RAIRO-Oper. Res. 56 (2022) 1823–1839
https://doi.org/10.1051/ro/2022066

RAIRO Operations Research
www.rairo-ro.org

RESTRICTED HAMMING–HUFFMAN TREES

Min C. Lin[1], Fabiano S. Oliveira[2], Paulo E. D. Pinto[2], Moysés S. Sampaio Jr.[3,*] and Jayme L. Szwarcfiter[2,3]

Abstract. We study a special case of Hamming–Huffman trees, in which both data compression and data error detection are tackled on the same structure. Given a hypercube Q_n of dimension n, we are interested in some aspects of its vertex neighborhoods. For a subset L of vertices of Q_n, the neighborhood of L is defined as the union of the neighborhoods of the vertices of L. The minimum neighborhood problem is that of determining the minimum neighborhood cardinality over all those sets L. This is a well-known problem that has already been solved. Our interest lies in determining optimal Hamming–Huffman trees, a problem that remains open and which is related to minimum neighborhoods in Q_n. In this work, we consider a restricted version of Hamming–Huffman trees, called $[k]$-HHTs, which admit symbol leaves in at most k different levels. We present an algorithm to build optimal [2]-HHTs. For uniform frequencies, we prove that an optimal HHT is always a [5]-HHT and that there exists an optimal HHT which is a [4]-HHT. Also, considering experimental results, we conjecture that there exists an optimal tree which is a [3]-HHT.

Mathematics Subject Classification. 68P30, 94Bxx, 05C90.

Received September 17, 2021. Accepted May 6, 2022.

1. Introduction

In information theory, there is a common trade-off that arises in data transmission processes, in which two goals are usually tackled independently: data compression and preparation for error detection. Paradoxically, these two goals have conflicting natures: while data compression shrinks the message as much as possible, data preparation for error detection adds redundancy to messages so that a receiver can detect corrupted bits, and fix them when possible. Data compression can be achieved using different strategies, often depending on the type of data being compressed. One of the most traditional methods is that of Huffman [3], which uses ordered trees, known as Huffman trees, to encode the symbols of a given message.

A Huffman tree assigns each symbol found on the message to be compressed to a new binary string, such that the total amount of data associated with the message, using this new encoding scheme, is as small as possible. Huffman trees achieve this goal by observing the frequencies of each symbol in the original message

Keywords. Hamming–Huffman codes, hypercube graphs, minimum neighborhood.

[1] Universidad de Buenos Aires, Buenos Aires, Argentina.
[2] Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brazil.
[3] Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil.
*Corresponding author: moysessj@cos.ufrj.br

and assigning smaller codifications to higher-frequency symbols. A relevant aspect of Huffman trees is that this type of tree is proved to yield optimal codes.

In 1980, Hamming proposed the union of both compression and error detection features through a data structure called Hamming–Huffman tree [2]. This data structure compresses data similarly to Huffman trees with the additional feature of enabling the detection of any 1-bit error due to error transmission.

In contrast to Huffman trees, building optimal Hamming–Huffman trees is still an open problem. An approximation algorithm having time $O(n \log^3 n)$ was presented in [9,10] with a low additive error with respect to the entropy. Hamming–Huffman trees inspired other codes, such as in [11,12], in which the authors proposed a code called even codes. Even codes are obtained by a special type of Huffman tree in which symbols are allowed to be encoded only with an even number of ones. In this type of tree, nodes associated with an odd number of ones are either internal nodes or error leaves. The authors of that code provided an algorithm to build such a code in time $O(n^3 \log n)$. In [14], the algorithm was improved to have time complexity of $O(n^3)$. They also presented two approximation algorithms for even codes having time complexity of $O(n \log n)$, the second achieving a code having cost 16.7% higher than that of Huffman trees. Related/constrained problems are studied in [15].

Due to its importance and relevance for practical applications, we aim to study exact algorithms to build optimal Hamming–Huffman trees. Since the problem is still wide open, our approach is by constraining the number of levels of the tree at which its leaves can appear. This approach was employed in [1] to study Hamming–Huffman trees in which the leaves lie at a single level.

In this paper, we define a more restricted version of the problem of building optimal Hamming–Huffman trees. We tackle the problem of building optimal Hamming–Huffman trees in which the leaves lie in exactly k distinct levels. If $k \leq 2$, we provide a polynomial time algorithm to solve the problem. Otherwise, we provide an algorithm to evaluate a lower bound on the optimal cost of such trees when the symbols have a uniform probability of occurrence. In this case, we also prove that there always exists an optimal Hamming–Huffman tree having their symbol leaves lying on at most 4 consecutive levels.

The paper is organized as follows. In Section 2, basic definitions and notations are presented. In Section 3, the problem of building Hamming–Huffman trees in which all symbol leaves lie on the same level is tackled. In Section 4, the problem of building Hamming–Huffman trees in which the leaves are distributed in two distinct levels is discussed. In Section 5, we prove that, for symbols with a uniform probability of occurrence, there is always an optimal Hamming–Huffman tree such that its symbol leaves lie on four consecutive levels. Also, we present an algorithm to evaluate a lower bound on the cost of such trees. In Section 6, we present some experimental results. Concluding remarks are presented in the last section.

2. Preliminaries

Let G be a simple graph and $u \in V(G)$. The *(open) neighborhood* of u, denoted by $N_G(u)$, is defined as $N_G(u) = \{v \in V(G) : (u, v) \in E(G)\}$. The *closed neighborhood* of u is denoted by $N_G[u] = \{u\} \cup N_G(u)$. Let $U \subseteq V(G)$. Define $N_G(U) = \bigcup_{u \in U} N_G(u)$ and $N_G[U] = \bigcup_{u \in U} N_G[u]$. When G is clear in the context, it may be omitted from the notation.

An *n-cube* or a *hypercube* with *dimension* n, is the graph Q_n having $V(Q_n)$ as the set of all binary strings with size n (and, therefore, $|V(Q_n)| = 2^n$). Moreover, $(u, v) \in E(Q_n)$ if the binary strings of u and v differ exactly in one position. Let $u, v \in V(Q_n)$, the *Hamming distance* between u and v, denoted by $d(u, v)$, is the number of positions in which the binary strings of u and v differ.

The *parity* of a binary string v is the parity of the number of 1's in v.

A subset $L \subseteq V(G)$ is called an *independent set* of G if, for all $u, v \in L$, $(u, v) \notin E(G)$. We define the *minimum neighborhood* over independent sets, with size ℓ, of Q_n as

$$\varphi(\ell, n) = \min\{|N(L)| : L \subset V(Q_n), |L| = \ell \text{ and } L \text{ is an independent set of } Q_n\}.$$

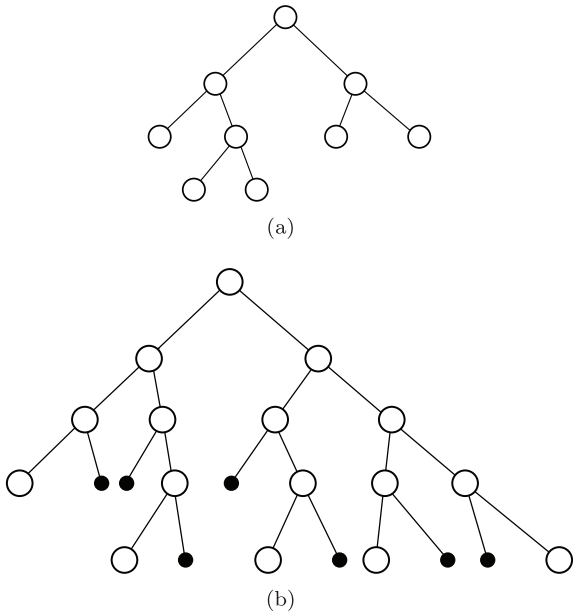

FIGURE 1. Examples of (a) uniform Huffman and (b) optimal uniform Hamming–Huffman trees, for 5 symbols.

A *strict binary tree* is a rooted tree such that each node has either two or zero children. The *level* of a node is the number of edges on the path from this node up to the root of the tree. A *full binary tree* is a strict binary tree in which all the leaves are at the same level. The *height* of a tree is the maximum level over all its nodes.

In the context of data compression techniques, an important data structure is the *Huffman tree*. A *Huffman tree* (HT) T is a rooted strict binary tree in which each edge (u, v), v being a left (resp. right) child of u, is labeled by 0 (resp. 1) and the set of leaves of T is Γ, the set of all distinct symbols of which a message M to be sent consists. Given T, each symbol a of M is sequentially encoded into a binary string $c(a)$. Such encoding is given by the sequence of 0's and 1's found on the edges of the directed path from the root of T to the leaf corresponding to a. In Figure 1a, for instance, the leaves are encoded, reading them from left to right, as 00, 010, 011, 10, and 11. Over all possible trees, the HT for M is a tree T such that its cost

$$c(T) = \sum_{a \in \Gamma} p(a)|c(a)|$$

is minimum, where $p(a)$ stands for the probability of occurrence of a in the message and $|c(a)|$ is the length of the string $c(a)$. We say that an HT is *uniform* if all of its symbols have a uniform probability of occurrence, that is, each symbol has a probability of occurrence of $\frac{1}{|\Gamma|}$. Figure 1a depicts a uniform HT T with $c(T) = 2.4$ on 5 symbols.

The concept of Hamming–Huffman trees generalizes that of Huffman trees. A *Hamming–Huffman tree* (HHT) is a strict binary tree holding the same properties of an HT, except that the set of leaves is partitioned into symbol and error leaves, such that the following properties hold:

- Every node e of T such that $d(c(e), c(a)) = 1$, for some symbol leaf $a \in \Gamma$, is a leaf of T called an *error leaf*;
- Every node of T is either an error leaf or an ancestor of a symbol leaf.

HHTs can be applied to detect errors which occurs in the transmission of messages. Under the assumption that when data is transmitted at most one bit can accidentally be flipped, HHTs detect such errors during the

decoding process: if an error leaf is hit, the data has been corrupted during transmission. Optimal HHTs are defined exactly the same as (optimal) HTs. Figure 1b depicts an optimal uniform HHT T with $c(T) = 3.8$ on 5 symbols. In the presented figures, error leaves are colored black.

In this work, we will weaken the concept of HHTs to enlarge the number of trees which could be called HHTs. This is convenient to our algorithms and proofs, without losing the motivation associated with the application related to detecting error capabilities.

If a tree T is an HHT as previously defined, we will say that T is a *strict HHT*. The weaker definition of HHTs is the following. Given a strict binary tree T, in which the set of leaves is partitioned into symbol and error leaves, T is an HHT if $contract(T)$ is a strict HHT, where $contract(T)$ is a tree transformation defined as follows: if T has no sibling error leaves, then no modification to the tree is done; otherwise, if the sibling node of an error leaf e is an error leaf, remove both e and its sibling from T and make the node p, parent of e in T, into an error leaf. Let T' be the resulting tree. Apply $contract(T')$ recursively to obtain the final transformation. Figure 2a presents an HHT T in this weaker sense, and Figure 2b shows the strict HHT resulting of $contract(T)$. Note that $contract(T)$ can be done in time $O(\ell)$.

The number of error leaves in HHTs is directly related to the encodings associated with the symbol leaves. In Figure 2a, symbol leaves are encoded as 000 and 111, resulting in 6 error leaves, whereas in Figure 2c, symbol leaves are encoded as 000, 011, 101, and 110, resulting in 4 corresponding error leaves. In this second HHT, two more symbols are being encoded using the same full binary tree as the one used in the first HHT.

Although HTs can be built efficiently in a greedy fashion [3], the construction of optimal HHTs is open since defined by Hamming in the eighties [2]. In this work, we approach the problem by defining a constrained version of it, namely that of determining an optimal HHT T in which the symbol leaves are placed in exactly k distinct levels. A tree for which this property holds will be called a *k-Hamming–Huffman tree*, or *k-HHT*. Formally, we define the k-HHT problem as

Problem:	k-HHT
Input:	A set of symbols Γ and, for all $a \in \Gamma$, the probability p(a) of occurrence of a in a message.
Output:	An HHT T in which all symbol leaves lie at exactly k levels of T and such that $c(T)$ is minimum.

We denote as *$[k]$-Hamming–Huffman tree* ($[k]$-*HHT*) a minimum cost HHT over all optimal k'-HHT, for all $1 \leq k' \leq k$.

In the following sections, the problem of k-HHT is discussed.

3. Hamming Huffman trees with leaves in one level

In this section, we tackle the 1-HHT problem. This problem can be reduced to that of deciding the minimum height of the full binary HHT for which the symbol leaves can be arranged in the last level.

First, note that there is an important relation between optimal 1-HHTs with ℓ symbols and minimum neighborhoods of independent sets with ℓ elements of Q_n. Consider the one-to-one mapping between the leaves of a full binary 1-HHT T having height n to the vertices of Q_n, in which a leaf a corresponds to $c(a) \in V(Q_n)$. The problem of finding the minimum number of error leaves, over all possible trees T, is equivalent to that of finding, over all independent sets L of cardinality $\ell = |\Gamma|$ in $V(Q_n)$, one that minimizes $|N(L)|$. This is so because

$$\begin{aligned} T \text{ is a } 1-\text{HHT} &\iff d(c(u), c(v)) \geq 2 \text{ for all distinct } u, v \in \Gamma \\ &\iff L = \{c(u) : u \in \Gamma\} \text{ is an independent set of } Q_n \end{aligned}$$

Thus, for a given 1-HHT T, the set of errors leaves of T is precisely $N(L)$ in Q_n and $L = \{c(u) : u \in \Gamma\}$.

The efficient computation of $\varphi(\ell, n)$ is possible with the aid of Theorem 3.2. Before presenting the theorem, we have to state the following auxiliar lemma.

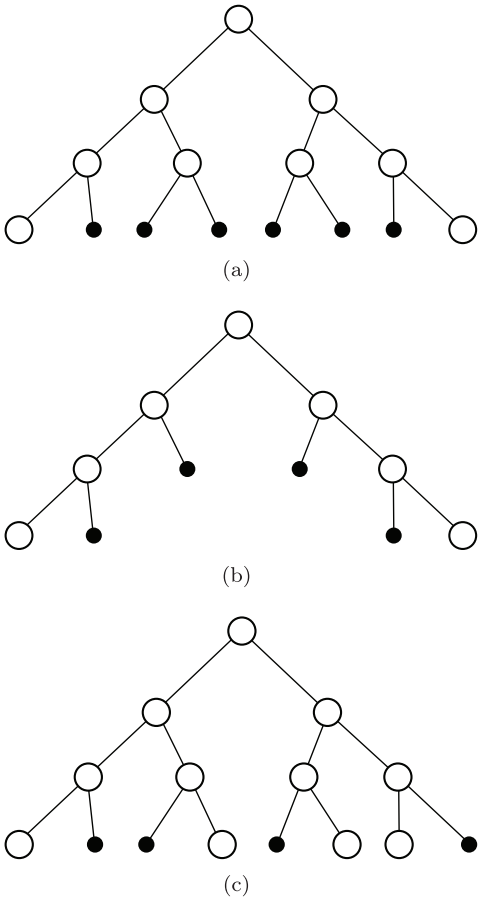

FIGURE 2. Examples of Hamming–Huffman trees.

Lemma 3.1 ([4, 8]). *For any given non-negative integers ℓ and n, $\ell < 2^n$, the number m has a unique representation*

$$\ell = \binom{n}{n} + \binom{n}{n-1} + \cdots + \binom{n}{k+1} + \binom{a_k}{k} + \cdots + \binom{a_t}{t}$$

such that

$$n > a_k > a_{k-1} > \cdots > a_t \geq t \geq 1.$$

The representation presented in Lemma 3.1 is defined as the *n-bounded canonical representation* of ℓ.

Also, in [4], Katona defined the function $G(\ell, n)$ as follows:

$$G(\ell, n) = \begin{cases} 0, & \text{if} \quad \ell \leq 0 \\ \binom{n}{n} + \binom{n}{n-1} + \cdots + \binom{n}{k+1} + \binom{n}{k} + \binom{a_k}{k-1} + \cdots + \binom{a_t}{t-1}, & \text{otherwise} \end{cases}$$

where the a's are those from Lemma 3.1 with respect to the given ℓ.

The following theorem was proven in [7].

Theorem 3.2 (Thm. 2 in [6] and [7]). *For every $\ell \leq 2^{n-1}$*

$$\varphi(\ell, n) = G(\ell, n-1).$$

A consequence of this theorem in [6], is that all the vertices belonging to the independent set L yielding $|N(L)| = \varphi(\ell, n)$ can be assumed to be, without loss of generality, of the same parity. This fact can be used directly to solve the 1-HHT problem, as shown in the following theorem. Let $h(\ell) = \lceil \log_2 \ell \rceil + 1$.

Theorem 3.3. *Let Γ be a set of ℓ symbols, each $a \in \Gamma$ having probability $p(a)$. The height and the cost of an optimal 1-HHT T are, respectively, $h(\ell)$ and*

$$c(T) = h(\ell) \sum_{a \in \Gamma} p(a).$$

Proof. To find an optimal 1-HHT T, it is necessary to determine the minimum height h of T such that the symbols and their corresponding error leaves lie all at this level, that is, $\ell + \varphi(\ell, h) \leq 2^h$. By [6], one may consider without loss of generality that the set of ℓ codifications consists of elements having a same parity. In order to choose ℓ vertices with a same parity, one may use at most half the symbol leaves of that level. That is, a full binary 1-HHT must have at least height $\lceil \log_2 \ell \rceil + 1$ to be able to contain ℓ leaves with a same parity thus, $h \geq \lceil \log_2 \ell \rceil + 1$. On the other hand, let L be a set of codifications having a same parity. First note that L is an independent set of Q_n and the corresponding error leaves have the opposite parity. Therefore, any set of ℓ symbols leaves at level $\lceil \log_2 \ell \rceil + 1$ having a same parity consists of a valid 1-HHT. Consequently, $h \leq \lceil \log_2 \ell \rceil + 1$, yielding that $h = \lceil \log_2 \ell \rceil + 1$ and $c(T) = (\lceil \log_2 \ell \rceil + 1) \sum_{a \in \Gamma} p(a)$. □

4. Hamming Huffman trees with leaves in two levels

In this section, we discuss optimal 2-HHTs. We show that, similarly to the 1-HHTs, it is possible to build optimal 2-HHTs efficiently. We provide an algorithm for building optimal 2-HHTs that runs in time $O(\ell \log^2 \ell)$, where $\ell = |\Gamma|$.

A motivation to study this specific case lies in the fact that, for symbols with uniform probabilities of occurrence, there is always a Huffman tree with symbols in at most two different levels. This follows from Section 2.3.4.5 of Knuth's Vol. 1 [5]. It is not known whether this is also the case for Hamming–Huffman trees. Experimental results were designed to investigate this hypothesis and the results are presented in Section 6. Furthermore, recall that the problem of building optimal general HHTs is open since the eighties. Thus, the approach of studying more restrictive cases is worthy, since a solution for a particular case may have practical value or lead to a solution for general HHTs.

The strategy for determining an optimal 2-HHT with leaves at two levels h_1 and h_2, with $h_1 < h_2$, is devised as follows. Note that by Theorem 3.3, a full binary tree having height $h(\ell)$ is enough to build a 1-HHT. Therefore, $h(\ell)$ is a natural upper bound on h_1, that is, $1 \leq h_1 < h(\ell)$.

First, consider any specific value for h_1. For such a value, let ℓ_1 be the number of symbol leaves that are placed at level h_1. Note that $1 \leq \ell_1 \leq \min\{\ell - 1, 2^{h_1 - 1}\}$, since 2^{h_1} is the maximum number of nodes at level h_1, and half of them have the same parity.

Once ℓ_1 symbol leaves are chosen to be placed at level h_1, there will be error nodes corresponding to such symbol leaves at this same level, and the remaining nodes will be free nodes from which the tree can grow to achieve larger levels (in particular, to achieve level h_2, where the remaining symbol leaves must lie). As seen in Figure 2, distinct sets of leaves lead to distinct sets of error leaves, the latter varying considerably in size. Clearly, to minimize the cost of the solution for the fixed values of (h_1, ℓ_1), it suffices to minimize the value of h_2. To do that, it suffices to distribute as uniformly as possible the remaining $\ell_2 = \ell - \ell_1$ symbols leaves over the subtrees rooted at the free nodes. Indeed, it is possible to arrange all symbol leaves at level h_2 of each subtree all having the same parity, ensuring that the leaves at level h_2 pairwise have Hamming distance of at least 2. Therefore, the aim is to choose the set of symbol leaves at level h_1 in such a way that the number of free nodes is maximized or, equivalently, that the number of error leaves is minimized. In other words, the algorithm must select a set of symbol leaves at level h_1 which produces $\varphi(\ell_1, h_1)$ corresponding error nodes. For this choice, the maximum number of free nodes, that will be denoted by $\rho(\ell_1, h_1)$, is given by

$$\rho(\ell_1, h_1) = 2^{h_1} - \ell_1 - \varphi(\ell_1, h_1). \tag{4.1}$$

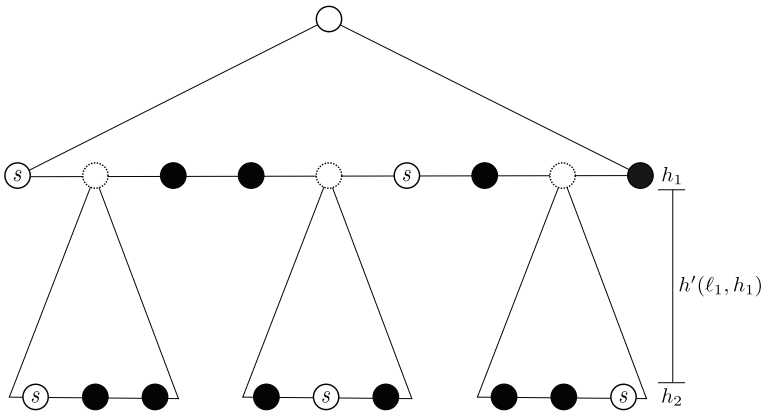

FIGURE 3. A Hamming–Huffman tree with leaves on two levels.

For the cost $c(T) = c(\ell, \ell_1, h_1)$ of this particular way of building a 2-HHT T having ℓ_1 symbols at level h_1, we present an algorithm to evaluate this value. If $\rho(\ell_1, h_1) = 0$, then it is not possible to grow the tree to allocate the remaining ℓ_2 symbols at level h_2. So, the choices of h_1, ℓ_1 turned out to lead to an unfeasible solution. If $\rho(\ell_1, h_1) > 0$, then the remaining ℓ_2 symbols are uniformly distributed in $\rho(\ell_1, h_1)$ subtrees rooted at the free nodes, each receiving at most $\left\lceil \frac{\ell_2}{\rho(\ell_1, h_1)} \right\rceil$ symbols, and one of them receiving exactly such an amount. Minimizing the common height in each subtree rooted at each free node is a 1-HHT problem. By using the result $h(\ell)$ of Theorem 3.3, we have that the minimum height $h'(\ell_1, h_1)$ required for each subtree to accommodate those symbols is given by

$$h'(\ell_1, h_1) = h\left(\left\lceil \frac{\ell_2}{\rho(\ell_1, h_1)} \right\rceil\right)$$

To determine $c(\ell, \ell_1, h_1)$ in this case, it is needed to assign the set of symbols Γ to the set of chosen symbols leaves. But to minimize such a cost, it clearly suffices to place at level h_1 the ℓ_1 symbols with the highest probability of occurrence. Assuming that $\Gamma = \{a_1, a_2, \ldots, a_\ell\}$ is ordered decreasingly according to their respective probability of occurrence, we have that

$$\begin{aligned} c(\ell, \ell_1, h_1) &= h_1 \sum_{i=1}^{\ell_1} p(a_i) + h_2 \sum_{i=\ell_1+1}^{\ell} p(a_i) \\ &= h_1 \sum_{i=1}^{\ell_1} p(a_i) + (h_1 + h'(\ell_1, h_1)) \sum_{i=\ell_1+1}^{\ell} p(a_i) \\ &= h_1 \sum_{i=1}^{\ell} p(a_i) + h'(\ell_1, h_1) \sum_{i=\ell_1+1}^{\ell} p(a_i) \\ &= h_1 + h'(\ell_1, h_1) \sum_{i=\ell_1+1}^{\ell} p(a_i). \end{aligned}$$

Figure 3 depicts this strategy. The nodes labeled with "s" represent symbol leaves, the black nodes represent the error leaves, and the dashed nodes represent the free nodes.

The optimal cost is the minimum cost obtained by varying h_1 and ℓ_1 over all possible values. Formally, the cost of an optimal 2-HHT T is given by

$$c(T) = \min\{c(\ell, \ell_1, h_1) : 1 \le h_1 < h(\ell)\ ,\ 1 \le \ell_1 \le \min\{\ell - 1, 2^{h_1 - 1}\}\}$$

where

$$c(\ell, \ell_1, h_1) = h_1 + h'(\ell_1, h_1) \sum_{i=\ell_1+1}^{\ell} p(a_i).$$

Concerning the computational complexity for determining the optimal cost, for each $1 \le h_1 < h(\ell)$, there are at most 2^{h_1-1} possible values for ℓ_1. Therefore, there are at most $1+2+2^2+\cdots+2^{\lceil \log_2 \ell \rceil - 1} = 2^{\lceil \log_2 \ell \rceil} - 1 = \Theta(\ell)$ distinct pairs (h_1, ℓ_1). Moreover, for the computation of each $c(\ell, \ell_1, h_1)$, the evaluation of $\varphi(\ell_1, h_1)$ is required. This evaluation can be computed in time $O(h_1^2) = O(\log^2 \ell)$ with the aid of a precomputed Pascal triangle. Besides that, a precomputed sum of values $\sum_{j=1}^{i} p(a_i)$ for all $1 \le i \le \ell$, which can be done in time $\Theta(\ell)$, can be used to obtain the summation present in $c(\ell, \ell_1, h_1)$ in constant time. Therefore, the complexity of evaluating the cost of an optimal tree is $O(\ell \log^2 \ell)$.

The results of this section can be summarized by the following theorem.

Theorem 4.1. *Let Γ be a set of ℓ symbols, each $a \in \Gamma$ having probability $p(a)$. The cost of an optimal 2-HHT T is*

$$c(T) = \min\{c(\ell, \ell_1, h_1) : 1 \le h_1 < h(\ell)\ ,\ 1 \le \ell_1 \le \min\{\ell - 1, 2^{h_1 - 1}\}\}$$

where

$$c(\ell, \ell_1, h_1) = h_1 + h'(\ell_1, h_1) \sum_{i=\ell_1+1}^{\ell} p(a_i)$$

and

$$h'(\ell_1, h_1) = h\left(\left\lceil \frac{\ell_2}{\rho(\ell_1, h_1)} \right\rceil\right)$$

Furthermore, this cost can be computed in time $O(\ell \log^2 \ell)$.

5. Uniform Hamming–Huffman trees

In this section, we discuss the problem of building optimal uniform HHTs. In contrast to 2-HHTs, even for the more restrictive case of uniform probabilities, an efficient algorithm for building an optimal uniform HHT will remain open. Let $\lambda(T)$ be the difference between the last and the first levels of T which have at least one symbol leaf at that level. For instance, for T as in Figure 1a, $\lambda(T) = 1$, where $\lambda(T) = 0$ for any tree T of Figure 2. We prove that $\lambda(T) \le 4$ for all optimal uniform HHT T. Moreover, we show that there is always an optimal uniform HHT T in which $\lambda(T) \le 3$. In addition, all optimal uniform HHTs are [5]-HHTs, and there exists an optimal uniform HHT which is a [4]-HHT. Finally, we present a dynamic programming algorithm to evaluate a lower bound on the cost of such a tree.

Recall that (optimal) HTs for symbols with uniform frequencies have all leaves in at most two levels. It is unknown whether the same holds for HHTs. We consider the conjecture that there is always an optimal uniform HHT in which the symbol leaves are distributed in $k \le 3$ distinct levels, and we provide empirical evidence in favor of it. Section 6 compares the lower bound of this section with the cost of optimal 2-HHTs, the latter being computed as presented in Section 4.

Let T be a uniform HHT on ℓ symbols. Consider the following operation over a symbol leaf s of T:

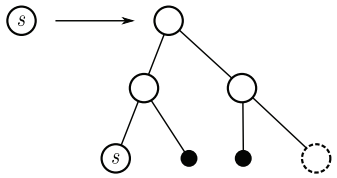

FIGURE 4. Operation *descend*(s), over a symbol leaf. The dotted node represents a free node to be used in the encoding of another symbol.

– *descend*(s) (see Fig. 4): replace the leaf s by a full binary HHT T_s having height two in such a way that this leaf becomes the root of T_s. The tree T_s is such that one of its leaves is the symbol leaf s. Note that there are exactly two error leaves associated with s among the leaves of T_s, besides one free node s', regardless which leaf corresponds to s. Next, transform s' into a symbol leaf associated with any symbol s'' that appears in the last level of T. Finally, transform the node of s'' into an error leaf. Let T' be the resulting tree. Apply *contract*(T') to obtain the final transformation.

Let T' be the resulting tree after applying *descend*(s). The following lemma proves that T' is also an HHT.

Lemma 5.1. *Let T be an HHT, s be one of its symbol leaves and T' be the tree obtained by the operation descend(s). The tree T' is an HHT.*

Proof. We shall prove that, in T', all the leaves with Hamming distance one to s are error leaves. As the root of T_s comes from a symbol leaf in T, all nodes with Hamming distance one to it in T' are error leaves. Moreover, as T_s is an HHT by construction, all the nodes in T' with Hamming distance one to s and s' are also error leaves in T'. Finally, the transformation carried out in the last step ensures that T' does not contain two sibling leaves which are both error leaves. That is, every node of T' is either an error leaf or an ancestor of a symbol leaf. Therefore, T' is an HHT. □

Let $p = \frac{1}{\ell}$ be the probability of occurrence of the symbol leaves of T and T' be the tree obtained by the operation *descend*(s), for some symbol leaf s of T. Let l_1 be the level of s and l_2 be the last level of T. Note that, the symbol of s'' was moved from level l_2 to level $l_1 + 2$. Moreover, the symbol of s was moved from level l_1 to level $l_1 + 2$. Therefore, the cost of T' can be written as a function of the cost of T as

$$c(T') = c(T) - p(\lambda(T) - 4). \tag{5.1}$$

Theorem 5.2. *Let T be a uniform HHT. If T is optimal, then $\lambda(T) \leq 4$. Moreover, there is always an optimal T for which $\lambda(T) \leq 3$.*

Proof. We prove that when $\lambda(T) > 4$ is always possible to obtain an HHT T' from T such that $c(T') < c(T)$, contradicting the optimality of T.

Let l_1 and l_k be the first and the last level of T containing symbol leaves, respectively. Let p be the probability of occurrence of each symbol associated to T. Apply *descend*(s) to some symbol leaf s at level l_1 to obtain T'. If $\lambda(T) > 4$, then $\lambda(T)p > 4p$ and, equivalently, $\lambda(T)p - 4p = p(\lambda(T) - 4) > 0$. Therefore, by (5.1), we have that $c(T') < c(T)$. Moreover, note that each application of *descend*(s) eliminates a leaf in level l_1 and a leaf in level l_k. By successive applications of descend(s) to symbol leaves, it is possible to obtain an optimal uniform HHT such that $\lambda(T) \leq 3$. □

We will proceed in the remaining of this section by providing a lower bound on the cost of uniform HHTs. For this, we need to generalize the concept of Hamming–Huffman trees. A *k-Hamming–Huffman forest*, or *k-HHF*, is a forest F of HHTs such that the symbol leaves are distributed among exactly k distinct levels of F. Note

that the trees of F may have a height greater than k since there might be some levels of F with no symbol leaves. The cost of a k-HHF is defined by the sum of the costs of its HHTs.

The strategy to derive the lower bound on the cost of a k-HHF for ℓ symbols and r trees is as follows. Consider h_1 to be the first level in which symbol leaves appear in F. Let ℓ_1 be the number of symbol leaves to be represented in the level h_1. Clearly, the most desirable arrangement for choosing ℓ_1 nodes at level h_1 is one in which the corresponding error leaves are minimized, that is, in which the free nodes are maximized. This is so because the remaining $\ell - \ell_1$ symbol leaves must be allocated as descendants of the resulting free nodes at level h_1, and the more resulting free nodes, the better. At this point, this strategy will deal with all those free nodes as independent trees of a $(k-1)$-HHF. But, some of them may actually be part of the same HHT of the k-HHF and, because of that, the symbol leaves allocated in a tree descending from a free node produce error leaves that may conflict with the allocation of symbol leaves descending from another free node. Since the possibility of conflict will not be dealt with, the resulting k-HHF may not be feasible and that is why this strategy yields a lower bound on the cost of this k-HHF. The lower bound, defined as $c_F(k, r, \ell)$, on the cost of an optimal k-HHF of r disjoint HHTs for ℓ symbols derived from this strategy is evaluated as follows.

First note that if $\ell = 0$, then $c_F(k, r, \ell) = 0$. Also, if $r = 0$ (resp. $k = 0$) and $\ell \geq 1$, it means that there are not enough free nodes to accommodate the remaining ℓ symbols. In other words, this scenario leads to an unfeasible solution and, therefore, $c_F(k, r, \ell) = +\infty$. If $k = 1$ and $\ell, r \geq 1$, the resulting problem is equivalent to the one of distributing ℓ symbols among r 1-HHTs with the same height. In particular, each one of these trees must have at least $h(\lceil \frac{\ell}{r} \rceil)$ leaves to be able to accommodate all the symbols. Thus, using the same reasoning as the one used in Theorem 3.3 and, as the symbols have an equal probability of occurrence, $c_F(k, r, \ell) = h(\lceil \frac{\ell}{r} \rceil)$. For the general case, the algorithm minimizes the cost over all possible pairs (ℓ_1, h_1). Note that despite the fact that there are only ℓ_1 symbols at level h_1, all the remaining $\ell - \ell_1$ symbols have a prefix with size h_1 in their codifications. Given that, the first part of the cost of the general case is given by $h_1(\ell_1 + (\ell - \ell_1)) = h_1\ell$.

For the remaining part of the cost, the algorithm uses pre-computed values to solve the problem of distributing $\ell - \ell_1$ symbols among a $(k-1)$-HHF in which the roots are the resulting free nodes, in a dynamic programming fashion. Formally, $c_F(k, r, \ell)$ can be expressed as

$$c_{\mathrm{F}}(k, r, \ell) = \begin{cases} 0, & \text{if} \quad \ell = 0 \\ +\infty, & \text{if} \quad \ell \geq 1, \\ & (r = 0 \text{ or } k = 0) \\ h(\lceil \frac{\ell}{r} \rceil), & \text{if} \quad \ell, r \geq 1, k = 1 \\ \min \left\{ \begin{array}{l} h_1\ell + c_F(k-1, \rho_F(r, \ell_1, h_1), \ell - \ell_1) : \\ \ell_1 \in D_1 \text{ and } h_1 \in D_2 \end{array} \right\}, & \text{otherwise} \end{cases}$$

where:

- $D_1 = \{1, \ldots, \ell - k + 1\}$,
- $D_2 = \{h(\lceil \frac{\ell_1}{r} \rceil), \ldots, h(\lceil \frac{\ell}{r} \rceil)\}$, and
- $\rho_F(r, \ell, h)$ denotes the maximum number of free nodes when ℓ symbol leaves are allocated at level h of an HHF consisting of r HHTs, and h is the first level having leaves. The computation of ρ_F will be discussed next.

For 2-HHTs, $c_F(2, 1, \ell)$ is exactly the cost of a uniform 2-HHT using the algorithm presented in Section 4. Moreover, considering general uniform HHTs, the cost of an optimal uniform HHT with ℓ symbol is at least

$$\min\{c_F(k, 1, \ell) : 1 \leq k \leq \ell\}. \tag{5.2}$$

Figure 5 depicts the strategy being adopted in the computation of c_F.

The computation of $\rho_F(r, \ell, h)$ will also be carried out by a dynamic programming algorithm. First, note that $\rho_F(r, \ell, h)$ equals the maximum number of free nodes when ℓ symbols are distributed among the leaves of r full HHTs with height h. So, the strategy to yield the recurrence is as follows. First, suppose that ℓ_1 symbols are to be allocated into a single HHT. Therefore, the remaining $\ell - \ell_1$ symbols have to be allocated among the

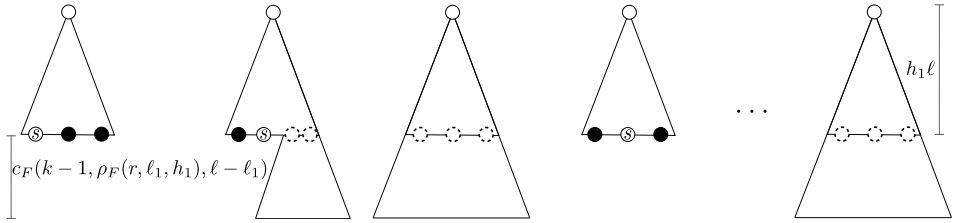

FIGURE 5. Example of the strategy used to evaluate $c_F(k, r, \ell)$.

leaves of $r-1$ full HHTs with height h. Both allocations must be done in such a way that the number of free nodes is maximized. The former can be computed with the aid of the formula given in (4.1). The latter can be determined using recursion. Formally, we have

$$\rho_F(r,\ell,h) = \begin{cases} r2^h, & \text{if} \quad \ell = 0 \\ -\infty, & \text{if} \quad \ell \geq 1, (r = 0 \text{ or} \\ & h = 0 \text{ or } \ell > r2^{h-1}) \\ \max\left\{\begin{matrix}\rho(\ell_1,h) + \rho_F(r-1,\ell-\ell_1,h) : \\ 0 \leq \ell_1 \leq \min\{2^{h-1},\ell\}\end{matrix}\right\}, & \text{otherwise.} \end{cases}$$

Precomputing the values of ρ_F requires a matrix whose number of elements is $O(\ell^2 \log \ell)$, as r is limited by ℓ and h is limited by $h(\ell)$. Moreover, as the processing of each cell of such matrix requires $O(\ell)$ steps, precomputing the values of ρ_F takes time $O(\ell^3 \log \ell)$. Assuming that the values of ρ_F are available at constant time, precomputing the values of c_F depends on a matrix whose number of elements is $O(\ell^2)$, as $k \leq 4$ by Theorem 5.2 and the remaining parameters are limited by ℓ. Furthermore, since processing each cell of such a matrix requires $O(D_1 D_2) = O(\ell \log \ell)$ steps, evaluating c_F takes time $O(\ell^3 \log \ell)$. Therefore, the proposed lower bound can be computed in time $O(\ell^3 \log \ell)$ and space $O(\ell^3)$.

6. Experimental results

In this section, we describe some experimental results which have been performed in the context of the previous sections.

We have conducted three experiments. The first one is related to the algorithm described in Section 4. It compares uniform [2]-HHTs with the lower bound described in Section 5. The second experiment compares general [2]-HHTs with the Huffman trees aiming to enlighten the tradeoffs of both strategies. In the third experiment we have implemented a backtracking that finds an optimal uniform Hamming–Huffman tree for $1 \leq \ell \leq 38$.

Implementations of such algorithms were executed on a notebook having a CPU Core i7, with 8 GB RAM, running Ubuntu 16.04 OS. The algorithms were implemented in C++. The results are presented next.

All the programs related to this section are available at [17].

6.1. Uniform [3]-HHT optimality hypothesis

As the first experiment, we tested the hypothesis that uniform [3]-HHTs are indeed optimal. In this case, for all $1 \leq \ell \leq 4096$, we have compared the costs of the algorithms in Sections 3 and 4 with those produced by the algorithm in Section 5. Some values of this comparison are presented in Table 1. The first column represents the number of symbols. The second shows the cost of the corresponding [2]-HHT. The third represents the cost of the lower bound described in Section 5. This column is divided into two parts. The first is the minimum k value that minimized the cost of the resulting tree and the second is the cost of the tree. The last column of the table gives the relative differences between the costs presented in the last two columns. These costs and their

TABLE 1. Comparison between the cost of optimal [2]-HHTs and the lower bound on the cost of k-HHTs.

ℓ	[2]-HHT cost	Lower bound k-HHT		% Diff	ℓ	[2]-HHT cost	Lower bound k-HHT		% Diff
		k	Cost				k	Cost	
3	3.000	1	3.000	0.000	44	6.841	3	6.750	1.347
85	7.741	3	7.624	1.543	126	8.000	1	8.000	0.000
167	8.725	3	8.545	2.102	208	8.928	3	8.832	1.089
249	9.000	1	9.000	0.000	290	9.421	3	9.286	1.448
331	9.689	3	9.498	2.004	372	9.831	3	9.664	1.725
413	9.908	3	9.797	1.137	454	9.965	3	9.905	0.600
495	9.996	2	9.996	0.000	536	10.164	3	10.121	0.424
577	10.385	3	10.255	1.268	618	10.560	3	10.371	1.826
659	10.663	3	10.472	1.826	700	10.744	3	10.561	1.731
741	10.810	3	10.641	1.585	782	10.859	3	10.712	1.373
823	10.903	3	10.776	1.173	864	10.936	3	10.834	0.940
905	10.962	3	10.887	0.690	946	10.981	3	10.936	0.416
987	10.995	3	10.980	0.138	1028	11.016	2	11.016	0.000
1069	11.153	3	11.102	0.463	1110	11.270	3	11.172	0.879
1151	11.379	3	11.237	1.260	1192	11.463	3	11.298	1.463
1233	11.540	3	11.354	1.636	1274	11.594	3	11.407	1.638
1315	11.651	3	11.457	1.693	1356	11.691	3	11.504	1.628
1397	11.722	3	11.548	1.506	1438	11.766	3	11.589	1.530
1479	11.800	3	11.628	1.477	1520	11.825	3	11.665	1.370
1561	11.853	3	11.700	1.303	1602	11.873	3	11.733	1.186
1643	11.890	3	11.765	1.066	1684	11.913	3	11.795	0.997
1725	11.930	3	11.824	0.897	1766	11.944	3	11.851	0.784
1807	11.957	3	11.877	0.676	1848	11.968	3	11.902	0.550
1889	11.978	3	11.926	0.439	1930	11.987	3	11.949	0.321
1971	11.993	3	11.971	0.186	2012	11.998	3	11.992	0.054
2053	12.010	2	12.010	0.000	2094	12.081	3	12.054	0.226
2135	12.144	3	12.091	0.438	2176	12.207	3	12.127	0.656
2217	12.263	3	12.162	0.834	2258	12.317	3	12.195	0.999
2299	12.371	3	12.227	1.177	2340	12.421	3	12.259	1.321
2381	12.459	3	12.289	1.388	2422	12.495	3	12.318	1.438
2463	12.527	3	12.346	1.470	2504	12.555	3	12.373	1.472
2545	12.580	3	12.399	1.458	2586	12.601	3	12.424	1.419
2627	12.628	3	12.449	1.437	2668	12.656	3	12.473	1.472
2709	12.680	3	12.496	1.474	2750	12.699	3	12.518	1.444
2791	12.719	3	12.540	1.429	2832	12.737	3	12.561	1.403
2873	12.756	3	12.582	1.383	2914	12.773	3	12.602	1.362
2955	12.787	3	12.621	1.314	2996	12.801	3	12.640	1.273
3037	12.811	3	12.658	1.210	3078	12.822	3	12.676	1.153
3119	12.841	3	12.693	1.164	3160	12.855	3	12.710	1.140
3201	12.867	3	12.727	1.097	3242	12.879	3	12.743	1.070
3283	12.889	3	12.759	1.017	3324	12.894	3	12.774	0.940
3365	12.908	3	12.789	0.934	3406	12.917	3	12.804	0.885
3447	12.922	3	12.818	0.815	3488	12.930	3	12.832	0.766
3529	12.934	3	12.845	0.693	3570	12.946	3	12.859	0.677
3611	12.954	3	12.872	0.639	3652	12.959	3	12.884	0.582
3693	12.966	3	12.897	0.535	3734	12.970	3	12.909	0.473
3775	12.976	3	12.921	0.428	3816	12.980	3	12.932	0.369
3857	12.983	3	12.943	0.302	3898	12.988	3	12.955	0.259
3939	12.992	3	12.965	0.204	3980	12.994	3	12.976	0.141
4021	12.997	3	12.987	0.080	4062	12.999	3	12.997	0.019

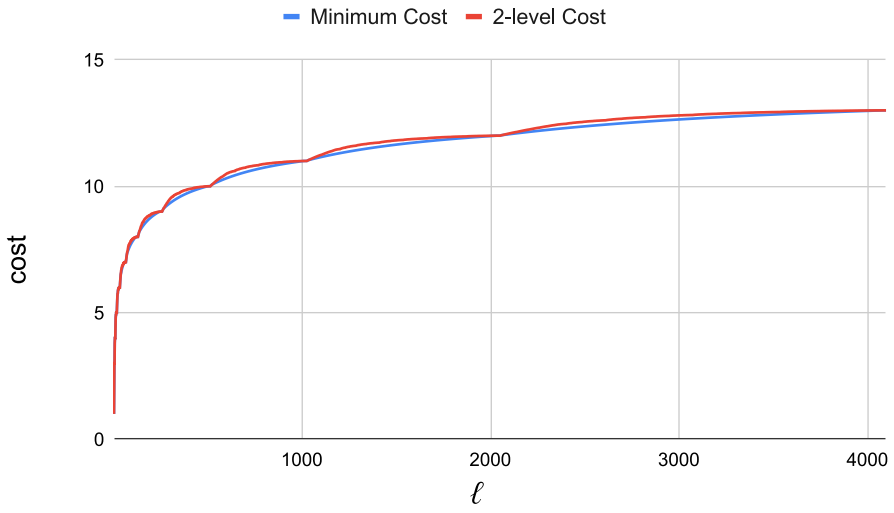

FIGURE 6. Costs of optimal uniform [2]-HHTs and the lower bound of uniform k-HHTs for ℓ symbols.

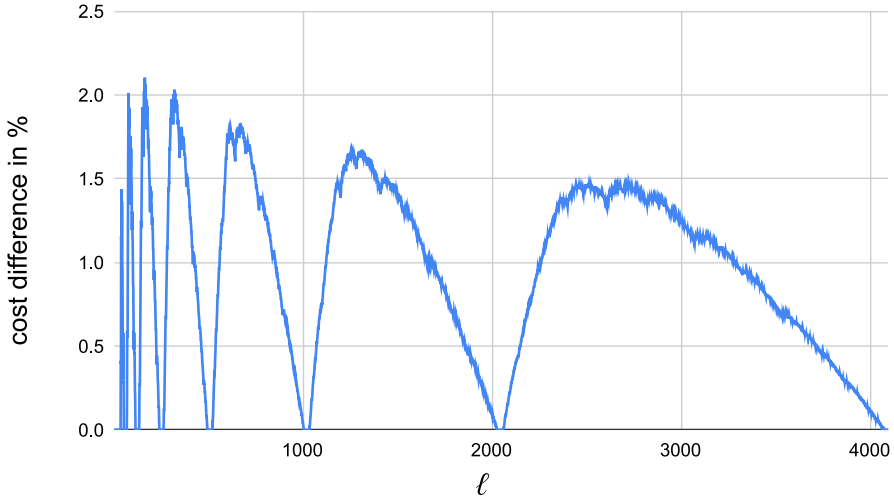

FIGURE 7. Difference in percent between the cost of optimal uniform [2]-HHTs and the cost of the lower bound of uniform k-HHTs for ℓ symbols.

relative differences are depicted in Figures 6 and 7, respectively. By observing the table and the figures one can note that, for symbols with uniform probability of occurrence, the cost of [2]-HHTs are very close to the ones of the lower bound, for all tested values of ℓ. The difference between these costs was no more than 2.1%. Moreover, all the trees obtained by the lower bound have symbol leaves in at most three different levels.

6.2. [2]-HHTs efficiency

In the second experiment, we have compared the costs and the error detection capabilities between [2]-HHTs and Huffman trees. The goal of this experiment is to present the tradeoffs of using [2]-HHTs instead of Huffman trees. In this comparison, we analyze their differences in compression and error detection rates.

Considering the compression, we have performed two tests. First, we compared the costs of uniform [2]-HHTs with the costs of uniform HTs. The second test compares the costs of [2]-HHTs and HTs for the Zipf distribution. The Zipf distribution is well-known for its empirical correspondence with the frequencies of words in natural languages [16]. This relation describes that the ith most frequent word in an alphabet occurs with frequency $\frac{1}{i}$. We use this distribution to simulate real-world compressions. Both these comparisons were done for $10 \leq \ell \leq 1111110$ and the results are shown in Table 2, which is organized similarly to Table 1. For both cases, the difference in the costs of the trees was inversely proportional to the number of symbols being encoded. Considering uniform trees, this difference converged to around 5% and, for the Zipf's distribution, this difference converged to around 25%.

Concerning error detection, we have compared optimal [2]-HHTs, HTs, and even trees. Notice that the Huffman trees have some sort of error detection capability. This occurs when, at the end of the process of decoding, the last node being processed by the HT is not a leaf. In this case, it means that some bits of the message have been corrupted. For this experiment, we build [2]-HHTs and HTs considering the Zipf distribution. The results reported for even trees are those from [13] in which a similar strategy of testing has been used. We have chosen a value for ℓ, in the range $10 \leq \ell \leq 500\,000$, in such a way that the related optimal [2]-HHT has the value

$$\frac{\text{Number of symbol leaves}}{\text{Number of error leaves}}$$

maximized. That is, the resulting tree minimizes the proportion of error leaves in comparison with symbols leaves, meaning that such a tree is the one that has the least capacity of error detection. For the given ℓ, we created an optimal [2]-HHT and an HT for ℓ symbols considering the Zipf distribution. For such trees, we have tested random messages with b symbols, $b \in \{10, 25, 50, 100, 250, 500, 1000, 2500, 5000\}$. For each one of these messages, we introduced i random errors in their bits, for all $1 \leq i \leq \min\{b, 20\}$. For each value of i, we ran the test one million times, counting how many times the tree could detect the error. The percentage of detection of each tree is presented in Table 3. In this table, one may observe that the error detection capability of HTs seems to decrease as b increases. Also, comparing even trees with the optimal [2]-HHT, one may note that in both trees the error detection capability seems to be proportional to b. Moreover, the optimal [2]-HHT seems to have a significantly greater detection capability. For instance, for a message with 500 symbols, the optimal [2]-HHT achieves an error detection rate that is achieved by the even tree only when the message has 5000 symbols.

6.3. Backtracking for optimal uniform HHTs

In the third experiment, we used backtracking to build an optimal uniform HHT for all $1 \leq \ell \leq 38$. We have concluded that, for these values of ℓ, there is always an optimal Hamming–Huffman tree with at most two levels with symbol leaves. Besides that, in some cases, there is also an optimal tree with more than two levels. For instance, for $\ell = 38$, there is also an optimal tree with three levels. Another interesting aspect of this experiment is the fact that, considering optimal uniform Hamming–Huffman trees for 5 symbols, the backtracking obtained the tree depicted in Figure 8 which has a different structure from the one depicted in Figure 1, presented in the literature.

7. Conclusion

In this work, we have presented a restricted case of the problem of building optimal Hamming–Huffman trees. Namely, the problem of building k-Hamming–Huffman trees (k-HHTs), which are Hamming–Huffman trees in which the symbol leaves are distributed in exactly k distinct levels. For $k \leq 2$, we presented a polynomial time algorithm to solve the problem. We showed that such a case is reduced to the problem of finding an independent

TABLE 2. Comparison between the costs of optimal [2]-HHTs and HTs.

Uniform Probabilities				Zipf Distribution			
ℓ	[2]-HHT cost	Huffman cost	% Diff	ℓ	[2]-HHT cost	Huffman cost	% Diff
10	4.800	3.400	41.176	10	4.327	2.934	47.479
20	5.750	4.400	30.682	20	5.263	3.669	43.442
30	6.000	4.933	21.622	30	5.732	4.102	39.757
40	6.700	5.400	24.074	40	6.016	4.403	36.634
50	6.940	5.720	21.329	50	6.271	4.636	35.270
60	7.000	5.933	17.978	60	6.497	4.825	34.645
70	7.343	6.171	18.981	70	6.659	4.988	33.498
80	7.675	6.400	19.922	80	6.811	5.126	32.854
90	7.822	6.578	18.919	90	6.959	5.246	32.652
100	7.920	6.720	17.857	100	7.045	5.352	31.622
110	7.973	6.836	16.622	110	7.145	5.447	31.158
210	8.933	7.781	14.810	210	7.854	6.073	29.331
310	9.584	8.348	14.799	310	8.277	6.436	28.601
410	9.905	8.751	13.183	410	8.589	6.695	28.293
510	10.000	8.996	11.160	510	8.833	6.897	28.066
610	10.538	9.321	13.050	610	9.015	7.062	27.656
710	10.765	9.558	12.629	710	9.159	7.201	27.198
810	10.894	9.736	11.894	810	9.301	7.323	27.014
910	10.965	9.875	11.039	910	9.446	7.430	27.134
1010	10.999	9.986	10.143	1010	9.538	7.525	26.758
1110	11.270	10.155	10.983	1110	9.658	7.610	26.912
2110	12.108	11.059	9.488	2110	10.372	8.187	26.695
3110	12.839	11.683	9.892	3110	10.731	8.534	25.750
4110	13.013	12.007	8.381	4110	11.052	8.780	25.871
5110	13.581	12.397	9.552	5110	11.300	8.973	25.937
6110	13.814	12.659	9.125	6110	11.451	9.130	25.420
7110	13.941	12.848	8.508	7110	11.652	9.263	25.786
8110	13.998	12.990	7.764	8110	11.789	9.378	25.707
9110	14.334	13.202	8.576	9110	11.944	9.480	25.986
10110	14.549	13.379	8.743	10110	12.035	9.572	25.733
11110	14.696	13.525	8.653	11110	12.129	9.654	25.633
21110	15.611	14.448	8.055	21110	12.817	10.212	25.513
31110	15.983	14.947	6.936	31110	13.230	10.544	25.474
41110	16.565	15.406	7.522	41110	13.542	10.782	25.600
51110	16.845	15.718	7.171	51110	13.737	10.965	25.280
61110	16.974	15.928	6.573	61110	13.942	11.115	25.430
71110	17.258	16.157	6.814	71110	14.149	11.243	25.852
81110	17.532	16.384	7.009	81110	14.264	11.353	25.641
91110	17.706	16.561	6.912	91110	14.371	11.450	25.517
101110	17.830	16.704	6.741	101110	14.466	11.536	25.392
111110	17.911	16.820	6.484	111110	14.582	11.615	25.543
211110	18.867	17.758	6.243	211110	15.269	12.149	25.678
311110	19.450	18.315	6.201	311110	15.722	12.471	26.069
411110	19.838	18.725	5.943	411110	15.985	12.701	25.851
511110	19.992	18.974	5.363	511110	16.225	12.881	25.955
611110	20.408	19.284	5.828	611110	16.466	13.028	26.385
711110	20.663	19.525	5.826	711110	16.596	13.153	26.174
811110	20.821	19.707	5.654	811110	16.706	13.261	25.973
911110	20.925	19.849	5.419	911110	16.837	13.357	26.049
1011110	20.987	19.963	5.131	1011110	16.960	13.443	26.163
1111110	21.183	20.113	5.321	1111110	17.080	13.520	26.331

TABLE 3. Comparison between the error detection capabilities of optimal HTs, even trees and [2]-HHTs.

b	Huffman Tree (%)	Even Tree (%)	Optimal [2]-HHT (%)
10	66.1303	90.3540	98.9303
25	73.9825	98.7620	99.8219
50	67.0522	99.0953	99.9172
100	42.4929	99.4894	99.9584
250	34.3946	99.8482	99.9835
500	38.6768	99.9150	99.9923
1000	9.4573	99.9595	99.9958
2500	1.7315	99.9831	99.9983
5000	2.3790	99.9922	99.9994

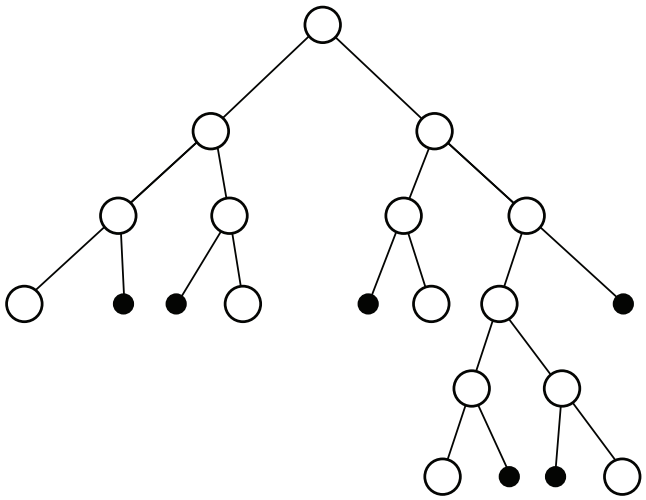

FIGURE 8. An optimal uniform Hamming–Huffman tree for 5 symbols.

set L with a certain size ℓ of a hypercube Q_n such that the cardinality of the neighborhood of L is minimum, over all such independent sets of size ℓ. The latter is a problem well-studied and has already been solved. For $k \geq 3$, we presented an algorithm to evaluate a lower bound on the cost of such trees when the symbols have a uniform probability of occurrence. Moreover, we proved that, for uniform frequencies, an optimal HHT is always a [5]-HHT and that there exists an optimal HHT which is a [4]-HHT.

Lastly, we have made some experiments to investigate the optimality of uniform [2]-HHTs and to measure the capabilities of compression and error detection of [2]-HHTs. Considering these experiments, we conjecture that there is always an optimal uniform HHT in which the leaves lie on at most three levels. We formalize this conjecture as follows.

Conjecture. Let Γ be a set of symbols having the same frequency. There exists an optimal Hamming–Huffman T tree associated with Γ such that T is a [3]-HHT.

Also, we conclude that 2-HHTs are indeed a viable solution to compress text data in real-world situations. In comparison with HTs, its cost is around 25% higher but it provides an excellent error detection rate. For instance, for block messages of size 5000, our experiment showed that the error detection rate is around 99.9994% for 2-HHTs.

Acknowledgements. We are grateful to the anonymous referees for the many insightful suggestions and thorough revision. They pointed out parts of the paper which needed to be rewritten, including some central concepts. This work was partially supported by UBACyT Grants 20020190100126BA and 20020170100495BA (Argentina), FAPERJ, CAPES and CNPq (Brazil).

References

[1] L. Faria, F.S. Oliveira, P.E.D. Pinto and M.S. Sampaio Jr., On the minimum neighborhood of independent sets in the n-cube. *Mat. Contemp.* **44** (2016) 1–10

[2] R.W. Hamming, Coding and Information Theory. Prentice-Hall (1986).

[3] D.A. Huffman, A method for the construction of minimum redundancy codes. *Proc. IRE* **40** (1951) 1098–1101

[4] G.O.H. Katona, The Hamming sphere has minimum boundary. *Stud. Sci. Math. Hung.* **10** (1977) 131–140.

[5] D.E. Knuth. Art of Computer Programming. Vol. 1. Addison-Wesley (2005).

[6] J. Körner and V.K. Wei, Odd and even Hamming spheres also have minimum boundary. *Discrete Math.* **51** (1984) 147–165.

[7] J. Körner and V.K. Wei, Addendum to "odd and even Hamming spheres also have minimum boundary". *Discrete Math.* **62** (1986) 105–106.

[8] J.B. Kruskal, The number of simplices in a complex. In Mathematical Optimization Techniques, edited by R. Bellman. University of California Press (1963) 251–278.

[9] A. Pessoa, An approximation algorithm for constructing error detecting prefix codes. *Optimization* Online (2006).

[10] A. Pessoa, A note on the construction of error detecting/correcting prefix codes. *Inf. Process. Lett.* **107** (2008) 34–38.

[11] P.E.D. Pinto, F. Protti and J.L. Szwarcfiter, A Huffman-based error detecting code. In: Proc. of the Experimental and Efficient Algorithms (WEA'2004). *Lecture Notes in Computer Science* **3059** (2004) 446–457.

[12] P.E.D. Pinto, F. Protti and J.L. Szwarcfiter, Parity codes. *RAIRO – Theor. Inf. Appl.* **39** (2005) 263–278.

[13] P.E.D. Pinto, F. Protti and J.L. Szwarcfiter, Exact and experimental algorithms for a huffman-based error detecting code. *Lect. Notes Comput. Sci.* **5532** (2009) 311–324.

[14] P.E.D. Pinto, F. Protti and J.L. Szwarcfiter, Exact and approximation algorithms for error-detecting even codes. *Theor. Comput. Sci.* **440,441** (2012) 60–72.

[15] T. Wenisch, P.F. Swaszek and A.K. Uht, Combined error correction and compression codes. In: IEEE International Symposium on Information Theorys (2001) 238.

[16] G.K. Zipf, Human Behaviour and the Principle of Least Effort. Addison-Wesley (1949).

[17] https://github.com/moysessj/restricted-hamming-huffman-trees.git (Accessed: May 17, 2022).