Balanced Mobiles with applications to phylogenetic trees and Huffman-like problems

Yassine Hamoudi, Sophie Laplante, Roberto Mantaci

▶ **To cite this version:**

Yassine Hamoudi, Sophie Laplante, Roberto Mantaci. Balanced Mobiles with applications to phylogenetic trees and Huffman-like problems. 2015. hal-04047256

HAL Id: hal-04047256
https://hal.science/hal-04047256

Preprint submitted on 27 Mar 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Balanced Mobiles with applications to phylogenetic trees and Huffman-like problems

Yassine Hamoudi[1], Sophie Laplante[1], and Roberto Mantaci[1]

1 **IRIF, Université Paris Diderot, France.**
{hamoudi,laplante,mantaci}@irif.fr

Abstract

The Colless index is a measure of balance used in phylogeny to study the shape of phylogenetic trees. In this paper, we study a more general class of objects, which we call mobiles, where leaves of a full binary tree have integer weights. We extend the Colless index to the more general problem of measuring balance in mobiles. We give a lower bound on the Colless index in the case of unit weights on n leaves, and give two classes of trees for which this bound is tight. We then turn to the more general case of mobiles, where we are given a list of integer weights and are asked to find a mobile of minimal Colless index. We identify instances of this problem that admit a polynomial-time algorithm, and show that in general it is in the parameterized class XP. The general problem is not known to be in P nor NP hard, a situation similar to the case of finding Huffman codes where the costs of the letters are not all equal. We give an integer linear program for both of these problems, that can easily be adapted to other Huffman-like problems with different cost evaluation functions.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Algorithms, phylogenetic trees, Colless index, generalized Huffman coding, integer linear programming.

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

1.1 The Colless index for phylogenetic trees

A *phylogenetic tree* is an abstract representation of the evolutionary relationships among species. It is represented as a rooted tree whose leaves are labelled with the species under study, and each internal node represents the most recent common ancestor of its descendants (see Figure 1 for an example). The task of phylogenetic reconstruction involves the study of morphological and molecular similarities. The resulting trees are often compared to other phylogenies, or to random evolutionary models, to assess their relevance. This has led to the development of a tree shape theory in phylogeny, that provides comparison techniques based on the tree topologies.

Among the most studied shape properties is the notion of *tree balance* [30, 25] which has been made popular by the *Sackin index* [29] and the *Colless index* [10]. The former is the sum of the depths of the leaves (often normalized to represent the average external path length), whereas the latter (over binary trees) computes the sum of the absolute difference $|x - y|$ at each internal node, where x (resp. y) is the number of leaves in the left (resp. right) subtree of the node considered. We also mention the *number of cherries* (pairs of adjacent leaves) [23], and the *total cophenetic index* defined more recently by Mir *et al.* [24, 7] as the sum over all pairs of leaves of the depth of their least common ancestor.


© Yassine Hamoudi, Sophie Laplante, Roberto Mantaci;
licensed under Creative Commons License CC-BY

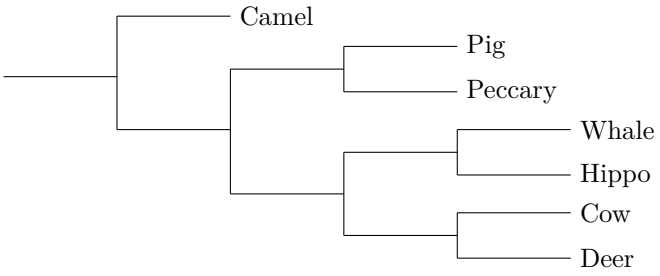


Figure 1 Example of a phylogenetic tree for the Artiodactyla.

The tree shape indices are used in practice to quantify the diversity skewness of phylogenetic trees, and to compare it to models of random tree growth [16, 30, 20, 25, 23, 2, 4, 17]. The expected value, variance, covariance, etc., of these indices are known for random evolutionary processes [20, 16, 23, 5, 24, 7]. Some normalization techniques have also been proposed, in order to compare trees with different number of leaves. Shao and Sokal [30] suggested for instance to replace an index I over n leaves by $(I - I_{\max})/(I_{\max} - I_{\min})$ where $I_{\max}$ (resp. $I_{\min}$) is the maximum (resp. minimum) possible value of the index over n leaves. Computing the maximum of the aforementioned indices is rather straightforward (it is $\mathcal{O}(n^2)$ for the Sackin and Colless indices, and $\mathcal{O}(n^3)$ for the total cophenetic number [24]). On the other hand, having closed-form expressions for the minimum can be more involved. Unlike the Sackin and total cophenetic indices [22, 24], such a formula was not known for the Colless index until now (Section 3 and Theorem 4). We summarize these results in Table 1.

The previous indices can also be expressed as the sum of costs $c(x, y)$ placed on the internal nodes of the trees (where $c(x, y) = x + y$, $c(x, y) = |x - y|$ and $c(x, y) = \binom{x}{2} + \binom{y}{2}$ for the Sackin, Colless and total cophenetic indices, respectively). This implies a natural linear time algorithm for computing the indices, as well as recursive properties that are given in Table 1 and will be studied in more detail for the Colless index in this paper.

Index	Cost	Formula for the Minimum value	Optimal trees
Sackin [26, 22]	$x + y$	$\mathcal{S}_n = (\lceil \log n \rceil + 1)n - 2^{\lceil \log n \rceil}$ $\mathcal{S}_n = \mathcal{S}_{\lceil n/2 \rceil} + \mathcal{S}_{\lfloor n/2 \rfloor} + n$ $\mathcal{S}_{n+1} = \mathcal{S}_n + \lvert n \rvert_0 + \lvert n \rvert_1 + 1$	- Complete
Colless (Section 3)	$\lvert x - y \rvert$	$\mathcal{C}_n = 2 \cdot (n \bmod 2^k) + \sum_{i=0}^{k-1} (-1)^{b_i} \cdot (n \bmod 2^{i+1})$ $\mathcal{C}_n = \mathcal{C}_{\lceil n/2 \rceil} + \mathcal{C}_{\lfloor n/2 \rfloor} + (n \bmod 2)$ $\mathcal{C}_{n+1} = \mathcal{C}_n + \lvert n \rvert_0 - \lvert n \rvert_1 + 1$	- Left-complete - Partition
Cophenetic [24]	$\binom{x}{2} + \binom{y}{2}$	$\Phi_n = \binom{n}{2} + \frac{1}{2} \cdot (\mathcal{C}_n - \mathcal{S}_n)$ $\Phi_n = \Phi_{\lceil n/2 \rceil} + \Phi_{\lfloor n/2 \rfloor} + \binom{\lceil n/2 \rceil}{2} + \binom{\lfloor n/2 \rfloor}{2}$ $\Phi_{n+1} = \Phi_n + n - \lvert n \rvert_1$	- Complete

Table 1 Minimal values of the Sackin ($\mathcal{S}_n$), Colless ($\mathcal{C}_n$) and Cophenetic (Φ_n) indices over n leaves. We denote $b_k b_{k-1} \dots b_0$ the binary representation of n, and $\lvert n \rvert_0$ (resp. $\lvert n \rvert_1$) the number of 0 (resp. 1) in it. The rightmost column outlines families of trees realizing the minimum value. A full characterization for the Colless index is not known. These families will be defined in Section 2.

1.2 Weights, Huffman coding and Balanced Mobiles

The indices from the previous section can be generalized to binary trees with positive weights on their leaves. We call such objects *"mobiles"*, by analogy with what can be found in modern art (Calder) or above toddler beds. The Sackin (resp. Colless) index of a mobile is computed as before, except that the internal cost $c(x, y) = x + y$ (resp. $c(x, y) = |x - y|$) takes as input the total weight of the leaves in the left and right subtrees of the node considered. The situation described in Section 1.1 corresponds to mobiles with leaves of weight 1. Note that a leaf of (integer) weight w can also be interpreted as a fixed subtree with w leaves, for which we do not want to include its internal imbalances in the total cost.

The Sackin index of a mobile with weights $w_1, \ldots, w_n$ has value $\sum_{i=1}^{n} w_i \cdot l_i$, where l_i is the depth of the leaf having weight w_i. Minimizing this quantity has been extensively studied in the context of coding theory, since it is equivalent to the following problem:

▶ **Problem 1** (Huffman Coding)**.** Let $\{a_1, a_2, \ldots, a_n\}$ be letters and w_i the frequency of a_i. Find a prefix code $\{c_1, c_2, \ldots, c_n\}$ minimizing the average code length $\sum_{i=1}^{n} w_i \cdot |c_i|$, where $|c_i|$ is the length of the encoding c_i of a_i.

This problem can be solved with the well-known Huffman algorithm [18] that recursively builds a mobile of optimal Sackin index by grouping the two smallest weights together. There has been much work attempting to characterize cost functions for which the Huffman algorithm is optimal. In the generalized setting, we are given a weight merging function w, a cost function c and a cost combination function F. Given a mobile M over n leaves, the weight and cost of an internal node are defined recursively as $w(x, y)$ and $c(x, y)$, where x and y are the weights of the left and right children of the node considered. The total cost of M is $F(c_1, \ldots, c_{n-1})$ where each c_i is the cost of a distinct internal node. In the standard Huffman problem (Sackin index), $w(x, y) = c(x, y) = x + y$ and F is the sum function.

There is an extensive literature [1] on the functions w, c and F for which the Huffman algorithm minimizes the total cost. One well-studied case for instance is when $w(x, y) = c(x, y) = t + \max(x, y)$ (where t is a constant) and $F = \max$ [15, 27]. The cost combination function F can also be $F(c_1, \ldots, c_{n-1}) = \sum_i f(c_i)$, where f is any nondecreasing concave function, if $w(x, y) = c(x, y) = x + y$ [11]. Parker [28] encompassed these results in a more general framework in which F must be a max, min or "Schur concave" function (which includes the sum), and w and c are some "quasilinear" functions (e.g. sum, max, etc.). From a more abstract point of view, Knuth [21] developed the notion of "Huffman algebra" in which the Huffman algorithm minimizes the evaluation cost of any given expression. A complete characterization of the cases of optimality of the Huffman algorithm is still unknown.

One of the main examples where Huffman's algorithm is not optimal is Huffman coding with unequal letter costs. The cost function is $c(x, y) = \alpha \cdot x + \beta \cdot y$ for some — possibly nonconstant — parameters α, β (in the basic Huffman problem, $\alpha = \beta = 1$). This problem was first studied by Karp who solved it in exponential time with integer linear programming [19]. Later, a result of Bradford *et al.* [6] (improving on a dynamic programming algorithm from Golin *et al.* [13]) gave an exact algorithm in time $\mathcal{O}\big(n^{\max(\alpha,\beta)}\big)$. Golin *et al.* [12] also provided a polynomial-time approximation scheme (PTAS). The special case of all-equal weights w_i (also known as Varn coding) has been solved in polynomial time [31, 14, 9], but no such algorithm is known for the general case, nor is it known to be NP-hard.

The generalized Colless index is another case of non-optimality for the Huffman algorithm (see a counterexample in Figure 7 of Appendix A). Most of this paper will be devoted to this problem, which does not seem to have been studied so far, and that we call Balanced Mobiles:

▶ **Problem 2** (Balanced Mobiles)**.** Given a (multi-)set of integer weights $\{w_1, w_2, \ldots, w_n\}$, find a mobile M whose n leaves have weights $w_1, w_2, \ldots, w_n$ (in any possible order) and whose Colless imbalance $\mathcal{C}_M$ is as small as possible.

One of our main contributions is to show that the Huffman algorithm (described in Appendix A.1) solves this problem in the particular case of power-of-two weights (Theorem 6). This does not fall within the scope of the previous optimality results for the Huffman algorithm.

Summary of results We start by studying the optimal value of the Colless index over trees with n leaves of unit weight (Section 3). We show that partition and left-complete trees are optimal, and give recursive and closed-form expressions for their imbalance $\mathcal{C}_n$.

We then turn to polynomial time instances of the Balanced Mobiles problem (Section 4). We prove that the Huffman algorithm is optimal when the weights are powers-of-two (Section 4.1), or for finding perfectly balanced mobiles (Section 4.3). We show that the Balanced Mobiles problem is in the parameterized class XP, using a relaxation of Huffman's algorithm (Section 4.2). The complexity of the given algorithm is roughly $\mathcal{O}\big(\log(n) n^{\mathcal{C}^\star+1}\big)$ where $\mathcal{C}^\star$ is the optimal imbalance.

Finally, we give an integer linear program for the Balanced Mobiles problem (Section 5.1) and we compare it to a naive algorithm based on a bijective characterisation of labeled trees (Section 5.2). These two algorithms can be used to solve other Huffman-like problems, such as Huffman coding with unequal letter costs.

2 Terminology of trees

A *rooted tree* is a connected acyclic graph with one node identified as the *root*. The *depth* of a node is its distance to the root, and the *level* k of a tree is the nodes at depth $k-1$. A node u is the *parent* of a node v (resp. v is a *child* of u) if they are connected by an edge, and the depth of v is greater than the depth of u. We also say that u is an *ancestor* of w (resp. w is a *descendant* of u) if there is a path from w to u in which each node is the parent of the previous one. Two nodes are *siblings* if they have the same parent. A *leaf* is a node which is not parent to any other node. A node which is not a leaf is called an *internal node*.

We focus on (rooted) *full binary* trees in which each internal node has exactly two children. Such a tree is *ordered* if the children of each parent are identified as *left* or *right*. We also sometimes refer to left and right children for *unordered* trees, in which case they can be chosen arbitrarily.

Some special cases of full binary trees are the *cherry* (binary tree over two leaves), the *perfect* trees over 2^k leaves (all the leaves have depth k), the *complete* trees (the leaves are concentrated on the last two levels), the *left-complete* trees (complete trees with the additional condition that all leaves in the last level are as far left as possible) and the *partition* trees (at each internal node, the number of leaves x and y in the left and right subtrees differ by at most 1). Since all the partition trees over n leaves are isomorphic, we will talk about *the* partition tree P_n over n leaves. We also denote L_n the left-complete tree over n leaves.

Figure 2 Partition tree P_{11} (on the left) and left-complete trees L_{11} (on the right).

A *mobile* is a full binary tree whose leaves are labelled with positive integer weights. As the Colless index is the only measure of imbalance studied in this paper, we fix the *local imbalance* of an internal node ν to be the difference (in absolute value) between the weights of the left and right subtrees of ν, and the (Colless) *imbalance* $\mathcal{C}_M$ of a mobile M to be the sum of its local imbalances. We also define the imbalance $\mathcal{C}_T$ of a tree T as the imbalance of T with unit weights on its leaves, and we denote $\mathcal{C}_n$ the smallest possible imbalance for a tree with n leaves. A mobile M is *optimal* if it is not possible to achieve an imbalance less than $\mathcal{C}_M$ on the same weights as M. If $\mathcal{C}_M = 0$ then M is said to be *perfectly balanced*. Figure 3 depicts two mobiles built on weights $\{2, 3, 5, 7\}$. The local imbalances are written next to the internal nodes.

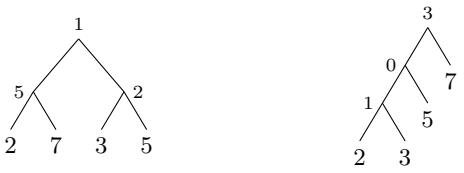


Figure 3 Two mobiles built on weights $\{2, 3, 5, 7\}$. The imbalance of the left mobile is 8. The imbalance of the right one is 4. The right mobile is optimal: one cannot build a mobile of imbalance less than 4 on weights $\{2, 3, 5, 7\}$.

3 Minimal values of the Colless index for unit weights

We study the Collex indices $\mathcal{C}_{P_n}$ and $\mathcal{C}_{L_n}$ of the partition and left-complete trees over n leaves. We prove that these trees have both minimal Colles index (i.e. $\mathcal{C}_{P_n} = \mathcal{C}_{L_n} = \mathcal{C}_n$). This allows us to give several characterizations of the optimal Colless index $\mathcal{C}_n$ over n leaves, including a closed-form expression.

The combinatorial structures of the partition and left-complete trees lead to two natural recursive definitions of their imbalance.

▶ **Proposition 1.** *The imbalance $\mathcal{C}_{P_n}$ of the partition tree P_n over n leaves is*

$$\begin{cases} \mathcal{C}_{P_1} = 0 \\ \mathcal{C}_{P_{2n}} = 2\mathcal{C}_{P_n} \\ \mathcal{C}_{P_{2n+1}} = 1 + \mathcal{C}_{P_n} + \mathcal{C}_{P_{n+1}} \end{cases}$$

Proof. The local imbalance of the root of a partition tree is zero if and only if the tree has $2n$ leaves (for some n). In this case, the left and right subtrees of the root have both n leaves. Thus $\mathcal{C}_{P_{2n}} = 2\mathcal{C}_{P_n}$. On the other hand, the local imbalance of the root is one if and only if the tree has $2n + 1$ leaves (for some n). In this case, the left and right subtrees of the root have n and $n + 1$ leaves respectively. Thus $\mathcal{C}_{P_{2n+1}} = 1 + \mathcal{C}_{P_n} + \mathcal{C}_{P_{n+1}}$. ◀

▶ **Proposition 2.** *The imbalance $\mathcal{C}_{L_n}$ of the left-complete tree L_n over n leaves is*

$$\begin{cases} \mathcal{C}_{L_1} = 0 \\ \mathcal{C}_{L_{n+1}} = \mathcal{C}_{L_n} + |n|_0 - |n|_1 + 1 \end{cases}$$

where $|n|_0$ (resp. $|n|_1$) is the number of 0 (resp. 1) in the binary representation of n.

Proof. The left-complete tree over $n + 1$ leaves is obtained from the left-complete tree over $n = 2^k + m$ leaves (where $0 \leq m < 2^k$) by replacing the leftmost leaf of depth k with a cherry. We show in Appendix B.1 that this operation increases the total imbalance by $|n|_0 - |n|_1 + 1$. ◀

Using the previous two propositions, it is easy to prove that the partition and left-complete trees have the same imbalance (i.e. $\mathcal{C}_{P_n} = \mathcal{C}_{L_n}$). This is shown in Theorem 4 below. Before, we prove that the partition trees have optimal Colless index, which will also imply the optimality of the left-complete trees.

▶ **Proposition 3.** *The partition trees have optimal Colless index.*

Proof. We show how to transform any optimal tree into a partition tree, without increasing the imbalance. The proof is done by induction on n in Appendix B.2. ◀

We can now conclude our study of the optimal Colles index $\mathcal{C}_n$ over trees with n leaves. The next theorem gives several characterizations of $\mathcal{C}_n$ in terms of trees, recursive formula and closed-form expressions.

▶ **Theorem 4.** *Denote by $b_k b_{k-1} \dots b_0$ the binary representation of n, and $|n|_0$ (resp. $|n|_1$) the number of 0 (resp. 1) in it. Then, the following definitions of $\mathcal{C}_n$ are equivalent:*

1. *$\mathcal{C}_n$ is the optimal Colless index for a tree with n leaves*
2. *$\mathcal{C}_n$ is the imbalance of the partition tree over n leaves*
3. *$\mathcal{C}_n$ is the imbalance of the left-complete tree over n leaves*
4. *$\mathcal{C}_1 = 0$, $\mathcal{C}_{2n} = 2\mathcal{C}_n$ and $\mathcal{C}_{2n+1} = 1 + \mathcal{C}_n + \mathcal{C}_{n+1}$*
5. *$\mathcal{C}_1 = 0$ and $\mathcal{C}_{n+1} = \mathcal{C}_n + |n|_0 - |n|_1 + 1$*
6. *$\mathcal{C}_n = 2 \cdot (n \bmod 2^k) + \sum_{i=0}^{k-1} (-1)^{b_i} \cdot (n \bmod 2^{i+1})$*

Proof. The equivalence between the first, second and fourth definitions is a direct consequence of Propositions 1 and 3. Using the characterization of $\mathcal{C}_{P_n}$ and $\mathcal{C}_{L_n}$ given in Propositions 1 and 2 respectively, we prove in Appendix B.3 that $\mathcal{C}_{P_n} = \mathcal{C}_{L_n}$. This shows the equivalence between the third, fourth and fifth definitions. Finally, the equivalence with the last item is also proved in Appendix B.3, using the fourth and fifth definitions. ◀

▶ Remark. These results about $\mathcal{C}_n$ and the Colless index are summarized and compared to other tree shape statistics in Table 1. In particular, note that $\binom{n+1}{2} + (\mathcal{C}_{n+1} - \mathcal{S}_{n+1})/2 = \left(\binom{n}{2} + (\mathcal{C}_n - \mathcal{S}_n)/2\right) + n - |n|_1$. Thus, $\Phi_n = \binom{n}{2} + (\mathcal{C}_n - \mathcal{S}_n)/2$, where $\mathcal{S}_n$ and Φ_n are respectively the optimal Sackin and total cophenetic indices defined in introduction.

The set of trees having optimal Colless index seems to be more complex than for the Sackin and total cophenetic indices, and we dot not have a full characterization of it. It might be tempting to propose the family of complete trees (as it includes the partition and left-complete trees). However, Figure 4 gives a counterexample to this conjecture.

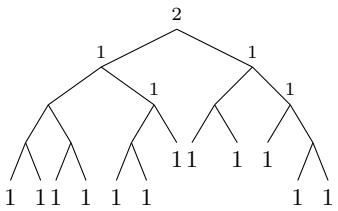


Figure 4 A non-optimal complete tree over 12 leaves (the imbalance is 6, whereas $\mathcal{C}_{12} = 4$).

4 Polynomial time instances of Balanced Mobiles

In the previous section, we studied the minimal Colless index for mobiles with unit weights. Here, we turn our attention to the Balanced Mobiles problem, that generalizes this problem to arbitrary input weights. We restrict to *coprime numbers*, as multiplying all the weights by a same factor does not change the shape of the optimal mobiles.

We concentrate on the Huffman algorithm for Balanced Mobiles, that recursively builds a mobile by grouping the two smallest weights together (see Appendix A.1). This algorithm is not optimal in general for solving Balanced Mobiles, nor it is an approximation algorithm (see Figure 7 of Appendix A). However, we show in Section 4.1 that it becomes optimal if the weights are restricted to be powers-of-two. We also prove that the general Balanced Mobiles problem is in the parameterized class XP, by giving an algorithm that finds optimal mobiles in polynomial time when the total imbalance is a fixed constant. In the case of perfectly balanced mobiles, which is studied in Section 4.3, this algorithm behaves exactly like Huffman.

4.1 Mobiles with powers-of-two weights

Observe first that the Huffman algorithm executed on n unit weights gives precisely the left-complete tree L_n. Consequently, using Theorem 4, the Huffman algorithm is optimal in this case.

▶ **Proposition 5.** *If the weights $w_1, \dots, w_n$ are all equal to one, then the Huffman algorithm builds an optimal mobile.*

We generalize this optimality result to powers-of-two weights.

▶ **Theorem 6.** *If the weights $w_1, \dots, w_n$ are powers of two, then the Huffman algorithm builds an optimal mobile in time $\mathcal{O}(n \log n)$.*

Proof. This is a proof by contradiction that uses the result of Proposition 5. We assume that Theorem 6 is false and choose a particular mobile $\mathcal{M}$ that must exist in this case and have some minimality properties. We then derive a contradiction by studying the two possible shapes of this mobile. See Appendix C for the details. ◀

As explained in introduction (Section 1.2), there is an extensive literature (notably [1, 28, 21]) on the Huffman-like problems that can be solved with the Huffman algorithm. None of these results applies to the case of Theorem 6. We hope it would help to characterize further families of problems solved with the Huffman algorithm.

4.2 Mobiles of constant imbalance

We relax the Huffman algorithm to construct optimal mobiles for arbitrary input weights. Our goal is to find a mobile of imbalance less than a given threshold δ (if one exists). Instead of always grouping the two smallest weights w_1 and w_2, as it is done in Huffman's algorithm, we also try to group w_1 with the other weights w_i such that $|w_1 - w_i| \leq \delta$ and we search recursively a mobile of imbalance less than $\delta - |w_1 - w_i|$ on $\{w_1 + w_i, w_2, \dots, w_{i-1}, w_{i+1}, \dots, w_n\}$. We call this algorithm R-Huffman. See Appendix A.2 for the formal description and the complexity analysis. This result shows that Balanced Mobiles is in the parameterized class XP.

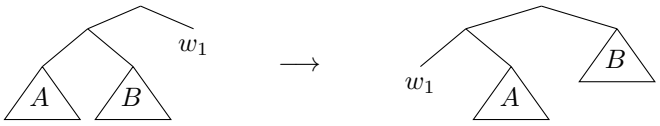
▶ **Theorem 7.** *Given weights $\{w_1, \dots, w_n\}$ and a threshold δ, R-Huffman finds a mobile of imbalance at most δ (if one exists) in time $\mathcal{O}\big(\log(n) n^{\min(\delta,n)+1}\big)$ and space $\mathcal{O}(n \log n)$.*

▶ **Corollary 8.** *The optimal imbalance $\mathcal{C}^\star$ over n weights $w_1, \dots, w_n$ can be found in time $\mathcal{O}\big(\log(n) n^{\min(\mathcal{C}^\star, n)+1}\big)$ by running the R-Huffman algorithm with $\delta = 0, 1, 2 \dots, \mathcal{C}^\star$. In particular, this is polynomial time if $\mathcal{C}^\star$ is known to be bounded by a constant.*

The proof is immediate from the following lemma.

▶ **Lemma 9.** *For any weights $w_1 \leq \cdots \leq w_n$, there exists an optimal mobile in which the sibling of the leaf of weight w_1 is also a leaf.*

Proof. Given two mobiles of weights A and B such that $w_1 \leq A \leq B$, the following rotation does not increase the imbalance:



Consequently, starting from an optimal mobile and repeating this operation several times, we can increase the depth of the leaf of weight w_1 until its sibling is another leaf. The resulting mobile is also optimal. ◀

4.3 Perfectly balanced mobiles

The R-Huffman and Huffman algorithms are equivalent if the input is $\delta = 0$. Thus, Theorem 7 implies that the Huffman algorithm is optimal for finding perfectly balanced mobiles.

▶ **Theorem 10.** *The Huffman algorithm builds a perfectly balanced mobile on weights $w_1, \dots, w_n$ in time $\mathcal{O}(n \log n)$, whenever such a mobile exists.*

▶ Remark. It is also easy to see from Theorem 7 that Huffman is optimal when the smallest imbalance is 1. However, Figure 7 in Appendix A.1 gives a counterexample for imbalance 2.

It turns out that the class of weights that admit such mobiles can be easily characterized.

▶ **Theorem 11.** *There exists a perfectly balanced mobile on the (coprime) weights $w_1, \dots, w_n$ if and only if all the weights w_i are powers of two and $\sum_{i=1}^{n} w_i$ is a power of two.*

Proof. The proof proceeds by induction. Details are provided in Appendix D. ◀

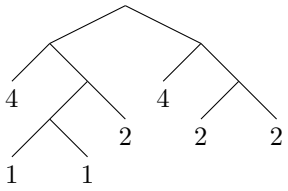


Figure 5 A perfectly balanced mobile on weights $\{1, 1, 2, 2, 2, 4, 4\}$.

5 Exponential algorithms for Balanced Mobiles

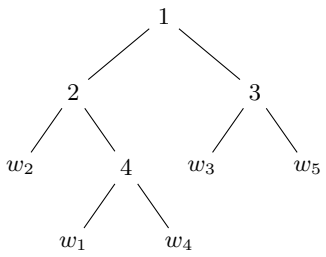
We conclude with an integer linear program that solves Balanced Mobiles, as well as many other Huffman-like problems. We also detail briefly an exhaustive search algorithm based on a bijective characterisation of labeled trees.

5.1 Integer Linear Programming

In this section, we add the extra condition that mobiles are ordered trees. We first describe what the variables of our program represent. Recall that any mobile built on n weights $w_1, \ldots, w_n$ has exactly $n-1$ internal nodes. We say that a numbering of these nodes is *increasing* if each path from the root has increasing numbers (the node v cannot be an ancestor of the node u if $u < v$) and no two nodes have the same number. For simplicity, we assume that these numbers are taken in $\{1, \ldots, n-1\}$ (in particular, the root is numbered 1). Given an increasing numbering, we associate with each weight w_i ($1 \leq i \leq n$) and each node u ($1 \leq u \leq n-1$) two boolean variables $\ell_{i,u}$ and $r_{i,u}$ assigned as follows:

- $\ell_{i,u} = 1$ if w_i is in the left subtree of the node u, 0 otherwise
- $r_{i,u} = 1$ if w_i is in the right subtree of the node u, 0 otherwise.

Figure 6 illustrates a mobile with the assignment of the corresponding variables.



weight \ node	1	2	3	4
w_1	$\ell_{1,1}=1$ $r_{1,1}=0$	$\ell_{1,2}=0$ $r_{1,2}=1$	$\ell_{1,3}=0$ $r_{1,3}=0$	$\ell_{1,4}=1$ $r_{1,4}=0$
w_2	$\ell_{2,1}=1$ $r_{2,1}=0$	$\ell_{2,2}=1$ $r_{2,2}=0$	$\ell_{2,3}=0$ $r_{2,3}=0$	$\ell_{2,4}=0$ $r_{2,4}=0$
w_3	$\ell_{3,1}=0$ $r_{3,1}=1$	$\ell_{3,2}=0$ $r_{3,2}=0$	$\ell_{3,3}=1$ $r_{3,3}=0$	$\ell_{3,4}=0$ $r_{3,4}=0$
w_4	$\ell_{4,1}=1$ $r_{4,1}=0$	$\ell_{4,2}=0$ $r_{4,2}=1$	$\ell_{4,3}=0$ $r_{4,3}=0$	$\ell_{4,4}=0$ $r_{4,4}=1$
w_5	$\ell_{5,1}=0$ $r_{5,1}=1$	$\ell_{5,2}=0$ $r_{5,2}=0$	$\ell_{5,3}=0$ $r_{5,3}=1$	$\ell_{5,4}=0$ $r_{5,4}=0$

Figure 6 An evaluation tree and the corresponding correct assignment.

We now address the reverse question: given a set of variables $(\ell_{i,u}, r_{i,u})_{1\leq i\leq n, 1\leq u\leq n-1}$: how can one characterize the *correct assignments* that represent existing mobiles? We first describe a set of conditions, and we translate them later into linear constraints.

▶ **Definition 12.** An assignment of the boolean variables $(\ell_{i,u}, r_{i,u})_{1\leq i\leq n, 1\leq u\leq n-1}$ is *correct* if it corresponds to a mobile whose leaves are distinctly labeled with $w_1, \ldots, w_n$, together with an increasing numbering of its internal nodes. An assignment is *proper* if it satisfies the following conditions:

(1) *Disjoint subtrees* For all i and u, either $\ell_{i,u} = 0$ or $r_{i,u} = 0$.
(2) *Root consistency* For all i, either $\ell_{i,1} = 1$ or $r_{i,1} = 1$.
(3) *Full binary tree* For all u, there exists i such that $\ell_{i,u} = 1$ and j such that $r_{j,u} = 1$.
(4) *Left leaf node* If $\ell_{i,u} = 1$ and $\ell_{i,v} = r_{i,v} = 0$ for all $v > u$, then $\ell_{j,u} = 0$ for all $j \neq i$.
Right leaf node If $r_{i,u} = 1$ and $\ell_{i,v} = r_{i,v} = 0$ for all $v > u$, then $r_{j,u} = 0$ for all $j \neq i$.
(5) *Common ancestors* If $\ell_{i,v} + r_{i,v} = \ell_{j,v} + r_{j,v} = 1$ then $\ell_{i,u} = \ell_{j,u}$ and $r_{i,u} = r_{j,u}$ for $u < v$.
(6) *Direct child node* If $\ell_{i,u} = \ell_{j,u} = 1$ or $r_{i,u} = r_{j,u} = 1$, then there exists $v > u$ such that $\ell_{i,v} + r_{i,v} = \ell_{j,v} + r_{j,v} = 1$ and $\ell_{i,w} = r_{i,w} = \ell_{j,w} = r_{j,w} = 0$ for all $u < w < v$.

We intend to show that the correct assignments are precisely the proper ones. We first prove that the previous conditions are necessary for an assignment to be correct.

▶ **Proposition 13.** *Any correct assignment is proper.*

Proof. Consider a correct assignment of the variables $(\ell_{i,u}, r_{i,u})_{1\leq i\leq n, 1\leq u\leq n-1}$ and a corresponding mobile M built on some weights $w_1, \ldots, w_n$. We make some basic observations on this mobile that explain why this assignment is also proper.

The leaf with weight w_i cannot be simultaneously in the left and right subtrees of the node u (Condition 1). All the leaves are either in the right or the left subtree of the root (Condition 2). Each internal node has at least one leaf in its left subtree and one leaf in its right subtree (Condition 3). If the left child of the node u is the leaf with weight w_i (i.e. $\ell_{i,u} = 1$ and $\ell_{i,v} = r_{i,v} = 0$ for all $v > u$, since we consider increasing numbering), then none of the other leaves can be in the left subtree of the node u (first case of Condition 4, the second one is similar). If v is a common ancestor to the leaves of weights w_i and w_j (i.e. $\ell_{i,v} + r_{i,v} = 1$ and $\ell_{j,v} + r_{j,v} = 1$), then they must also have the same ancestors from node 1 to v (Condition 5). If the leaves of weights w_i and w_j are in the left subtree of the node u (i.e. $\ell_{i,u} = \ell_{j,u} = 1$), then the left child v of the node u is a common ancestor to these leaves ($\ell_{i,v} + r_{i,v} = \ell_{j,v} + r_{j,v} = 1$) and none of the other nodes w, for $u < w < v$, can be an ancestor to them (first case of Condition 6, the second one is similar). ◀

We now show that these conditions are sufficient for an assignment to be correct.

▶ **Proposition 14.** *Any proper assignment is correct.*

Proof. We show how to split the set of variables into two parts, for which the two restrictions of the initial proper assignment are also proper. Using a well chosen induction hypothesis, it implies that these two restricted assignments are correct. Finally, the initial assignment is proved to be also correct, since it corresponds to a mobile whose left and right subtrees are associated to the two previous sets of variables. See Appendix E.1 for the details. ◀

It is now easy to convert the conditions of Definition 12 into linear constraints.

▶ **Proposition 15.** *The boolean assignments that satisfy the following integer constraints are the correct assignments.*

$$
\begin{array}{ll}
\textit{Constraint 0:} & \forall i, u, 0 \leq \ell_{i,u} \leq 1 \textit{ and } 0 \leq r_{i,u} \leq 1 \\
\textit{Constraint 1:} & \forall i, u, \ell_{i,u} + r_{i,u} \leq 1 \\
\textit{Constraint 2:} & \forall i, \ell_{i,1} + r_{i,1} = 1 \\
\textit{Constraint 3:} & \forall u, \sum_i \ell_{i,u} > 0 \textit{ and } \sum_i r_{i,u} > 0 \\
\textit{Constraint 4:} & \forall i \neq j, \forall u, \begin{cases} (1-\ell_{i,u}) + \sum_{v>u} (\ell_{i,v} + r_{i,v}) \geq \ell_{j,u} \\ (1-r_{i,u}) + \sum_{v>u} (\ell_{i,v} + r_{i,v}) \geq r_{j,u} \end{cases} \\
\textit{Constraint 5:} & \forall i \neq j, \forall u < v, \begin{cases} \ell_{i,u} + (\ell_{i,v} + r_{i,v} + \ell_{j,v} + r_{j,v}) \leq 2 + \ell_{j,u} \\ r_{i,u} + (\ell_{i,v} + r_{i,v} + \ell_{j,v} + r_{j,v}) \leq 2 + r_{j,u} \end{cases} \\
\textit{Constraint 6:} & \forall i \neq j, \forall u < u', \begin{cases} 2 - \ell_{i,u} - \ell_{j,u} + \sum_{w=u+1}^{u'} (\ell_{i,w} + r_{i,w}) \geq \ell_{j,u'} + r_{j,u'} \\ 2 - r_{i,u} - r_{j,u} + \sum_{w=u+1}^{u'} (\ell_{i,w} + r_{i,w}) \geq \ell_{j,u'} + r_{j,u'} \end{cases}
\end{array}
$$

Moreover, this set of constraints involves $\mathcal{O}(n^2)$ 0–1 variables and $\mathcal{O}(n^4)$ constraints.

Proof. Translating the conditions of Definition 12 into these constraints is straightforward. See Appendix E.2 for the details. ◀

Finally, the choice of the objective function in the linear program follows from the cost function used for Balanced Mobiles.

▶ **Theorem 16.** *The optimal mobiles for the Balanced Mobiles problem with weights $w_1, \ldots, w_n$ are those that satisfy the constraints of Proposition 15 and minimize the objective function*

$$\sum_{u=1}^{n-1} \left| \sum_{i=1}^{n} (\ell_{i,u} - r_{i,u}) \cdot w_i \right|$$

The solutions to this program can be computed (naively) in time $2^{\mathcal{O}(n^2)}$ *and space* $\mathcal{O}(n^2)$.

Proof. The cost associated to each internal node u of a mobile is $\left|\sum_{i \in L_u} w_i - \sum_{i \in R_u} w_i\right|$, where L_u (resp. R_u) are the leaves in the left (resp. right) subtree of node u. We also have $\sum_{i \in L_u} w_i = \sum_{i=1}^{n} \ell_{i,u} \cdot w_i$ and $\sum_{i \in R_u} w_i = \sum_{i=1}^{n} r_{i,u} \cdot w_i$. The total cost is the sum over all the internal nodes of these quantities. ◀

Note that the constraints specified in Proposition 15 only express the fact that the mobiles are correctly constructed. It does not say anything about the imbalance. Consequently, other Huffman-like problems can be cast in this linear program, using other objective functions. This is the case for instance of Huffman coding with unequal letter costs.

▶ **Theorem 17.** *The optimal mobiles for the problem Huffman coding with unequal letter costs* α, β *and weights* $w_1, \ldots, w_n$ *are those that satisfy the constraints of Proposition 15 and minimize the objective function*

$$\sum_{u=1}^{n-1} \sum_{i=1}^{n} (\alpha \cdot \ell_{i,u} + \beta \cdot r_{i,u}) \cdot w_i$$

▶ **Corollary 18.** *Balanced Mobiles and Huffman coding with unequal letter costs can be recast as integer linear programs with* $\mathcal{O}(n^2)$ *variables and* $\mathcal{O}(n^4)$ *constraints. Moreover, the program for Huffman coding with unequal letter costs contains only binary variables.*

Proof. The objective function for Balanced Mobiles is $\sum_{u=1}^{n-1} |\sum_{i=1}^{n} (\ell_{i,u} - r_{i,u}) \cdot w_i|$, which is not linear. However, we can introduce new variables $x_1, \ldots, x_{n-1}$ together with constraints $x_u \geq \sum_{i=1}^{n} (\ell_{i,u} - r_{i,u}) \cdot w_i$ and $x_u \geq -\sum_{i=1}^{n} (\ell_{i,u} - r_{i,u}) \cdot w_i$ (for all u) and a new objective function $\sum_{u=1}^{n-1} x_u$. This program has the same solutions as the original one, and it is linear.

The objective function for Generalized Huffman Coding is already linear. ◀

▶ Remark. We point out that the LP relaxations of the linear programs for these problems admit a zero solution (obtained by setting all the $\ell_{i,u}$ and $r_{i,u}$ variables to $1/2$). This rules out the possibility of using a relaxation to obtain approximation algorithms.

Our 0-1 integer linear program for solving Huffman coding with unequal letter costs is an alternative to the one proposed by Karp [19]. The latter contains $\mathcal{O}(\max(\alpha, \beta) \cdot n^2)$ variables and $\mathcal{O}(\max(\alpha, \beta) \cdot n)$ constraints in the worst case (Karp suggests some heuristics to decrease these numbers). Also note that the heuristic methods used in practice for solving integer linear programs may often be better than the above mentioned $2^{\mathcal{O}(n^2)}$ time complexity.

5.2 Efficient enumeration of trees

We give a second (naive) exponential time algorithm, based on a bijection between leaf-labeled, unordered, full binary trees and matchings. It uses exhaustive search by efficiently enumerating all mobiles, and finding the one with minimum imbalance. It also works for any other Huffman-like problem, by changing the cost function which is evaluated on the enumerated trees.

▶ **Proposition 19.** *There is an algorithm that solves Balanced Mobiles exactly in* $\mathcal{O}(2^{(n \log n)/2})$ *time and* $\mathcal{O}(n \log n)$ *space.*

Proof. See Appendix F. ◀

The space complexity is polynomial and the algorithm is highly parallelizable. The worst-case complexity is also better than for the integer linear program. However, it could be the case that the input weights and the heuristics of the LP solvers used in practice make the latter more efficient.

6 Concluding remarks

This work leaves open many questions. The first is to find a polynomial-time algorithm that solves the Balanced Mobiles problem, or to prove that it is NP-hard. This is also open for other Huffman-like problems, such as Huffman coding with unequal letter costs. More generally, what are the cost functions for which there is a polynomial time algorithm, an approximation scheme or a fixed parameter tractable algorithm? Can we describe all the situations in which the Huffman algorithm is optimal?

One direction that could be explored further is to use the bijection between labeled trees and matchings to devise a new integer linear program, and then use a rounding argument to get an approximation algorithm for some cost functions.

References

1 J. Abrahams. Code and parse trees for lossless source encoding. In *Proceedings. Compression and Complexity of Sequences*, pages 145–171, 1997.

2 P. Agapow and A. Purvis. Power of eight tree shape statistics to detect nonrandom diversification: A comparison by simulation of two models of cladogenesis. *Systematic Biology*, 51(6):866–872, 2002.

3 O. Bernardi, B. Duplantier, and P. Nadeau. A bijection between well-labelled positive paths and matchings. In *Séminaire Lotharingien de Combinatoire*, volume 63, page B63e, 2010.

4 M. G. B. Blum and O. François. On statistical tests of phylogenetic tree imbalance: The sackin and other indices revisited. *Mathematical Biosciences*, 195(2):141 – 153, 2005.

5 M. G. B. Blum, O. François, and S. Janson. The mean, variance and limiting distribution of two statistics sensitive to phylogenetic tree balance. *Ann. Appl. Probab.*, 16(4):2195–2214, 2006.

6 P. G. Bradford, M. J. Golin, L. L. Larmore, and W. Rytter. Optimal prefix-free codes for unequal letter costs: Dynamic programming with the Monge property. *Journal of Algorithms*, 42(2):277–303, 2002.

7 G. Cardona, A. Mir, and F. Rossello. Exact formulas for the variance of several balance indices under the yule model. *Journal of Mathematical Biology*, 67(6):1833–1846, 2013.

8 W. Y. C. Chen. A general bijective algorithm for trees. *Proc. Nat. Acad. Sci. U.S.A.*, 87(24):9635—9639, 1990.

9 V. Choi and M. J. Golin. Lopsided trees, I: Analyses. *Algorithmica*, 31(3):240–290, 2001.

10 D. H. Colless. Review of "Phyogenetics: The theory and practice of phylogenetic systematics". *Systematic Zoology*, 31:100–104, 1982.

11 C. R. Glassey and R. M. Karp. On the optimality of huffman trees. *SIAM Journal on Applied Mathematics*, 31(2):368–378, 1976.

12 M. J. Golin, C. Mathieu, and N. E. Young. Huffman coding with letter costs: A linear-time approximation scheme. *SIAM J. Comput.*, 41(3):684–713, 2012.

13 M. J. Golin and G. Rote. A dynamic programming algorithm for constructing optimal prefix-free codes with unequal letter costs. *IEEE Transactions on Information Theory*, 44(5):1770–1781, 1998.

14 M. J. Golin and N. Young. Prefix codes: Equiprobable words, unequal letter costs. *SIAM Journal on Computing*, 25(6):1281–1292, 1996.

15 M. C. Golumbic. Combinatorial merging. *IEEE Trans. Computers*, 25(11):1164–1167, 1976.

16 S. B. Heard. Patterns in tree balance among cladistic. phenetic, and randomly generated phylogenetic trees. *Evolution*, 46(6):1818–1826, 1992.

17 S. B. Heard and G. H. Cox. The shapes of phylogenetic trees of clades, faunas, and local assemblages: Exploring spatial pattern in differential diversification. *The American Naturalist*, 169(5):E107–E118, 2007.

18 D.A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

19 R. M. Karp. Minimum-redundancy coding for the discrete noiseless channel. *Information Theory, IRE Transactions on*, 7(1):27–38, 1961.

20 M. Kirkpatrick and M. Slatkin. Searching for evolutionary patterns in the shape of a phylogenetic tree. *Evolution*, 47(4):1171–1181, 1993.

21 D. E. Knuth. Huffman's algorithm via algebra. *Journal of Combinatorial Theory, Series A*, 32(2):216 – 224, 1982.

22 D. E. Knuth. *The Art of Computer Programming, Volume 3: (2nd Ed.) Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998.

23 A. McKenzie and M. Steel. Distributions of cherries for two models of trees. *Mathematical Biosciences*, 164:81–92, 2000.

24 A. Mir, F. Rossello, and L. Rotger. A new balance index for phylogenetic trees. *Mathematical Biosciences*, 241(1):125 – 136, 2013.

25 A. Mooers and S. Heard. Inferring evolutionary process from phylogenetic tree shape. *Quarterly Review of Biology*, 72:31–54, 1997.

26 R. Morris. Some theorems on sorting. *SIAM Journal on Applied Mathematics*, 17(1):1–6, 1969.

27 Jr Parker, D. S. Combinatorial merging and huffman's algorithm. *IEEE Trans. Comput.*, 28(5):365–367, 1979.

28 Jr Parker, D. S. Conditions for optimality of the huffman algorithm. *SIAM J. Comput.*, 27(1):317–317, 1998.

29 M. J. Sackin. "good" and "bad" phenograms. *Systematic Biology*, 21(2):225–226, 1972.

30 K. Shao and R. Sokal. Tree balance. *Systematic Zoology*, 39(3):266–276, 1990.

31 B. Varn. Optimal variable length codes (arbitrary symbol cost and equal code word probability). *Information and Control*, 19(4):289 – 301, 1971.

A Algorithms

A.1 Huffman algorithm for the Colless index

Algorithm 1: Huffman algorithm for the Colless index

Input: weights $w_1 \leq \cdots \leq w_n$
Output: imbalance $\mathsf{Huffman}(w_1, \ldots, w_n)$
if $n = 2$ **then**
  Return $|w_2 - w_1|$
else
  Return $|w_2 - w_1| + \mathsf{Huffman}(\mathsf{Sort}(w_1 + w_2, w_3, \ldots, w_n))$

In this procedure, $w_1 + w_2$ is not always the smallest weight among $w_1 + w_2, w_3, \ldots, w_n$. This is why they are sorted in increasing order. In practice, it is better to use a min-heap so as to extract the two smallest weights and then re-insert their sum efficiently. Thus, the running

time can be made $\mathcal{O}(n \log n)$. The algorithm can also be modified to output a mobile that achieves the resulting imbalance. Ties can be handled arbitrarily, as it may change the shape of the mobile but not its imbalance.

The Huffman algorithm does not solve the general Balanced Mobiles problem, nor it is an approximation algorithm, as shown in Figure 7.

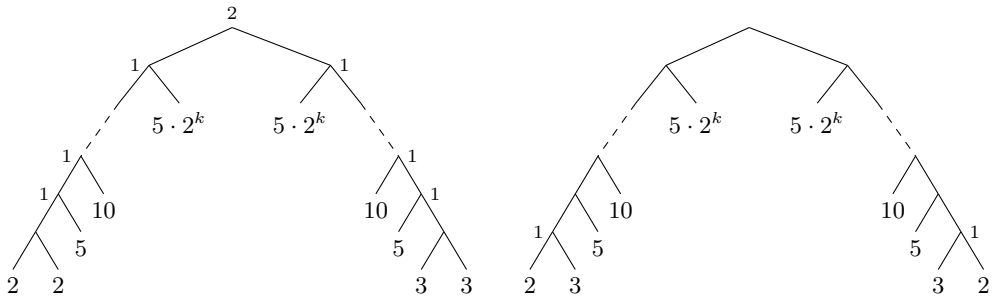


Figure 7 Two mobiles built on weights $\{2, 2, 3, 3\} \cup \{5 \cdot 2^j, 5 \cdot 2^j\}_{j=0}^{k}$. The left mobile is obtained with Huffman's algorithm, it has imbalance $4 + 2k$. The right mobile is optimal, it has imbalance 2.

A.2 Relaxation of Huffman for small imbalance: R-Huffman

Algorithm 2: R-Huffman algorithm

Input: weights $w_1 \leq \cdots \leq w_n$, threshold δ
Output: True if there exists a mobile of imbalance less than δ on weights $\{w_1, \ldots, w_n\}$, False otherwise

```
1 if n = 2 then
2     Return the boolean value |w_1 − w_2| ≤ δ
3 else
4     foreach i ≥ 2 such that |w_1 − w_i| ≤ δ do
5         if R-Huffman(Sort(w_1 + w_i, w_2, ..., w_{i−1}, w_{i+1}, ... w_n), δ − |w_1 − w_i|) then
6             Return True
7     Return False
```

Let us replace $\{w_1, \ldots, w_n\}$ by a set $\{(q_j, n_j)\}$ in which the q_j are the distinct weights of the multiset $\{w_1, \ldots, w_n\}$ and the n_j are their multiplicities ($\sum n_j = n$).

The line 4 is executed at most n times. The insertion in the sorted list (line 5) can then be done in $\mathcal{O}(\log n)$. Thus, apart from the recursive calls, the running time is $\mathcal{O}(n \log n)$.

Since the q_j are all distinct, the recursive calls of the line 5 cannot be run twice on the same threshold. Denote by $T(n, \delta)$ the complexity of R-Huffman$(\{w_1, \ldots, w_n\}, \delta)$ (for a worst-case input $\{w_1, \ldots, w_n\}$). We obtain $T(n, \delta) \leq \mathcal{O}(n \log n) + \sum_{i=0}^{\min(\delta,n)} T(n-1, i)$. Moreover, $T(n, 0) = \mathcal{O}(n \log n)$. It is easy to prove by induction that $T(n, \delta) = \mathcal{O}\big(\log(n) n^{\min(\delta,n)+1}\big)$.

Recall that we don't take into account the magnitude of the weights for the space complexity. The algorithm uses a stack of depth $\mathcal{O}(n)$ and $\mathcal{O}(\log n)$ space on each level of the stack to store the position i of the element which is grouped with w_1 (line 4). Overall the space needed is $\mathcal{O}(n \log n)$.

B Proofs for Section 3 (Colless index for unit weights)

B.1 Proof of Proposition 2 (Imbalance of left-complete trees)

We want to show that the imbalance $\mathcal{C}_{L_n}$ of the left-complete tree L_n over n leaves is

$$\begin{cases} \mathcal{C}_{L_1} = 0 \\ \mathcal{C}_{L_{n+1}} = \mathcal{C}_{L_n} + |n|_0 - |n|_1 + 1 \end{cases}$$

where $|n|_0$ (resp. $|n|_1$) is the number of 0 (resp. 1) in the binary representation of n.

Consider the left-complete tree L_n over $n = 2^k + m$ leaves, where $0 \leq m < 2^k$, and replace the leftmost leaf ℓ of depth k by a cherry (so as to obtain L_{n+1}). During this operation, the local imbalances evolve as follows (see Figure 2 for an example):

- the local imbalances of the nodes which are not ancestors of the leaf ℓ do not change
- for each ancestor ν of ℓ, if ℓ is in the right (resp. left) subtree of ν then the local imbalance at node ν decreases (resp. increases) by 1

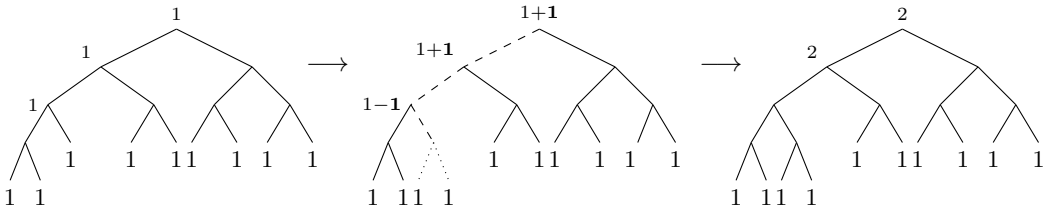


Figure 8 Evolution of the local imbalances between L_9 and L_{10}.

In other words, $\mathcal{C}_{L_{n+1}}$ is obtained from $\mathcal{C}_{L_n}$ by adding $+1$ when going to the left on the path from the root to leaf ℓ, and -1 when going to the right. If we label the nodes at depth k by their binary representation with k bits (the leftmost node is labeled by k zeros, and the rightmost one by k ones), then the quantity to be added is the number of zeros minus the number of ones in the label of node ℓ.

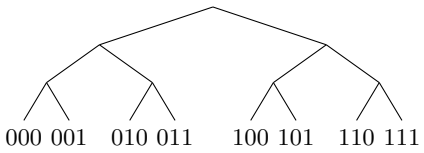


Figure 9 Numbering of the nodes at depth three on 3 bits. When going from 9 to 10 leaves, the imbalance increases of $+1$ since the node to be replaced with a cherry has label 001.

However, the label of the node ℓ is precisely the binary representation of m over k bits. Denote by $\textsc{Pad}(m,k)$ the binary representation of m padded with 0 on the left to be of length k. We have:

$$\begin{aligned} \mathcal{C}_{L_{n+1}} &= \mathcal{C}_{L_n} + |\textsc{Pad}(m,k)|_0 - |\textsc{Pad}(m,k)|_1 \\ &= \mathcal{C}_{L_n} + |2^k + m|_0 - |2^k + m|_1 + 1 \\ &= \mathcal{C}_{L_n} + |n|_0 - |n|_1 + 1 \end{aligned}$$

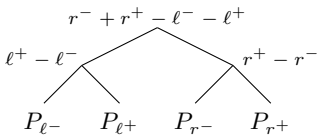
B.2 Proof of Proposition 3 (Optimality of partition trees)

We prove by induction on n that the partition trees have optimal Colless index. The base cases $n = 2$ and $n = 3$ are trivial. We take some $n \geq 4$ and assume that our hypothesis is true up to $n - 1$. We want to show that it is also the case for n.

Consider an optimal tree T with n leaves. Denote by T_L and T_R the left and right subtrees of T, and let ℓ and r be the number of leaves in T_L and T_R respectively ($\ell + r = n$, and $|\ell - r|$ is the local imbalance of the root). We show how to transform T into the partition tree P_n without increasing the imbalance (i.e. $\mathcal{C}_T = \mathcal{C}_{P_n}$).

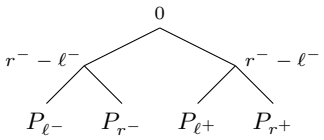
The induction hypothesis is applied first to ℓ and r: the partition trees P_ℓ and P_r on ℓ and r leaves respectively are optimal. We denote by T' the tree T in which T_L and T_R are replaced by P_ℓ and P_r. We must have $\mathcal{C}_T = \mathcal{C}_{T'}$ since T is optimal, $\mathcal{C}_{P_\ell} \leq \mathcal{C}_{T_L}$ and $\mathcal{C}_{P_r} \leq \mathcal{C}_{T_R}$.

If $\ell = 1$ or $r = 1$ then the local imbalance of the root of T' is $n - 2$. However, using Proposition 1, it is easy to prove that $\mathcal{C}_{P_n} < n-2$ (when $n \geq 4$). Consequently, we must have $\ell, r \geq 2$ (since $\mathcal{C}_{T'} \leq \mathcal{C}_{P_n}$). We denote P_ℓ^- and P_r^- the left subtrees of P_ℓ and P_r respectively, and P_ℓ^+ and P_r^+ their right subtrees. We also let ℓ^-, r^-, ℓ^+, r^+ be the number of leaves in all of these subtrees ($\ell^- + \ell^+ = \ell$ and $r^- + r^+ = r$). We can assume *w.l.o.g.* that $\ell \leq r$, $\ell^- \leq \ell^+$ and $r^- \leq r^+$. Since T_ℓ and T_r are partition trees, we must have $\ell^+ - \ell^- \leq 1$ and $r^+ - r^- \leq 1$. The tree T' is as follows:

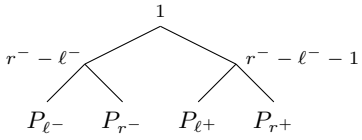


Consider now the three possible cases:

- $\ell^- = \ell^+$ and $r^- = r^+$. The imbalance of T' is: $\mathcal{C}_T = \mathcal{C}_{T'} = \mathcal{C}_{P_{\ell^-}} + \mathcal{C}_{P_{\ell^+}} + \mathcal{C}_{P_r^-} + \mathcal{C}_{P_r^+} + 2(r^- - \ell^-)$. We rotate T' into T'' as follows:



  The new imbalance is $\mathcal{C}_{T''} = \mathcal{C}_{P_{\ell^-}} + \mathcal{C}_{P_{\ell^+}} + \mathcal{C}_{P_{r^-}} + \mathcal{C}_{P_{r^+}} + 2(r^- - \ell^-) = \mathcal{C}_T$. Thus, T'' is still optimal, but the local imbalance of the root is 0 now. Finally, we replace the left and right subtrees of T'' by $P_{\ell^- + r^-}$ so as to obtain P_n. According to the induction hypothesis, this does not increase the imbalance. Thus $\mathcal{C}_T = \mathcal{C}_{P_n}$.
- $\ell^+ = \ell^- + 1$, $r^- = r^+$. The imbalance of T' is: $\mathcal{C}_T = \mathcal{C}_{T'} = \mathcal{C}_{P_{\ell^-}} + \mathcal{C}_{P_{\ell^+}} + \mathcal{C}_{P_{r^-}} + \mathcal{C}_{P_{r^+}} + 2(r^- - \ell^-)$. We rotate T' into T'' as follows:



  T'' is still optimal ($\mathcal{C}_{T''} = \mathcal{C}_T$), but the local imbalance of the root is 1 now. We reuse the induction hypothesis on the left and right subtrees of T'' to turn it into P_n without increasing the imbalance. Thus $\mathcal{C}_T = \mathcal{C}_{P_n}$.
- We use the same kind of reasoning for the last case: $\ell^+ = \ell^- + 1$ and $r^+ = r^- + 1$.

B.3 Proof of Theorem 4 (Optimal imbalance for unit weights)

We first prove that $\mathcal{C}_{2n} = 2\mathcal{C}_n$ and $\mathcal{C}_{2n+1} = 1 + \mathcal{C}_n + \mathcal{C}_{n+1}$ implies $\mathcal{C}_{n+1} = \mathcal{C}_n + |n|_0 - |n|_1 + 1$. Equivalently, we want to show that $\lambda_n = |n|_0 - |n|_1 + 1$, where $\lambda_n = \mathcal{C}_{n+1} - \mathcal{C}_n$. The proof is by induction on n. The base case is true since $\lambda_1 = \mathcal{C}_2 - \mathcal{C}_1 = 0 - 0 = |1|_0 - |1|_1 + 1$. Consider now $n > 1$ and assume that the result is true up to $n-1$. There are two cases:

- If $n = 2m$ (n even) then $\lambda_{2m} = \mathcal{C}_{2m+1} - \mathcal{C}_{2m} = 1 + \mathcal{C}_m + \mathcal{C}_{m+1} - 2\mathcal{C}_m = 1 + \lambda_m$ (second case of Theorem 4). So, using the induction hypothesis, $\lambda_{2m} = 1 + |m|_0 - |m|_1 + 1$. However $|n|_0 = |2m|_0 = |m|_0 + 1$ and $|n|_1 = |2m|_1 = |m|_1$. Thus, $\lambda_n = |n|_0 - |n|_1 + 1$.
- The other case $n = 2m + 1$ (n odd) can be proved similarly.

We now show $\mathcal{C}_n = 2 \cdot (n \bmod 2^k) + \sum_{i=0}^{k-1} (-1)^{b_i} \cdot (n \bmod 2^{i+1})$ using the fourth and fifth definitions of Theorem 4. The proof is by induction on n. Observe that it is true for all $n \leq 8$, and consider the binary representation $b_k b_{k-1} \dots b_0$ of an integer $n > 8$. There are two cases:

- If $n = 2m$ (n is even) then $b_0 = 0$. According to the fourth definition of Theorem 4, we have $\mathcal{C}_n = 2 \cdot \mathcal{C}_m$. Thus, using the induction hypothesis:

$$\begin{aligned}
\mathcal{C}_n &= 2 \cdot \left(2 \cdot (m \bmod 2^{k-1}) + \sum_{i=0}^{k-2} (-1)^{b_{i+1}} \cdot (m \bmod 2^{i+1}) \right) \\
&= 2 \cdot (n \bmod 2^k) + \sum_{i=0}^{k-2} (-1)^{b_{i+1}} \cdot (n \bmod 2^{i+2}) \quad \text{since } b_0 = 0 \\
&= 2 \cdot (n \bmod 2^k) + \sum_{i=0}^{k-1} (-1)^{b_i} \cdot (n \bmod 2^{i+1})
\end{aligned}$$

- If $n = 2m + 1$ (n is odd) then $b_0 = 1$. Assume that there exists i such that $b_i = 0$ (otherwise it is easy to directly prove that the result is true for $n = 2^{k+1} - 1$). Let i_0 be the position of the rightmost 0 (i.e. $n = b_k \dots b_{i_0+1} 01 \dots 1$). According to the fifth definition of Theorem 4, we have $\mathcal{C}_n = \mathcal{C}_m + \mathcal{C}_{m+1} + 1$. Thus, using the induction hypothesis:

$$\begin{aligned}
\mathcal{C}_n = &\left(2 \cdot (m \bmod 2^{k-1}) + \sum_{i=0}^{k-2} (-1)^{b_{i+1}} \cdot (m \bmod 2^{i+1}) \right) + \left(2 \cdot (m+1 \bmod 2^{k-1}) \right. \\
&\left. + \sum_{i=0}^{i_0-1} (-1)^{1+b_{i+1}} \cdot (m+1 \bmod 2^{i+1}) + \sum_{i=i_0}^{k-2} (-1)^{b_{i+1}} \cdot (m+1 \bmod 2^{i+1}) \right) + 1
\end{aligned}$$

However, for all $i < i_0$ we have $(m+1 \bmod 2^i) = 0$, and $(m+1 \bmod 2^{i_0}) = 2^{i_0 - 1}$. Thus:

$$\begin{aligned}
\mathcal{C}_n = &\left(2 \cdot (m \bmod 2^{k-1}) + \sum_{i=0}^{k-2} (-1)^{b_{i+1}} \cdot (m \bmod 2^{i+1}) \right) + \left(2 \cdot (m+1 \bmod 2^{k-1}) \right. \\
&\left. - 2^{i_0 - 1} + \sum_{i=i_0}^{k-2} (-1)^{b_{i+1}} \cdot (m+1 \bmod 2^{i+1}) \right) + 1
\end{aligned}$$

Moreover, $\sum_{i=0}^{i_0-1}(-1)^{b_{i+1}} \cdot (m \bmod 2^{i+1}) = -2^{i_0-1} + i_0$. Consequently:

$$\mathcal{C}_n = 4 \cdot (m \bmod 2^{k-1}) + 2 \cdot \sum_{i=0}^{k-2}(-1)^{b_{i+1}} \cdot (m \bmod 2^{i+1}) + \sum_{i=i_0}^{k-2}(-1)^{b_{i+1}} - i_0 + 3$$

Finally, $\sum_{i=0}^{i_0-1}(-1)^{b_{i+1}} = 2 - i_0$. It implies that:

$$\begin{aligned}
\mathcal{C}_n &= 4 \cdot (m \bmod 2^{k-1}) + 2 \cdot \sum_{i=0}^{k-2}(-1)^{b_{i+1}} \cdot (m \bmod 2^{i+1}) + \sum_{i=0}^{k-2}(-1)^{b_{i+1}} + 1 \\
&= 4 \cdot b_{k-1} \dots b_1 + 2 \cdot \sum_{i=0}^{k-2}(-1)^{b_{i+1}} \cdot b_{i+1} \dots b_1 + \sum_{i=0}^{k-2}(-1)^{b_{i+1}} + 1 \\
&= 2 \cdot (b_{k-1} \dots b_1 0 + 1) + \sum_{i=0}^{k-2}(-1)^{b_{i+1}} \cdot (b_{i+1} \dots b_1 0 + 1) - 1 \\
&= 2 \cdot (n \bmod 2^k) + \sum_{i=0}^{k-2}(-1)^{b_{i+1}} \cdot (n \bmod 2^{i+2}) - 1 \\
&= 2 \cdot (n \bmod 2^k) + \sum_{i=0}^{k-1}(-1)^{b_i} \cdot (n \bmod 2^{i+1})
\end{aligned}$$

C Proof of Theorem 6 (Powers-of-two weights)

▶ **Definition 20.** A mobile M is said to be *irregular* if it verifies the following conditions:

- M is an optimal mobile built on powers-of-two weights
- M cannot be built by the Huffman algorithm
- the imbalance of M is *less* than the one obtained by the Huffman algorithm on the same weights.

Theorem 6 precisely states that if the weights are powers of 2, then there is no irregular mobile. We prove it by contradiction. Assume from now on that such mobiles exist. We are going to choose one of them that verifies some minimality conditions.

▶ **Definition 21.** For any mobile M, define the *maximum* $\max M$ of M to be the greatest weight on the leaves on M. Let m be the smallest possible maximum for irregular mobiles. We choose $\mathcal{M}$ to be an irregular mobile whose maximum is m, and whose number of leaves is as small as possible among all the irregular mobiles of maximum m.

We also define $\mathcal{M}_L$ and $\mathcal{M}_R$ to be the left and right subtrees of $\mathcal{M}$. We state several lemmas allowing to prove that $\mathcal{M}$ cannot exist.

▶ **Lemma 22.** *We have:* $m \geq 2$.

Proof. According to Proposition 5, there is no irregular mobile whose weights are all equal to one. ◀

▶ **Lemma 23.** *We can assume w.l.o.g. that* $\mathcal{M}_L$ *and* $\mathcal{M}_R$ *are built by the* Huffman *algorithm.*

Proof. Note that $\mathcal{M}_L$ is an optimal mobile whose maximum is at most m, and that has fewer leaves than $\mathcal{M}$. Given the minimality conditions on $\mathcal{M}$, the mobile $\mathcal{M}_L$ cannot be irregular. According to Definition 20, this implies that we can replace $\mathcal{M}_L$ by a mobile built by the Huffman algorithm on the same weights without increasing the imbalance. The same reasoning applies to $\mathcal{M}_R$. ◀

▶ **Lemma 24.** *The mobile $\mathcal{M}$ has either one or two leaves of weight one.*

Proof. If $\mathcal{M}$ does not contain any leaf of weight one then we could divide all the weights by two so as to obtain an irregular mobile of maximum $m/2$, which is a contradiction.

Suppose now that $\mathcal{M}$ has at least three leaves of weight one. We can assume without loss of generality that $\mathcal{M}_L$ contains at least two such leaves. However, according to Lemma 23, $\mathcal{M}_L$ is built by the Huffman algorithm. Thus, there must be two siblings leaves of unit weights in $\mathcal{M}_L$. Replace them by a single leaf of weight two so as to obtain a new irregular mobile $\mathcal{M}'$. According to Lemma 22, we have $\max \mathcal{M} = \max \mathcal{M}' = m$. However, $\mathcal{M}'$ has fewer leaves than $\mathcal{M}$, which contradicts the choice of Definition 21. ◀

▶ **Lemma 25.** *If $\mathcal{M}_L$ (resp. $\mathcal{M}_R$) has a leaf of weight one, then it contains at least two leaves.*

Proof. Assume that $\mathcal{M}_L$ consists of a single leaf of weight one. Denote by $w_1 \leq \cdots \leq w_n$ the weights on the leaves of $\mathcal{M}$ (we have $w_1 = 1$). The imbalance of $\mathcal{M}$ is $\mathcal{C}_{\mathcal{M}} = (\sum_{i=2}^{n} w_i - 1) + \mathcal{C}_{\mathcal{M}_R}$.

Let $\mathcal{M}'$ be the mobile obtained as follows from the mobile $\mathcal{M}_R$:

- replace the leaf of weight w_2 by a cherry
- put the weights $w_1 = 1$ and w_2 on the leaves of this subtree.

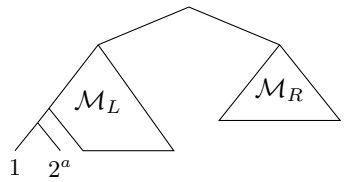
This operation adds a leaf of weight 1, a node of imbalance $w_2 - 1$ and increases each of the other local imbalances in $\mathcal{M}_R$ by at most one. Since there are $n - 2$ nodes in $\mathcal{M}_R$ we have $\mathcal{C}_{\mathcal{M}'} \leq (w_2 - 1) + \mathcal{C}_{\mathcal{M}_R} + n - 2$.

According to Lemma 24, $\mathcal{M}$ contains at most two leaves of weight one. Thus, for $n \geq 3$ we have $(w_2 - 1) + n - 2 < (\sum_{i=2}^{n} w_i) - 1$. Consequently $\mathcal{C}_{\mathcal{M}'} < \mathcal{C}_{\mathcal{M}}$. However, $\mathcal{M}$ is an optimal mobile (since it is irregular). This is a contradiction. ◀

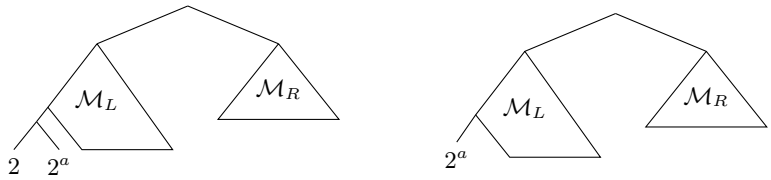
We study the two possibilities of Lemma 24 in the next two sections. For each of these cases, we show that the mobile $\mathcal{N}$ built by the Huffman algorithm on the same weights as $\mathcal{M}$ is also optimal, which is a contradiction.

C.1 First case of the proof

We suppose that $\mathcal{M}$ contains a single leaf of weight one (first case of Lemma 24). Assume without loss of generality that this leaf is in $\mathcal{M}_L$. According to Lemmas 23 and 25, there exists some $a \geq 1$ such that $\mathcal{M}$ is as follows:



Denote by $\mathcal{M}^+$ the mobile $\mathcal{M}$ in which the leaf of weight one has been replaced by a leaf of weight two. Similarly, let $\mathcal{M}^-$ be the mobile $\mathcal{M}$ in which the cherry with weights 1 and 2^a is replaced by a leaf of weight 2^a. These mobiles are as follows ($\mathcal{M}^+$ is on the left and $\mathcal{M}^-$ on the right):



▶ **Lemma 26.** *$\mathcal{M}^+$ and $\mathcal{M}^-$ cannot be irregular.*

Proof. If $\mathcal{M}^+$ (resp. $\mathcal{M}^-$) is irregular then we can divide all its weights by two so as to obtain an irregular mobile of maximum $m/2$, which contradicts the minimality assumptions made on $\mathcal{M}$. ◀

▶ **Lemma 27.** *We have: $\mathcal{C}_{\mathcal{M}} = \frac{1}{2}\mathcal{C}_{\mathcal{M}^+} + \frac{1}{2}\mathcal{C}_{\mathcal{M}^-} + 2^{a-1}$.*

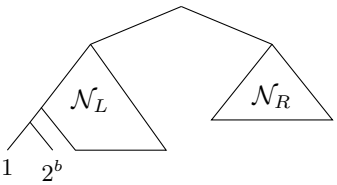
Proof. For each node ν, denote by $c(\nu)$ the local imbalance of ν in $\mathcal{M}$. Similarly, let $c^+(\nu)$ and $c^-(\nu)$ be the local imbalances in $\mathcal{M}^+$ and $\mathcal{M}^-$. Recall that $\mathcal{C}_{\mathcal{M}}$ is the sum of the $c(\nu)$ for all the nodes ν of $\mathcal{M}$.

Let us study how the local imbalances evolve in $\mathcal{M}^+$ and $\mathcal{M}^-$ with respect to those in the mobile $\mathcal{M}$. First, note that they do not change for those nodes that are not ancestors of the leaf of weight one: $c(\nu) = c^+(\nu) = c^-(\nu)$. Thus, for such nodes we have $c(\nu) = \frac{1}{2}c^+(\nu) + \frac{1}{2}c^-(\nu)$.

Consider now the ancestor nodes of the leaf of weight one, excluding its direct parent ν_0. For each of these nodes ν, the subtree of ν containing the leaf of weight one has an odd weight, whereas the other subtree has an even weight. Consequently $c(\nu) > 0$. It implies that $c^+(\nu) = c(\nu) + 1$ and $c^-(\nu) = c(\nu) - 1$, or $c^+(\nu) = c(\nu) - 1$ and $c^-(\nu) = c(\nu) + 1$. Thus, we must have $c(\nu) = \frac{1}{2}c^+(\nu) + \frac{1}{2}c^-(\nu)$. It is also easy to see that for the direct parent ν_0 of ν (that disappears in $\mathcal{M}^-$), we have $c(\nu_0) = \frac{1}{2}c^+(\nu_0) + 2^{a-1}$.

We sum up all the previous equalities so as to obtain $\mathcal{C}_{\mathcal{M}} = \frac{1}{2}\mathcal{C}_{\mathcal{M}^+} + \frac{1}{2}\mathcal{C}_{\mathcal{M}^-} + 2^{a-1}$. ◀

Consider now the mobile $\mathcal{N}$ built by the Huffman algorithm on the same weights as $\mathcal{M}$. This mobile is as follows (for some $b \geq 1$):



Once again, we define $\mathcal{N}^+$ and $\mathcal{N}^-$ to be the mobiles obtained from $\mathcal{N}$ by replacing the weight one by two, or deleting the leaf of weight one.

▶ **Lemma 28.** *$\mathcal{N}^+$ and $\mathcal{N}^-$ can be built by the Huffman algorithm.*

Proof. Let $w_1 < w_2 \leq \cdots \leq w_n$ be the weights of $\mathcal{N}$ (recall that $w_1 = 1$ and the weights are powers of two). Whenever the Huffman algorithm compares the weights of two (disjoint) subsets S_1 and S_2 of $\{w_1, \ldots, w_n\}$, the output of the comparison does not change if w_1 is set to 0 or 2. Indeed, assume for instance that S_1 contains the weight w_1. All of the other weights

are powers of two greater than one. Thus, the weight W_{S_1} of S_1 is odd whereas W_{S_2} is even. It implies that W_{S_1} and W_{S_2} cannot be equal. Therefore, $W_{S_1} \leq W_{S_2} \Leftrightarrow W_{S_1} \pm 1 \leq W_{S_2}$ and $W_{S_2} \leq W_{S_1} \Leftrightarrow W_{S_2} \leq W_{S_1} \pm 1$.

Consequently, the results of the comparisons performed by the Huffman algorithm during the construction of $\mathcal{N}$ do not change if w_1 is set to 0 or 2 (in the first case, w_1 is in fact removed). Therefore, $\mathcal{N}^+$ and $\mathcal{N}^-$ are correctly built by the Huffman algorithm. ◀

The reasoning carried out in the proof of Lemma 27 also applies to $\mathcal{N}$:

▶ **Lemma 29.** *We have:* $\mathcal{C}_{\mathcal{N}} = \frac{1}{2}\mathcal{C}_{\mathcal{N}^+} + \frac{1}{2}\mathcal{C}_{\mathcal{N}^-} + 2^{b-1}$.

We can now conclude the first case of our proof. According to Lemma 26, $\mathcal{M}^+$ and $\mathcal{M}^-$ cannot be irregular. On the other hand, we have exhibited two mobiles $\mathcal{N}^+$ an $\mathcal{N}^-$ that have the same leaf set as $\mathcal{M}^+$ and $\mathcal{M}^-$ respectively, and that can built by the Huffman algorithm (Lemma 28). Using Definition 20, it implies $\mathcal{C}_{\mathcal{N}^+} \leq \mathcal{C}_{\mathcal{M}^+}$ and $\mathcal{C}_{\mathcal{N}^-} \leq \mathcal{C}_{\mathcal{M}^-}$. Moreover, since $\mathcal{N}$ and $\mathcal{M}$ have the same leaf set and $\mathcal{N}$ is built by the Huffman algorithm, 2^b must be the second smallest weight in $\mathcal{M}$, thus $2^b \leq 2^a$. Using the equalities of Lemmas 27 and 29 we obtain:

$$\mathcal{C}_{\mathcal{N}} = \frac{1}{2}\mathcal{C}_{\mathcal{N}^+} + \frac{1}{2}\mathcal{C}_{\mathcal{N}^-} + 2^{b-1} \leq \frac{1}{2}\mathcal{C}_{\mathcal{M}^+} + \frac{1}{2}\mathcal{C}_{\mathcal{M}^-} + 2^{a-1} = \mathcal{C}_{\mathcal{M}}$$

However, since $\mathcal{M}$ is irregular we must have $\mathcal{C}_{\mathcal{N}} > \mathcal{C}_{\mathcal{M}}$. This is a contradiction.

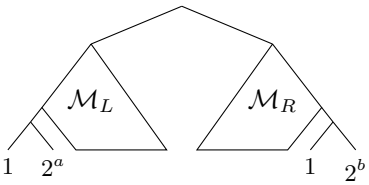
C.2 Second case of the proof

Assume now that $\mathcal{M}$ contains exactly two leaves of weight one (second case of Lemma 24). We first prove that these leaves cannot be in the same subtree $\mathcal{M}_L$ or $\mathcal{M}_R$ of $\mathcal{M}$.

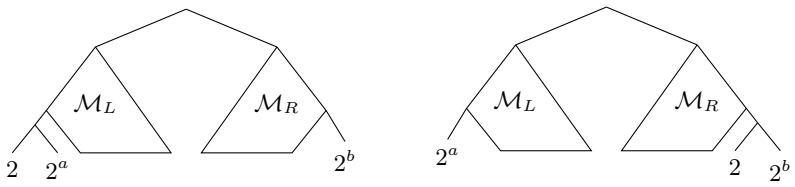
▶ **Lemma 30.** *Neither $\mathcal{M}_L$ nor $\mathcal{M}_R$ can contain two leaves of weight one.*

Proof. Suppose for instance that there are two leaves of weight one in $\mathcal{M}_L$. According to Lemma 23, we can assume that $\mathcal{M}_L$ is built by Huffman. Thus, it must have two sibling leaves of weight one. Replace these siblings by a single leaf of weight two. We get an irregular mobile with fewer leaves than $\mathcal{M}$ and whose maximum is m. This is a contradiction. ◀

According to Lemmas 23, 25 and 30 there are some $a, b \geq 1$ such that $\mathcal{M}$ is as follows:



Let $\mathcal{M}^+$ be the mobile $\mathcal{M}$ in which the leaf of weight one in $\mathcal{M}_L$ has been replaced with a leaf of weight two, and the leaf of weight one in $\mathcal{M}_R$ has been removed. Define also $\mathcal{M}^-$ using the reverse operation. These mobiles are as follows ($\mathcal{M}^+$ is on the left and $\mathcal{M}^-$ on the right):



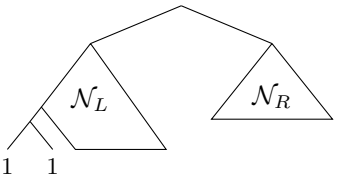
We state several lemmas similar to the ones described in the previous section.

▶ **Lemma 31.** *$\mathcal{M}^+$ and $\mathcal{M}^-$ cannot be irregular.*

▶ **Lemma 32.** *We have: $\mathcal{C}_{\mathcal{M}} \geq \frac{1}{2}\mathcal{C}_{\mathcal{M}^+} + \frac{1}{2}\mathcal{C}_{\mathcal{M}^-} + 2^{a-1} + 2^{b-1} - 2$.*

Proof. We study the local imbalances as in the proof of Lemma 27. The only difference is the root ν of $\mathcal{M}$ for which we can only state that $c^+(\nu), c^-(\nu) \leq c(\nu) + 2$. Consequently, $c(\nu) \geq \frac{1}{2}c^+(\nu) + \frac{1}{2}c^+(\nu) - 2$. ◀

Define now the mobile $\mathcal{N}$ built by Huffman on the same weights as $\mathcal{M}$. This mobile is as follows:



Let $\mathcal{N}'$ be the mobile obtained from $\mathcal{N}$ by replacing the two sibling leaves of weight one by a single leaf of weight two (the mobiles $\mathcal{N}'$, $\mathcal{M}^+$ and $\mathcal{M}^-$ have the same weights on their leaves).

▶ **Lemma 33.** *$\mathcal{N}'$ can be built by Huffman and $\mathcal{C}_{\mathcal{N}} = \mathcal{C}_{\mathcal{N}'}$.*

We conclude the second case of our proof. Since $\mathcal{M}^+$ and $\mathcal{M}^-$ cannot be irregular and $\mathcal{N}'$ is optimal (Lemma 33), we have $\mathcal{C}_{\mathcal{N}'} \leq \mathcal{C}_{\mathcal{M}^+}$ and $\mathcal{C}_{\mathcal{N}'} \leq \mathcal{C}_{\mathcal{M}^-}$. According to Lemmas 32 and 33 we obtain:

$$\mathcal{C}_{\mathcal{N}} = \frac{1}{2}\mathcal{C}_{\mathcal{N}'} + \frac{1}{2}\mathcal{C}_{\mathcal{N}'} \leq \frac{1}{2}\mathcal{C}_{\mathcal{M}^+} + \frac{1}{2}\mathcal{C}_{\mathcal{M}^-} + 2^{a-1} + 2^{b-1} - 2 \leq \mathcal{C}_{\mathcal{M}}$$

However, since $\mathcal{M}$ is irregular, $\mathcal{C}_{\mathcal{N}} > \mathcal{C}_{\mathcal{M}}$. This is a contradiction. It concludes the proof of Theorem 6.

D Proof of Theorem 11 (Perfectly balanced mobiles)

We prove the result by induction on n. The base case $n = 2$ is trivial. Assume that it is true up to $n - 1$ for some $n \geq 3$. We want to show that is is also true for n.

We first consider a set of powers-of-two weights $w_1 \leq \cdots \leq w_n$ whose sum $\sum_{i=1}^n w_i$ is also a power of two. Since the weights are coprime, we must have $w_1 = 1$. Moreover, since $\sum_{i=1}^n w_i$ is a power of two, the weight w_2 is also equal to 1. Thus, there exists a perfectly balanced mobile on weights $\{w_1, \ldots, w_n\}$ if and only if there exists one on weights $\{2, w_3, \ldots, w_n\}$. It is easy to see that the two conditions of the statement of Theorem 11 still hold for weights $\{2, w_3, \ldots, w_n\}$. According to our induction hypothesis, it implies that there exists a perfectly balanced mobile M on these weights. Finally, it suffices to replace a leaf of weight 2 in M by two leaves of weights 1 so as to obtain a perfectly balanced mobile on weights $\{w_1, \ldots, w_n\}$.

We now take a perfectly balanced mobile M on some weights $w_1 \leq \cdots \leq w_n$. We want to show that these weights verify the two conditions of the statement of the theorem. First remark that we can assume that the leaves of weights w_1 and w_2 are siblings, and $w_1 = w_2$ (otherwise the imbalance cannot be zero). Thus, if we replace these two leaves by a single leaf of weight $2w_1$ we obtain a perfectly balanced mobile on weights $\{2w_1, w_3, \ldots, w_n\}$.

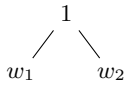
According to the induction hypothesis, both the weights in $\{2w_1, w_3, \dots, w_n\}$ and the sum $2w_1 + w_3 + \cdots + w_n$ are powers of two. It implies directly that the weights $\{w_1, \dots, w_n\}$ verify the two points of Theorem 11.

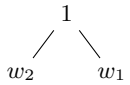
E Proofs for Section 5.1 (Integer Linear Programming)

E.1 Proof of Proposition 14

We prove by induction on n a stronger proposition P_n: for any $m \geq 1$, if an assignment of the boolean variables $(\ell_{i,u}, r_{i,u})_{1 \leq i \leq n, 1 \leq u \leq m}$ is proper then $m = n - 1$ and it is a correct assignment.

We first show that P_2 is true. Let us consider a proper assignment of the variables $(\ell_{i,u}, r_{i,u})_{1 \leq i \leq 2, 1 \leq u \leq m}$ that verifies the conditions of Definition 12. Suppose that $m \geq 2$. According to Conditions 1 and 3, either $\ell_{1,2} = 1$ and $r_{2,2} = 1$, or $r_{1,2} = 1$ and $\ell_{2,2} = 1$. It implies that $\ell_{1,1} = \ell_{2,1}$ and $r_{1,1} = r_{2,1}$ (Condition 5). Thus, according to Condition 1, $\ell_{1,1} = \ell_{2,1} = 0$ or $r_{1,1} = r_{2,1} = 0$, which contradicts Condition 3 on node $u = 1$. Consequently, we must have $m = 1$. There are four variables $\ell_{1,1}, r_{1,1}, \ell_{2,1}, r_{2,1}$ and it is easy to see that Conditions 1 to 3 imply only two possible assignments:

- $\ell_{1,1} = r_{2,1} = 1$ and $r_{1,1} = \ell_{2,1} = 0$, which is correct since it corresponds to 

- $\ell_{1,1} = r_{2,1} = 0$ and $r_{1,1} = \ell_{2,1} = 1$, which is correct since it corresponds to 

Thus, proposition P_2 is true.

Let us now take $n > 2$ and assume that $P_{n'}$ is true for all $n' < n$. We consider a proper assignment of the variables $\Omega = (\ell_{i,u}, r_{i,u})_{1 \leq i \leq n, 1 \leq u \leq m}$. According to Conditions 1, 2 and 3 we can partition $\{1, \dots, n\}$ into two non-empty sets $L = \{i : \ell_{i,1} = 1\}$ and $R = \{i : r_{i,1} = 1\}$ such that $L \cap R = \varnothing$ and $L \cup R = \{1, \dots, n\}$. We define $L' = \{u > 1 : \exists i \in L \text{ s.t. } \ell_{i,u} \neq 0 \text{ or } r_{i,u} \neq 0\}$ and $\Omega_L = (\ell_{i,u}, r_{i,u})_{i \in L, u \in L'}$. Similarly, let $R' = \{u > 1 : \exists i \in R \text{ s.t. } \ell_{i,u} \neq 0 \text{ or } r_{i,u} \neq 0\}$ and $\Omega_R = (\ell_{i,u}, r_{i,u})_{i \in R, u \in R'}$. Note that none of the $u \in \{2, \dots, m\}$ can be simultaneously in L' and R', otherwise it would contradict Condition 5 ($\ell_{i,u} + r_{i,u} = \ell_{j,u} + r_{j,u} = 1$ for some $i \in L$, $j \in R$ would imply $\ell_{i,1} = \ell_{j,1}$ and $r_{i,1} = r_{j,1}$, which is not possible by definition of L and R). Thus, L' and R' are a partition of $\{2, \dots, m\}$.

Using our induction hypothesis, we intend to show that the restrictions of the initial assignment to Ω_L and Ω_R correspond respectively to two valid mobiles M_L and M_R, whose weights are indexed by L and R, and nodes by L' and R'. Consequently, the initial assignment of Ω will be a correct representation of the mobile M whose left and right subtrees are M_L and M_R.

We carry out the proof for M_L only, and $|L| \geq 2$ (the case $|L| = 1$ is trivial). Knowing that Conditions 1 to 6 are true for the assignment of Ω, we want to show that it is still the case for the restriction to Ω_L:

- Conditions 1, 4, 5 and 6 are trivially true in Ω_L (since they are for Ω)
- If Condition 3 does not hold for Ω_L, then there exists $u \in L'$ such that $\ell_{i,u} = 0$ for all $i \in L$, or $r_{i,u} = 0$ for all $i \in L$. Assume that $\ell_{i,u} = 0$ for all $i \in L$. There must exist $i \in L$ such that $r_{i,u} = 1$ (by definition of L and L'). Moreover, there is $j \in R$ such that $\ell_{j,u} = 1$ (Condition 2). Thus, we must have $\ell_{i,1} = \ell_{j,1}$ and $r_{i,1} = r_{j,1}$ (Condition 5), which is a contradiction since $i \in L$ and $j \in R$. Condition 3 holds in Ω_L.
- Let $u_{\min}$ be the smallest element of L', and $i \in L$ such that $\ell_{i,u_{\min}} = 1$ or $r_{i,u_{\min}} = 1$. For all $1 < w < u_{\min}$, we must have $\ell_{i,w} = r_{i,w} = 0$. Therefore, according to Condition 6, for

all $j \in L$, $\ell_{j,u_{\min}} + r_{j,u_{\min}} = 1$ (since $\ell_{i,1} = \ell_{j,1} = 1$). In other words, for all $j \in L$, either $\ell_{j,u_{\min}} = 1$ or $r_{j,u_{\min}} = 1$. Thus Condition 2 holds in Ω_L (where the role of the number 1 is played by the smallest element $u_{\min}$ of L').

Thus we showed that Conditions 1 to 6 hold for Ω_L up to a scaling of the nodes' numbers over $\{1, \dots, |L'|\}$ (for instance $u_{\min}$ must be interpreted as the number 1 in Definition 12). Since $|L| < n$, we can apply our induction hypothesis $P_{|L|}$ to Ω_L, which proves that $|L'| = |L| - 1$ and the restriction of the initial assignment to Ω_L is correct. In other words, there exists a mobile M_L (whose root is the node $u_{\min}$), together with a consistent numbering of its internal nodes, that corresponds to the assignment of Ω_L.

Similarly, we can prove that $|R'| = |R| - 1$ and the assignment of Ω_R corresponds to a mobile M_R (whose root is the node $v_{\min} = \min R'$), together with a consistent numbering of its internal nodes. Finally, $m = 1 + |L'| + |R'| = |L| + |R| - 1 = n - 1$ and our initial assignment of Ω is correct since it corresponds to a mobile whose left subtree is M_L and right subtree is M_R. Moreover, the resulting numbering of the internal node is increasing since the root has number 1 and its two subtrees have increasing (and disjoint) numbering of their nodes. Proposition P_n is true.

E.2 Proof of Proposition 15

Constraints 1 to 3 are easily deducted from Conditions 1 to 3 of Definition 12.

Consider the first part of Condition 4. If $\ell_{i,u} = 1$ and $\ell_{i,v} = r_{i,v} = 0$ for all $v > u$ (i.e. $\sum_{v>u}(\ell_{i,v} + r_{i,v}) = 0$) then Constraint 4 correctly implies $\ell_{j,u} = 0$. On the other hand, assume that the first part of Condition 4 does not hold. Then, there exists $i \neq j$ and u such that $\ell_{i,u} = 1$ and $\ell_{i,v} = r_{i,v} = 0$ for all $v > u$, but $\ell_{j,u} = 1$, which contradicts Constraint 4. Thus, Condition 4 and Constraint 4 are equivalent.

Similarly, for the first part of Condition 5, if $\ell_{i,v} + r_{i,v} = 1$ and $\ell_{j,v} + r_{j,v} = 1$ then Condition 5 implies $\ell_{i,u} \leq \ell_{j,u}$ and $\ell_{j,u} \leq \ell_{i,u}$ (we switch the role of i and j). Thus $\ell_{i,u} = \ell_{j,u}$. The converse implication is also trivial. Consequently, Condition 5 and Constraint 5 are equivalent.

According to Condition 4, if $\ell_{i,u} = \ell_{j,u} = 1$ (or $r_{i,u} = r_{j,u} = 1$) then there exist $v_i, v_j > u$ such that $\ell_{i,v_i} + r_{i,v_i} = \ell_{j,v_j} + r_{j,v_j} = 1$ and $\ell_{i,v'_i} + r_{i,v'_i} = \ell_{j,v'_j} + r_{j,v'_j} = 0$ for all $u < v'_i < v_i$ and $u < v'_j < v_j$. Thus, Condition 6 must only express the fact that $v_i = v_j$. According to Constraint 6, if $\ell_{i,u} = \ell_{j,u} = 1$ (or $r_{i,u} = r_{j,u} = 1$) and $\ell_{i,w} + r_{i,w} = 0$ (for all $u < w \leq u'$) then $\ell_{j,u'} + r_{j,u'} = 0$. It implies correctly that $v_i = v_j$.

F Proof of Proposition 19 (Enumeration of trees)

The algorithm is based on a bijection between leaf-labeled, unordered, full binary trees with n leaves, and perfect matchings over $N = 2(n-1)$ vertices [8, 3]. The number of perfect matchings over N vertices is known to be $(N-1)!!$, where is $N!! = N \cdot (N-2) \cdots 3 \cdot 1$ when N is odd, and $N!! = N \cdot (N-2) \cdots 4 \cdot 2$ when N is even. Our algorithm proceeds as follows.

- Generate all perfect matchings over N vertices, using a standard backtracking algorithm.
- For each matching, use the bijection to generate a mobile, and evaluate it (in linear time)
- Return the minimal value obtained.

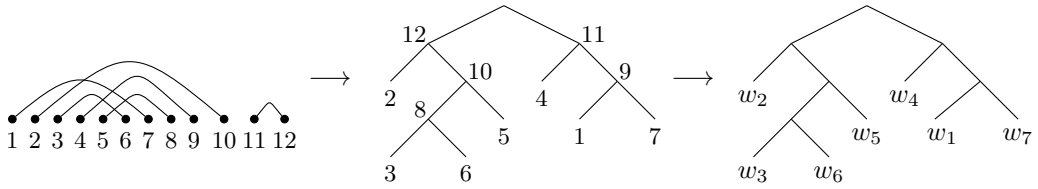


Figure 10 An example of how to obtain a mobile with $n = 7$ leaves from a matching on $2n-2 = 12$ vertices. Vertices 1–7 in the matching represent leaves, and vertices 8–12 represent internal nodes of the tree, excluding the root. Pairs in the matching are siblings in the tree. The pairs are assigned a parent in the following order: $(11, 12), (2, 10), (4, 9), (5, 8), (1, 7), (3, 6)$.

We describe how to obtain a mobile from a perfect matching, using the bijection in [8, 3]. (See Figure 10 for an example.) Let $(i_j, j_1), \ldots, (i_{n-1}, j_{n-1})$ be a perfect matching over $N = 2n-2$ vertices, numbered $1, \ldots, N$. Vertices 1 through n represent the leaves of the tree, and vertices $n+1, \ldots, N$ are the internal nodes (excluding the root). Order the pairs so that $i_k < j_k$ for all k and $j_1 < j_2 < \cdots < j_{n-1}$. The children of the root are labeled by i_{n-1}, j_{n-1}. Taking the pairs (i_k, j_k) in descending order of k, assign them to be the children of the largest vertex already in the tree (that is not a leaf) that does not have any children. Notice that the procedure operates in linear time and space.

Altogether, the algorithm requires $(2n-3)!! = \mathcal{O}\big(2^{(n \log n)/2}\big)$ iterations, each taking linear time. The backtracking algorithm used to generate matchings has a stack of depth of $\mathcal{O}(n)$ and uses $\mathcal{O}(\log n)$ space on each level of the stack to store one pair of vertices in the matching. Overall the space needed is $\mathcal{O}(n \log n)$. This algorithm is also naturally parallelizable.