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

Yunghsiang S. Han, *Member, IEEE*,
Carlos R. P. Hartmann, *Fellow, IEEE*,
and Kishan G. Mehrotra

***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 correspondence 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 computational effort of the algorithm. A suboptimal decoding algorithm is also proposed. The performance of this suboptimal decoding 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 of the optimal performance for the $(128, 64)$ binary BCH code, respectively.**

***Index Terms*—Block codes, decoding, Dijkstra's algorithm, maximum-likelihood, soft-decision, suboptimal.**

I. 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 [2], [5], [20], [25] have presented techniques for decoding linear block codes that convert the decoding problem into a graph-search problem on a trellis derived from a parity-check matrix of the code. Thus the *maximum-likelihood decoding* (MLD) rule can be implemented by applying the Viterbi algorithm [24] 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 [16]. Some coset decoding schemes have been proposed [10], [18], [19], [23]; however, they depend on the selection of a specific subcode. An efficient algorithm has also been proposed for long high-rate codes, and short- and moderate-length codes [4].

We recently proposed a novel maximum-likelihood soft-decision decoding algorithm for linear block codes [11]–[13]. This algorithm uses a generalization of Dijkstra's algorithm (GDA) [17] to search through the trellis for a code equivalent to the transmitted code. The use of this priority-first search strategy drastically reduces the decoding search space and results in an efficient optimal soft-decision decoding algorithm for linear block codes. Furthermore, in contrast with Wolf's algorithm [25], the decoding effort of our algorithm is adaptable to the noise level.

In Section II we review the MLD of linear block codes, describe the code tree for a linear code, and briefly state the decoding algorithm proposed in [13]. In Section III we give an upper bound on the computational effort of this algorithm, and in Section IV we present a suboptimal decoding algorithm. Simulation results for the $(48, 24)$, the $(104, 52)$ binary extended quadratic residue codes, and the $(128, 64)$ binary extended Bose–Chaudhuri–Hocquengham (BCH) code are given in Section V, and concluding remarks in Section VI.

Manuscript received April 11, 1994; revised September 24, 1997. This work was supported in part by the National Science Foundation under Grant NCR-9205422 and used the computational facilities of the Northeast Parallel Architectures Center (NPAC) at Syracuse University. The work of Y. S. Han was supported in part by the National Science Council R.O.C. under Grant NSC 84-2213-E-211-004. The material in this correspondence was presented in part at the IEEE International Symposium on Information Theory, Trondheim, Norway, June 1994.

Y. S. Han is with the Department of Computer Science and Information Engineering, National Chi Nan University, Taiwan, R.O.C.

C. R. P. Hartmann and K. G. Mehrotra are with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244-4100 USA.

Publisher Item Identifier S 0018-9448(98)02361-X.

II. Preliminaries

Let $\boldsymbol{C}$ be a binary (n, k) linear block code with generator matrix $\boldsymbol{G}$, and let $\boldsymbol{c} = (c_0, c_1, \cdots, c_{n-1})$ be a codeword of $\boldsymbol{C}$ transmitted over a time-discrete memoryless channel with output alphabet $\boldsymbol{B}$. Furthermore, let $\boldsymbol{r} = (r_0, r_1, \cdots, r_{n-1})$, $r_j \in \boldsymbol{B}$ denote the received vector, and assume that $\Pr(r_j|c_i) > 0$ for all $r_j \in \boldsymbol{B}$ and $c_i \in \mathrm{GF}(2)$. Let $\widehat{\boldsymbol{c}}$ 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 [3], [22], [23]

$$\text{set } \widehat{\boldsymbol{c}} = c_\ell, \qquad \text{where } \boldsymbol{c}_\ell \in \boldsymbol{C}$$

and

$$\sum_{j=0}^{n-1} (\phi_j - (-1)^{c_{\ell j}})^2 \le \sum_{j=0}^{n-1} (\phi_j - (-1)^{c_{ij}})^2 \tag{1}$$

for all $\boldsymbol{c_i} \in \boldsymbol{C}$ where

$$\phi_j = \ln \frac{\Pr(r_j|0)}{\Pr(r_j|1)}. \tag{2}$$

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

Our decoding algorithm (presented in [13]) uses the priority-first search strategy, thus avoiding traversing the entire trellis. Guided by an evaluation function f, it 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 $\boldsymbol{\phi}$. Let $\boldsymbol{G}^*$ be a generator matrix of $\boldsymbol{C}^*$ whose first k columns form a $k \times k$ identity matrix. In this decoding algorithm, the vector $\boldsymbol{\phi}^* = (\phi_0^*, \phi_1^*, \cdots, \phi_{n-1}^*)$ 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 $\boldsymbol{G}$ are permuted to obtain $\boldsymbol{G}^*$.

Since the probability is very small that our decoding algorithm will revisit a node of the trellis, our implementation did not check for repeated nodes [13]. In this case, the graph where the search is performed is 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 . In the code tree, every path is totally distinct from every other path. The leftmost node is called the *start node*, which is at level 0. There are two branches, labeled 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.

To determine the sequence of labels encountered when traversing a path from a node at level k to a goal node, let $\boldsymbol{G}^*$ be a generating

matrix of $\boldsymbol{C}^*$ whose first k columns form the $k \times k$ identity matrix. Furthermore, let $c_0, c_1, \cdots, c_{k-1}$ be the sequence of labels encountered when traversing a path from the start node to a node m at level k. Then $c_k, c_{k+1}, \cdots, c_{n-1}$, the sequence of labels encountered when traversing a path from node m to a goal node, can be obtained as follows:

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

The cost of the branch from a node at level t to a node at level $t+1$ is assigned the value $(\phi_t^* - (-1)^{c_t^*})^2$, where c_t^* is the label of the branch. Thus the solution of the decoding problem is converted into finding a lowest cost path from the start node to a goal node. Such a path will be called an optimal path.

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 branch costs encountered while constructing this path. The GDA requires that for all nodes m_i and m_j such that node m_j is an immediate successor of node m_i

$$h(m_i) \leq h(m_j) + c(m_i, m_j) \tag{3}$$

where $c(m_i, m_j)$ is the branch cost between node m_i and node m_j. This requirement guarantees that the GDA will find an optimal path. In the GDA, the next node to be expanded is the 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. Every time the GDA expands a node, it calculates values f of two immediate successors of this node and then inserts these two successors into list OPEN. In this process the GDA visits are these two successors. All the nodes on list OPEN also keep the labels of the paths from the start node to them, which can be used to calculate function f. When the algorithm chooses to expand a goal node, it is time to stop because the algorithm has constructed a path with minimum cost.

For practical applications there may exist many functions h that satisfy inequality (3). Following are some results presented in [13] that can be used to design a suitable function h to reduce the number of nodes visited.

Theorem 1: Let $h^*(m)$ be the minimum cost among all the paths from node m to goal nodes and

$$f^*(m) = g(m) + h^*(m).$$

For every node m, if $h(m)$ satisfies inequality (3), then

$$h(m) \leq h^*(m).$$

Theorem 2: Let two functions

$$f_1(m) = g(m) + h_1(m)$$

and

$$f_2(m) = g(m) + h_2(m)$$

satisfy

$$h_1(m) \leq h_2(m) \leq h^*(m)$$

for every non-goal node m. Furthermore, there exists a unique optimal path. Then the GDA, using evaluation function f_2, will never expend more nodes than the algorithm using evaluation function f_1.

Theorem 3: Assume that there exists a unique optimal path and that $f^*(m_{\text{start}})(= h^*(m_{\text{start}}))$ is the cost of the optimal path, where m_{start} is the start node. Then, for any node m selected for expansion (open) by the GDA

$$f(m) \leq f^*(m_{\text{start}}).$$

From the above theorems, we intend to design a function h such that the value $h(m)$ for any non-goal node m is as large as possible; however, the computational effort of $h(m)$ is usually higher when $h(m)$ is larger. The best function h we may have is h^*. Usually, the computation of $h^*(m)$ involves the search of a path from node m to a goal node with minimum cost, and such a search is intractable. Thus there is a tradeoff between the number of nodes visited and the computation complexity of function h. Normally, to define a good function h we need to have some knowledge of the structure of the graph where the search is performed.

We use some properties of linear block codes to define our function h [13] that satisfies inequality (3). For every received vector $\boldsymbol{\phi}$, since we order the components in $\boldsymbol{\phi}$ according to their reliability, the properties that we use to define function h must be invariant under any permutation of the positions of the codewords with which we obtain $\boldsymbol{C}^*$ from $\boldsymbol{C}$; otherwise, we need to define different function h for every received vector, which is impractical. Since the Hamming distance between any two codewords is invariant under the permutations with which we obtain $\boldsymbol{C}^*$ from $\boldsymbol{C}$, our heuristic function is designed to take into consideration the fact that the Hamming distance between any two codewords of $\boldsymbol{C}^*$ must belong to HW, where $HW = \{w_i | 0 \leq i \leq I\}$ is the set of all distinct Hamming weights that codewords of $\boldsymbol{C}$ may have. Furthermore, assume $w_0 < w_1 < \cdots < w_I$. Let $\boldsymbol{c}^*$ be a given codeword of $\boldsymbol{C}^*$. Our function h is defined with respect to $\boldsymbol{c}^*$, which is called the seed of the decoding algorithm.

1) For nodes at level ℓ, with $0 \leq \ell \leq k-1$:

   Let m be a node at level ℓ, and let $\overline{v}_0, \overline{v}_1, \cdots, \overline{v}_{\ell-1}$ be the labels of the path $\boldsymbol{P}'_m$ from the start node to node m. Let the set $T(m)$ contain all binary n-tuples $\boldsymbol{v}$ such that their first ℓ 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,

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

   Note that $T(m) \neq \emptyset$. This can easily be seen by considering the binary k-tuple $\boldsymbol{u} = (\overline{v}_0, \overline{v}_1, \cdots, \overline{v}_{\ell-1}, 0, \cdots, 0)$ and noting that $\boldsymbol{u} \cdot \boldsymbol{G}^* \in T(m)$.

   We now define function h as

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

2) For nodes at level ℓ, with $k \leq \ell \leq n$:

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

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

   where $v_\ell^*, v_{\ell+1}^*, \cdots, v_{n-1}^*$ 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 $h(m) = 0$. Since we can construct the only path from any node m at level ℓ, with $\ell \geq k$, to the goal node using $\boldsymbol{G}^*$, the estimate $h(m)$ computed here is always exact. Furthermore, $h(m) = h^*(m)$ since there is only one path from node m to a goal node and $h(m)$ is the cost of this path.

An algorithm to calculate $h(m)$ for node m at level ℓ, with $0 \le \ell \le k-1$, whose time complexity is $O(n)$, is presented in [12] and [13]. In [13] it is shown that our decoding algorithm is a depth-first search type that will search the code tree only up to level k. The labels of the combined paths from the start node to node m at level k, and from node m to a goal node, correspond to a codeword. Therefore, the cost of this path $f(m)$ can be used as an upper bound (UB) on the cost of an optimal path. Therefore, we can use this UB to reduce the size of list OPEN. 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. More details about this decoding algorithm can be found in [12] and [13], where we also described other speedup techniques such as the stopping criterion and changing the seed during the decoding procedure. It is shown by the simulation results in [12] and [13] that speedup techniques reduce the number of nodes visited by our decoding algorithm. Therefore, it is worthwhile to briefly describe the speedup techniques here.

The search procedure can be stopped at any time when we know that a generated codeword $\boldsymbol{c}_{\boldsymbol{\ell}}^{*} = (c_{\ell 0}^{*}, c_{\ell 1}^{*}, \cdots, c_{\ell(n-1)}^{*})$ satisfies inequality (1). The following criterion can be used to indicate this fact.

Criterion: Let m_{start} be the start node.

$$\text{If } h(m_{\text{start}}) = \sum_{j=0}^{n-1} (\phi_j^{*} - (-1)^{c_{\ell j}^{*}})^2$$

where $h(m_{\text{start}})$ is calculated with respect to seed $\boldsymbol{c}_{\boldsymbol{\ell}}^{*}$, then $\boldsymbol{c}_{\boldsymbol{\ell}}^{*}$ satisfies inequality (1).

Hence, during the decoding procedure, if the algorithm generates a new codeword and the cost of the path whose labels correspond to this codeword is the lowest found so far by the algorithm, then we may check whether the new codeword satisfies the criterion or not. If so, this codeword satisfies inequality (1) and the decoding procedure stops.

Another speedup technique is not fixing seed $\boldsymbol{c}^{*}$ during the decoding of $\boldsymbol{\phi}^{*}$. When seed $\boldsymbol{c}^{*}$ is allowed to change, we have an adaptive decoding procedure. In order to avoid increasing computational complexity when the seed is changed at some stage of the decoding procedure, we may not want to recalculate the values of function h with respect to this new seed for every node on list OPEN. Under these circumstances, nodes on list OPEN may calculate the values of function h with respect to different seeds.

Since all the theorems and simulation results are obtained when we assume that code $\boldsymbol{C}$ is transmitted over a memoryless, additive white Gaussian noise (AWGN) channel, we describe the channel here. 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

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

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 $\gamma_b = E_b/N_0 = \gamma n/k = \gamma/R$, where $R = k/n$ is the code rate.

III. Analysis of the Computational Effort of the Algorithm

Our decoding algorithm can be considered a branch-and-bound algorithm. In general, it is difficult to know how well a branch-and-bound algorithm will perform on a given problem [6]; 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}.

One important measure of the computational effort of an algorithm is its time complexity [1]. If the complexity is taken as the "average" complexity over all inputs of fixed size, then the complexity is called the expected (average) complexity. In a decoding problem, the inputs are received vectors. The time complexity of our decoding algorithm is the multiplication of the number of nodes visited and the time complexity to calculate the function $f(m)$ [11]. Since the time complexity to calculate the function $f(m)$ in our decoding algorithm is $O(n)$, the average complexity of our decoding algorithm is determined by the average numbers of nodes visited [12], [13].

In order to derive an upper bound on the average number of nodes visited by our decoding algorithm, we will define another heuristic function, h_s, that satisfies the condition

$$h_s(m) \le h_p(m) \text{ for every node of the code tree}$$

where h_p is the function defined in the preceding section. Thus by Theorem 2, the decoding algorithm using the function h_p will never open more nodes than the decoding algorithm using the function h_s.

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

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

and

$$f_s(m) = g(m) + h_s(m)$$

where

$$g(m) = \sum_{i=0}^{\ell-1} \left(\phi_i - (-1)^{\overline{v}_i}\right)^2.$$

For a node at a level greater than $k-1$, the function h_s will be defined as in the previous section. It is easy to see that $h_s(m) \le h_p(m)$ for every node of the code tree. By the above definition, when the decoding algorithm is calculating $f_s(m)$ and $h_s(m)$, ϕ_i, with $0 \le i \le n-1$, and $\overline{v}_i$, with $0 \le i \le \ell - 1$, must be known. Before the decoding procedure starts, ϕ_i can be obtained by (2) and the received vector. Hence, ϕ_i can be stored and used when the algorithm calculates $f_s(m)$ and $h_s(m)$ for any node m. When the decoding algorithm visits node m, the path from the start node to node m in the code tree is known, since all nodes on list OPEN keep the labels of the paths from the start node to them. Therefore, the labels on this path, $\overline{v}_i$, are known. Since the calculation of $f_s(m)$ and $h_s(m)$ involves only n terms, at most, of known values for which the algorithm does not need to calculate or search, the time complexities of calculating $h_s(m)$ and $f_s(m)$ are $O(n)$. However, for any node m at level ℓ, with $\ell \le k-1$, if the value of function f_s of its immediate predecessor m' is known, we may obtain $f_s(m)$ from $f_s(m')$ by the simple computation given below.

Assume that node m is not the start node m_{start}, and that node m is at level ℓ, with $\ell \le k-1$. Let node m' be the immediate predecessor of node m. Furthermore, let $\boldsymbol{y} = (y_0, y_1, \cdots, y_{n-1})$ be the hard decision of $\boldsymbol{\phi}$. That is,

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

Let $\overline{v}_{\ell-1}$ be the label of the branch between node m' and node m and let $\oplus$ denote a modula 2 addition. Since

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

then

$$\begin{aligned} f_s(m) &= g(m) + h_s(m) \\ &= g(m') + (\phi_{\ell-1} - (-1)^{\overline{v}_{\ell-1}})^2 + h_s(m') - (|\phi_{\ell-1}| - 1)^2. \end{aligned}$$

Consequently,

$$f_s(m) = f_s(m') + (y_{\ell-1} \oplus \overline{v}_{\ell-1})(4 \cdot |\phi_{\ell-1}|). \tag{4}$$

Thus when $f_s(m')$ is calculated, $f_s(m)$ can be obtained by (4). Since the start node m_{start} has no predecessor, we cannot use (4) to obtain the value $f_s(m_{\text{start}})$.

In order to obtain an upper bound of the computational effort of our decoding algorithm, we first derive an upper bound of the computational effort 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.

For a given received vector $\boldsymbol{\phi}$, by Theorem 3, if node m is selected for expansion, then $f(m) \leq f^*(m_{\text{start}})$, where $f^*(m_{\text{start}})$ is the cost of the optimal path. Since we do not order the positions of $\boldsymbol{\phi}$, the components of $\boldsymbol{\phi}$ are independent random variables. Furthermore, since the cost of the path whose labels correspond to the transmitted codeword is greater than or equal to $f^*(m_{\text{start}})$, by the central limit theorem we can calculate an upper bound on the probability of a node being expanded. Consequently, we may calculate an upper bound on the average number of nodes visited by the SDA.

We now state the main results of the computational effort of the SDA when code $\boldsymbol{C}$ is transmitted over the AWGN channel given in Section II. Since the result is derived by using the central limit theorem, it only holds when n is large.

Theorem 4: Let $\overline{N}_s$ be the average number of nodes visited by the SDA and let $Q(\cdot)$ be the standard normal distribution. Then, for a large n

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

where

$$\widetilde{N} = 2\left[k + \sum_{\ell=0}^{k-1}\sum_{\overline{d}=1}^{\ell}\binom{\ell}{\overline{d}} Q\left(-\frac{\overline{\mu}(\ell,\overline{d})}{\overline{\sigma}(\ell,\overline{d})}\right)\right] \tag{5}$$

$$\begin{aligned} \overline{\mu}(\ell,\overline{d}) = \sqrt{N_0}\Bigg\{ & 2\overline{d}\sqrt{R\gamma_b} + (n-\ell) \\ & \cdot \left[2\sqrt{R\gamma_b}Q(-\sqrt{2R\gamma_b}) - \frac{1}{\sqrt{\pi}}e^{-R\gamma_b}\right]\Bigg\} \end{aligned}$$

and

$$\begin{aligned} \overline{\sigma}^2(\ell,\overline{d}) = N_0\Bigg\{ & 2\overline{d} + (n-\ell)\Bigg[(4R\gamma_b+2)Q(-\sqrt{2R\gamma_b}) \\ & - 2\sqrt{\frac{R\gamma_b}{\pi}}e^{-R\gamma_b} \\ & - \left(2\sqrt{R\gamma_b}Q(-\sqrt{2R\gamma_b}) - \frac{1}{\sqrt{\pi}}e^{-R\gamma_b}\right)^2\Bigg]\Bigg\}. \end{aligned}$$

The proof of Theorem 4 is given in Appendix A.

The first term on the right-hand side of (5) is due to the assumption that the path whose labels correspond to the transmitted codeword will be expanded. In the second term, $Q(-\overline{\mu}(\ell,\overline{d})/\overline{\sigma}(\ell,\overline{d}))$ is an upper bound on the probability of a node being expanded, where the node is at level ℓ and the sequence of the labels of the path from the start node to this node have Hamming distance $\overline{d}$ to the transmitted codeword.

In the decoding algorithm proposed in [13] we ordered the positions of $\boldsymbol{\phi}$ to obtain $\boldsymbol{\phi}^*$, which is assumed to be the "received vector." Next, we prove that if the k-most reliable positions of $\boldsymbol{\phi}$ are linearly independent, then the reordering will not increase the computational effort of the SDA.

Theorem 5: 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 the SDA when $\boldsymbol{\phi}^*$ and $\boldsymbol{\phi}$ are decoded, respectively.

The proof of Theorem 5 can be found in Appendix B.

Let $N(\boldsymbol{\phi}^*)$ be the number of nodes visited by the decoding algorithm proposed in [13] when it decodes $\boldsymbol{\phi}^*$ and let $\overline{N}$ be the average number of nodes visited by this decoding algorithm. By Theorems 2, 4, and 5 as well as the Markov inequality [21], we have the following result.

Theorem 6: If the k-most reliable positions of $\boldsymbol{\phi}$ are linearly independent, then

$$\begin{gathered} N(\boldsymbol{\phi}^*) \leq N_s(\boldsymbol{\phi}^*) \leq N_s(\boldsymbol{\phi}) \\ \overline{N} \leq \overline{N}_s \leq \widetilde{N} \end{gathered}$$

and

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

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 in [13].

When the GDA (SDA) searches for an optimal path in a code tree of an (n, k) linear block code, the minimum number of nodes visited by the GDA (SDA) is $2k$, which is the number of nodes visited while the GDA (SDA) searches along the optimal path only. Therefore, the average number of nodes visited by the SDA and $\widetilde{N}$ are greater than or equal to $2k$. However, the average number of nodes visited by the decoding algorithm proposed in [13] may be less than $2k$ due to the effect of the stopping criterion. Since the computation complexity of the stopping criterion is the same order as the computation complexity of the SDA that searches along the optimal path only [13], it is reasonable that we compare $\widetilde{N}$ with the average number of nodes visited by the decoding algorithm in [13] without using the stopping criterion when $\widetilde{N}$ is close to $2k$.

The values of $\widetilde{N}$ for the $(48, 24)$ code for γ_b equal to 2, 3, 4, 5, 6, 7, 8, 9, and 10 dB are given in Fig. 1. In this figure are also given the simulation results of the average number of nodes visited by the SDA and by the decoding algorithm proposed in [13] with and without using the stopping criterion. These averages were obtained by simulating 10 000 samples. The 10 000 samples were generated randomly by a codeword generator and then transmitted over the AWGN channel described in Section II. For the AWGN channel, by the result given in Appendix A we can substitute $\boldsymbol{r}(\boldsymbol{r}^*)$ for $\boldsymbol{\phi}(\boldsymbol{\phi}^*)$ in the decoding algorithm. Since

$$\Pr(r_i|0) = \frac{1}{\sqrt{\pi N_0}} \exp\left[-\frac{(r_i - \sqrt{E})^2}{N_0}\right]$$

for the AWGN channel, the bit error probability of uncoded data (P_e) is the probability that $r_i < 0$ [7]. It can be shown that this probability P_e is given by

$$P_e = Q(\sqrt{2R\gamma_b}). \tag{6}$$

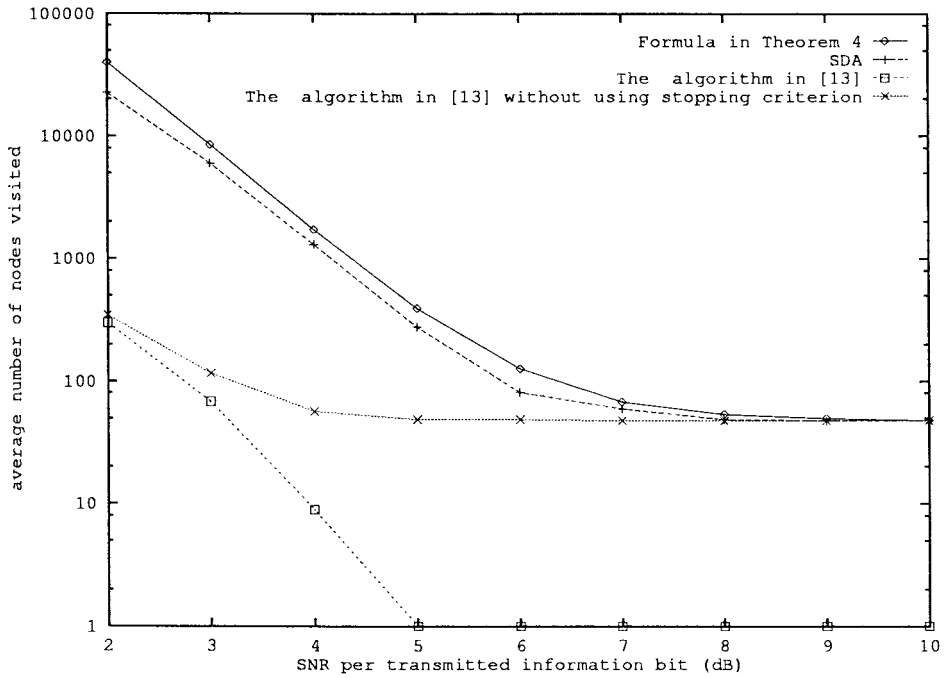

Fig. 1. Average number of nodes visited for the $(48, 24)$ code.

When the probability of error of uncoded data is less than 10^{-3} ($P_e \leq 10^{-3}$) and a $1/2$ rate code is transmitted over the AWGN channel, by (6), $\gamma_b \geq 6.8$ dB.

By the results given in Fig. 1, the values of $\widetilde{N}$ for the $(48, 24)$ code are very tight to the average numbers of nodes visited by the SDA that are obtained by computer simulations. Furthermore, the values of $\widetilde{N}$ are very tight to the average numbers of nodes visited by the decoding algorithm proposed in [13] without using the stopping criterion when SNR is greater than 6 dB. However, because of the simplifying assumptions we had to make, the values of $\widetilde{N}$ are not tight to the average numbers of nodes visited by the decoding algorithm proposed in [13] with and without using the stopping criterion when SNR is less than 6 dB.

In Figs. 2 and 3 we give the values of $\widetilde{N}$ for the $(104, 52)$ code and the $(128, 64)$ code for γ_b from 2 to 10 dB, respectively. We also give the average numbers of nodes visited by the decoding algorithm proposed in [13] with and without using the stopping criterion for γ_b from 5 to 10 dB. By the results presented in Figs. 2 and 3, the values of $\widetilde{N}$ are very tight to the average numbers of nodes visited by the decoding algorithm proposed in [13] without using the stopping criterion for SNR greater than 6.8 dB. Furthermore, the values of $\widetilde{N}$ are closed to $2k$ for SNR greater than 6.8 dB. Thus we may conclude that the decoding algorithms proposed in [13] are efficient for codes of moderate lengths for most practical communication systems where the probability of error is less than 10^{-3} when we assume a $1/2$ rate code is transmitted over the AWGN channel. Even though this upper bound is not tight for SNR of less than 6.8 dB, we still have complexity gains $10^{52-5} = 10^{47}(10^{64-6} = 10^{58})$ on SNR = 5 dB for the $(104, 52)$ code (the $(128, 64)$ code) compared with Wolf's algorithm.

It should be mentioned here that we could not get simulation results when we applied the decoding algorithm given in [13] to the $(104, 52)$ and the $(128, 64)$ for γ_b under 5 dB. Due to the limitations of the memory of the computer, we encountered a received vector, generated by our simulation program, that could not be decoded before the computer crashed. However, to the authors' best knowledge, this algorithm is still the only feasible optimal decoding algorithm for these two codes, even for γ_b greater than 5 dB.

IV. Suboptimal Decoding Algorithm

In the previous section we showed that the GDA is quite efficient for codes of moderate lengths for most practical communication systems where probability of error is less than 10^{-3}; however, by the results given in Figs. 2 and 3, for codes $(104, 52)$ and $(128, 64)$, the number of nodes on list OPEN in the decoding algorithm presented in [13] (GDA) is still too large for the algorithm to have practical applications for low SNR's.

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 which we limit the size of list OPEN by using the following two criteria.

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. That is, we can make sure that every node that stays on list OPEN has a high probability that an optimal path goes through it.

Next, we describe in detail how to use these criteria.

Memory requirements are usually a crucial factor in the practical implementation of any decoding algorithm, especially in the VLSI implementation. Since, in the worst case, for any (n, k) code the maximum size of list OPEN is 2^{k-1}, the GDA is impractical even for the $(48, 24)$ code for low SNR's. However, simulation results indicate that the required size of list OPEN may be much smaller than 2^{k-1} if a small degradation on the performance of the GDA is

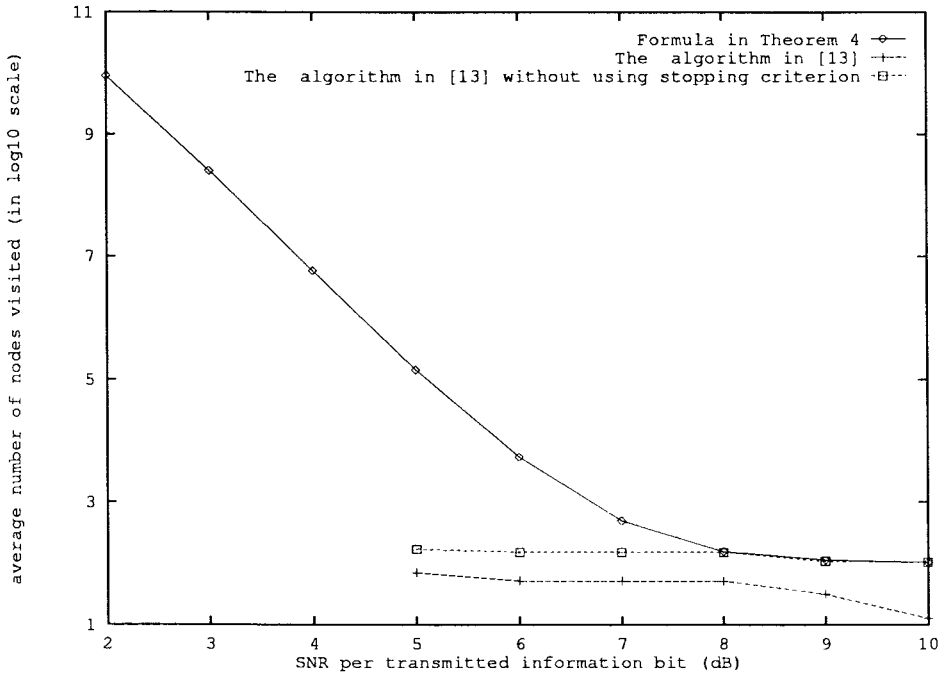

Fig. 2. Average number of nodes visited for the $(104, 52)$ code.

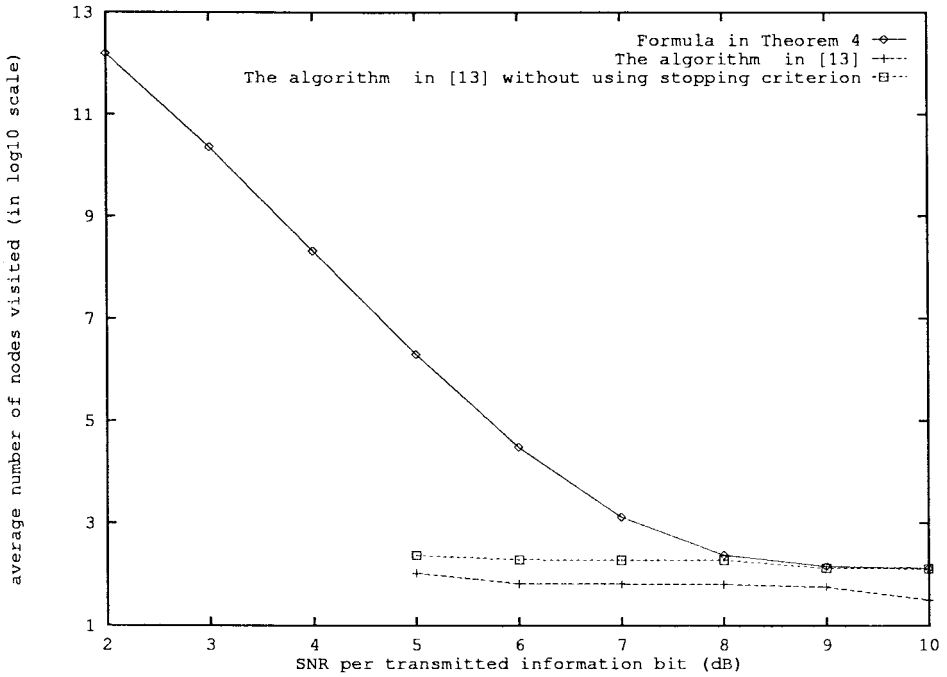

Fig. 3. Average number of nodes visited for the $(128, 64)$ code.

tolerated. 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.

While the GDA searches for an optimal path in a code tree, it calculates $f(m)$ for every node m visited. If $f(m)$ is large, then node m has a low probability of being expanded before the optimal path is found. In other words, when $f(m)$ is large, the probability that the optimal path goes through node m is low and we can discard node m before the optimal path is found without degrading the performance of the GDA much. Therefore, to use the second criterion we need to calculate the probability that an optimal path goes through a node. We now demonstrate how to calculate this probability for an AWGN channel. For any received vector ϕ^*, if an optimal decoding algorithm decodes it to a nontransmitted codeword, then it is almost impossible for a suboptimal decoding algorithm to decode it to the transmitted codeword. Thus when an optimal decoding algorithm decodes a received vector to a nontransmitted codeword, we do not care which codeword a suboptimal decoding algorithm decodes 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. Under this assumption we have the following theorem.

Theorem 7: Let a codeword of an (n, k) code $\boldsymbol{C}$ be transmitted over an AWGN channel. Furthermore, assume the branch cost assigned to the branch from a node at level t to a node at level $t+1$ in the code tree is replaced with the value

$$\left(\frac{N_0}{4\sqrt{E}}\phi_t^* - \sqrt{E}(-1)^{c_t^*}\right)^2$$

where c_t^* is the label of the branch. When no decoding error occurs, the probability distribution of the cost of an optimal path, $h^*(m_{\text{start}})$, 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 7 is given in Appendix C and the theorem holds only for large n.

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. By Theorem 1, for any node m of the code tree, $h(m) \le h^*(m)$. Thus if an optimal path goes through node m, then $h(m) \le h^*(m) \le h^*(m_{\text{start}})$. Thus the probability that an optimal path goes through node m is less than or equal to $\Pr(h(m) \le h^*(m_{\text{start}}) \le UB)$. This leads us to the following theorem.

Theorem 8: 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 $T \le T_{UB}$, where

$$T_{UB} = \frac{1}{\sigma\sqrt{2\pi}}\int_{h(m)}^{UB}\exp\left[-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2\right]dt$$

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. We remark here that to use the second criterion we need to replace the branch cost from a node at level t to a node at level $t+1$ in the code tree with the value

$$\left(\frac{N_0}{4\sqrt{E}}\phi_t^* - \sqrt{E}(-1)^{c_t^*}\right)^2.$$

Now we describe the outline of our decoding algorithm. In our suboptimal decoding algorithm we will fix the maximum number of nodes MB 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 MB, 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 speedup techniques described in Section II, such as stopping criterion and changing the seed during the decoding procedure, can be applied to the suboptimal decoding algorithm.

V. Simulation Results for the AWGN Channel

In order to verify the performance of our suboptimal decoding algorithm, we present simulation results for the $(48, 24)$, the $(104, 52)$ binary extended quadratic residue codes, and the $(128, 64)$ binary extended BCH code when these codes are transmitted over the AWGN channel described in Section II.

For the $(48, 24)$ code, $HW = \{0, 12, 16, 20, 24, 28, 32, 36, 48\}$. We do not know HW for the $(104, 52)$ and the $(128, 64)$ 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 [15]. Thus for this code the superset used is $\{x|(x$ is divisible by 4 and $20 \le x \le 84)$ or $(x = 0)$ or $(x = 104)\}$; for $(128, 64)$, the superset used is $\{x|$ $(x$ is even and $22 \le x \le 106)$ or $(x = 0)$ or $(x = 128)\}$, since this code has $d_{\min} = 22$.

We have implemented a suboptimal version of the adaptive decoding algorithm presented in [13]. Let $\boldsymbol{y} = (y_0, y_1, \cdots, y_{n-1})$ be the hard decision of $\boldsymbol{r}^*$. In the optimal version of the adaptive decoding algorithm presented in [13], the initial seed c_0^* is constructed by $c_0^* = \boldsymbol{u}\cdot\boldsymbol{G}^*$, where $u_i = y_i$ for $0 \le i \le k-1$. Although the received vector $\boldsymbol{r}$ was reordered to get $\boldsymbol{r}^*$ according to the reliability of its components, some errors may occur in the first k components of $\boldsymbol{y}$ when the channel is noisy. Thus the initial seed $\boldsymbol{c}_0^*$ is constructed by considering the 16 codewords as follows. Let

$$S = \{\boldsymbol{u}|\boldsymbol{u} = (u_0, u_1, \cdots, u_{k-1}) \text{ and } u_i = y_i \text{ for } 0 \le i \le k-5 \text{ and } u_i = 0 \text{ or } 1 \text{ for } k-4 \le i \le k-1\}.$$

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 $h(m_{\text{start}})$, calculated with respect to $\boldsymbol{c}^*$, is the largest among all the 16 codewords. We remark here that the selection of these codewords is based on a simulation observation that y_{k-4}, y_{k-3}, y_{k-2}, and y_{k-1} are the four components among $y_0, y_1, \cdots$, and y_{k-1} with higher probability to contain errors. The rule of updating seed is as follows [12], [13]. Let $\boldsymbol{c}_s^*$ be the seed constructed so far by the decoding algorithm. Whenever a codeword $\boldsymbol{c}_n^*$ is generated during the decoding procedure, the algorithm calculates $h(m_{\text{start}})$ with respect to $\boldsymbol{c}_n^*$. If this value is greater than the $h(m_{\text{start}})$ calculated with respect to $\boldsymbol{c}_s^*$, then $\boldsymbol{c}_n^*$ will be the new seed.

The simulation results for the $(48, 24)$ code for γ_b equal to 1, 2, 3, and 4 dB are given in Fig. 4 and in Table I for three MB's; δ is equal to 0.0. Bit error probability of the uncoded data (P_e) is also given.

From the results given in Fig. 4, for the $(48, 24)$ code the performance of the suboptimal decoding algorithm with $MB = 500$ is the same as that of the optimal decoding algorithm whose $MB = 2^{24} = 8388608$. When $MB = 250$, the performance of the suboptimal decoding algorithm is slightly worse than that of the optimal decoding algorithm. From the results given in Table I, the average number of nodes visited is smaller when MB is smaller. Furthermore, when we do not limit the size of list OPEN in the optimal decoding, the maximum number of nodes in list OPEN will grow to 3961, which is far smaller than 8388608, the possible largest size of list OPEN. However, it is still very large if we compare it with 500. Therefore, for code $(48, 24)$ to limit MB to 500 seems a feasible solution for practical application when the SNR is low. Since the average number of nodes visited by the suboptimal decoding algorithm with $MB = 500$ is small (754), it is not necessary to use the second criterion given in Section IV.

The simulation results for the $(104, 52)$ code for γ_b equal to 1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0, and 3.25 dB are given in Fig. 5 and in Table II for three threshold values. MB is equal to 3000. In Fig. 5 we also give a lower bound on the bit error probability of the maximum-likelihood decoding algorithm. This lower bound is obtained as follows [8]. For every sample, when the suboptimal

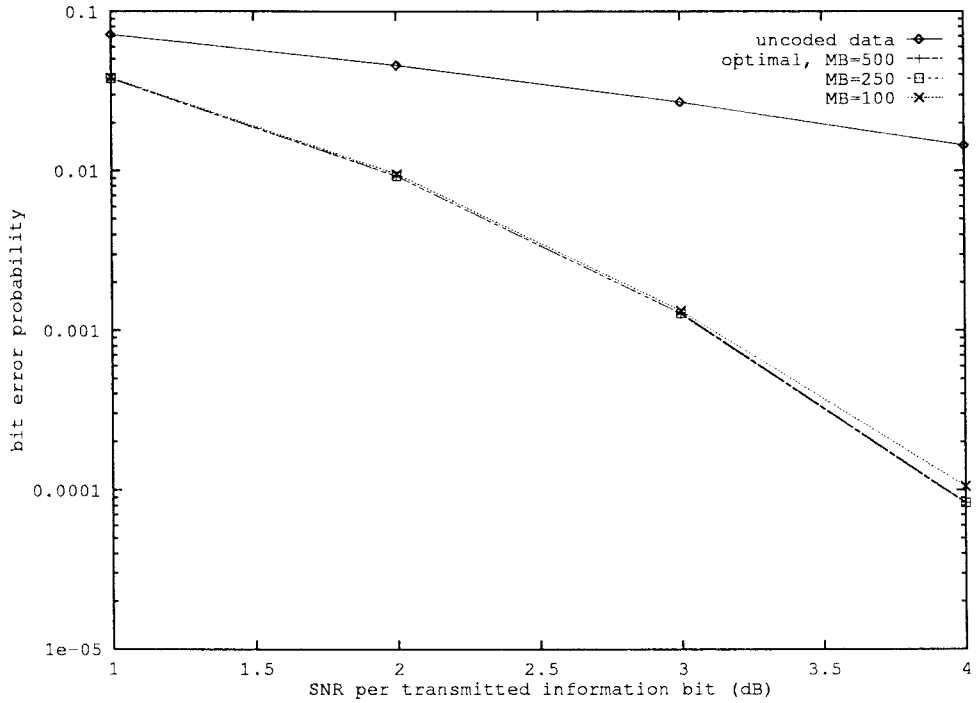

Fig. 4. Performance of suboptimal decoding algorithm for the $(48, 24)$ code for several MB's.

TABLE I
THE AVERAGE NUMBER OF NODES VISITED AND THE MAXIMUM SIZE OF LIST OPEN REQUIRED DURING THE DECODING OF $(48, 24)$ CODE

γ_b	1 dB		2 dB		3 dB		4 dB	
	ave	$\|OPEN\|$	ave	$\|OPEN\|$	ave	$\|OPEN\|$	ave	$\|OPEN\|$
Optimal	843	3810	301	3916	71	2495	9	919
MB=500	754	500	277	500	68	500	9	500
MB=250	631	250	244	250	63	250	9	250
MB=100	398	100	171	100	50	100	8	100

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 to the transmitted codeword, then any optimal decoding algorithm will also decode the received vector to a nontransmitted codeword. Thus we assume that the optimal decoding algorithm will decode to the codeword obtained from the suboptimal decoding algorithm and report if a decoding error occurs. Bit error probability of the uncoded data is also given in Fig. 5.

From Fig. 5, 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 0.65 dB of the performance of an optimal decoding algorithm; and the performance of the suboptimal decoding algorithm with $\delta = 0.5$ is within 1.025 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 Fig. 2.

The simulation results for the $(128, 64)$ code for γ_b equal to 1.0, 1.25, 1.5, 1.75, and 2.0 dB are given in Fig. 6 and Table III for three threshold values. MB is equal to 6000. In Fig. 6 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 Fig. 6, for the $(128, 64)$ code the performance of the suboptimal decoding algorithm 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 Fig. 3.

VI. CONCLUSIONS

In this correspondence we present an upper bound on the average number of nodes visited by the maximum-likelihood soft-decision decoding algorithm given in [13] (GDA). Since this upper bound is derived by applying the central limit theorem to a simplified version of the GDA, the results hold only for large code lengths. However, from the results presented in Section III, this upper bound shows that the GDA is efficient for codes of moderate lengths when the probability of error of the channel is less than 10^{-3}. For low SNR's,

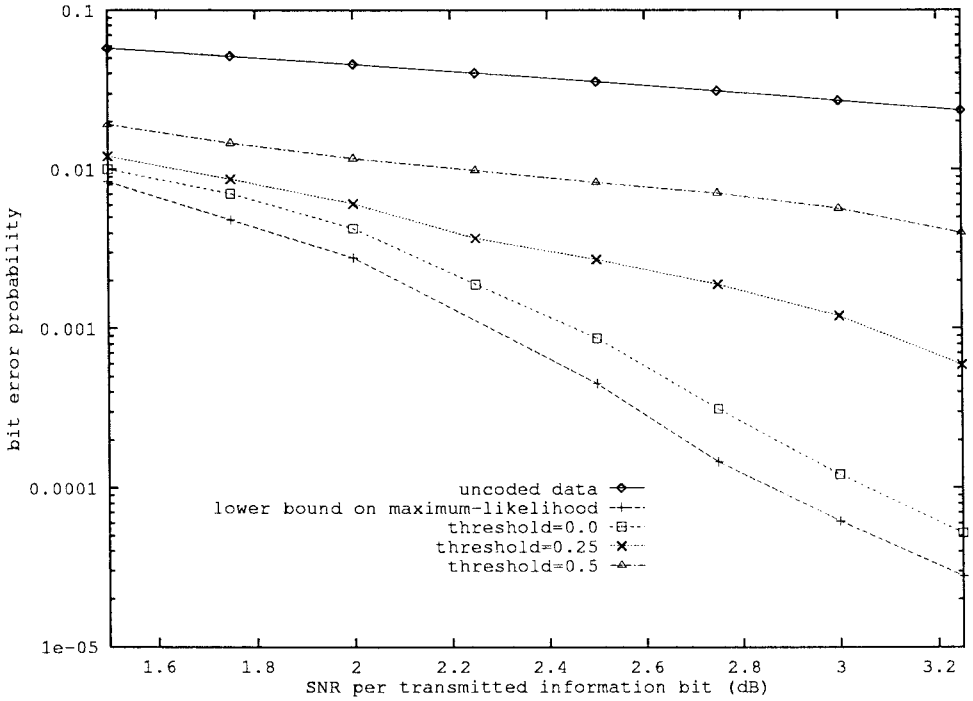

Fig. 5. Performance of suboptimal decoding algorithm for the $(104, 52)$ code.

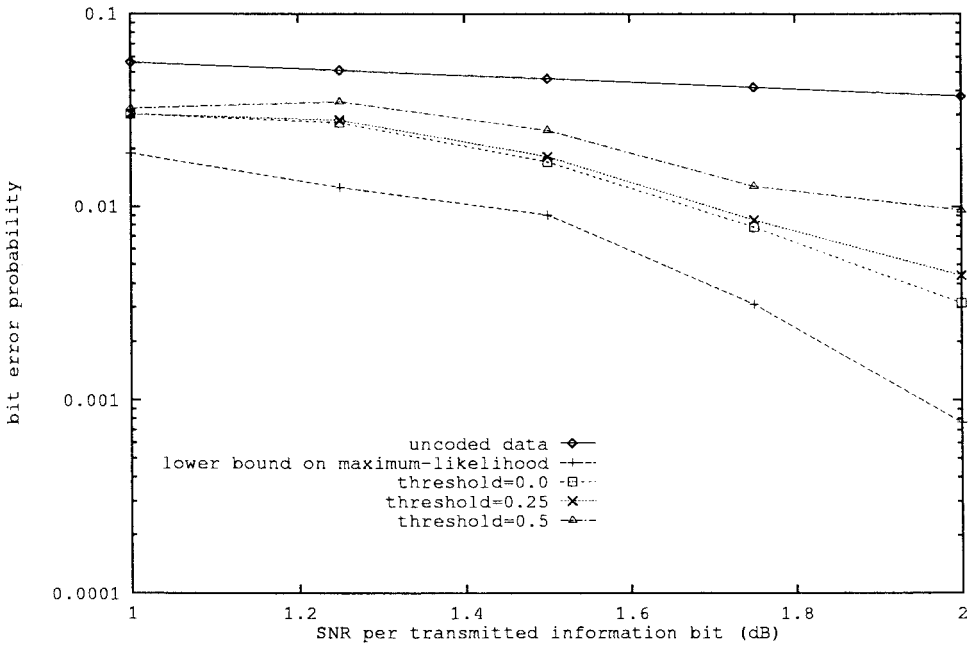

Fig. 6. Performance of suboptimal decoding algorithm for the $(128, 64)$ code.

TABLE II
THE AVERAGE NUMBER OF NODES VISITED DURING THE DECODING OF $(104, 52)$ CODE

threshold	1.5 dB	1.75 dB	2.0 dB	2.25 dB	2.5 dB	2.75 dB	3.0 dB	3.25 dB
0.0	26357	23909	18366	13240	10070	6698	4281	2612
0.25	10976	9643	6481	3980	2879	1579	821	435
0.5	3166	2827	1818	950	703	344	185	101

the GDA becomes impractical for these codes. In order to solve this problem, we also give a suboptimal version of the GDA that reduces the decoding complexity, but compensates for a loss in performance.

The branch cost assigned to the branch from a node at level t to a node at level $t+1$ in the code tree, presented in Section II may be replaced with the value $-(-1)^{c_t^*}\phi_t^*$ to save computation power. However, the designed heuristic function cannot violate the requirement of inequality (3) in order to guarantee that the GDA will find an optimal path. For example, in the case that $h(m) = 0$ for any node m at level ℓ, with $\ell \leq k-1$, the branch cost cannot be changed

TABLE III
THE AVERAGE NUMBER OF NODES VISITED DURING THE DECODING OF $(128, 64)$ CODE

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

to the new value since it violate the requirement of inequality (3). More discussion of the new branch costs can be found in Appendix D.

It is interesting to verify the performance of the decoding algorithm given in [13] when all simulated samples contain at least one error in the hard decision of the received vector for high SNR's. We simulated 10 000 samples and the samples containing at least one error among these 10 000 samples for SNR were equal to 7 and 8 dB. There are 4489 and 2510 samples containing errors among the samples simulated for SNR = 7 dB and SNR = 8 dB, respectively. For SNR = 7 dB, the average number of nodes visited for both cases are 0.0164 and 0.0232, respectively. For SNR = 8 dB, the average number of nodes visited for both cases was 0.0. Therefore, for high SNR's, the decoding algorithm given in [13] is still very efficient when we apply it to those received vectors whose hard decisions contain at least one error.

APPENDIX A
PROOF OF THEOREM 4

Let an (n, k) code $\boldsymbol{C}$ be transmitted over an AWGN channel. It is easy to show that the MLD rule can be formulated as [7]

$$\text{set } \widehat{\boldsymbol{c}} = \boldsymbol{c}_\ell, \quad \text{where } \boldsymbol{c}_\ell \in \boldsymbol{C} \text{ and}$$

$$\sum_{j=0}^{n-1} (a\phi_j - b(-1)^{c_{\ell j}})^2 \le \sum_{j=0}^{n-1} (a\phi_j - b(-1)^{c_{ij}})^2 \tag{7}$$

for all $\boldsymbol{c}_i \in \boldsymbol{C}$, where a and b are any positive nonzero real number.

By the above MLD rule, the branch cost assigned to the branch from a node at level t to a node at level $t+1$ in the code tree may be replaced with the value $(a\phi_t - b(-1)^{c_t})^2$, where c_t is the label of the branch. Furthermore, we may substitute $(a\phi_i - b(-1)^{c_i})^2$ for the value $(\phi_i - (-1)^{c_i})^2$ in the definition of h_p. The proof that h_p satisfies inequality (3) is very similar to that given in [13] where we use old branch costs. Hence, we omit the proof here.

From [14]

$$\Pr(r_i|o) = \frac{1}{\sqrt{\pi N_0}} \exp\left[-\frac{(r_i - \sqrt{E})^2}{N_0}\right]$$
$$\Pr(r_i|1) = \frac{1}{\sqrt{\pi N_0}} \exp\left[-\frac{(r_i + \sqrt{E})^2}{N_0}\right]$$

and

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

Thus

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

and we can substitute $\boldsymbol{r}$ for $\boldsymbol{\phi}$ in our decoding algorithm when $\boldsymbol{C}$ is transmitted over the AWGN channel [3], [7] if we set $a = \frac{N_0}{4\sqrt{E}}$ and $b = 1$. Furthermore, without loss of generality we can assume that all zero codewords $\mathbf{0}$ are transmitted over the AWGN channel.

Let $\boldsymbol{P}'_{\mathbf{0}}$ be the path from the start node m_{start} to a goal node whose labels are all zero. Let us define the cost of the path $\boldsymbol{P}'_0$ as $g(\boldsymbol{P}'_{\mathbf{0}})$. That is,

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

From the definition of $f_s^*(m_{\text{start}})$ which is the cost of an optimal path, we have

$$g(\boldsymbol{P}'_{\mathbf{0}}) \ge f_s^*(m_{\text{start}}).$$

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_{start} to node m, be $\overline{v}_0, \overline{v}_1, \cdots, \overline{v}_{\ell-1}$. Let $S' = \{i | \overline{v}_i = 1, 0 \le i \le \ell - 1\}$ and $|S'| = \overline{d}$. From the definition of function f_s

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

Now we want to calculate the probability that node m is expanded by the algorithm. By Theorem 3, if the GDA expands the node m then $f_s(m) \le f_s^*(m_{\text{start}})$. Thus this probability will be less than or equal to the probability that $f_s(m) \le f_s^*(m_{\text{start}})$, i.e., $\Pr(f_s(m) \le f_s^*(m_{\text{start}}))$. Since

$$g(\boldsymbol{P}'_{\mathbf{0}}) \ge f^*(m_{\text{start}})$$

then

$$\Pr(f_s(m) \le f_s^*(m_{\text{start}})) \le \Pr(f_s(m) \le g(\boldsymbol{P}'_{\mathbf{0}})).$$

Furthermore,

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

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,

$$\Pr(r_i) = \frac{1}{\sqrt{\pi N_0}} \exp\left[-\frac{(r_i - \sqrt{E})^2}{N_0}\right].$$

Thus $E(r_i)$ is $\sqrt{E}$ and $\text{Var}(r_i)$ is $N_0/2$. Then

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

and

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

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

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

Let

$$x = \frac{t - \sqrt{E}}{\sqrt{\frac{N_0}{2}}}$$

then

$$\begin{aligned} E(Z_i') &= \frac{\sqrt{\frac{N_0}{2}}}{\sqrt{\pi N_0}} \int_{-\infty}^{-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}} 2\left(\sqrt{\frac{N_0}{2}}x + \sqrt{E}\right) e^{-\frac{x^2}{2}}\, dx \\ &= \frac{2\sqrt{\frac{N_0}{2}}}{\sqrt{2\pi}} \int_{-\infty}^{-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}} x e^{-\frac{x^2}{2}}\, dx \\ &\quad + \frac{2\sqrt{E}}{\sqrt{2\pi}} \int_{-\infty}^{-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}} e^{-\frac{x^2}{2}}\, dx \\ &= \frac{2\sqrt{\frac{N_0}{2}}}{\sqrt{2\pi}} \int_{-\infty}^{-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}} x e^{-\frac{x^2}{2}}\, dx + 2\sqrt{E} Q\left(-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}\right) \end{aligned}$$

where $Q(\cdot)$ is the standard normal distribution,

If $y = x^2/2$, then $dy = xdx$. Thus

$$E(Z_i') = 2\sqrt{E} Q\left(-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}\right) - \sqrt{\frac{N_0}{\pi}}\, e^{-(E/N_0)}.$$

Similarly,

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

Now let us define a new random variable X as

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

By Lindeberg's central limit theorem [9], 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

$$\begin{aligned} \overline{\mu}(\ell, \overline{d}) &= \overline{d} E(Z_i) + (n - \ell) E(Z_i') \\ &= 2\overline{d}\sqrt{E} + (n - \ell)\left\{2\sqrt{E} Q\left(-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}\right) - \sqrt{\frac{N_0}{\pi}} e^{-\frac{E}{N_0}}\right\} \\ &= \sqrt{N_0}\left\{2\overline{d}\sqrt{R\gamma_b} + (n - \ell)\right. \\ &\quad \left. \cdot \left[2\sqrt{R\gamma_b} Q(-\sqrt{2R\gamma_b}) - \frac{1}{\sqrt{\pi}} e^{-R\gamma_b}\right]\right\} \end{aligned}$$

$$\begin{aligned} \overline{\sigma}^2(\ell, \overline{d}) &= \overline{d}\,\mathrm{Var}\,(Z_i) + (n - \ell)\,\mathrm{Var}\,(Z_i') \\ &= 2\overline{d} N_0 + (n - \ell) \\ &\quad \cdot \left\{2(2E + N_0) Q\left(-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}\right) - 2\sqrt{\frac{EN_0}{\pi}} e^{-\frac{E}{N_0}}\right. \\ &\quad \left. - \left(2\sqrt{E} Q\left(-\frac{\sqrt{E}}{\sqrt{\frac{N_0}{2}}}\right) - \sqrt{\frac{N_0}{\pi}} e^{-\frac{E}{N_0}}\right)^2\right\} \\ &= N_0\left\{2\overline{d} + (n - \ell)\right. \\ &\quad \cdot \left[(4R\gamma_b + 2) Q(-\sqrt{2R\gamma_b}) - 2\sqrt{\frac{R\gamma_b}{\pi}} e^{-R\gamma_b}\right. \\ &\quad \left.\left. - \left(2\sqrt{R\gamma_b} Q(-\sqrt{2R\gamma_b}) - \frac{1}{\sqrt{\pi}} e^{-R\gamma_b}\right)^2\right]\right\}. \end{aligned}$$

Thus

$$\Pr\left(f_s(m) \le f_s^*(m_{\mathrm{start}})\right) \le \Pr\left(X \le 0\right) = Q\left(-\frac{\overline{\mu}(\ell, \overline{d})}{\overline{\sigma}(\ell, \overline{d})}\right).$$

Since $f_s(m_0) \le g(\boldsymbol{P}_0')$ 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_{start} to it will contain at least one 1. Consider those nodes at level ℓ whose paths contain $\overline{d}$ ones, where $1 \le \overline{d} \le \ell$ and $0 \le \ell \le k - 1$. From the above argument, the probability of these nodes being expanded are $Q(-\frac{\overline{\mu}(\ell, \overline{d})}{\overline{\sigma}(\ell, \overline{d})})$. The total number of these nodes is $\binom{\ell}{\overline{d}}$. 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

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

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

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

Appendix B
Proof of Theorem 5

Let $\boldsymbol{\phi} = (\phi_0, \phi_1, \cdots, \phi_{n-1})$ be the received vector and let $\boldsymbol{\phi}^* = (\phi_0^*, \phi_1^*, \cdots, \phi_{n-1}^*)$ 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 $\phi_0^* = \phi_{\pi(0)}$, $\phi_1^* = \phi_{\pi(1)}, \cdots,$ and $\phi_{n-1}^* = \phi_{\pi(n-1)}$, where π is a position permutation. We now prove that $N_s(\boldsymbol{\phi}^*) \le N_s(\boldsymbol{\phi})$ 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 $f_s(m_1) \le f_s(m_2)$. Let the labels of the path from the start node to node m_1 at level ℓ be $c_0, c_1, \cdots,$ and $c_{\ell-1}$.

Let us define $S_s(\ell)$, $S_a(\ell)$, $S_b(\ell)$, $S_c(\ell)$, $S_d(\ell)$, which are subsets of $\{0, 1, 2, \cdots, n-1\}$ as follows:

$$\begin{aligned}
S_s(\ell) &= \{x | x \le \ell - 1 \text{ and } \pi(x) \le \ell - 1\} \\
S_a(\ell) &= \{x | x \le \ell - 1\} - \{\pi(x) | x \in S_s(\ell)\} \\
S_b(\ell) &= \{x | x \le \ell - 1\} - \{x | x \in S_s(\ell)\} \\
S_c(\ell) &= \{\pi(x) | x \in S_b(\ell)\}, \text{ and} \\
S_d(\ell) &= \{x | \pi(x) \in S_a(\ell)\}.
\end{aligned}$$

It is clear that

$$|S_a(\ell)| = |S_b(\ell)| = |S_c(\ell)| = |S_d(\ell)|.$$

Now let us define the labels $c_0^*, c_1^*, \cdots, c_{\ell-1}^*$ from the start node to node m_2 as follows:

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

where

$$\begin{aligned}
y_i &= 0, \qquad \text{when} \quad \phi_i \ge 0 \\
&= 1, \qquad \text{otherwise} \\
y_i^* &= 0, \qquad \text{when} \quad \phi_i^* \ge 0 \\
&= 1, \qquad \text{otherwise}
\end{aligned}$$

and $q(\boldsymbol{\phi}, \ell)$ 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 correspondence to node m_1.

We next prove that $f_s(m_1) \le f_s(m_2)$.

$$\begin{aligned}
& f(m_2) - f(m_1) \\
&= \sum_{i=0}^{\ell-1} (\phi_i^* - (-1)^{c_i^*})^2 + \sum_{i=\ell}^{n-1} (|\phi_i^*| - 1)^2 \\
&\quad - \sum_{i=0}^{\ell-1} (\phi_i - (-1)^{c_i})^2 - \sum_{i=\ell}^{n-1} (|\phi_i| - 1)^2 \\
&= \sum_{i \in S_b(\ell)} (\phi_i^* - (-1)^{c_i^*})^2 + \sum_{i \in S_d(\ell)} (|\phi_i^*| - 1)^2 \\
&\quad - \sum_{i \in S_a(\ell)} (\phi_i - (-1)^{c_i})^2 - \sum_{i \in S_c(\ell)} (|\phi_i| - 1)^2 \\
&= \left[\sum_{i \in S_b(\ell)} (\phi_i^* - (-1)^{c_i^*})^2 - \sum_{i \in S_c(\ell)} (|\phi_i| - 1)^2 \right] \\
&\quad - \left[\sum_{i \in S_a(\ell)} (\phi_i - (-1)^{c_i})^2 - \sum_{i \in S_d(\ell)} (|\phi_i^*| - 1)^2 \right] \\
&= \sum_{i \in S_f(\ell)} [(|\phi_i^*| + 1)^2 - (|\phi_i^*| - 1)^2] \\
&\quad - \sum_{i \in S_e(\ell)} [(|\phi_i| + 1)^2 - (|\phi_i| - 1)^2] \\
&= 4 \left[\sum_{i \in S_f(\ell)} |\phi_i^*| - \sum_{i \in S_e(\ell)} |\phi_i| \right]
\end{aligned}$$

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

$$c_x^* = y_x^* \oplus y_{q(\boldsymbol{\phi}, \ell)(x)} \oplus c_{q(\boldsymbol{\phi}, \ell)(x)}, \qquad x \in S_b(\ell)$$

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 linearly independent, it follows that $|\phi_i^*| \ge |\phi_j|$ for any $i \in S_f(\ell)$ and $j \in S_e(\ell)$. Thus $f_s(m_2) \ge f_s(m_1)$. By Theorem 3, the GDA will expand the node m only when $f_s(m) \le f_s^*(m_{\text{start}})$. Furthermore, the cost of the optimal path is the same, no matter when the GDA decodes $\boldsymbol{\phi}$ or $\boldsymbol{\phi}^*$. Therefore, we have $N_s(\boldsymbol{\phi}^*) \le N_s(\boldsymbol{\phi})$. □

Appendix C
Proof of Theorem 7

Let an (n, k) code $\boldsymbol{C}$ be transmitted over an AWGN channel. If we assume that no decoding error occurs, then the decoded codeword that forms an optimal path is the transmitted codeword. Assume the transmitted codeword is $(c_0, c_1, \cdots, c_{n-1})$. By inequality (7), given in Appendix A, if we set $a = \frac{N_0}{4\sqrt{E}}$ and $b = \sqrt{E}$, then

$$\begin{aligned}
h^*(m_{\text{start}}) &= f^*(m_{\text{start}}) \\
&= \sum_{i=0}^{n-1} (a\phi_i - b(-1)^{c_i})^2 \\
&= \sum_{i=0}^{n-1} \left(r_i - (-1)^{c_i} \sqrt{E} \right)^2 \\
&= \sum_{i=0}^{n-1} \left(\left(e_i + (-1)^{c_i} \sqrt{E} \right) - (-1)^{c_i} \sqrt{E} \right)^2 \\
&= \sum_{i=0}^{n-1} e_i^2
\end{aligned}$$

where for each $0 \le i \le n-1$, e_i's are independent and identically distributed (i.i.d.) and e_i is a normal random variable with mean 0 and variance $N_0/2$. Consequently,

$$\frac{2}{N_0} \sum_{i=0}^{n-1} e_i^2 = \frac{2}{N_0} h^*(m_{\text{start}})$$

will be distributed as a chi–square random variable with n degrees of freedom. From [21] it follows that

$$\mu = E(h^*(m_{\text{start}})) = n \frac{N_0}{2}$$

and $\sigma^2 = \text{Var}\,(h^*(m_{\text{start}})) = n \frac{N_0^2}{2}$.

However, by the central limit theorem, for large values of n, the probability distribution of $h^*(m_{\text{start}})$ is approximately a normal distribution with mean μ and variance σ^2, given above.

Appendix D

In this appendix we show that the branch cost assigned to the branch from a node at level t to a node at level $t+1$ in the code tree may be replaced with the value $-(-1)^{c_t^*} \phi_t^*$. First, we derive another form of MLD rule which contains the value $-(-1)^{c_t^*} \phi_t^*$. Next, we prove that the function h_p defined in this correspondence still satisfies inequality (3).

Another form of MLD rule can formulated as follows [3], [14]:

$$\text{set } \widehat{\boldsymbol{c}} = \boldsymbol{c}_\ell, \quad \text{where} \quad \boldsymbol{c}_\ell \in \boldsymbol{C}$$

and

$$\sum_{j=0}^{n-1} -(-1)^{c_{\ell j}} \phi_j \le \sum_{j=0}^{n-1} -(-1)^{c_{ij}} \phi_j, \qquad \text{for all } \boldsymbol{c_i} \in \boldsymbol{C} \quad (8)$$

where

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

Proof: Since

$$\sum_{j=0}^{n-1}(\phi_j - (-1)^{c_{\ell j}})^2 \le \sum_{j=0}^{n-1}(\phi_j - (-1)^{c_{ij}})^2$$

iff

$$\sum_{j=0}^{n-1}\phi_j^2 - 2\sum_{j=0}^{n-1}(-1)^{c_{\ell j}}\phi_j + n \le \sum_{j=0}^{n-1}\phi_j^2 - 2\sum_{j=0}^{n-1}(-1)^{c_{ij}}\phi_j + n$$

iff

$$\sum_{j=0}^{n-1} -(-1)^{c_{\ell j}}\phi_j \le \sum_{j=0}^{n-1} -(-1)^{c_{ij}}\phi_j$$

then the result holds directly from the MLD rule given in Section II. □

By the above MLD rule we may substitute $-(-1)^{v_i}\phi_i^*$ $(-|\phi_i|)$ for the value $(\phi_i^* - (-1)^{v_i})^2$ $((|\phi_i| - 1)^2)$ in the definition of h_p (h_s). Next, we prove that h_p satisfies inequality (3).

Let node m_2 at level ℓ be an immediate successor of node m_1. Furthermore, let $\overline{v}_{\ell-1}$ be the label of the branch from node m_1 to node m_2 and $c(m_1, m_2) = -(-1)^{\overline{v}_{\ell-1}}\phi_{\ell-1}^*$. We now prove that

$$h_p(m_1) \le h_p(m_2) + c(m_1, m_2).$$

Case 1. $\ell \le k-1$*:* Let

$$\boldsymbol{v} = (\overline{v}_0, \overline{v}_1, \cdots, \overline{v}_{\ell-1}, v_\ell, v_{\ell+1}, \cdots v_{n-1}) \in T(m_2)$$

such that

$$h_p(m_2) = \sum_{i=\ell}^{n-1} -(-1)^{v_i}\phi_i^*.$$

Since

$$\boldsymbol{v} \in T(m_2)$$

then

$$\boldsymbol{v} \in T(m_1).$$

Thus

$$\sum_{i=\ell}^{n-1} -(-1)^{v_i}\phi_i^* + c(m_1, m_2) \ge h_p(m_1)$$

i.e.,

$$h_p(m_2) + c(m_1, m_2) \ge h_p(m_1).$$

Case 2. $\ell = k$*:*

$$h_p(m_1) \le h_p^*(m_1)$$

and

$$h_p(m_2) = h_p^*(m_2).$$

Since

$$h_p^*(m_1) - c(m_1, m_2) \le h_p^*(m_2)$$

then

$$h_p(m_1) \le h_p^*(m_2) + c(m_1, m_2) = h_p(m_2) + c(m_1, m_2).$$

Case 3. $\ell > k$*:*

$$h_p(m_1) = h_p^*(m_1)$$

and

$$h_p(m_2) = h_p^*(m_2).$$

Since

$$h_p^*(m_1) - c(m_1, m_2) = h_p^*(m_2)$$

then

$$h_p(m_1) = h_p(m_2) + c(m_1, m_2).$$

Since the proof that h_s satisfies inequality (3) is easy we omit it here.

ACKNOWLEDGMENT

The authors wish to thank Elaine Weinman for her invaluable help in the preparation of this manuscript and on the language check. In addition, the authors wish to thank the reviewers for their invaluable suggestions, which helped to improve the presentation of the correspondence.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Reading, MA: Addison-Wesley, 1974.
[2] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.
[3] Y. Be'ery and J. Snyders, "Optimal soft decision block decoders based on fast Hadamard transform," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 355–364, 1986.
[4] Y. Berger and Y. Be'ery, "Soft trellis-based decoder for linear block codes," *IEEE Trans. Inform. Theory*, vol. 40, pp. 764–773, May 1994.
[5] R. W. D. Booth, M. A. Herro, and G. Solomon, "Convolutional coding techniques for certain quadratic residue codes," in *Proc. 1975 Int. Telemetering Conf.*, 1975, pp. 168–177.
[6] G. Brassard and P. Bratley, *Algorithmics Theory and Practice.* Englewood Cliffs, NJ: Prentice-Hall, 1988.
[7] G. C. Clark, Jr., and J. B. Cain, *Error-Correction Coding for Digital Communications.* New York: Plenum, 1981.
[8] B. G. Dorsch, "A decoding algorithm for binary block codes and J-ary output channels," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 391–394, May 1974.
[9] W. Feller, *An Introduction to Probability Theory and Its Applications.* New York: Wiley, 1966.
[10] G. D. Forney, Jr., "Coset codes–Part II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1152–1187, Sept. 1988.
[11] Y. S. Han, "Efficient soft-decision decoding algorithms for linear block codes using algorithm A^*," Ph.D. dissertation, Sch. Comp, Inform. Sci., Syracuse Univ., Syracuse, NY 13244, 1993.
[12] Y. S. Han and C. R. P. Hartmann, "Designing efficient maximum-likelihood soft-decision decoding algorithms for linear block codes using algorithm A^*," Sch. Com. Inform. Sci., Syracuse Univ., Syracuse, NY 13244, June 1992, Tech. Rep. SU-CIS-92-10.
[13] 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 Trans. Inform. Theory*, vol. 39, pp. 1514–1523, Sept. 1993.
[14] T.-Y. Hwang, "Decoding linear block codes for minimizing word error rate," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 733–737, Nov. 1979.
[15] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes.* New York: Elsevier, 1977.
[16] K. R. Matis and J. W. Modestino, "Reduced-search soft-decision trellis decoding of linear block codes," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 349–355, Mar. 1982.
[17] N. J. Nilsson, *Principle of Artificial Intelligence.* Palo Alto, CA: Tioga, 1980.
[18] J. Snyders, "Reduced lists of error patterns for maximum likelihood soft decoding," *IEEE Trans. Inform. Theory*, vol. 37, pp. 1194–1200, July 1991.

[19] J. Snyders and Y. Be'ery, "Maximum likelihood soft decoding of binary block codes and decoders for the Golay codes," *IEEE Trans. Inform. Theory*, vol. 35, pp. 963–975, Sept. 1989.
[20] G. Solomon and H. C. A. van Tilborg, "A connection between block and convolutional codes," *SIAM J. Appl. Math.*, pp. 358–369, 1979.
[21] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications.* Englewood Cliffs, NJ: Prentice-Hall, 1982.
[22] A. Vardy and Y. Be'ery, "Bit-level soft decision decoding of Reed–Solomon codes," *IEEE Trans Commun.*, vol. 37, pp. 440–445, 1991.
[23] ——, "More efficient soft-decision decoding of the Golay codes," *IEEE Trans. Inform. Theory*, vol. 37, pp. 667–672, 1991.
[24] A. J. Viterbi, "Error bound for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260–269, Apr. 1967.
[25] J. K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 76–80, Jan. 1978.

A Proof of the Fisher Information Inequality via a Data Processing Argument

Ram Zamir, *Member, IEEE*

***Abstract*— The Fisher information $J(X)$ of a random variable X under a translation parameter appears in information theory in the classical proof of the Entropy-Power Inequality (EPI). It enters the proof of the EPI via the De-Bruijn identity, where it measures the variation of the differential entropy under a Gaussian perturbation, and via the convolution inequality $J(X+Y)^{-1} \geq J(X)^{-1} + J(Y)^{-1}$ (for independent X and Y), known as the Fisher Information Inequality (FII). The FII is proved in the literature directly, in a rather involved way. We give an alternative derivation of the FII, as a simple consequence of a "data-processing inequality" for the Cramer–Rao lower bound on parameter estimation.**

***Index Terms*—Cramer–Rao bound, data processing inequality, entropy-power inequality, Fisher information, linear modeling, non-Gaussian noise, prefiltering.**

I. Introduction

The data processing inequality (or the data processing theorem) is used in information theory for proving the converse channel-coding theorem [4, Secs. V.3, V.4], [6, Secs. II.8, VIII.9]. This inequality asserts that if the random variables $W - X - Y$ form a Markov chain in this order, then the mutual informations between them satisfy

$$I(W;Y) \leq I(W;X). \tag{1}$$

In the special case where Y is given by a deterministic function ϕ of X, (1) becomes

$$I(W;\phi(X)) \leq I(W;X) \tag{2}$$

Manuscript received June 1, 1995; revised October 1, 1997. This work was supported in part by the Wolfson Research Awards administered by the Israel Academy of Science and Humanities. The material in this correspondence was presented in part at the Information Theory Workshop on Multiple Access and Queuing, St. Louis, MO, April 1995.

The author is with the Department of Electrical Engineering–Systems, Tel Aviv University, Tel Aviv 69978, Israel.

Publisher Item Identifier S 0018-9448(98)02377-3.

with equality if $W - \phi(X) - X$ form a Markov chain, e.g., if $\phi(\cdot)$ is an invertible function. The proof of (1) follows straightforwardly from the chain rule and the positivity of the mutual information [6].

The name "data processing inequality" apparently came from the analogy to the problem of optimal filtering. Suppose that W, X, Y are real variables. In analogy with (1) and (2), it is clear and easy to verify that the conditional variance, i.e., the mean-squared error of the conditional mean estimator of W, satisfies the data processing inequalities

$$\mathrm{VAR}(W|Y) \geq \mathrm{VAR}(W|X)$$

and

$$\mathrm{VAR}(W|\phi(X)) \geq \mathrm{VAR}(W|X) \tag{3}$$

where

$$\mathrm{VAR}(W|X) \triangleq E[W - E(W|X)]^2.$$

When the estimated quantity is a parameter $\boldsymbol{\theta}$ (i.e., not a random variable), it is impossible to use the conditional variance as a measure for the goodness of the optimal estimator. Instead, it is common to use the Fisher Information matrix (FI) of the measurement $\boldsymbol{X}$ relative to the parameter vector $\boldsymbol{\theta}$, defined as [4], [6], [10]

$$\begin{aligned}\boldsymbol{J}(\boldsymbol{X};\boldsymbol{\theta}) &\triangleq \mathrm{COV}\left\{\frac{\partial}{\partial\boldsymbol{\theta}}\ \ln(f_{\boldsymbol{\theta}}(\boldsymbol{X}))\right\} \\ &= \int \frac{1}{f_{\boldsymbol{\theta}}(\boldsymbol{x})}\left(\frac{\partial f_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial\boldsymbol{\theta}}\right)\cdot\left(\frac{\partial f_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial\boldsymbol{\theta}}\right)^{t} d\boldsymbol{x}\end{aligned} \tag{4}$$

where $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_m)$, the set $\{f_{\boldsymbol{\theta}}(\boldsymbol{x})\}$ is a family of densities of $\boldsymbol{X}$ parameterized by $\boldsymbol{\theta}$, $\partial/\partial\boldsymbol{\theta}$ denotes the gradient (i.e., a column vector of partial derivatives) with respect to the parameters $\theta_1, \cdots, \theta_m$, $\ln(\cdot)$ denotes the natural logarithm, and $\mathrm{COV}\{\cdot\}$ denotes the $m \times m$ covariance matrix calculated relative to the distribution of $\boldsymbol{X}$. Here $\boldsymbol{X}$ may either be a single measurement or a vector of n measurements. The importance of the matrix $\boldsymbol{J}(\boldsymbol{X};\boldsymbol{\theta})$ follows from the Cramer–Rao Bound (CRB), [4], [6], [10], saying that for any unbiased estimator $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}(\boldsymbol{x})$ (i.e., estimator for which $E\{\hat{\boldsymbol{\theta}}(\boldsymbol{X})\} = \boldsymbol{\theta}$) the error vector $\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}$ satisfies

$$\mathrm{COV}\{\hat{\boldsymbol{\theta}}(\boldsymbol{X})\} \geq \boldsymbol{J}(\boldsymbol{X};\boldsymbol{\theta})^{-1} \tag{5}$$

where throughout the correspondence an inequality between (nonnegative definite) matrices means that the difference matrix is nonnegative definite. As it turns out (see Lemma 3 below), the notion of data processing extends easily to the FI; if $\boldsymbol{\theta} - \boldsymbol{X} - \boldsymbol{Y}$ satisfy a chain relation of the form $f(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{\theta}) = f_{\boldsymbol{\theta}}(\boldsymbol{x})f(\boldsymbol{y}|\boldsymbol{x})$ (i.e., the conditional distribution of Y given X is independent of $\boldsymbol{\theta}$), then we have the data processing inequality

$$\boldsymbol{J}(\boldsymbol{Y};\boldsymbol{\theta}) \leq \boldsymbol{J}(\boldsymbol{X};\boldsymbol{\theta}) \tag{6}$$

whose deterministic version (in analogy with (2)) is

$$\boldsymbol{J}(\phi(\boldsymbol{X});\boldsymbol{\theta}) \leq \boldsymbol{J}(\boldsymbol{X};\boldsymbol{\theta}). \tag{7}$$

Equality in (7) holds if $\phi(X)$ is a *sufficient statistic* relative to the family $\{f_{\boldsymbol{\theta}}(x)\}$, i.e., $\boldsymbol{\theta} - \phi(\boldsymbol{X}) - \boldsymbol{X}$ form a chain [6, Sec. II.10].[1]

In the context of information-theoretic inequalities, e.g., in the derivation of the Entropy Power Inequality (EPI), there appears a

[1] An alternative ("Bayesian") way to express the equality condition is that $\boldsymbol{\Theta} - \phi(\boldsymbol{X}) - \boldsymbol{X}$ for a Markov chain for any distribution on the parameter θ.