3-1-1994

Decoding Linear Block Codes Using a Priority-First Search: Performance Analysis and Suboptimal Version

Yunghsiang S. Han
Syracuse University

Carlos R.P. Hartmann
Syracuse University, chartman@syr.edu

Kishan Mehrotra
Syracuse University, mehrotra@syr.edu

SU-CIS-94-4

Decoding Linear Block Codes Using a Priority-First Search: Performance Analysis and Suboptimal Version

Yunghsiang S. Han, Carlos R.P. Hartmann,
and Kishan G. Mehrotra

March 1994

School of Computer and Information Science
Syracuse University
Suite 4-116, Center for Science and Technology
Syracuse, NY 13244-4100

Decoding Linear Block Codes Using a Priority-First Search: Performance Analysis and Suboptimal Version[1]

Yunghsiang S. Han[2]

Carlos R. P. Hartmann[3]

Kishan G. Mehrotra[4]

Index Terms: block codes, decoding, Dijkstra's algorithm, maximum-likelihood, priority-first search, soft-decision, trellis, performance analysis, suboptimal

---

[1]This work was partially supported by the National Science Foundation under Grant NCR-9205422, and used the computational facilities of the Northeast Parallel Architectures Center (NPAC) at Syracuse University.

[2]Y. S. Han was with the School of Computer and Information Science at Syracuse University, Syracuse, NY 13244-4100. He is now with the Dept. of Electronic Engineering, HuaFan College of Humanities and Technology, 1., HuaFan Rd., Shihtin Hsiang, Taipei Hsien, Taiwan.

[3]C. R. P. Hartmann is with the School of Computer and Information Science at Syracuse University, Syracuse, NY 13244-4100 (e-mail: hartmann@top.cis.syr.edu; Fax: (315) 443-1122.).

[4]K. G. Mehrotra is with the School of Computer and Information Science at Syracuse University, Syracuse, NY 13244-4100 (e-mail: kishan@top.cis.syr.edu; Fax: (315) 443-1122.).

Abstract

An efficient maximum-likelihood soft-decision decoding algorithm for linear block codes using a generalized Dijkstra's Algorithm was proposed by Han, Hartmann, and Chen. In this report we prove that this algorithm is efficient for most practical communication systems where the probability of error is less than 10^{-3} by finding an upper bound of the computation performance of the algorithm. A suboptimal decoding algorithm is also proposed. The performance of this suboptimal decoding algorithm is within 0.25 dB and 0.5 dB of the performance of an optimal decoding algorithm for the $(104, 52)$ binary extended quadratic residue code and the $(128, 64)$ binary extended BCH code, respectively.

1 Introduction

The use of block codes is a well-known error-control technique for reliable transmission of digital information over noisy communication channels. Linear block codes with good coding gains have been known for many years; however, these block codes have not been used in practice for lack of an efficient soft-decision decoding algorithm.

Several researchers [1, 16, 13] have presented techniques for decoding linear block codes that convert the decoding problem into a graph-search problem on a trellis derived from the parity-check matrix of the code. Thus the *maximum-likelihood decoding* (MLD) rule can be implemented by applying the Viterbi Algorithm [15] to this trellis. In practice, however, this breadth-first search scheme can be applied only to codes with small redundancy or to codes with a small number of codewords [10].

We recently proposed a novel maximum-likelihood soft-decision decoding algorithm for linear block codes [6, 7]. This algorithm uses a generalization of Dijkstra's algorithm (GDA) [11] to search through the trellis for a code equivalent to the transmitted code. The use of this priority-first search strategy for decoding drastically reduces the search space and results in an efficient optimal soft-decision decoding algorithm for linear block codes. Furthermore, in contrast with Wolf's algorithm [16], the decoding efforts of our decoding algorithm are adaptable to the noise level.

In Section 2 we review MLD of linear block codes, describe the code tree for a linear code, and briefly state the decoding algorithm proposed in [7]. In Section 3 we give an upper bound on the computation performance of this algorithm. In the next section we present a suboptimal decoding algorithm. Simulation results for the (104,52) binary extended quadratic residue code and the (128,64) binary extended BCH code are given in Section 5. Concluding remarks are presented in Section 6.

2 Preliminaries

Let $\boldsymbol{C}$ be a binary (n, k) linear code with generator matrix $\boldsymbol{G}$, and let $\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1})$ be a codeword of $\boldsymbol{C}$ transmitted over a time-discrete memoryless channel with output alphabet $\boldsymbol{B}$. Furthermore, let $\boldsymbol{r} = (r_0, r_1, \ldots, r_{n-1})$, $r_j \in \boldsymbol{B}$ denote the received vector, and assume

that $\boldsymbol{Pr}(r_j|c_i) > 0$ for $r_j \in \boldsymbol{B}$ and $c_i \in \boldsymbol{GF}(2)$. Let $\hat{\boldsymbol{c}}$ be an estimate of the transmitted codeword $\boldsymbol{c}$.

The *maximum-likelihood decoding rule* (MLD rule) for a time-discrete memoryless channel can be formulated as

$$\text{set } \hat{\boldsymbol{c}} = \boldsymbol{c}_\ell, \text{ where } \boldsymbol{c}_\ell = (c_{\ell 0}, c_{\ell 1}, \ldots, c_{\ell(n-1)}) \in \boldsymbol{C} \text{ and}$$

$$\sum_{j=0}^{n-1} (\phi_j - (-1)^{c_{\ell j}})^2 \leq \sum_{j=0}^{n-1} (\phi_j - (-1)^{c_{ij}})^2 \quad \text{for all } \boldsymbol{c}_i = (c_{i0}, c_{i1}, \ldots, c_{\ell(n-1)}) \in \boldsymbol{C},$$

$$\text{where } \phi_j = \ln \frac{\boldsymbol{Pr}(r_j|0)}{\boldsymbol{Pr}(r_j|1)} .$$

We, therefore, may consider that the "received vector" is $\boldsymbol{\phi} = (\phi_0, \phi_1, \ldots, \phi_{n-1})$. In the special case where the codewords of $\boldsymbol{C}$ have equal probability of being transmitted, the MLD rule minimizes error probability.

We now give a short description of our decoding algorithm presented in [7]. This algorithm uses the priority-first search strategy, thus avoiding traversing the entire trellis, and, guided by an evaluation function f, searches through a graph that is a trellis for a code $\boldsymbol{C}^*$, which is equivalent to code $\boldsymbol{C}$. $\boldsymbol{C}^*$ is obtained from $\boldsymbol{C}$ by permuting the positions of codewords of $\boldsymbol{C}$ in such a way that the first k positions of codewords in $\boldsymbol{C}^*$ correspond to the "most reliable linearly independent" positions in the received vector $\boldsymbol{\phi}$. $\boldsymbol{G}^*$ is a generator matrix of $\boldsymbol{C}^*$ whose first k column forms a $k \times k$ identity matrix. In our decoding algorithm the vector $\boldsymbol{\phi}^* = (\phi_0^*, \phi_1^*, \ldots, \phi_{n-1}^*)$ is used as the "received vector." It is obtained by permuting the positions of $\boldsymbol{\phi}$ in the same manner in which the columns of $\boldsymbol{G}$ can be permuted to obtain $\boldsymbol{G}^*$.

Since the probability that our decoding algorithm will revisit a node of the trellis is very small, our implementation of this decoding algorithm did not check for repeated nodes [7]. In this case, the graph where the search is performed is an expanded version of a trellis that is denoted by a code tree. A code tree is a way to represent every codeword of an (n, k) code $\boldsymbol{C}^*$ as a path through a tree containing $n+1$ levels . The code tree can be treated as an expanded version of the trellis, where every path is totally distinct from every other path. The leftmost node is called the *start node*, which is at level -1. There are two branches, labeled by 0 and 1, respectively, that leave each node at the first k levels. After the k levels, there is only one branch leaving each node. The 2^k rightmost nodes are called *goal nodes*, which are at level $n-1$.

Next, we describe how to determine the sequence of labels encountered when traversing a path from a node at level k to a goal node. Let $c_0, c_1, \ldots, c_{k-1}$ be the sequence of labels encountered when traversing a path from the start node to a node m at level $k-1$. Then $c_k, c_{k+1}, \ldots, c_{n-1}$, the sequence of labels encountered when traversing a path from node m to a goal node, can be obtained as follows:

$$(c_0, c_1, \ldots, c_k, c_{k+1}, \ldots, c_{n-1}) = (c_0, c_1, \ldots, c_{k-1})\boldsymbol{G}^*.$$

We first specify the arc costs in the code tree of $\boldsymbol{C}^*$. The cost of the arc from a node at level $t-1$ to a node at level t is assigned the value $(\phi_t^* - (-1)^{c_t^*})^2$, where c_t^* is the label of the

arc. Thus the solution of the decoding problem is converted into finding a path from the start node to a goal node, that is, a codeword $\boldsymbol{c}^* = (c_0^*, c_1^*, \ldots, c_{n-1}^*)$ such that $\sum_{i=0}^{n-1}(\phi_i^* - (-1)^{c_i^*})^2$ is minimum among all paths from the start node to goal nodes. Such a path is denoted as an optimal path.

Now we define the evaluation function f for every node m in the code tree as $f(m) = g(m) + h(m)$, where $g(m)$ is the cost of the path from the start node to node m and $h(m)$ is an estimate of the minimum cost among all the paths from node m to goal nodes. The cost of a path is obtained by summing all the arc costs encountered while constructing this path. GDA requires that for all nodes m_i and m_j such that node m_j is an immediate successor of node m_i,

$$h(m_i) \leq h(m_j) + c(m_i, m_j), \tag{1}$$

where $c(m_i, m_j)$ is the arc cost between node m_i and node m_j. This requirement guarantees that GDA will find an optimal path. In GDA, the next node to be expanded is one with the smallest value of f on the list of all leaf nodes (list OPEN) of the subtree constructed so far by the algorithm. Thus, list OPEN must be kept ordered according to the values f of its nodes. When the algorithm chooses to expand a goal node, it is time to stop, because the algorithm has constructed a path with minimum cost, i.e., an optimal path.

We now define our function h, which satisfies (1). In order to define a function h that is a "good" estimator, we must use properties of the linear block code that are invariant under any permutation of the positions of the codewords.

Let $HW = \{w_i | 0 \leq i \leq I\}$ be the set of all distinct Hamming weights that codewords of $\boldsymbol{C}$ may have. Furthermore, assume $w_0 < w_1 < \cdots < w_I$. Our heuristic function is defined to take into consideration the linear property of $\boldsymbol{C}^*$ and that the Hamming distance between any two codewords of $\boldsymbol{C}^*$ must belong to HW.

Let $\boldsymbol{c}^*$ be a given codeword of $\boldsymbol{C}^*$. Our function h will be defined with respect to $\boldsymbol{c}^*$, which is called the seed of the decoding algorithm.

1. For nodes at level $\ell, -1 \leq \ell < k-1$:

   Let m be a node at level ℓ, and let $\overline{v}_0, \overline{v}_1, \ldots, \overline{v}_\ell$ be the labels of the path $\boldsymbol{P}'_m$ from the start node to node m. We now construct the set, $T(m)$, of all binary n-tuples $\boldsymbol{v}$ such that their first $\ell+1$ entries are the labels of $\boldsymbol{P}'_m$ and $d_H(\boldsymbol{v}, \boldsymbol{c}^*) \in HW$, where $d_H(\boldsymbol{x}, \boldsymbol{y})$ is the Hamming distance between $\boldsymbol{x}$ and $\boldsymbol{y}$. That is,

   $$T(m) = \{\boldsymbol{v} | \boldsymbol{v} = (\overline{v}_0, \overline{v}_1, \ldots, \overline{v}_\ell, v_{\ell+1}, \ldots, v_{n-1}) \text{ and } d_H(\boldsymbol{v}, \boldsymbol{c}^*) \in HW\}.$$

   Note that $T(m) \neq \emptyset$. This can easily be seen by considering the binary k-tuple $\boldsymbol{u} = (\overline{v}_0, \overline{v}_1, \ldots, \overline{v}_\ell, 0, \ldots, 0)$ and noting that $\boldsymbol{u} \cdot \boldsymbol{G}^* \in T(m)$.

   We now define function h as

   $$h(m) = \min_{\boldsymbol{v} \in T(m)} \left\{ \sum_{i=\ell+1}^{n-1} (\phi_i^* - (-1)^{v_i})^2 \right\}.$$

2. For nodes at level $\ell, k-1 \leq \ell < n$:

   Let m be a node at level ℓ. We define function h as

$$h(m) = \sum_{i=\ell+1}^{n-1} \left(\phi_i^* - (-1)^{v_i^*}\right)^2,$$

   where $v_{\ell+1}^*, v_{\ell+2}^*, \ldots, v_{n-1}^*$ are the labels of the only path $\boldsymbol{P}_m$ from node m to a goal node. Note that if node m is a goal node, then $h(m) = 0$.

In [6, 7] it is shown that our decoding algorithm is a depth-first search type algorithm. Thus, upper bounds (UBs) on the cost of an optimal path are obtained whenever a codeword is generated. These UBs can be used to reduce the size of list OPEN. More details about this decoding algorithm can be found in [6, 7] where we also described other speed-up techniques. Furthermore, the algorithm will still find an optimal path even if in the computation of function h the algorithm considers all the Hamming weights of any superset of HW.

3 Analysis of the performance of the algorithm

Our decoding algorithm can be considered as a branch-and-bound type of algorithm. In general, it is difficult to form an idea of how well a branch-and-bound algorithm will perform on a given problem [2]; however, we can derive an upper bound on the average number of nodes visited by our decoding algorithm, which shows that this decoding algorithm is very efficient for most practical communication systems where the probability of error is less than 10^{-3}.

In order to derive this upper bound we will define another heuristic function h_s that satisfies the condition $h_s(m) \leq h_p(m)$ for every node of the code tree, where function h_p is defined in the previous section. By Theorem 7 in Chapter 3 of [12] the decoding algorithm using function h_p will never open more nodes than the decoding algorithm using function h_s.

We now define function h_s. Let m be a node at level $\ell < k-1$, and let $\overline{v}_0, \overline{v}_1, \ldots, \overline{v}_\ell$ be the labels of the path $\boldsymbol{P}'_m$ from the start node to node m. Define h_s as

$$h_s(m) = \sum_{i=\ell+1}^{n-1} \left(|\phi_i| - 1\right)^2.$$

For a node at a level greater than $k-2$, the function h_s will be defined as in the previous section. It is easy to see that $h_s(m) \leq h_p(m)$ for every node of the code tree.

We now show that if m is not start node m_s, and m is at level $\ell < k-1$, then the time complexity of calculating $h_s(m)$ is a constant. Let node m be an immediate successor of node m', which is on path $\boldsymbol{P}'_m$. Furthermore, let $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$ be the hard-decision of $\boldsymbol{\phi}$. That is,

$$y_i = \begin{cases} 1 & \text{if } \phi < 0, \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$h_s(m) = h_s(m') - (|\phi_\ell| - 1)^2.$$

Consequently,

$$f(m) = f(m') + (y_\ell \oplus \overline{v}_\ell)(4 \times |\phi_\ell|),$$

where $\overline{v}_\ell$ is the label of the arc between node m' and node m and $\oplus$ denotes a modulo 2 addition. Thus, the time complexity of calculating $f(m)$ is a constant when node m is not start node m_s.

In order to obtain an upper bound of the computational performance of our decoding algorithm, we first derive an upper bound of the computational performance of a simplified version of it, which we denote by SDA. In this version

1. we do not order the positions of $\boldsymbol{\phi}$;

2. we use function h_s as the heuristic function.

We now state the main results of the computational performance of SDA when code $\boldsymbol{C}$ is transmitted over the AWGN channel. In order to account for the redundancy in codes of different rates, we use the SNR per transmitted information bit $\gamma_b = E_b/N_0 = \gamma n/k$.

Theorem 1 *Let $\overline{N}_s$ be the average number of nodes visited by SDA, and let G be the standard normal distribution. Then*

$$\overline{N}_s \leq \widetilde{N},$$

where

$$\begin{aligned}
\widetilde{N} &= 2\left[k + \sum_{\ell=0}^{k-2}\sum_{\overline{d}=1}^{\ell+1}\binom{\ell+1}{\overline{d}} G\left(-\frac{\overline{\mu}(\ell,\overline{d})}{\overline{\sigma}(\ell,\overline{d})}\right)\right], \\
\overline{\mu}(\ell,\overline{d}) &= \sqrt{N_0}\left\{2\overline{d}\sqrt{\frac{k}{n}\gamma_b} + (n-\ell-1)\left[2\sqrt{\frac{k}{n}\gamma_b}G(-\sqrt{2\frac{k}{n}\gamma_b}) - \frac{1}{\sqrt{\pi}}e^{-\frac{k}{n}\gamma_b}\right]\right\}, \\
&\textit{and} \\
\overline{\sigma}^2(\ell,\overline{d}) &= N_0\left\{2\overline{d} + (n-\ell-1)\left[(4\frac{k}{n}\gamma_b + 2)G(-\sqrt{2\frac{k}{n}\gamma_b}) - 2\sqrt{\frac{\frac{k}{n}\gamma_b}{\pi}}e^{-\frac{k}{n}\gamma_b}\right.\right. \\
&\qquad \left.\left. - \left(2\sqrt{\frac{k}{n}\gamma_b}G(-\sqrt{2\frac{k}{n}\gamma_b}) - \frac{1}{\sqrt{\pi}}e^{-\frac{k}{n}\gamma_b}\right)^2\right]\right\}.
\end{aligned}$$

The proof of Theorem 1 is given in Appendix A.

Recall that in the decoding algorithm proposed in [7] we ordered the position of $\boldsymbol{\phi}$ to obtain $\boldsymbol{\phi}^*$, which is assumed to be the "received vector."

Theorem 2 *If the k-most reliable positions of $\boldsymbol{\phi}$ are linearly independent, then*

$$N_s(\boldsymbol{\phi}^*) \leq N_s(\boldsymbol{\phi}),$$

where $N_s(\boldsymbol{\phi}^*)$ and $N_s(\boldsymbol{\phi})$ are the number of nodes visited by SDA when $\boldsymbol{\phi}^*$ and $\boldsymbol{\phi}$ are decoded, respectively.

The proof of Theorem 2 can be found in Appendix B.

Let $N(\boldsymbol{\phi}^*)$ be the number of nodes visited by the decoding algorithm proposed in [7] when it decodes $\boldsymbol{\phi}^*$. By Theorem 7 in Chapter 3 of [12] we have the following result.

Theorem 3 *If the k-most reliable positions of $\boldsymbol{\phi}$ are linearly independent, then*

$$N(\boldsymbol{\phi}^*) \leq N_s(\boldsymbol{\phi}^*).$$

Now for every (n, k) linear code $\boldsymbol{C}$ we define the set

$$S(\boldsymbol{C}) = \{\boldsymbol{\phi} | \text{ the } k\text{-most reliable positions of } \boldsymbol{\phi} \text{ are linearly independent.}\}$$

From the above theorems and Chebyshev's inequality [5], we have the following result.

Theorem 4 *For any $\boldsymbol{\phi} \in S(\boldsymbol{C})$,*

$$\boldsymbol{Pr}(N(\boldsymbol{\phi}) \geq L) \leq \frac{\widetilde{N}}{L},$$

where L is any positive real number.

We remark here that it is not always true that the k-most reliable positions of $\boldsymbol{\phi}$ are linearly independent. In this case, we cannot guarantee that $N(\boldsymbol{\phi}^*) \leq N_s(\boldsymbol{\phi}^*)$. However, in our simulations we have never encountered a case where $N(\boldsymbol{\phi}^*) > N_s(\boldsymbol{\phi}^*)$. Therefore, we can take $\widetilde{N}$ to be a good estimator of an upper bound on $\overline{N}$, the average number of nodes visited by our decoding algorithm [7].

The values of $\widetilde{N}$ for the $(48, 24)$ binary extended quadratic residue code for γ_b equal to 2 dB, 3 dB, 4 dB, 5 dB, 6 dB, 7 dB, and 8 dB are given in Figure 1. In this figure is also given the simulation results of the average number of nodes visited by the SDA, and by the decoding algorithm proposed in [7]. These averages were obtained by simulating 10,000 samples. We remark here that the upper bound on $\overline{N}$ that we derived is not tight, because of the simplifying assumptions we had to make.

In Figure 2 we give the values of $\widetilde{N}$ for the $(104, 52)$ binary extended quadratic residue code and the $(128, 64)$ binary extended BCH code for γ_b from 2 dB to 8 dB, respectively. Even though this upper bound is not tight, we may conclude that the decoding algorithms proposed in [7] are efficient for codes of moderate lengths for most practical communication systems where the probability of error is less than 10^{-3} (γ_b greater than 6.8 dB).

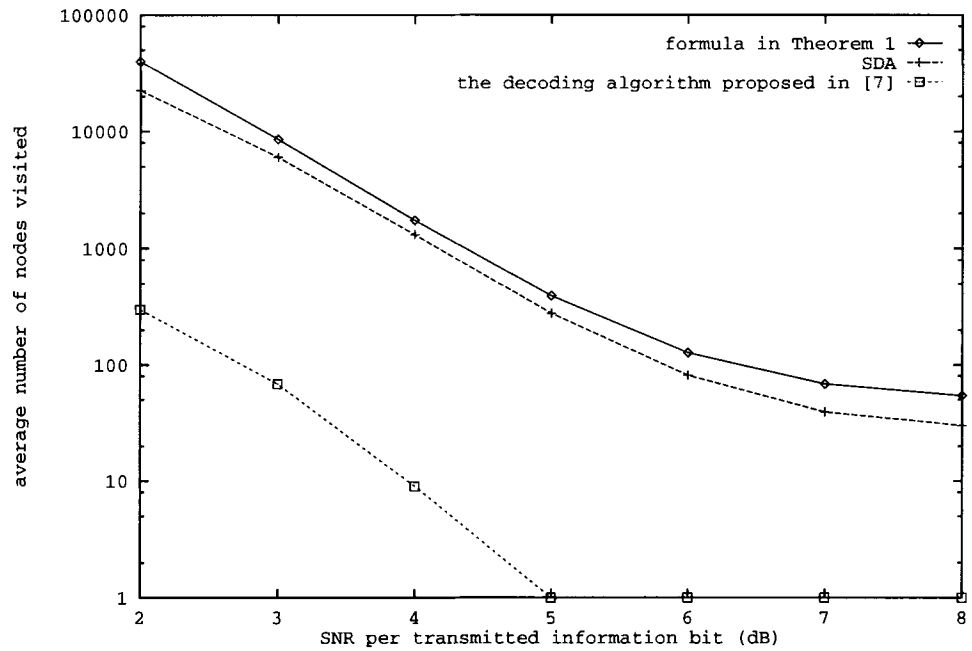

Figure 1: Average number of nodes visited for the $(48, 24)$ code

4 Suboptimal decoding algorithm

In the previous section we showed that GDA is very efficient for codes of moderate lengths for most practical communication systems where probability of error is less than 10^{-3}; however, for low SNRs the number of nodes on list OPEN is still too large for the algorithm to have practical application.

The results of our simulations have shown that the number of nodes that need to be stored on list OPEN before an optimal path is found is considerably smaller than the total number of nodes stored before the algorithm stops. Thus we may limit the search with small degradations on the performance of the algorithm.

In this section we present a suboptimal soft-decision decoding algorithm. In this algorithm we limit the size of list OPEN using two criteria that we will describe next.

1. If a node m needs to be stored on list OPEN when the size of list OPEN has reached a given upper bound, then we discard the node with larger f value between node m and the node on list OPEN with the maximum value of function f.

2. If the probability that an optimal path goes through a node is smaller than a given parameter, then we do not store this node.

Memory requirement is usually a crucial factor in the practical implementation of any

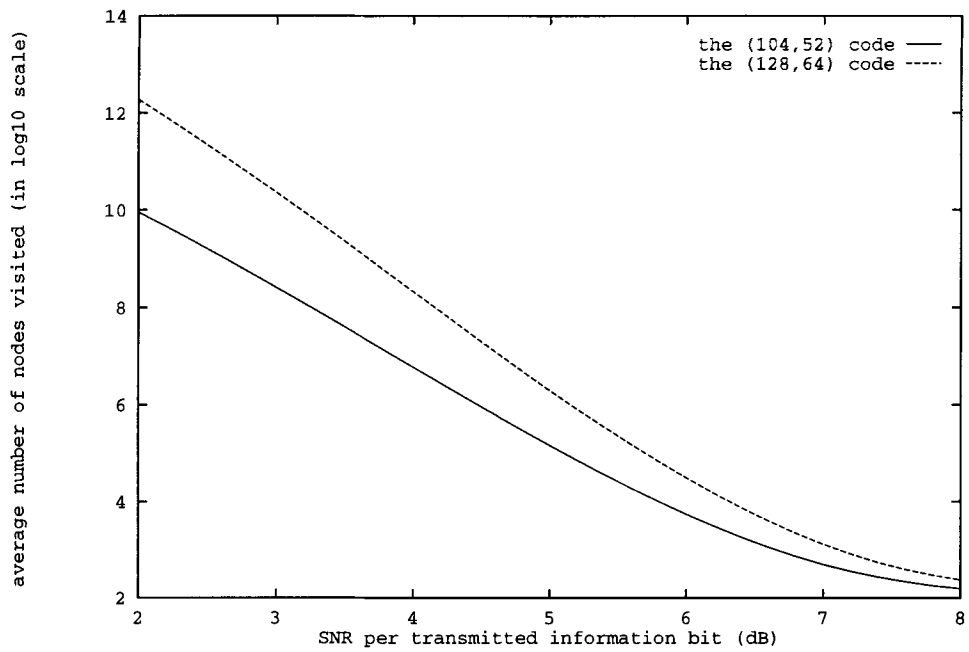

Figure 2: $\widetilde{N}$ for the $(104, 52)$ code and the $(128, 64)$ code

decoding algorithm. Thus, in the first criterion we limit the size of list OPEN by giving an upper bound on the maximum number of nodes that can be stored on list OPEN.

To use the second criterion we need to calculate the probability that an optimal path goes through a node. Now we will demonstrate how to calculate this probability for the AWGN channel. For any received vector $\boldsymbol{\phi}^*$, if an optimal decoding algorithm decodes it to a non-transmitted codeword, then it is very difficult for a suboptimal decoding algorithm to decode it to the transmitted codeword. Thus, when an optimal decoding algorithm decodes a received vector to a non-transmitted codeword we do not care which codeword a suboptimal decoding algorithm will decode to. Therefore, it is reasonable to consider only those received vectors that will be decoded to transmitted codewords by an optimal decoding algorithm. That is, when we derive the probability that an optimal path goes through a node, we will assume that no decoding error will occur if we employ an optimal decoding algorithm.

Let $h^*(m_s)$ note the cost of an optimal path. Under this assumption we have the following theorem.

Theorem 5 *Let an (n, k) code $\boldsymbol{C}$ be transmitted over the AWGN channel. When no decoding error occurs, the probability distribution of $h^*(m_s)$ is approximately a normal distribution with mean μ and variance σ^2, where*

$$\mu = n\frac{N_0}{2},$$

$$\sigma^2 = n\frac{N_0^2}{2}.$$

The proof of Theorem 5 is given in Appendix C.

Let node m be a node in the code tree of the transmitted (n, k) code $\boldsymbol{C}$ and let UB be the lowest upper bound on the cost of an optimal path found so far by the algorithm. In [7] it is shown that for any node m of the code tree, $h(m) \leq h^*(m)$ where $h^*(m)$ is the actual cost of the minimum cost path among all the paths from node m to goal nodes. Thus, if an optimal path goes through node m, then $h(m) \leq h^*(m) \leq h^*(m_s)$. Thus, the probability that an optimal path goes through node m is less than or equal to $\boldsymbol{Pr}(h(m) \leq h^*(m_s) \leq UB)$. This leads us to the following theorem.

Theorem 6 *Let T be the probability that an optimal path goes through node m. Furthermore, let UB be an upper bound on the cost of an optimal path. Then $T \leq T_{UB}$, where*

$$T_{UB} = \frac{1}{\sigma\sqrt{2\pi}} \int_{h(m)}^{UB} e^{-\frac{1}{2}(\frac{t-\mu}{\sigma})^2} dt,$$

where

$$\mu = n\frac{N_0}{2},$$
$$\sigma^2 = n\frac{N_0^2}{2}.$$

Thus, when a node is visited, the algorithm calculates T_{UB} for this node. If this value is less than a given threshold, then we will discard this node.

Now we describe the outline of our decoding algorithm. In our suboptimal decoding algorithm we will fix the maximum number of nodes, M_B, allowed on list OPEN. As in an optimal decoding algorithm, list OPEN is always kept ordered. When a node m is visited, the algorithm calculates T_{UB} for this node. If T_{UB} is less than a given threshold δ, then discard this node. Otherwise, we need to insert this node into list OPEN. If the number of nodes on list OPEN is equal to M_B, then the algorithm discards the node with larger f value between node m and the node with the largest f value on list OPEN. The algorithm inserts the remaining node into list OPEN.

We remark here that all the speed-up techniques in [7] can be applied to the suboptimal decoding algorithm.

5 Simulation results for the AWGN channel

In order to verify the performance of our suboptimal decoding algorithm, we present simulation results for the $(104, 52)$ binary extended quadratic residue code and the $(128, 64)$ binary extended BCH code when these codes are transmitted over the AWGN channel. We assume that antipodal signaling is used in the transmission so that the jth components of the transmitted codeword $\boldsymbol{c}$ and received vector $\boldsymbol{r}$ are

$$c_j = (-1)^{c_j}\sqrt{E} \quad \text{and} \quad r_j = (-1)^{c_j}\sqrt{E} + e_j,$$

threshold	1.5 dB	1.75 dB	2.0 dB	2.25 dB	2.5 dB	2.75 dB
0.0	26357	23909	18366	13240	10070	6698
0.25	10976	9643	6481	3980	2879	1579
0.5	3166	2827	1818	950	703	344

Table 1: The average number of nodes visited during the decoding of $(104, 52)$ code

respectively, where E is the signal energy per channel bit and e_j is a noise sample of a Gaussian process with single-sided noise power per hertz N_0. The variance of e_j is $N_0/2$ and the signal to noise ratio (SNR) for the channel is $\gamma = E/N_0$. In order to account for the redundancy in codes of different rates, we used the SNR per transmitted information bit $\gamma_b = E_b/N_0 = \gamma n/k$ in our simulation. For the AWGN channel, $\boldsymbol{\phi} = \frac{4\sqrt{E}}{N_0}\,\boldsymbol{r}$ [8], so we can substitute $\boldsymbol{r}(\boldsymbol{r}^*)$ for $\boldsymbol{\phi}(\boldsymbol{\phi}^*)$ in our decoding algorithm.

We do not know HW for these two codes, so we use a superset for them. For (104,52) we know that $d_{\min} = 20$ and that the Hamming weight of any codeword is divisible by 4 [9]. Thus for this code the superset used is $\{x|(x \text{ is divisible by } 4 \text{ and } 20 \leq x \leq 84) \text{ or } (x = 0) \text{ or } (x = 104)\}$; for (128,64), the superset used is $\{x|\ (x \text{ is even and } 22 \leq x \leq 106) \text{ or } (x = 0) \text{ or } (x = 128)\}$, since this code has $d_{\min} = 22$.

We have implemented a suboptimal version of the adaptive decoding algorithm presented in [7]. Since this suboptimal decoding algorithm is performed on low SNR, the initial $\boldsymbol{c}_0^*$ is constructed by considering the 16 codewords as follows. Let $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$ be the hard-decision of $\boldsymbol{r}^*$. Furthermore, let $S = \{\boldsymbol{u}|\boldsymbol{u} = (u_0, u_1, \ldots, u_{k-1}) \text{ and } u_i = y_i \text{ for } 0 \leq i \leq k-5\}$. For every element $\boldsymbol{u}$ in S, we get a codeword $\boldsymbol{c}^* = \boldsymbol{u} \cdot \boldsymbol{G}^*$. Now we let $\boldsymbol{c}_0^* = \boldsymbol{c}^*$, where the value of $h(m_s)$, calculated with respect to $\boldsymbol{c}^*$, is the largest among all the 16 codewords. The rule of updating seed is the same as that in [7].

The simulation results for the $(104, 52)$ code for γ_b equal to 1.5 dB, 1.75 dB, 2.0 dB, 2.25 dB, 2.5 dB, and 2.75 dB are given in Figure 3 and in Table 1 for three threshold values. M_B is equal to 3000.

In Figure 3 we also give a lower bound on the bit error probability of the maximum-likelihood decoding algorithm. This lower bound is obtained as follows [4]. For every sample, when suboptimal decoding algorithm terminates, we have a codeword that is obtained from the algorithm. If this codeword is closer with respect to Euclidean distance to the received vector than the transmitted codeword, then any optimal decoding algorithm will also decode the received vector to a non-transmitted codeword. Thus, we assume that the optimal decoding algorithm will decode to the codeword obtained from the suboptimal decoding algorithm and report a decoding error occurs. Bit error probability of the uncoded data is also given in Figure 3.

From Figure 3, for the $(104, 52)$ code the performance of the suboptimal decoding algorithm with $\delta = 0.0$ is within 0.25 dB of the performance of an optimal decoding algorithm; the performance of the suboptimal decoding algorithm with $\delta = 0.25$ is within 0.5 dB of the performance of an optimal decoding algorithm; and the performance of the suboptimal

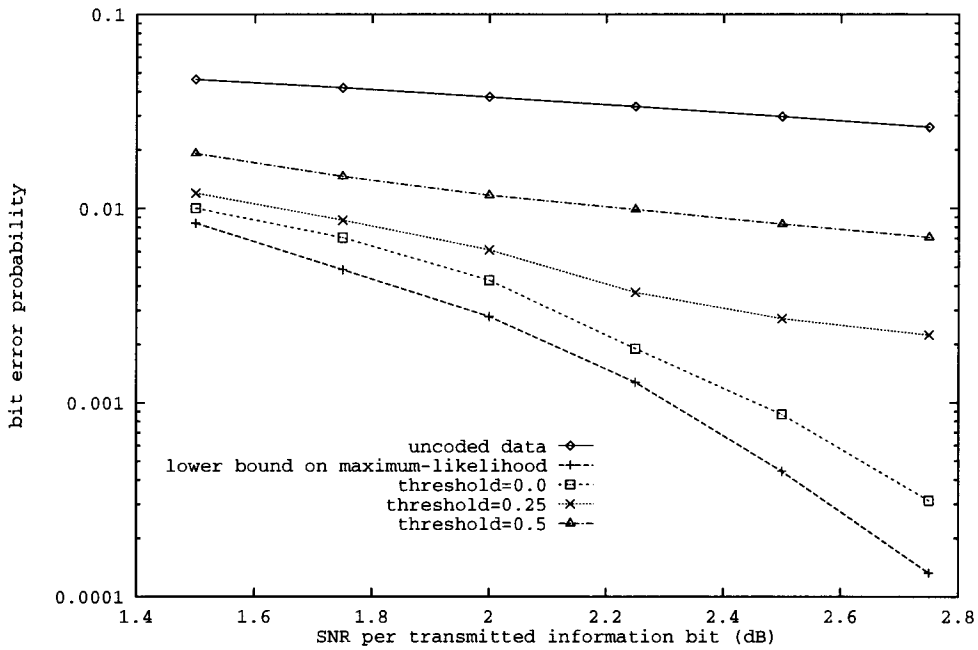

Figure 3: Performance of suboptimal decoding algorithm for the $(104, 52)$ code

decoding algorithm with $\delta = 0.5$ is within 1 dB of the performance of an optimal decoding algorithm. Thus, for the samples tried, limiting the size of list OPEN to 3000 nodes introduced only a small degradation on the performance of the algorithm for the $(104, 52)$ code. However, the average number of nodes visited for the sample tried is several orders of magnitude smaller than the upper bound given in Figure 2.

The simulation results for the $(128, 64)$ code for γ_b equal to 1.0 dB, 1.25 dB, 1.5 dB, 1.75 dB, and 2.0 dB are given in Figure 4 and in Table 2 for three threshold values. M_B is equal to 6000.

In Figure 4 we also give a lower bound on the bit error probability of the maximum-likelihood decoding algorithm and bit error probability of the uncoded data.

From Figure 4, for the $(128, 64)$ code the performance of the suboptimal decoding algo-

threshold	1.0 dB	1.25 dB	1.5 dB	1.75 dB	2.0 dB
0.0	88325	82650	75905	65223	55474
0.25	54416	41694	35613	29554	23162
0.5	22294	16705	13478	10389	6910

Table 2: The average number of nodes visited during the decoding of $(128, 64)$ code

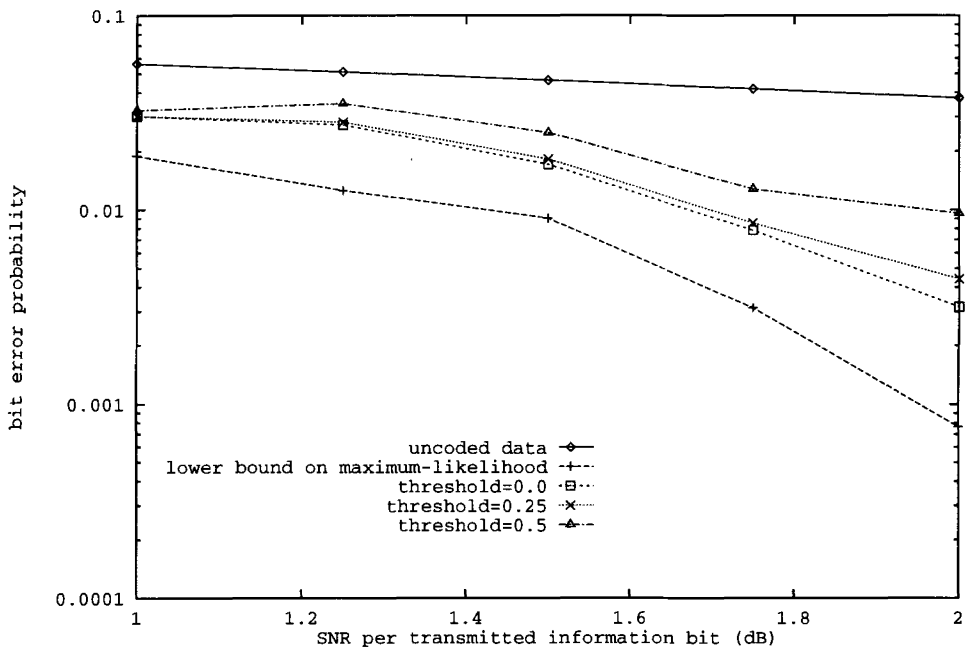

Figure 4: Performance of suboptimal decoding algorithm for the $(128, 64)$ code

rithm with $\delta = 0.0$ is within 0.5 dB of the performance of an optimal decoding algorithm; the performance of the suboptimal decoding algorithm with $\delta = 0.25$ is within 0.6 dB of the performance of an optimal decoding algorithm; and the performance of the suboptimal decoding algorithm with $\delta = 0.5$ is within 0.75 dB of the performance of an optimal decoding algorithm. Thus, for the samples tried, limiting the size of list OPEN to 6000 nodes introduced only a small degradation on the performance of the algorithm for the $(128, 64)$ code. However, the average number of nodes visited for the samples tried is several orders of magnitude smaller than the upper bound given in Figure 2.

6 Conclusions

In this report we determine an upper bound of the probability distribution of the number of nodes visited by the algorithm proposed in [7]. Even though this bound is not tight, Figure 2 shows that this decoding algorithm is efficient for most practical communication systems where the probability of error is less than 10^{-3}.

For low SNRs we propose a suboptimal decoding algorithm. From Figures 3 and 4, for the SNRs we tried, the performance of this algorithm is within 0.25 dB of the performance of an optimal decoding algorithm for the $(104, 52)$ binary extended quadratic residue code and within 0.5 dB for the $(128, 64)$ binary extended BCH code.

Appendix A

Proof of Theorem 1

Let an (n, k) code $\boldsymbol{C}$ be transmitted over an AWGN channel. In this case, from [8]

$$\phi_i = \ln \frac{\boldsymbol{Pr}(r_i|0)}{\boldsymbol{Pr}(r_i|1)} = \frac{4\sqrt{E}}{N_0} r_i.$$

Thus,

$$\boldsymbol{\phi} = \frac{4\sqrt{E}}{N_0} \boldsymbol{r},$$

and, for fixed SNR, $\frac{4\sqrt{E}}{N_0}$ can be treated as a positive constant. Since any positive constant multiplied to $\boldsymbol{\phi}$ will not affect the decoding procedure, we can substitute $\boldsymbol{r}$ for $\boldsymbol{\phi}$ in our decoding algorithm when $\boldsymbol{C}$ is transmitted over an AWGN channel [3]. Furthermore, without loss of generality we can assume that $\mathbf{0}$ is transmitted over an AWGN channel.

Let $\boldsymbol{P'_0}$ be the path from start node m_s to a goal node whose labels are all zero. Let us define the cost of the path $\boldsymbol{P'_0}$ as $g(\boldsymbol{P'_0})$. That is

$$g(\boldsymbol{P'_0}) = \sum_{i=0}^{n-1} (r_i - 1)^2.$$

From the definition of $f^*(m_s)$ which is the cost on an optimal path, we have

$$g(\boldsymbol{P'_0}) \geq f^*(m_s).$$

Now let node m be a node at level ℓ in the code tree and the labels of path $\boldsymbol{P'_m}$, the path from node m_s to node m, are $\overline{v}_0, \overline{v}_1, \ldots, \overline{v}_\ell$. Let $S' = \{i | \overline{v}_i = 1, 0 \leq i \leq \ell\}$ and $|S'| = \overline{d}$. From the definition of function f

$$\begin{aligned} f(m) &= g(m) + h_s(m) \\ &= \sum_{i=0}^{\ell} \left(r_i - (-1)^{\overline{v}_i}\right) + \sum_{i=\ell+1}^{n-1} \left(|r_i| - 1|\right)^2. \end{aligned}$$

Now we want to calculate the probability that node m is expanded by the algorithm. From Result 5 in Chapter 2 of [11], this probability will be less than or equal to the probability that $f(m) \leq f^*(m_s)$, i.e., $\boldsymbol{Pr}(f(m) \leq f^*(m_s))$. Since

$$g(\boldsymbol{P'_0}) \geq f^*(m_s),$$

then

$$\boldsymbol{Pr}(f(m) \leq f^*(m_s)) \leq \boldsymbol{Pr}(f(m) \leq g(\boldsymbol{P'_0})).$$

Furthermore,

$$
\begin{aligned}
f(m) \leq g(\boldsymbol{P}'_{\mathbf{0}}) \quad \text{iff} \quad & \sum_{i=0}^{\ell} \left(r_i - (-1)^{\bar{v}_i}\right)^2 + \sum_{i=\ell+1}^{n-1} \left(|r_i| - 1|\right)^2 \leq \sum_{i=0}^{n-1} (r_i - 1)^2 \\
\text{iff} \quad & \sum_{i \in S'} 4r_i + \sum_{i=\ell+1}^{n-1} 2(r_i - |r_i|) \leq 0 \\
\text{iff} \quad & \sum_{i \in S'} 2r_i + \sum_{i=\ell+1}^{n-1} (r_i - |r_i|) \leq 0.
\end{aligned}
$$

Now let us define two new random variables Z_i and Z'_i as

$$Z_i = 2r_i \text{ and } Z'_i = r_i - |r_i|.$$

Since $\mathbf{0}$ is transmitted,

$$\boldsymbol{Pr}(r_i) = \boldsymbol{Pr}(r_i|0).$$

Let $E(X)$ be the mean of random variable X and let $Var(X)$ be the variance of X. Thus, $E(r_i)$ is $\sqrt{E}$ and $Var(r_i)$ is $\frac{N_0}{2}$. Then

$$E(Z_i) = 2\sqrt{E}$$

and

$$
\begin{aligned}
Var(Z_i) &= 4Var(r_i) \\
&= 2N_0.
\end{aligned}
$$

Now let us calculate $E(Z'_i)$ and $Var(Z'_i)$. We first note that $Z'_i = 2r_i$ if $r_i < 0$ and $Z'_i = 0$ if $r_i \geq 0$, where r_i is normally distributed with mean $\sqrt{E}$ and variance $N_0/2$.

$$E(Z'_i) = \frac{1}{\sqrt{\pi N_0}} \int_{-\infty}^{0} 2te^{-\frac{(t-\sqrt{E})^2}{N_0}} dt.$$

Let $x = \frac{t-\sqrt{E}}{\sqrt{\frac{N_0}{2}}}$, then $dx = \frac{dt}{\sqrt{\frac{N_0}{2}}}$. Thus

$$
\begin{aligned}
E(Z'_i) &= \frac{\sqrt{\frac{N_0}{2}}}{\sqrt{\pi N_0}} \int_{-\infty}^{-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}} 2(\sqrt{\frac{N_0}{2}}x + \sqrt{E})e^{-\frac{x^2}{2}} dx \\
&= \frac{2\sqrt{\frac{N_0}{2}}}{\sqrt{2\pi}} \int_{-\infty}^{-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}} xe^{-\frac{x^2}{2}} dx + \frac{2\sqrt{E}}{\sqrt{2\pi}} \int_{-\infty}^{-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}} e^{-\frac{x^2}{2}} dx \\
&= \frac{2\sqrt{\frac{N_0}{2}}}{\sqrt{2\pi}} \int_{-\infty}^{-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}} xe^{-\frac{x^2}{2}} dx + 2\sqrt{E}G(-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}),
\end{aligned}
$$

where G is the standard normal distribution.

Let $y = \frac{x^2}{2}$, then $dy = xdx$. Thus,

$$E(Z_i') = 2\sqrt{E}G(-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}) - \sqrt{\frac{N_0}{\pi}}e^{-\frac{E}{N_0}}.$$

Similarly,

$$\begin{aligned}
Var(Z_i') &= E(Z_i^2) - E^2(Z_i) \\
&= \frac{1}{\sqrt{\pi N_0}}\int_{-\infty}^{0}(2t)^2 e^{-\frac{(t-\sqrt{E})^2}{N_0}}dt - E^2(Z_i) \\
&= 2(2E+N_0)G(-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}) - 2\sqrt{\frac{EN_0}{\pi}}e^{-\frac{E}{N_0}} - E^2(Z_i) \\
&= 2(2E+N_0)G(-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}) - 2\sqrt{\frac{EN_0}{\pi}}e^{-\frac{E}{N_0}} \\
&\quad - \left(2\sqrt{E}G(-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}) - \sqrt{\frac{N_0}{\pi}}e^{-\frac{E}{N_0}}\right)^2.
\end{aligned}$$

Now let us define a new random variable X as

$$X = \sum_{i\in S'} Z_i + \sum_{i=\ell+1}^{n-1} Z_i'.$$

By Lindeberg's central limit theorem [5], the probability distribution of X is approximately a normal distribution with mean $\overline{\mu}(\ell,\overline{d})$ and variance $\overline{\sigma}^2(\ell,\overline{d})$, where

$$\begin{aligned}
\overline{\mu}(\ell,\overline{d}) &= \overline{d}E(Z_i) + (n-\ell-1)E(Z_i') \\
&= 2\overline{d}\sqrt{E} + (n-\ell-1)\left\{2\sqrt{E}G(-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}) - \sqrt{\frac{N_0}{\pi}}e^{-\frac{E}{N_0}}\right\}, \\
\overline{\sigma}^2(\ell,\overline{d}) &= \overline{d}Var(Z_i) + (n-\ell-1)Var(Z_i') \\
&= 2\overline{d}N_0 + (n-\ell-1)\left\{2(2E+N_0)G(-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}) - 2\sqrt{\frac{EN_0}{\pi}}e^{-\frac{E}{N_0}}\right. \\
&\quad \left. - \left(2\sqrt{E}G(-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}) - \sqrt{\frac{N_0}{\pi}}e^{-\frac{E}{N_0}}\right)^2\right\}.
\end{aligned}$$

Thus,

$$\boldsymbol{Pr}(f(m) \le f^*(m_s)) \le \boldsymbol{Pr}(X \le 0) = G(-\frac{\overline{\mu}(\ell,\overline{d})}{\overline{\sigma}(\ell,\overline{d})}).$$

Since $f(m)_0 \le g(\boldsymbol{P_0'})$ for any node m_0 on path $\boldsymbol{P_0'}$, we can assume that node m_0 will be expanded. There are k nodes on this path that will be expanded. We now consider those

nodes that are not on this path. It is easy to see that, for any node that is not on path $\boldsymbol{P}'_0$, the labels of the path from node m_s to it will contain at least one 1. Consider those nodes at level ℓ whose paths contain $\overline{d}$ ones, where $1 \leq \overline{d} \leq \ell + 1$ and $0 \leq \ell \leq k-2$. From the above argument, the probability of these nodes being expanded are $G\left(-\frac{\overline{\mu}(\ell,\overline{d})}{\overline{\sigma}(\ell,\overline{d})}\right)$. The total number of these nodes is $\binom{\ell+1}{\overline{d}}$. Since the first k positions of any codeword are information bits, the average number of nodes expanded by the algorithm is less than or equal to

$$\left[k + \sum_{\ell=0}^{k-2} \sum_{\overline{d}=1}^{\ell+1} \binom{\ell+1}{\overline{d}} G\left(-\frac{\overline{\mu}(\ell,\overline{d})}{\overline{\sigma}(\ell,\overline{d})}\right)\right].$$

Since when a node is expanded by the algorithm, the algorithm will visit two nodes, the average number of nodes visited is less than or equal to

$$2\left[k + \sum_{\ell=0}^{k-2} \sum_{\overline{d}=1}^{\ell+1} \binom{\ell+1}{\overline{d}} G\left(-\frac{\overline{\mu}(\ell,\overline{d})}{\overline{\sigma}(\ell,\overline{d})}\right)\right].$$

□

Appendix B

Proof of Theorem 2

Let $\boldsymbol{\phi} = (\phi_0, \phi_1, \ldots, \phi_{n-1})$ be the received vector and let $\boldsymbol{\phi}^* = (\phi_0^*, , \phi_1^*, \ldots, \phi_{n-1}^*)$ be obtained by permuting the positions of $\boldsymbol{\phi}$ such that the first k positions are the "most reliable linearly independent" positions in $\boldsymbol{\phi}$. Furthermore, let $\phi_0^* = \phi_{\pi(0)}, \phi_1^* = \phi_{\pi(1)}, \ldots,$ and $\phi_{n-1}^* = \phi_{\pi(n-1)}$. We now prove that $N_s(\boldsymbol{\phi}^*) \leq N_s(\boldsymbol{\phi})$ by proving that, for every node m_1 in the search tree generated by the decoding algorithm when it decodes $\boldsymbol{\phi}$, we can find a one-to-one correspondent node m_2 in the search tree generated by the decoding algorithm when it decodes $\boldsymbol{\phi}^*$ such that $f(m_1) \leq f(m_2)$. Let the labels of the path from the start node to node m_1 that is at level ℓ be $c_0, c_1, \ldots,$ and c_ℓ. Let us define $S_s(\ell), S_a(\ell), S_b(\ell), S_c(\ell), S_d(\ell)$, which are subsets of $\{0, 1, 2, \ldots, n-1\}$ as follows:

$$\begin{aligned}
S_s(\ell) &= \{x | x \leq \ell \text{ and } \pi(x) \leq \ell\}, \\
S_a(\ell) &= \{x | x \leq \ell\} - \{\pi(x) | x \in S_s(\ell)\}, \\
S_b(\ell) &= \{x | x \leq \ell\} - \{x | x \in S_s(\ell)\}, \\
S_c(\ell) &= \{\pi(x) | x \in S_b(\ell)\}, \text{ and} \\
S_d(\ell) &= \{x | \pi(x) \in S_a(\ell)\}.
\end{aligned}$$

It is clear that $|S_a(\ell)| = |S_b(\ell)| = |S_c(\ell)| = |S_d(\ell)|$. Now let us define the labels $c_0^*, c_1^*, \ldots, c_\ell^*$ from the start node to node m_2 as follows:

$$\begin{aligned}
c_x^* &= c_{\pi(x)} \text{ for } x \in S_s(\ell), \\
c_x^* &= y_x^* \oplus y_{q(\boldsymbol{\phi},\ell)(x)} \oplus c_{q(\boldsymbol{\phi},\ell)(x)} \text{ for } x \in S_b(\ell),
\end{aligned}$$

where

$$\begin{aligned} y_i &= 0 \text{ when } \phi_i \geq 0, \\ &= 1 \text{ otherwise,} \\ y_i^* &= 0 \text{ when } \phi_i^* \geq 0, \\ &= 1 \text{ otherwise, and} \end{aligned}$$

$q(\boldsymbol{\phi}, \ell)$ is a bijection from $S_b(\ell)$ to $S_a(\ell)$.

It is easy to see that for any node m_1, node m_2 is a one-to-one correspondent to node m_1.

We next prove that $f(m_1) \leq f(m_2)$.

$$f(m_2) - f(m_1) = \sum_{i=0}^{\ell} \left(\phi_i^* - (-1)^{c_i^*}\right)^2 + \sum_{i=\ell+1}^{n-1} (|\phi_i^*| - 1)^2$$

$$- \sum_{i=0}^{\ell} (\phi_i - (-1)^{c_i})^2 - \sum_{i=\ell+1}^{n-1} (|\phi_i| - 1)^2 = \sum_{i \in S_b(\ell)} \left(\phi_i^* - (-1)^{c_i^*}\right)^2 + \sum_{i \in S_d(\ell)} (|\phi_i^*| - 1)^2$$

$$- \sum_{i \in S_a(\ell)} (\phi_i - (-1)^{c_i})^2 - \sum_{i \in S_c(\ell)} (|\phi_i| - 1)^2$$

$$= \left[\sum_{i \in S_b(\ell)} \left(\phi_i^* - (-1)^{c_i^*}\right)^2 - \sum_{i \in S_c(\ell)} (|\phi_i| - 1)^2\right] - \left[\sum_{i \in S_a(\ell)} (\phi_i - (-1)^{c_i})^2 - \sum_{i \in S_d(\ell)} (|\phi_i^*| - 1)^2\right]$$

$$= \sum_{i \in S_f(\ell)} \left[(|\phi_i^*| + 1)^2 - (|\phi_i^*| - 1)^2\right]$$

$$- \sum_{i \in S_e(\ell)} \left[(|\phi_i| + 1)^2 - (|\phi_i| - 1)^2\right] = 4 \left[\sum_{i \in S_f(\ell)} |\phi_i^*| - \sum_{i \in S_e(\ell)} |\phi_i|\right],$$

where $S_f(\ell) = \{x | x \in S_b(\ell) \text{ and } c_x^* \oplus y_x^* = 1\}$ and $S_e(\ell) = \{x | x \in S_a(\ell) \text{ and } c_x \oplus y_x = 1\}$.

Since $c_x^* = y_x^* \oplus y_{q(\boldsymbol{\phi},\ell)(x)} \oplus c_{q(\boldsymbol{\phi},\ell)(x)}$ for $x \in S_b(\ell)$ and $q(\boldsymbol{\phi}, \ell)$ is a bijection from $S_b(\ell)$ to $S_a(\ell)$, then $|S_f(\ell)| = |S_e(\ell)|$. Furthermore, since the k-most reliable positions in $\boldsymbol{\phi}$ are linear independent, then $|\phi_i^*| \geq |\phi_j|$ for any $i \in S_f(\ell)$ and $j \in S_e(\ell)$. Thus $f(m_2) \geq f(m_1)$. By Theorem 7 in Chapter 3 of [12], we have $N_s(\boldsymbol{\phi}^*) \leq N_s(\boldsymbol{\phi})$. □

Appendix C

Proof of Theorem 5

Let an (n, k) code $\boldsymbol{C}$ be transmitted over an AWGN channel. If we assume that no decoding error occurs,

$$\begin{aligned} f^*(m_s) &= \sum_{i=0}^{n-1} \left((-1)^{c_i}\sqrt{E} - (e_i + (-1)^{c_i}\sqrt{E})\right)^2 \\ &= \sum_{i=0}^{n-1} e_i^2, \end{aligned}$$

where for each $0 \leq i \leq n-1, e_i$'s are *i.i.d.* and e_i is a normal random variable with mean 0 and variance $N_0/2$. Consequently, $\frac{2}{N_0}\sum_{i=0}^{n-1} e_i^2 = \dfrac{2}{N_0} f^*(m_s)$ will be distributed as a chi-square random variable with n degrees of freedom. From [14] it follows that

$$E(f^*(m_s)) = n\frac{N_0}{2},$$

and $\mathrm{Var}(f_m^*(m_s)) = n\frac{N_0^2}{2}$.

However, for large value of n, the probability distribution of $f^*(m_s)$ is approximately a normal distribution with mean μ and variance σ^2, given above. □

Acknowledgment

The authors would like to thank Elaine Weinman for her invaluable help in the preparation of this manuscript.

References

[1] R.W. D. Booth, M. A. Herro, and G. Solomon, "Convolutional Coding Techniques for Certain Quadratic Residue Codes," *Proc. 1975 Int. Telemetering Conf.*, pp. 168–177, 1975.

[2] G. Brassard and P. Bratley, *Algorithmics Theory and Practice.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1988.

[3] G. C. Clark, Jr. and J. B. Cain, *Error-Correction Coding for Digital Communications.* New York, NY: Plenum Press, 1981.

[4] B. G. Dorsch, "A Decoding Algorithm for Binary Block Codes and J-ary Output Channels," *IEEE Transactions on Information Theory*, pp. 391–394, May 1974.

[5] W. Feller, *An Introduction to Probability Theory and its Applications.* New York, NY: John Wiley and Sons, 1966.

[6] Y. S. Han and C. R. P. Hartmann, "Designing Efficient Maximum-Likelihood Soft-Decision Decoding Algorithms for Linear Block Codes Using Algorithm A^*," Technical Report SU-CIS-92-10, School of Computer and Information Science, Syracuse University, Syracuse, NY 13244, June 1992.

[7] Y. S. Han, C. R. P. Hartmann, and C.-C. Chen, "Efficient Priority-First Search Maximum-Likelihood Soft-Decision Decoding of Linear Block Codes," *IEEE Transactions on Information Theory*, pp. 1514–1523, September 1993.

[8] T.-Y. Hwang, "Decoding Linear Block Codes for Minimizing Word Error Rate," *IEEE Transactions on Information Theory*, pp. 733–737, November 1979.

[9] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes.* New York, NY: Elsevier Science Publishing Company, Inc., 1977.

[10] K. R. Matis and J. W. Modestino, "Reduced-Search Soft-Decision Trellis Decoding of Linear Block Codes," *IEEE Transactions on Information Theory*, pp. 349–355, March 1982.

[11] N. J. Nilsson, *Principle of Artificial Intelligence.* Palo Alto, CA: Tioga Publishing Co., 1980.

[12] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Reading, MA: Addison-Wesley Publishing Company, 1984.

[13] G. Solomon and H. C. A. van Tilborg, "A Connection Between Block and Convolutional Codes," *SIAM J. Appl. Math.*, pp. 358–369, 1979.

[14] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications.* Englewood Cliffs, NJ: Prentice-Hall Inc., 1982.

[15] A. J. Viterbi, "Error Bound for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Transactions on Information Theory*, pp. 260–269, April 1967.

[16] J. K. Wolf, "Efficient Maximum Likelihood Decoding of Linear Block Codes Using a Trellis," *IEEE Transactions on Information Theory*, pp. 76–80, January 1978.

Figure Captions

Figure 1. Average number of nodes visited for the $(48, 24)$ code

Figure 2. $\widetilde{N}$ for the $(104, 52)$ code and the $(128, 64)$ code

Figure 3. Performance of suboptimal decoding algorithm for the $(104, 52)$ code

Figure 4. Performance of suboptimal decoding algorithm for the $(128, 64)$ code