant j < \lambda(R)} \left[2^{-d_R(\ell_j^R)} \dot{-} 2^{-d_R(\ell_{j+1}^R)}\right], \tag{5}$$

where $R = \underline{T}_h$ (the truncation of T at level h). Notice that the summation in (4) is over the leaves of T while in (5) it is over the, typically fewer, leaves of R. Equations (4) and (5) are equivalent since consecutive $-h$ values in the leaf h-leveled weight sequence U_1 (i.e., where $d_T(\ell_j^T) \geqslant h$) make no contribution to the potential increase.

It follows that

$$\begin{aligned} \Phi(U_h) &= \Phi(U_1) + \left[\Phi(U_h) - \Phi(U_1)\right] \\ &\leqslant 1 + \big(L - \lambda(R)\big)2^{-h} + 1/2 - 2^{-d_R(\ell_1^R)} \\ &\quad + \sum_{1 \leqslant j < \lambda(R)} \left[2^{-d_R(\ell_j^R)} \dot{-} 2^{-d_R(\ell_{j+1}^R)}\right] \end{aligned}$$

where $R = \underline{T}_h$.

So, to establish an upper bound on $\Phi(U_h)$ as a function of L and h, it will suffice to determine, among *all* trees R of height h, one that maximizes the function $f(R)$ given by

$$f(R) = -2^{x_1} - \lambda(R)2^{-h} + \sum_{1 \leqslant j < \lambda(R)} \left[2^{x_j} \dot{-} 2^{x_{j+1}}\right], \tag{6}$$

where $x_j = -d_R(\ell_j^R)$.

We begin by showing that any tree R that maximizes $f(R)$ can be assumed to have a particular structure. Two recursively defined families of trees play a role in this argument. A k-chain is defined as follows: a 0-chain is just an isolated vertex and, for $k > 0$, a k-chain is the binary tree whose left subtree is a singleton leaf and whose right subtree is a $(k-1)$-chain. A *Fibonacci tree* of height $k \geqslant 0$, $\mathcal{F}_k$, is defined as follows: for $0 \leqslant k \leqslant 4$, $\mathcal{F}_k$ is just a k-chain and, for $k > 4$, $\mathcal{F}_k$ is the binary tree whose left subtree is $\mathcal{F}_{k-1}$ and whose right subtree has a leaf as its left subtree and $\mathcal{F}_{k-2}$ as its right subtree. Figure 4 illustrates this construction. A tree of height h is said to be *k-fringed* if each of its subtrees rooted at a node of depth $h - k$ is either a singleton node (leaf) or a k-chain. (Note that every binary tree is 0-fringed, every tree of height $h < k$ is (trivially) k-fringed, and every Fibonacci tree of height $h \geqslant 4$ is 4-fringed.)

By straightforward induction on h, we get the following:

Lemma 6.5. *For* $0 \leqslant h \leqslant 3$, $f(\mathcal{F}_h) = -(h+2)2^{-h}$, *and for all* $h \geqslant 4$, $f(\mathcal{F}_h) = 1 - (F_{h+2} + 3F_{h+1} - 1)2^{-h}$, *where* F_k *is the* k*th Fibonacci number* ($F_0 = 0, F_1 = 1, \ldots$).

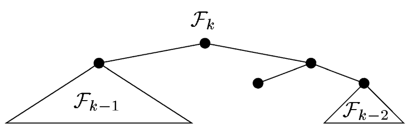


Fig. 4. A Fibonacci tree of height $k > 4$.

We first observe that, for $0 \leqslant h \leqslant 3$, $\mathcal{F}_h$ maximizes $f(R)$, and, for every $h \geqslant 4$, there is a 4-fringed tree that maximizes $f(R)$. (This is proved by demonstrating, for i running from 2 to 4, that if there exists an $(i-1)$-fringed tree that maximizes $f(R)$ then there is an i-fringed tree that maximizes $f(R)$. In each step the rightmost node at level $h-i$ that is neither a leaf nor the root of an i-chain, while such a node exists, has its subtree replaced by an i-chain. It suffices to observe that each such replacement does not decrease $f(R)$.) Thus,

$$\Phi(U_h) \leqslant 3/2 + (L-h-2)2^{-h} \tag{7}$$

when $1 \leqslant h \leqslant 3$.

Continuing with $h \geqslant 4$, we observe that if R is a 4-fringed tree that maximizes $f(R)$ then the tree $S = \underline{R}_{h-4}$ (formed by removal of the 4-fringe) can be assumed to satisfy the additional three properties:

(1) $\min\{y_j, y_{j+1}\} = -h+4$ for $1 \leqslant j < \lambda(S)$,
(2) $y_1 = -h+4$, and
(3) $y_{\lambda(S)} = -h+4$,

where $y_j = -d_S(\ell_j^S)$.

Suppose that S has two consecutive shallow (depth less than $h-4$) leaves, or a shallow first or last leaf. Let ℓ^* denote the deeper of a consecutive pair of shallow leaves (or a shallow first or last leaf) and let k denote the depth of ℓ^* in S. (Note that ℓ^* must be a leaf of R as well.) It is straightforward to confirm that the tree R' formed from R by replacing leaf ℓ^* by the subtree $\mathcal{F}_{h-k}$ satisfies

$$f(R') \geqslant f(R) + 2^{-k} f(\mathcal{F}_{h-k}) + 2^{-h} + 2^{-h+3} > f(R).$$

The first inequality expresses the change in Eq. (6) due to the removal of ℓ^* (2^{-h}) and its replacement by $\mathcal{F}_{h-k}$ ($2^{-k} f(\mathcal{F}_{h-k})$). The value 2^{-h+3} represents the minimum additional contribution to $f(R')$ arising from the first leaf in $\mathcal{F}_{h-k}$. The second inequality is equivalent to $f(\mathcal{F}_{h-k}) > -9 \cdot 2^{-h+k}$ which follows from Lemma 6.5.

A tree R is said to be an *h-comb* for $h \geqslant 4$ if R is a k-chain, for $0 \leqslant k \leqslant 3$, or R is a 4-fringed tree of height h whose associated (unfringed) tree $S = \underline{R}_{h-4}$ satisfies property (1). (Note that an isolated node is an h-comb for all h.) Thus, when $h \geqslant 4$, it suffices to find an h-comb R, whose associated tree $S = \underline{R}_{h-4}$ has both of its extreme leaves at depth $h-4$, and which maximizes $g(R)$ defined by

$$g(R) = -\lambda(R)2^{-h} + \sum_{1 \leqslant j \leqslant \lambda(R)} \left[2^{x_j} \dot{-} 2^{x_{j+1}}\right] \tag{8}$$

(note the change in summation boundary) where

$$x_j = \begin{cases} -d_R(\ell_j^R) & \text{if } j \leqslant \lambda(R), \\ -h+3 & \text{if } j = \lambda(R)+1. \end{cases}$$

If A_h denotes this maximum value then

$$\Phi(U_h) \leqslant 3/2 + L2^{-h} - 2^{-h+3} + A_h \tag{9}$$

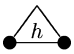
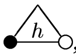
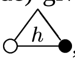
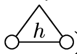
for all $T \in \mathcal{T}_L$, when $h \geqslant 4$.

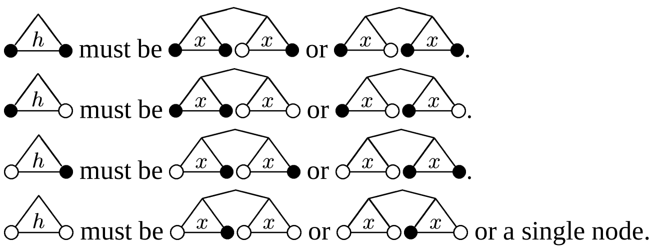
To determine A_h, it helps to consider two related optimizations. Let B_h denote the maximum value of $g(R)$ over all h-combs R at least one of whose extreme leaves has depth h, and let C_h denote the maximum value of $g(R)$ over all h-combs without constraints on their extreme leaves.

Lemma 6.6.

$$A_h = 1 - (F_{h+2} + 3F_{h+1} - 9)2^{-h}, \qquad B_h = 1 - (F_{h+1} + 3F_h)2^{-h},$$
$$C_h = 1 - 9 \cdot 2^{-h},$$

for all $h \geqslant 4$, where F_k is the kth Fibonacci number ($F_0 = 0$, $F_1 = 1, \ldots$).

Proof (by induction on h). For $h = 4$, the unique 4-comb of height 4 gives $A_4 = B_4 = 2^{-3}$. The 4-comb of height 0 (an isolated node) gives $C_4 = 7 \cdot 2^{-4}$. Suppose that $h > 4$ and denote by  (respectively , , ) an h-comb of height h whose associated unfringed tree has both (respectively at least its left, at least its right, possibly neither) extreme leaf at depth $h - 4$. Then by the comb property we have (for $x = h - 1$):



Hence,

$$A_h = A_{h-1}/2 + B_{h-1}/2, \qquad B_h = \max\{B_{h-1}/2 + B_{h-1}/2, C_{h-1}/2 + A_{h-1}/2\},$$
$$C_h = \max\{B_{h-1}/2 + C_{h-1}/2, 1 - 9 \cdot 2^{-h}\}.$$

The result follows by straightforward calculation. □

We conclude from Lemma 6.6 and Eq. (9) that

$$\Phi(U_h) \leqslant 5/2 + (L - F_{h+2} - 3F_{h+1} + 1)2^{-h} \tag{10}$$

for $h \geqslant 4$.

By choosing $T \in \mathcal{T}_L$ arbitrarily and considering particular ranges of h, we can extend the results of Observation 6.1. Specifically, if $1 \leqslant h \leqslant 3$ and $L \leqslant 2^{h+k} - 2^h + h + 1$, for some integer $k \geqslant 1$, then it follows from Eq. (7) that

$$2\Phi(U_h) \leqslant 3 + (L - h - 2)2^{-h+1} \leqslant 3 + \left(2^{h+k} - 2^h - 1\right)2^{-h+1}$$
$$= 2^{k+1} - 2^{-h+1} + 1$$

and so, by Lemma 6.1 and Eq. (3),

$$\Xi_h^0(T) \leqslant \alpha(W) = \left\lceil \lg \left\lfloor 2\Phi(U_h) \right\rfloor \right\rceil - 1 \leqslant k.$$

Similarly, if $h \geqslant 4$ and $L \leqslant 2^h(2^k - 2) + F_{h+2} + 3F_{h+1} - 2$, for some integer $k \geqslant 1$, then it follows from Eq. (10) that

$$\begin{aligned} 2\Phi(U_h) &\leqslant 5 + (L - F_{h+2} - 3F_{h+1} + 1)2^{-h+1} \leqslant 5 + \bigl(2^h\bigl(2^k - 2\bigr) - 1\bigr)2^{-h+1} \\ &= 2^{k+1} + 1 - 2^{-h+1} \end{aligned}$$

and so $\Xi_h^0(T) \leqslant k$, as above.

In summary, if

$$L_{h,t} = \begin{cases} (t+1)2^h + h + 2 & \text{if } 1 \leqslant h \leqslant 4, \\ F_{h+2} + 3F_{h+1} - 1 + t2^h & \text{if } h > 4, \end{cases}$$

then

$$\Xi_h^0(L) \leqslant \begin{cases} 0 & \text{if } L < h+2, \\ 1 & \text{if } h+2 \leqslant L < L_{h,0}, \\ k & \text{if } L_{h,2^{k-1}-2} \leqslant L \leqslant L_{h,2^k-2} \text{ and } k > 1, \\ \lceil \lg L \rceil - h & \text{if } h \leqslant 1 < \lceil \lg L \rceil. \end{cases} \tag{11}$$

In all cases, the Fibonacci trees introduced earlier can be used to construct examples of trees that realize our upper bounds on $\Xi_h^0(L)$. Since $\Xi_h^0(L)$ is monotonically non-decreasing it suffices to demonstrate, for each $h > 1$ and $k \geqslant 1$, a tree T, whose number of leaves $\lambda(T)$ satisfies $\lambda(T) = L_{h,2^k-2}$, for which $\Xi_h^0(T) = k + 1$.

An *extended Fibonacci tree* of height $h \geqslant 1$ and potential $t \geqslant 0$, denoted $\mathcal{F}_h^*\langle t\rangle$, is defined as follows: for $1 \leqslant h \leqslant 4$, $\mathcal{F}_h^*\langle t\rangle$ is just a $((t+1)2^h + h + 1)$-chain and, for $h > 4$, $\mathcal{F}_h^*\langle t\rangle$ is the binary tree whose left subtree is $\mathcal{F}_{h-1}^*\langle 0\rangle$ and whose right subtree has a leaf as its left subtree and $\mathcal{F}_{h-2}^*\langle 4t\rangle$ as its right subtree. Figure 5 illustrates this construction. See Figs. 7 and 8 for examples of extended Fibonacci trees.

By straightforward induction on h, we have:

Lemma 6.7. *For all $h > 1$ and $t \geqslant 0$, $\lambda(\mathcal{F}_h^*\langle t\rangle) = L_{h,t}$.*

Furthermore,

Lemma 6.8. *For every $h > 1$ and $k \geqslant 1$, $\Xi_h^0(\mathcal{F}_h^*\langle 2^k - 2\rangle) = k + 1$.*

Proof. It suffices to confirm, by induction on h, that for every $h \geqslant 1$, if W is the h-leveled weight sequence associated with $\mathcal{F}_h^*\langle t\rangle$ then W_h (the hth weight sequence in the reduction sequence associated with W) has the form

$$(-1)\bigl(0(-1)\bigr)^{2t+5}.$$

Thus, by Lemma 6.1 and Corollary 6.3, $\Xi_h^0(\mathcal{F}_h^*\langle 2^k - 2\rangle) = \lceil \lg\bigl(2^{k+1} + 1\bigr)\rceil - 1 = k + 1$.

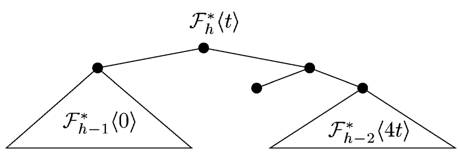


Fig. 5. An extended Fibonacci tree of height $h > 4$.

Thus, the upper bounds on $\Xi_h^0(L)$ given in Eq. (11) are tight. Using Lemma 4.2, these bounds translate into the following exact bounds on the leaf h-leveling cost (thereby, establishing Theorem 2.1)

$$\Delta_h^0(L) = \begin{cases} 0 & \text{if } L < h+2, \\ 1 & \text{if } h+2 \leqslant L < L_{h-1,0}, \\ 2 & \text{if } L \geqslant L_{h-1,0} \text{ and } \lceil \lg L \rceil < h, \\ \lceil \lg L \rceil - 1 & \text{if } \lceil \lg L \rceil = h, \\ \infty & \text{if } \lceil \lg L \rceil > h. \end{cases}$$

6.2. General h-leveled weight sequences

For any given $c \geqslant 1$ and $h \geqslant 0$ we are interested in determining the size of the smallest tree T satisfying $\Xi_h^*(T) = c$. By Lemma 6.1 this is equivalent to asking for the smallest tree T whose h-leveled weight sequence W satisfies $\alpha(W) = c$. Let W_z be the weight sequence in the reduction sequence associated with W that has $\min(\widetilde{W_z}) = 1$. The thickness of W_z is zero and by Corollary 6.3, $\alpha(W) = \lceil \lg r \rceil$ where $|W_z| = 2r-1$. Let $\Lambda(r,h)$ equal the minimum L such that there exists an h-leveled weight sequence W with $|W| = 2L-1$ whose associated sequence W_z has length $2r-1$. Then

$$\Xi_h^*(L) = c, \quad \text{for all } L \text{ satisfying } \Lambda\big(2^{c-1}+1, h\big) \leqslant L \leqslant \Lambda\big(2^c, h\big). \tag{12}$$

The action of the Phased Alphabetic Minimax Algorithm on an h-leveled weight sequence associated with a tree T can be understood in terms of its action on the weight sequences associated with the left and right subtrees of T. This gives rise to the following recurrence for $\Lambda(r,h)$.

Lemma 6.9.

$$\Lambda(r,0) = r, \qquad \Lambda(1,h) = 1,$$

and for $r \geqslant 2$ and $h \geqslant 1$,

$$\Lambda(r,h) = \min_{a,b:\ \lceil (a+b)/2 \rceil = r} \big\{ \Lambda(a,h-1) + \Lambda(b,h-1) \big\}.$$

Proof. If $h = 0$ then any h-leveled weight sequence W has the form 0^{2L-1}, thus $\min(\widetilde{W_1}) = 1$ and $|W_1| = 2L-1$ which implies $\Lambda(r,0) = r$.

If $r = 1$ then $\Lambda(1,h) = 1$ since $W = (0)$ has $\min(\widetilde{W_0}) = 1$ and $|W_0| = 1$.

Let $T \in \mathcal{T}_L$ and let T^L and T^R denote the subtrees of T rooted at the left and right children of the root of T. If W is the h-leveled weight sequence associated with T, for some $h \geqslant 1$, then W has the form $(W^L-1)0(W^R-1)$, where W^L (respectively W^R) denotes the $(h-1)$-leveled weight sequence associated with T^L (respectively T^R).[2]

[2] If S is a sequence of integers then $(S-1)$ denotes the same sequence with each element decreased by 1.

Recall that W_z is the weight sequence in the reduction sequence associated with W that has $\min(\widetilde{W_z}) = 1$. For $r \geqslant 2$ and $h \geqslant 1$, it is straightforward to confirm that the weight sequence W_{z-1} has the form $(W^L_{z^L} - 1)0(W^R_{z^R} - 1)$, where $W^L_{z^L}$ (respectively $W^R_{z^R}$) denotes the sequence in the reduction sequence associated with W^L (respectively W^R) with $\min(\widetilde{W^L_{z^L}}) = 1$ (respectively $\min(\widetilde{W^R_{z^R}}) = 1$). If $|W^L_{z^L}| = 2a - 1$ and $|W^R_{z^R}| = 2b - 1$ then W_{z-1} has the form $(-1)(0(-1))^{a+b-1}$ and, by Lemma 6.4, W_z has length $2\lceil (a+b)/2 \rceil - 1$. Thus, if weight sequence W realizes $\Lambda(r,h)$ then sequences W^L and W^R realize $\Lambda(a, h-1)$ and $\Lambda(b, h-1)$ for some a and b satisfying $\lceil (a+b)/2 \rceil = r$. Hence,

$$\Lambda(r,h) = \min_{a,b:\ \lceil (a+b)/2 \rceil = r} \bigl\{ \Lambda(a, h-1) + \Lambda(b, h-1) \bigr\}. \qquad \square$$

Lemma 6.10.

$$\Lambda(r,h) = \sum_{i=0}^{r-1} \binom{h}{i} (r-i).$$

Proof (by induction on r and h). For $h = 0$, the lemma states

$$\Lambda(r,0) = \sum_{i=0}^{r-1} \binom{0}{i} (r-i),$$

which is r, since $\binom{0}{i} = 0$ for $i > 0$.

For $r = 1$, the lemma states

$$\Lambda(1,h) = \sum_{i=0}^{0} \binom{h}{i} (1-i),$$

which is 1.

We claim that the minimum of $\Lambda(a, h-1) + \Lambda(b, h-1)$ over all values a, b such that $\lceil (a+b)/2 \rceil = r$ is achieved when $a = r-1$ and $b = r$ (or $a = r$ and $b = r-1$). Since $\Lambda(r,h)$ is an increasing function in r, we need only consider a and b such that $a + b = 2r - 1$.

Let

$$f(t) = \sum_{i=0}^{t-1} \binom{h-1}{i} (t-i) + \sum_{i=0}^{2r-2-t} \binom{h-1}{i} (2r-1-t-i).$$

Our claim, and the lemma, follows if $f(t)$ achieves its minimum at $t = r-1$. Consider $f(t) - f(t+1)$.

$$f(t) - f(t+1) = \sum_{i=0}^{2r-2-t} \binom{h-1}{i} - \sum_{i=0}^{t} \binom{h-1}{i}. \tag{13}$$

If $t < r - 1$ then $2r - 2 - t > t$ and $f(t) - f(t+1)$ is positive. If $t > r - 1$ then $2r - 2 - t < t$ and $f(t) - f(t+1)$ is negative. Thus, $f(t)$ is minimized when $t = r - 1$. $\square$

By Lemma 6.10 and Eq. (12), $\Xi_h^*(L) = \lceil \lg \rho(L, h) \rceil$ where $\rho(L, h) = \max\{r\colon \sum_{i=0}^{r-1} \binom{h}{i} (r - i) \leqslant L\}$. We then use Lemma 4.1 to obtain Theorem 2.2.

Acknowledgments

The authors express their appreciation to Prosenjit Bose, both for having posed the central question investigated in this paper and for numerous helpful discussions.

Appendix A. Restructuring examples

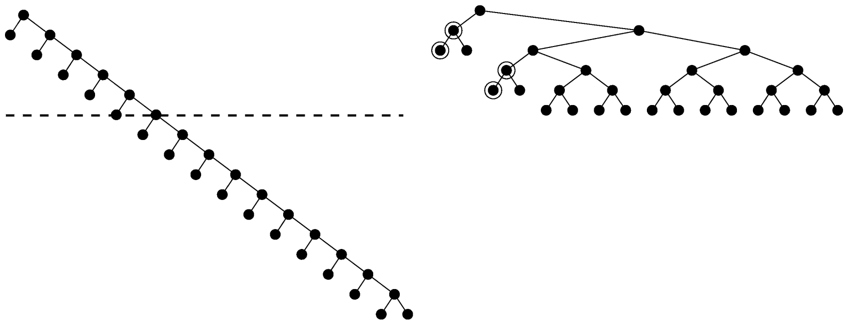


Fig. 6. Chain (left) with $L = 16$ that is restructured (right) to have height at most 5. Nodes whose depth increased are circled.

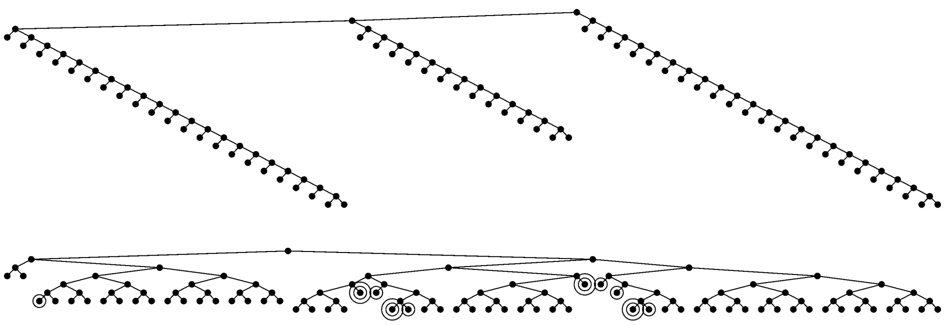


Fig. 7. Smallest tree (top) whose leaf-restricted h-leveling cost is 2 when $h = \lceil \lg L \rceil + 1$ ($L = 59$), and the resulting restructured tree (bottom) obtained by our algorithm. Leaves are circled a number of times equal to their depth increase.

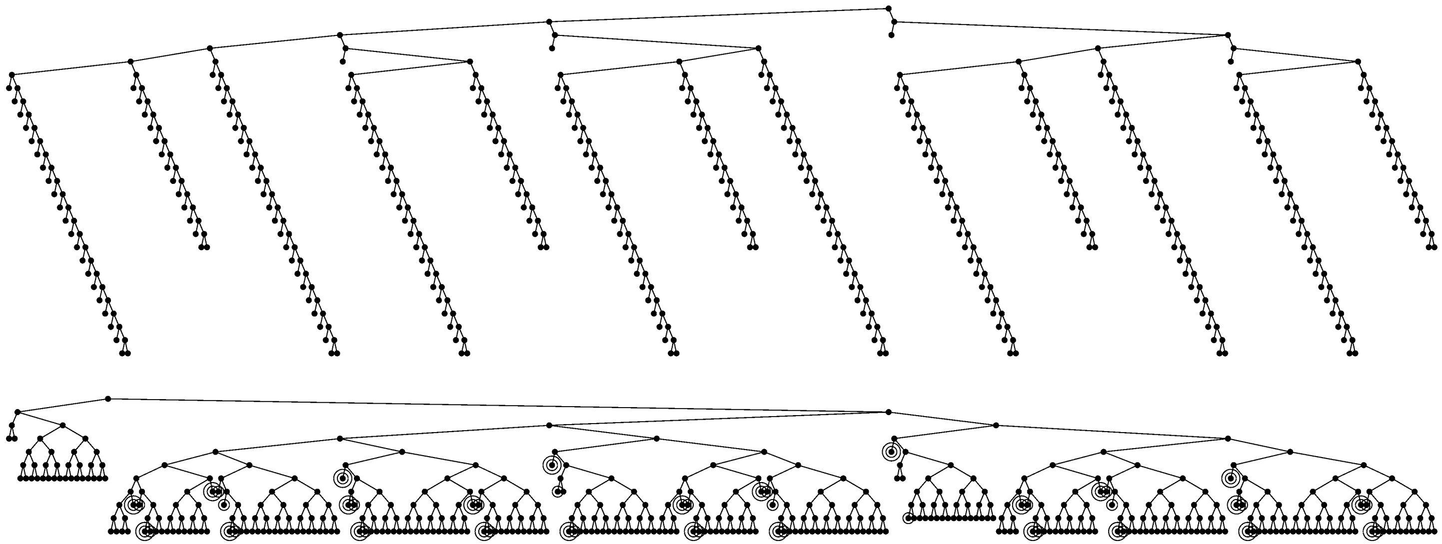


Fig. 8. Smallest tree (top) whose leaf-restricted h-leveling cost is 2 when $h = \lceil \lg L \rceil + 2$ ($L = 253$), and the resulting restructured tree (bottom) obtained by our algorithm. Leaves are circled a number of times equal to their depth increase.

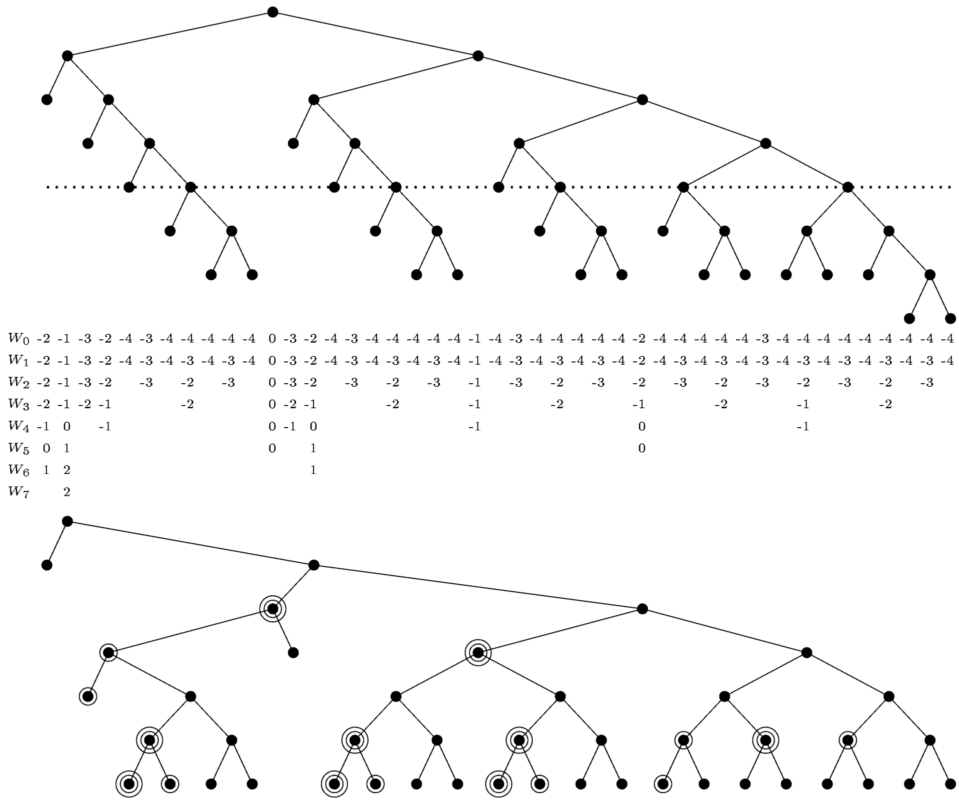


Fig. 9. An example of near-4-leveling, the resulting weight sequence, the weight sequences produced by the phased alphabetic minimax algorithm, and the resulting "balanced" tree. The original tree is a smallest tree ($L = 23$) whose h-leveling cost is 2 when $h = \lceil \lg L \rceil + 1$. Nodes are circled a number of times equal to their depth increase.

References

[1] M.R. Garey, Optimal binary search trees with restricted maximal depth, SIAM J. Comput. 3 (2) (1974) 101–110.
[2] E.N. Gilbert, Codes based on inaccurate source probabilities, IEEE Trans. Inform. Theory IT–17 (3) (1971) 304–314.
[3] T.C. Hu, D.J. Kleitman, J.K. Tamaki, Binary trees optimum under various criteria, SIAM J. Appl. Math. 37 (1979) 246–256.
[4] T.C. Hu, A.C. Tucker, Optimal computer search trees and variable length alphabetic codes, SIAM J. Appl. Math. 21 (4) (1971) 514–532.
[5] D.A. Huffman, A method for the construction of minimum-redundancy codes, Proc. IRE 40 (9) (1952) 1098–1101.
[6] A. Itai, Optimal alphabetic trees, SIAM J. Comput. 5 (1) (1976) 9–18.
[7] D.G. Kirkpatrick, M.M. Klawe, Alphabetic minimax trees, SIAM J. Comput. 14 (3) (1985) 514–526.
[8] D. Knuth, The Art of Computer Programming, vol. 3. Sorting and Searching, Addison–Wesley, 1973.
[9] L.L. Larmore, Height restricted optimal alphabetic trees, SIAM J. Comput. 16 (1987) 1115–1123.
[10] L.L. Larmore, D.S. Hirschberg, A fast algorithm for optimal length-limited Huffman codes, J. ACM 37 (3) (1990) 464–473.

[11] L.L. Larmore, T.M. Przytycka, A fast algorithm for optimal height-limited alphabetic binary trees, SIAM J. Comput. 23 (6) (1994) 1283–1312.
[12] K. Mehlhorn, Best possible bounds on the weighted path length of optimal binary search trees, SIAM J. Comput. 6 (2) (1977) 235–239.
[13] K. Mehlhorn, Data Structures and Algorithms, vol. 1: Sorting and Searching, in: EATCS Monographs, Springer-Verlag, 1984.
[14] R.L. Milidiú, E.S. Laber, Improved bounds on the inefficiency of length-restricted prefix codes, Monografias em Ciênciada Computação 33, Departamento de Informática, PUC-Rio, 1997.
[15] D.D. Sleator, R.E. Tarjan, Self-adjusting binary search trees, J. ACM 32 (3) (1985) 652–686.
[16] R.L. Wessner, Optimal alphabetic search trees with restricted maximal height, Inform. Process. Lett. 4 (4) (1976) 90–94.