RESTRICTED HAMMING-HUFFMAN TREES

Min C. Lin¹, Fabiano S. Oliveira², Paulo E. D. Pinto², Moysés S. Sampaio Jr³ and Jayme L. Szwarcfiter²,³

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.

Keywords: Hamming-Huffman codes, hypercube graphs, minimum neighborhood

Mathematics Subject Classification. 68P30, 94Bxx, 05C90
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 [2], 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 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 [4]. 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 [10,11] with a low additive error with respect to the entropy. Hamming-Huffman trees inspired other codes, such as in [12,13], 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 [15], 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 [16].

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 [3] 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 1, basic definitions and notations are presented. In Section 2, the problem of building Hamming-Huffman trees in which all symbol leaves lie on the same level is tackled. In Section 3, the problem of building Hamming-Huffman trees in which the leaves are distributed in two distinct levels is discussed. In Section 4, 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 5, we present some experimental results. Concluding remarks are presented in the last section.

1. 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 $\ell$, of $Q_n$ as

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

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 $\Gamma$, 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 1(a), 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 1(a) depicts a uniform HT $T$ with $c(T) = 2.4$ on 5 symbols.

![Figure 1(a)](image)

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 $d(e, c(a)) = 1$, 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 1(b) 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 $\text{contract}(T)$ is a strict HHT, where $\text{contract}(T)$ 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 $\text{contract}(T')$ recursively to obtain the final transformation. Figure 2(a) presents an HHT $T$ in this weaker sense, and Figure 2(b) shows the strict HHT resulting of $\text{contract}(T)$. Note that $\text{contract}(T)$ 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 2(a), symbol leaves are encoded as $000$ and $111$, resulting in 6 error leaves, whereas in Figure 2(c), 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 [2], the construction of optimal HHTs is open since defined by Hamming in the eighties [4]. 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

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

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.
2. 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 $\ell$ symbols and minimum neighborhoods of independent sets with $\ell$ 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
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 1. Before presenting the theorem, we have to state the following auxiliary lemma.

**Lemma 1** ([7], [5]). For any given non-negative integers $\ell$ and $n$, $\ell < 2^n$, the number $m$ has a unique representation

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

such that

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

The representation presented in Lemma 1 is defined as the $n$-bounded canonical representation of $\ell$.

Also, in [5], Katona defined the function $G(\ell, n)$ as follows:

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

where the $a$'s are those from Lemma 1 with respect to the given $\ell$.

The following theorem was proven in [9].

**Theorem 1** (Theorem 2 in [8] and [9]). For every $\ell \leq 2^n - 1$

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

A consequence of this theorem in [8], 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 2**. Let $\Gamma$ be a set of $\ell$ 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 [8], one may consider without loss of generality that the set of $\ell$ codifications consists of elements having a same parity.
In order to choose $\ell$ 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 $\ell$ 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 $\ell$ 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 $c(T) = (\lceil \log_2 \ell \rceil + 1) \sum_{a \in \Gamma} p(a)$.

3. 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 [6]. 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 5. 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 2, 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 $\ell_1$ 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 $\ell_1$ 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 $(h_1, \ell_1)$, 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

\[
\rho(\ell_1, h_1) = 2^{h_1} - \ell_1 - \varphi(\ell_1, h_1)
\]  

For the cost \( c(T) = c(\ell, \ell_1, h_1) \) of this particular way of building a \( 2 \)-HHT \( T \) having \( \ell_1 \) 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 \( \ell_2 \) symbols at level \( h_2 \). So, the choices of \( h_1, \ell_1 \) turned out to lead to an unfeasible solution. If \( \rho(\ell_1, h_1) > 0 \), then the remaining \( \ell_2 \) 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 2, 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 \( \Gamma \) to the set of chosen symbols leaves. But to minimize such a cost, it clearly suffices to place at level \( h_1 \) the \( \ell_1 \) 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

\[
c(\ell, \ell_1, h_1) = h_1 \sum_{i=1}^{\ell_1} p(a_i) + h_2 \sum_{i=\ell_1 + 1}^{\ell} p(a_i) \\
= h_1 \sum_{i=1}^{\ell_1} p(a_i) + (h_1 + h'(\ell_1, h_1)) \sum_{i=\ell_1 + 1}^{\ell} p(a_i) \\
= h_1 \sum_{i=1}^{\ell} p(a_i) + h'(\ell_1, h_1) \sum_{i=\ell_1 + 1}^{\ell} p(a_i) \\
= h_1 + h'(\ell_1, h_1) \sum_{i=\ell_1 + 1}^{\ell} p(a_i)
\]

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 \( \ell_1 \) over all possible values. Formally, the cost of an optimal \( 2 \)-HHT \( T \) is given by

\[
c(T) = \min\{c(\ell, \ell_1, h_1) : 1 \leq h_1 < h(\ell), 1 \leq \ell_1 \leq \min\{\ell - 1, 2^{h_1 - 1}\}\}
\]
Figure 3. A Hamming-Huffman tree with leaves on two levels.

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 \( 1 \leq h_1 < h(\ell) \), there are at most \( 2^{h_1-1} \) possible values for \( \ell_1 \). Therefore, there are at most \( 1 + 2 + 2^2 + \ldots + 2^{\log_2 \ell} = 2^{\log_2 \ell} - 1 = \Theta(\ell) \) distinct pairs \((h_1, \ell_1)\). 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 \( \sum_{i=1}^{\ell} p(a_i) \) for all \( 1 \leq i \leq \ell \), 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 3.** Let \( \Gamma \) be a set of \( \ell \) symbols, each \( a \in \Gamma \) having probability \( p(a) \). The cost of an optimal \( 2\text{-HHT} \) \( T \) is

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

where

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

and

\[ h'(\ell_1, h_1) = h\left(\left\lceil \frac{\ell}{\rho(\ell_1, h_1)} \right\rceil \right) \]

Furthermore, this cost can be computed in time \( O(\ell \log^2 \ell) \).
4. 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 1(a), \( \lambda(T) = 1 \), where \( \lambda(T) = 0 \) for any tree \( T \) of Figure 2. We prove that \( \lambda(T) \leq 4 \) for all optimal uniform HHT \( T \). Moreover, we show that there is always an optimal uniform HHT \( T \) in which \( \lambda(T) \leq 3 \). 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 \( k \leq 3 \) distinct levels, and we provide empirical evidence in favor of it. Section 5 compares the lower bound of this section with the cost of optimal 2-HHTs, the latter being computed as presented in Section 3.

Let \( T \) be a uniform HHT on \( \ell \) symbols. Consider the following operation over a symbol leaf \( s \) of \( T \):

- \( \text{descend}(s) \) (see Figure 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 \( T_s \). Finally, transform the node of \( s'' \) into an error leaf. Let \( T' \) be the resulting tree. Apply \( \text{contract}(T') \) to obtain the final transformation.

![Figure 4](image-url) Operation \( \text{descend}(s) \), over a symbol leaf. The dotted node represents a free node to be used in the encoding of another symbol.

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

**Lemma 2.** Let \( T \) be an HHT, \( s \) be one of its symbol leaves and \( T' \) be the tree obtained by the operation \( \text{descend}(s) \). 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. \( \square \)

Let \( p = \frac{1}{2} \) be the probability of occurrence of the symbol leaves of \( T \) and \( T' \) be the tree obtained by the operation \( \text{descend}(s) \), for some symbol leaf \( s \) of \( T \). Let \( l_1 \) be the level of \( s \) and \( l_2 \) 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

\[
c(T') = c(T) - p(\lambda(T) - 4)
\]  

(2)

**Theorem 4.** 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 \( \text{descend}(s) \) 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 (2), we have that \( c(T') < c(T) \). Moreover, note that each application of \( \text{descend}(s) \) eliminates a leaf in level \( l_1 \) and a leaf in level \( l_k \). By successive applications of \( \text{descend}(s) \) to symbol leaves, it is possible to obtain an optimal uniform HHT such that \( \lambda(T) \leq 3 \). \( \square \)

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 \( \ell \) symbols and \( r \) trees is as follows. Consider \( h_1 \) to be the first level in which symbol leaves appear in \( F \). Let \( \ell_1 \) be the number of symbol leaves to be represented in the level \( h_1 \). Clearly, the most desirable arrangement for choosing \( \ell_1 \) 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 \(\ell\) 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 \(\ell\) 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 \(\ell\) 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 2 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 \((\ell_1, h_1)\). Note that despite the fact that there are only \(\ell_1\) 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 \(h_1(\ell_1 + (\ell - \ell_1)) = h_1 \ell_1\).

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

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

where:

- \(D_1 = \{1, \ldots, \ell - k + 1\}\),
- \(D_2 = \{h(\lceil \frac{\ell}{r} \rceil), \ldots, h(\lceil \frac{\ell}{r} \rceil)\}\), and
- \(\rho_F(r, \ell, h)\) denotes the maximum number of free nodes when \(\ell\) 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 \(\rho_F\) 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 3. Moreover, considering general uniform HHTs, the
cost of an optimal uniform HHT with $\ell$ symbol is at least
\[
\min\{c_F(k, 1, \ell) : 1 \leq k \leq \ell\}
\] (3)

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 $\ell$ 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 $\ell_1$ symbols are to be allocated into a single HHT. Therefore, the remaining $\ell - \ell_1$ symbols have to be allocated among the 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 (1). The latter can be determined using recursion. Formally, we have

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

Precomputing the values of $\rho_F$ requires a matrix whose number of elements is $O(\ell^2 \log \ell)$, as $r$ is limited by $\ell$ 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 $\rho_F$ takes time $O(\ell^3 \log \ell)$. Assuming that the values of $\rho_F$ 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 4 and the remaining parameters are limited by $\ell$. 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 $O(\ell^3)$. 
5. 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 3. It compares uniform $[2]$-HHTs with the lower bound described in Section 4. 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 [1].

5.1. 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 Section 2 and Section 3 with those produced by the algorithm in Section 4. 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 4. 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 relative differences are depicted in Figure 6 and Figure 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 $\ell$. 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.

5.2. $[2]$-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 [17]. This relation describes that the $i$-th most frequent word
Table 1. Comparison between the cost of optimal $k$-HHTs and the lower bound on the cost of $k$-HHTs.
in an alphabet occurs with frequency $\frac{1}{\ell}$. 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 [14] in which a similar strategy of testing has been used. We have chosen a value for \( \ell \), in the range \( 10 \leq \ell \leq 500000 \), 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 \( \ell \), we created an optimal [2]-HHT and an HT for \( \ell \) 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.

5.3. 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 \( \ell \), 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.
<table>
<thead>
<tr>
<th>Uniform Probabilities</th>
<th>Zipf Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell$</td>
<td>[2]-HHT cost</td>
</tr>
<tr>
<td>10</td>
<td>4.800</td>
</tr>
<tr>
<td>20</td>
<td>5.750</td>
</tr>
<tr>
<td>30</td>
<td>6.000</td>
</tr>
<tr>
<td>40</td>
<td>6.700</td>
</tr>
<tr>
<td>50</td>
<td>6.940</td>
</tr>
<tr>
<td>60</td>
<td>7.000</td>
</tr>
<tr>
<td>70</td>
<td>7.343</td>
</tr>
<tr>
<td>80</td>
<td>7.675</td>
</tr>
<tr>
<td>90</td>
<td>7.822</td>
</tr>
<tr>
<td>100</td>
<td>7.920</td>
</tr>
</tbody>
</table>

<p>| Table 2. Comparison between the costs of optimal [2]-HHTs and HTs. |</p>
<table>
<thead>
<tr>
<th>b</th>
<th>Huffman Tree (%)</th>
<th>Even Tree (%)</th>
<th>Optimal [2]-HHT (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>66.1303</td>
<td>90.3540</td>
<td>98.9303</td>
</tr>
<tr>
<td>25</td>
<td>73.9825</td>
<td>98.7620</td>
<td>99.8219</td>
</tr>
<tr>
<td>50</td>
<td>67.0522</td>
<td>99.0953</td>
<td>99.9172</td>
</tr>
<tr>
<td>100</td>
<td>42.4929</td>
<td>99.4894</td>
<td>99.9584</td>
</tr>
<tr>
<td>250</td>
<td>34.3946</td>
<td>99.8482</td>
<td>99.9835</td>
</tr>
<tr>
<td>500</td>
<td>38.6768</td>
<td>99.9150</td>
<td>99.9923</td>
</tr>
<tr>
<td>1000</td>
<td>9.4573</td>
<td>99.9595</td>
<td>99.9958</td>
</tr>
<tr>
<td>2500</td>
<td>1.7315</td>
<td>99.9931</td>
<td>99.9983</td>
</tr>
<tr>
<td>5000</td>
<td>2.3790</td>
<td>99.9922</td>
<td>99.9994</td>
</tr>
</tbody>
</table>

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

Figure 8. An optimal uniform Hamming-Huffman tree for 5 symbols.

6. 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 set $L$ with a certain size $\ell$ of a hypercube $Q_n$ such that the cardinality of the neighborhood of $L$ is minimum, over all such independent sets of size $\ell$. 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 1.** Let $\Gamma$ be a set of symbols having the same frequency. There exists an optimal Hamming-Huffman $T$ tree associated with $\Gamma$ 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.

**REFERENCES**


