Hardness of Balanced Mobiles

Virginia Ardévol Martínez, Romeo Rizzi, Florian Sikora

Hardness of Balanced Mobiles

Virginia Ardévol Martínez[1][0000−0002−3703−2335], Romeo Rizzi[2][0000−0002−2387−0952], and Florian Sikora[1][0000−0003−2670−6258]

[1] Université Paris-Dauphine, PSL University, CNRS, LAMSADE, 75016 Paris, France {virginia.ardevol-martinez,florian.sikora}@dauphine.fr
[2] Department of Computer Science, University of Verona, Italy
romeo.rizzi@univr.it

Abstract. Measuring tree dissimilarity and studying the shape of trees are important tasks in phylogenetics. One of the most studied shape properties is the notion of tree imbalance, which can be quantified by different indicators, such as the Colless index. Here, we study the generalization of the Colless index to mobiles, i.e., full binary trees in which each leaf has been assigned a positive integer weight. In particular, we focus on the problem Balanced Mobiles, which given as input n weights and a full binary tree on n leaves, asks to find an assignment of the weights to the leaves that minimizes the Colless index, i.e., the sum of the imbalances of the internal nodes (computed as the difference between the total weight of the left and right subtrees of the node considered). We prove that this problem is strongly NP-hard, answering an open question given at IWOCA 2016.

Keywords: Phylogenetic trees · Colless Index · Balanced Mobiles· Strong NP-hardness.

1 Introduction

Phylogenetics is the study of evolutionary relationship among biological entities (taxa). Its main task is to infer trees whose leaves are bijectively labeled by a set of taxa and whose patterns of branching reflect how the species evolved from their common ancestors (*phylogenetic trees*). The inferred trees are often studied by comparing them to other phylogenetic trees or to existing models. Thus, it is important to be able to formally quantify how different trees differ from each other and to have measures that give information about the shape of the trees. With respect to the latter, one of the most studied shape properties of phylogenetic trees is that of *tree balance*, measured by metrics such as the Sackin index [12] or the Colless index [3] (see also the survey of Fischer et al. [6]). The Colless index is defined for binary trees as the sum, over all internal nodes v of the tree, of the absolute value of the difference of the number of leaves in the two children of v. It is one of the most popular and used metrics, see for example [1,11,5,2,13].

The natural generalization with *weights* on the leaves has later been studied within *mobiles*[3], defined as full binary trees with positive weights on their leaves. In particular, given a set of n integer weights $\{w_1, \ldots, w_n\}$, the problem Balanced Mobiles asks to find a mobile whose n leaves have weights $w_1, \ldots, w_n$, and which minimizes the total Colless index (i.e., the sum of the imbalances $|x - y|$ of every internal node, where x and y represent the total weight of the leaves on the left and right subtrees of the node considered) [9]. Despite being a natural generalization, the complexity of this problem is still not yet known. In fact, it was proposed as an open problem by Hamoudi, Laplante and Mantaci in IWOCA 2016 [8].

Still, some results are known for some specific cases. For example, if all the leaves have unit weight, it is known that building a partition tree or a left complete tree are both optimal solutions, and their imbalance can be computed in polynomial time using a recursive formula. On the other hand, if all the weights are powers of two or if a perfectly balanced mobile can be constructed, the well known Huffman's algorithm [10] is optimal. This algorithm recursively builds a mobile by grouping the two smallest weights together (where the weight of the constructed subtree is added to the list of weights in each step).

With respect to the complexity, it is only known that the problem is in the parameterized class XP, parameterized by the optimal imbalance [9] (i.e. it is polynomial for constant values of the parameter). This result was obtained by using a relaxation of Huffman's algorithm, which gives an algorithm of complexity $\mathcal{O}(\log(n)n^{\mathcal{C}^*})$, where C^* is the optimal imbalance. An ILP is also given to solve the problem [9]. However, no polynomial time approximation algorithm has been proposed for this problem, although it is known that Huffman's algorithm does not construct an approximate solution in the general case, being arbitrarily far away from the optimum for some instances [9].

In this paper, we shed some light into the complexity of the problem by showing that Balanced Mobiles is strongly NP-hard when both the full binary tree and the weights are given as input.

2 Preliminaries

We first give the necessary definitions to present the problem.

Definition 1. *A* full binary tree *is a rooted tree where every node that has at least one child has precisely two children. A full binary tree is said to be* perfect *when all its leaves are at the same depth. The depth $d(v)$ of a node v is defined by*

$$d(v) := \begin{cases} 0 \text{ if } v = r, \text{ the root,} \\ 1 + d(F(v)) \text{ otherwise,} \end{cases}$$

[3] The term "mobile" comes from what can be found in modern art (e.g. the ones of Calder, well known in TCS being the illustration of the cover of the famous book CLRS' Introduction to Algorithms [4]) or the toy above toddler beds [9].

where $F(v)$ denotes the father of node v. Also, for every non-leaf node v, $L(v)$ (resp., $R(v)$) denotes the left (resp., right) child of node v.

Definition 2. *A binary tree is said to be* leaf-weighted *when a natural number $w(v)$ is assigned to each one of its leaf nodes v. Then, the recurrence $w(v) := w(L(v)) + w(R(v))$ extends w defining it also on every internal node v as the total weight over the leaves of the subtree rooted at v. A leaf-weighted full binary tree is also called a* mobile.

In this paper, we focus only on the Colless index to measure the balance of mobiles. Thus, we will just refer to the cost at each node as *imbalance*, and to the total Colless index of the tree as the *total cost*.

Definition 3. *The* imbalance *of an internal node v is defined as $imb(v) := |w(L(v)) - w(R(v))|$. The* total cost *of a leaf-weighted full binary tree (mobile) is the sum of the imbalances over the internal nodes. If the total cost is equal to 0, the mobile is said to be* perfectly balanced.

We can now define the problem Balanced Mobiles studied in this paper.

Balanced Mobiles
Input: n natural numbers and a full binary tree $\mathcal{T}$ with n leaves.
Task: Assign each number to a different leaf of the given full binary tree in such a way that the sum of the imbalance over the internal nodes of the resulting leaf-weighted binary tree is minimum.

3 Balanced Mobiles is NP-hard in the strong sense

We prove that Balanced Mobiles as formulated above is NP-hard in the strong sense.

To do so, we will reduce from ABC-partition, a variant of the problem 3-partition which we define below.

ABC-partition
Input: A target integer T, three sets A, B, C containing n integers each such that the total sum of the $3n$ numbers is nT.
Task: Construct n triplets, each of which contains one element from A, one from B and one from C, and such that the sum of the three elements of each triplet is precisely the target value T.

The ABC-partition problem is known to be strongly NP-hard, that is, it is NP-hard even when restricted to any class of instances in which all numbers have magnitude $\mathcal{O}(poly(n))$. This fact is reported in [7], where the problem, labeled as [SP16], is also called Numerical 3-D Matching, as it can also be reduced to the 3-Dimensional Matching problem.

3.1 Preliminary steps on the ABC-partition problem

As a first step in the reduction, given an instance of ABC-PARTITION, we will reduce it to an equivalent instance of the same problem with some specific properties that will be useful for the final reduction.

A class of instances is called *shallow* when it comprises only instances all of whose numbers have magnitude $\mathcal{O}(poly(n))$. Since we aim at proving strong NP-hardness of the target problem, we need to make sure that, starting from any shallow class of instances, the classes of instances produced at every step remain shallow.

We start with some easy observations.

Observation 1 *For any natural constant k, we can assume that all numbers are divisible by 2^k, simply by multiplying all of them by 2^k.*

Note that, since k is a constant, after this first reduction we are still dealing with a shallow class of instances.

For the next step, we assume all numbers are greater than 1, which follows from the above with $k = 1$.

Observation 2 *We can then assume that n is a power of two, otherwise, let h be the smallest natural such that $2^h > n$, we can just add $2^h - n$ copies of the number $T - 2$ to the set A, and $2^h - n$ copies of the number 1 to both sets B and C.*

Note that we are still dealing with a shallow class of instances.

The next step requires to be more formal. Assume the three given sets of natural numbers to be $A = \{a_1^0, a_2^0, \ldots, a_n^0\}$, $B = \{b_1^0, b_2^0, \ldots, b_n^0\}$ and $C = \{c_1^0, c_2^0, \ldots, c_n^0\}$, the target value to be T^0 and let M^0 be the maximum number in $A \cup B \cup C$. Here, $M^0 = \mathcal{O}(poly(n))$ since this generic instance is taken from a shallow class. Consider the instance of the problem where the n input numbers and the target value T^0 are transformed as follows:

$$
\begin{aligned}
a_i^1 &:= a_i^0 + 8n^2M^0 && \textit{for every } i = 1, 2, ..., n \\
b_i^1 &:= b_i^0 + 4n^2M^0 && \textit{for every } i = 1, 2, ..., n \\
c_i^1 &:= c_i^0 + 2n^2M^0 && \textit{for every } i = 1, 2, ..., n \\
T^1 &:= T^0 + 14n^2M^0 \\
M^1 &:= T^1
\end{aligned}
$$

Notice that the new M^1 does not represent any longer the maximum value of the numbers of the input instance because it is equal to $T^0 + 14n^2M^0$, while the value of ever number is bounded above by $a_i^0 + 8n^2M^0$. The role that parameter M^1 plays in our reduction will be seen only later.

Clearly, this new instance is equivalent to the previous one in the sense that either both or none of them are yes-instances of ABC-PARTITION. Moreover, this

reduction yields a shallow class of instances $\mathcal{C}^1$ when applied to a shallow class of instances $\mathcal{C}^0$. Therefore, thanks to Observation 2 and this transformation, we can assume that we are dealing with a shallow class of instances each of which satisfies the following two properties:

$$n \text{ is a power of two} \tag{1}$$

$$b > c \text{ and } a > b + c \text{ for every } a \in A,\ b \in B \text{ and } c \in C. \tag{2}$$

3.2 ABCDE-partition problem

Once Properties (1) and (2) are in place, we create a next equivalent instance through one further reduction, this time yielding an instance of a slightly different version of the multi-dimensional partition problem, the ABCDE-partition problem, which we define below.

ABCDE-Partition
Input: A target integer T, five sets A, B, C, D, E, with n integers in each, such that the sum of the numbers of all sets is nT.
Task: Construct n 5-tuples, each of which contains one element of each set, with the sum of these five elements being precisely T.

If not known, the next transformation in our reduction proves that this variant is also strongly NP-hard. In fact, where $a_i^1, b_i^1, c_i^1, M^1, T^1$ comprise the modified input of the ABC-partition problem after the last transformation detailed just above, consider the equivalent instance of the ABCDE-partition problem where the input numbers and the target value are defined as follows:

$$\begin{aligned}
a_i &:= a_i^1 + 8n^2M^1 && \textit{for every } i = 1, 2, ..., n \\
b_i &:= b_i^1 + 4n^2M^1 && \textit{for every } i = 1, 2, ..., n \\
c_i &:= c_i^1 + 2n^2M^1 && \textit{for every } i = 1, 2, ..., n \\
d_i &:= n^2M^1 && \textit{for every } i = 1, 2, ..., n \\
e_i &:= n^2M^1 && \textit{for every } i = 1, 2, ..., n \\
T &:= T^1 + 16n^2M^1 \\
M &:= M^1
\end{aligned}$$

Notice that, once again, the new M parameter does not represent the maximum value of the numbers comprising the new input instance. In fact, it is significantly smaller.

Thanks to this last transformation, we see that the ABCDE-Partition problem is NP-hard even when restricted to a shallow class of instances each of which satisfies the following three properties:

$$n \text{ is a power of two (say } n = 2^h) \tag{3}$$

$$\text{the numbers in } D \cup E \text{ are all equal} \tag{4}$$

$$c > d + e, b > c + d + e \text{ and } a > b + c + d + e, \text{ for every } (a, b, c, d, e) \in A \times B \times C \times D \times E \tag{5}$$

This instance also possesses other useful properties that we will exploit in the reduction to the Balanced Mobiles problem we are going to describe next.

3.3 Reduction to Balanced Mobiles

To the above instance (T, A, B, C, D, E) of ABCDE-Partition, we associate the following instance $(\mathcal{T}, W)$ of Balanced Mobiles:

Weights (W). Besides the weights a_i, b_i, c_i, d_i, e_i defined above for every $i = 1, 2, ..., n$, we also introduce $n = 2^h$ copies of the number T. Notice that all these numbers have magnitude $\mathcal{O}(poly(n))$ since it was assumed that $M = \mathcal{O}(poly(n))$.

Full binary tree ($\mathcal{T}$). Before describing how to construct the tree $\mathcal{T}$, which completes the description of the instance and the reduction, we still have one proviso.

While describing how to obtain the instance of the target problem from the instance of the source problem, it often helps to describe simultaneously how to obtain a yes-certificate for the target instance from a hypothetical yes-certificate for the source instance. Hence, let σ_B and σ_C be two permutations in S_n meant to encode a generic possible solution to the generic ABCDE-partition problem instance (since all the elements in D and E are equal, it is enough to consider these two permutations). The pair (σ_B, σ_C) is a truly valid solution, i.e., a yes-certificate, iff $a_i + b_{\sigma_B(i)} + c_{\sigma_C(i)} + d_i + e_i = T$ for every $i = 1, 2, ..., n$. We are now going to describe not only how to construct an instance of the target problem but also a solution $S = S(\sigma_B, \sigma_C)$ for it, which depends solely on the hypothetical yes-certificate (σ_B, σ_C).

The tree $\mathcal{T}$ and the solution $S = S(\sigma_B, \sigma_C)$ are constructed as follows:

1. Start with a perfect binary tree of height $h+1$, with $2n = 2^{(h+1)}$ leaves. Its n internal nodes at depth h are called test nodes, denoted $t_i, i \in [n]$ (also called r_i^0). This tree is a full binary tree thanks to Property 3. Moreover, each test node t_i will next become the root of a subtree of depth 5, all these subtrees having the very same topology, described in the following and illustrated in Figure 1.
2. For $i = 1, ..., n$, the left child of the test node t_i is a leaf of the subtree rooted at t_i (and hence also of the global tree under construction). The certificate assigns one different copy of T to each left child of a test node. These n nodes will be the only leaves at depth $2n = 2^{(h+1)}$ in the final tree under construction. However, the right child of t_i, called r_i^1, has two children described next.

3. The left child of r_i^1 is a leaf, and the certificate assigns to this leaf the number a_i. On the other hand, the right child of r_i^1, which we denote r_i^2, will not be a leaf, which means that it has two children, described next.
4. In the next step, we also let the left child of r_i^2 be a leaf. The certificate assigns the number $b_{\sigma_B(i)}$ to the left child. On the other hand, the right child of r_i^2, called r_i^3, will have two children, described next.
5. As before, the left child of r_i^3 is also a leaf, and the certificate assigns the number $c_{\sigma_C(i)}$ to it. The right child of r_i^3, called r_i^4, will also have two children, but, finally, both of them are leaves: to the left (resp., right) child leaf, the certificate associates the number d_i (resp., e_i).

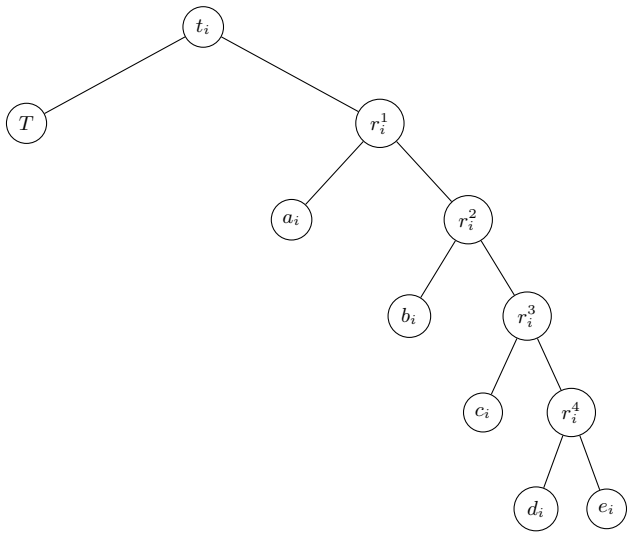

Fig. 1. Subtree rooted at the test node t_i with a canonical weight assignment. Recall that in the full tree $\mathcal{T}$, a full binary tree connects all the test nodes t_i.

The set I of the internal nodes of $\mathcal{T}$ partitions into $I_<$ and $I_\geq$, those of depth less than h and those of depth at least h, respectively. In other words, all the strict ancestors of test nodes t_i versus $I_\geq = \bigcup_{i=1,\ldots,n}\{t_i, r_i^1, r_i^2, r_i^3, r_i^4\}$. We also define $I_>$ as the set of internal nodes at depth strictly greater than h.

Definition 4. *A weight assignment is called* canonical *if it is of the form $S(\sigma_B, \sigma_C)$ for some pair (σ_B, σ_C). Equivalently, the canonical assignments are those where all $2n$ leaf nodes at depth $h+5$ have been assigned one different copy of weight Mn^2, and all n leaf nodes at depth $h+1$ (resp., $h+2$, $h+3$, or $h+4$) have weight precisely T (resp., falling in the interval $(8n^2M, 8n^2M+M)$, $(4n^2M, 4n^2M+M)$, or $(2n^2M, 2n^2M+M)$).*

The NP-hardness result now follows from the following discussion.

Lemma 5. *The total imbalance cost of $S(\sigma_B, \sigma_C)$ in the nodes $I_< \cup \{t_i \mid i \in [n]\}$ is equal to 0 if and only if $S(\sigma_B, \sigma_C)$ encodes a yes-certificate.*

Proof. First of all, the imbalance at the internal node t_i is equal to 0 if and only if

$$T = a_i + b_i + c_i + d_i + e_i$$

or equivalently,

$$T^1 = a_i^1 + b_i^1 + c_i^1$$

for every $i \in [n]$. That is, every 5-tuple (resp., every triplet) needs to sum up to T (resp., T^1), the target value. To complete the proof, we just need to observe that nodes at depth $h-1$ have as children two test nodes. Thus, their imbalance is 0 if and only if the two test nodes have exactly the same weight (equivalently, the triplets associated to them sum to the same value). Going up the (full binary) tree, it is easy to see that we need all the test nodes to have exactly the same weight, i.e., all the 5-tuples to sum up to the same value. □

Lemma 6. *The total imbalance cost of $S(\sigma_B, \sigma_C)$ is greater or equal to $\sum_{i=1}^{n}(a_i^1 - c_i^1)$ and equality holds if and only if $S(\sigma_B, \sigma_C)$ encodes a yes-certificate.*

Proof. We have already seen in Lemma 5 that $\sum_{v \in I_< \cup \{t_i\}} imb(v) = 0$ if and only if $S(\sigma_B, \sigma_C)$ encodes a yes-certificate. We will now prove that $\sum_{v \in I_>} imb(v) = \sum_{i=1}^{n}(a_i^1 - c_i^1)$ for canonical assignments, from where the result follows. First, for any canonical assignment, $imb(r_i^4) = n^2M - n^2M = 0$ for every $i = 1, \ldots, n$. We will next see that, for every canonical assignment, $\sum_{v \in \{r_i^1, r_i^2, r_i^3 : i=1,\ldots,n\}} imb(v) = \sum_{i=1}^{n}(a_i^1 - c_i^1)$.

First of all,

$$\sum_{i=1}^{n} imb(r_i^3) = \sum_{i=1}^{n} |c_i - w(r_i^4)| = \sum_{i=1}^{n} |(c_i^1 + 2n^2M) - 2n^2M| = \sum_{i=1}^{n} c_i^1$$

Similarly,

$$\sum_{i=1}^{n} imb(r_i^2) = \sum_{i=1}^{n} |b_i - w(r_i^3)| = \sum_{i=1}^{n} |b_i^1 - c_i^1| = \sum_{i=1}^{n} (b_i^1 - c_i^1)$$

where the last equality follows from the property that $b_i > c_i$. Finally,

$$\sum_{i=1}^{n} imb(r_i^1) = \sum_{i=1}^{n} |a_i - w(r_i^2)| = \sum_{i=1}^{n} |a_i^1 - (b_i^1 + c_i^1)| = \sum_{i=1}^{n} (a_i^1 - b_i^1 - c_i^1)$$

where again we use the property that $a_i^1 > b_i^1 + c_i^1$. Thus, summing all the costs below every test node, we get that the total cost is

$$\sum_{i=1}^{n} ((a_i^1 - b_i^1 - c_i^1) + (b_i^1 - c_i^1) + c_i^1) = \sum_{i=1}^{n} (a_i^1 - c_i^1)$$

□

Before continuing, note that $\sum_{i=1}^{n}(a_i^1 - c_i^1) \ll n^2M$. Indeed,

$$\sum_{i=1}^{n}(a_i^1 - c_i^1) \leq \sum_{i=1}^{n}(M^0 + 8n^2M^0 - 2n^2M^0) \leq nM$$

The last inequality follows since $M = T^0 + 14n^2M^0$, which is clearly greater than the term in the sum.

Lemma 7. *Any assignment f of cost less than n^2M is canonical.*

Proof. Let f be any assignment of cost less than n^2M. Notice that the constructed tree has precisely n internal nodes whose two children are both leaves (those labeled r_i^4). At the same time, we have $2n$ weights of value n^2M, whereas all other weights exceed $2n^2M$. Therefore, all copies of weight n^2M should be assigned to the leaves that are children of some r_i^4, that is, to the $2n$ nodes of largest depth $h+5$. Indeed, if at least one of the copies of weight n^2M were assigned to a node which is not at largest depth, then the imbalance at its parent node would be $|n^2M - w_r|$, with w_r being the weight of the right child, which is always greater or equal than $2n^2M$. Thus, the imbalance would be already at least n^2M.

After this, we can go up the tree. The n nodes of weight in the interval $[2n^2M, 2n^2M + M]$ are mapped to nodes that are left children of r_i^3 nodes (all leaf nodes at depth $h+4$). Otherwise, the weight of that node would be at least $4n^2M$, yielding an imbalance of at least $2n^2M$. Similarly, the n nodes of weight in the interval $[4n^2M, 4n^2M + M]$ are mapped to nodes that are left children of r_i^2 nodes (all leaf nodes at depth $h+3$), and the n nodes of weight in the interval $[8n^2M, 8n^2M + M]$ are mapped to nodes that are left children of r_i^2 nodes (all leaf nodes at depth $h+2$). Finally, the n nodes of weight T are mapped to nodes that are left children of test nodes t_i (all leaf nodes at depth $h+1$).

This shows that if we want an assignment of cost less than n^2M, then every weight, while it can be assigned to the leaves of a subtree rooted at any of the test nodes t_i, it has to be assigned to a leaf node of the right depth/category. But then f is a canonical assignment. □

Theorem 8. Balanced Mobiles *is strongly* NP*-hard.*

Proof. We have described a log-space reduction that, given a generic instance I of $ABCDE$-partition yields an instance $(\mathcal{T}, W)$ of Balanced Mobiles and a lower bound $L := \sum_{i=1}^{n}(a_i^1 - c_i^1)$ such that:

1. Every possible solution to $(\mathcal{T}, W)$ has total imbalance cost at least L.
2. $(\mathcal{T}, W)$ admits a solution of cost L iff I is a yes-instance of the $ABCDE$-partition problem.

This already shows that the Balanced Mobiles optimization problem is NP-hard. The Balanced Mobiles problem is strongly NP-hard because the $ABCDE$-partition problem is strongly NP-complete, and when the reduction is applied on top of any shallow class of instances of $ABCDE$-partition, it yields a shallow class of instances of Balanced Mobiles. □

Note that this implies that the decision version of the problem is (strongly) NP-complete. Indeed, one can check that the problem is in NP because given a potential solution, it can be verified in polynomial time whether it is valid or not.

4 Conclusion

We have shown that Balanced Mobiles is strongly NP-hard when the full binary tree is given as input. However, note that the complexity when the tree is not given remains open. Indeed, our reduction cannot be directly extended to this case since then, there is no structure to ensure that weights of set A are grouped with weights of sets B and C. On the other hand, the complexity when the weights are constant is also unknown, as in our proof, the constructed weights depend on n. Finally, with respect to the parameterized complexity, as we mentioned before, it is only known that the problem is in the parameterized class XP, parameterized by the optimal imbalance [9], so other future work includes to study whether there exists a fixed parameter algorithm or not.

Acknowledgements Part of this work was conducted when RR was an invited professor at Université Paris-Dauphine. This work was partially supported by the ANR project ANR-21-CE48-0022 ("S-EX-AP-PE-AL").

References

1. K. Bartoszek, T. M. Coronado, A. Mir, and F. Rosselló. Squaring within the Colless index yields a better balance index. *Mathematical Biosciences*, 331:108503, 2021.
2. M. G. Blum, O. François, and S. Janson. The mean, variance and limiting distribution of two statistics sensitive to phylogenetic tree balance. *The Annals of Applied Probability*, 16(4):2195–2214, 2006.
3. D. H. Colless. Phylogenetics: The theory and practice of phylogenetic systematics., 1982.
4. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 3rd Edition.* MIT Press, 2009.
5. T. M. Coronado, M. Fischer, L. Herbst, F. Rosselló, and K. Wicke. On the minimum value of the colless index and the bifurcating trees that achieve it. *Journal of Mathematical Biology*, 80(7):1993–2054, 2020.
6. M. Fischer, L. Herbst, S. Kersting, L. Kühn, and K. Wicke. Tree balance indices: a comprehensive survey, 2021.
7. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.
8. Y. Hamoudi, S. Laplante, and R. Mantaci. Balanced mobiles, 2016. www.iwoca.org, Problems Section.
9. Y. Hamoudi, S. Laplante, and R. Mantaci. Balanced mobiles with applications to phylogenetic trees and Huffman-like problems. Technical report, 2017. `https://hal.science/hal-04047256`.

10. D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.
11. A. Mir, L. Rotger, and F. Rosselló. Sound Colless-like balance indices for multifurcating trees. *PloS one*, 13(9):e0203401, 2018.
12. M. J. Sackin. "Good" and "bad" phenograms. *Systematic Biology*, 21(2):225–226, 1972.
13. K.-T. Shao and R. R. Sokal. Tree Balance. *Systematic Biology*, 39(3):266–276, 09 1990.