Neural Networks, Error-Correcting Codes, and Polynomials over the Binary n-Cube

JEHOSHUA BRUCK, MEMBER, IEEE, AND MARIO BLAUM, MEMBER, IEEE

Abstract —Several ways of relating the concept of error-correcting codes to the concept of neural networks are presented. Performing maximum likelihood decoding in a linear block error-correcting code is shown to be equivalent to finding a global maximum of the energy function of a certain neural network. Given a linear block code, a neural network can be constructed in such a way that every local maximum of the energy function corresponds to a codeword and every codeword corresponds to a local maximum. The connection between maximization of polynomials over the n-cube and error-correcting codes is also investigated; the results suggest that decoding techniques can be a useful tool for solving problems of maximization of polynomials over the n-cube. The results are generalized to both nonbinary and nonlinear codes.

I. Introduction

OUR MAIN GOAL is to explore the connections between the three concepts appearing in the title. A neural network is a computational model type that has recently been attracting much interest because it seems to have properties that are similar to those of both biological and physical systems. The usual computation performed in a neural network is the maximization of the so-called energy function. The state space of a neural network can be described by the topography defined by the energy function associated with the network.

The main problem in the field of error-correcting codes is to design good codes: codes that can correct many errors and whose encoding and decoding procedures are computationally efficient. An error-correcting code can be described by a topography, with the peaks of the topography being the codewords. The decoding of a corrupted word (a point in the topography that is not a peak) is then equivalent to looking for the closest peak in the topography.

The analogy between the two subjects just described was the initial motivation for this work. It turns out that both neural networks and error-correcting codes can be described by polynomials over the n-cube. Thus the connection between the two concepts can be established. The representation of error-correcting codes using polynomials over the n-cube also provides a new perspective to the subject that enables us to derive some new proofs to already known results. The problem of the maximization of polynomials over the n-cube is known in operations research and computer science. The connection with error-correcting codes suggests a new tool for solving these problems, namely, decoding techniques. The other direction, that is, maximum likelihood decoding using known techniques of maximization of polynomials, is not addressed in this paper but is an interesting subject for future research.

Manuscript received December 7, 1987; revised November 2, 1988. This work was supported in part by the U.S. Air Force Office of Scientific Research. This paper was presented in part at the IEEE Symposium on Information Theory, Kobe, Japan, June 1988.

The authors are with the IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, CA 95120-6099.

IEEE Log Number 8930427.

The paper is organized as follows. In Section II we present some background on neural networks. We review the basic definitions of the Hopfield model. We discuss stable states and the different modes of operation of the network. We conclude the section by proving that finding a global maximum of the energy function of the network is equivalent to finding a minimum cut in a certain graph. The generalization to energy functions of a higher degree is also reviewed.

In Section III we establish a connection between the Hopfield model and graph theoretic codes. We prove that maximum likelihood decoding in a graph theoretic code is equivalent to finding the minimum cut in a certain graph. By the description in the previous section, this implies that maximum likelihood decoding in a graph theoretic code is equivalent to finding a maximum of the energy in a neural network.

In Section IV we extend the results of Section III to general linear block codes. The key idea is to represent the binary symbols $\{0,1\}$ by the symbols $\{1,-1\}$ with the operation being multiplication instead of exclusive OR. A general energy function, not necessarily quadratic, is defined based on the generator matrix of a given linear block code. We show that finding the global maximum of this energy function is equivalent to maximum likelihood decoding in the code. A generalization of the results to nonbinary codes is presented in Appendix I while a generalization of the results to nonlinear codes is presented in Appendix II.

In Section V we study the energy function associated with the parity check matrix of a code. When this matrix is written in systematic form, we show that each codeword corresponds to a local maximum of the polynomial associated with the parity check matrix and that each local maximum corresponds to a codeword. We interpret the results of this section as dual to the ones in Section IV for defining the maximum likelihood problem.

In Section VI we consider the problem of solving unconstrained nonlinear 0–1 programs. This is basically the problem of maximizing a polynomial on n variables, with each variable being either zero or one. It is known that this problem is NP-hard. The known solvable cases use the concept of the conflict graph. We found that the family of polynomials associated with Hamming codes results in a conflict graph which is not bipartite in general (i.e., for which an efficient algorithm is not known). For the family of polynomials associated with Hamming codes, efficient recognition and maximization techniques that are based on decoding techniques are presented.

A note regarding the notation is in order. Since G denotes a graph in graph theory and a generator matrix in coding theory, we put carets on all letters that are used to denote graphs. Thus, a graph is denoted by $\hat{G}=(\hat{V},\hat{E})$, while a generator matrix of a code is denoted by G.

II. Background on Neural Networks

The neural network model is a discrete-time system that can be represented by a weighted undirected graph. There is a weight attached to each edge of the graph and a threshold value attached to each node (neuron) of the graph. The *order* of the network is the number of nodes in the corresponding graph. Let N be a neural network of order n; then N is uniquely defined by (W,T) where

- W is an $n\times n$ symmetric zero-diagonal matrix where W_{ij} is equal to the weight attached to edge (i,j),
- T is a vector of dimension n where T_i denotes the threshold attached to node i.

Every node (neuron) can be in one of two possible states, either 1 or -1. The state of node i at time t is denoted by $V_i(t)$. The *state* of the neural network at time t is the vector $V(t)$. The next state of a node is computed by

$$V_i(t+1)=\operatorname{sgn}(H_i(t))=\begin{cases}1, & \text{if } H_i(t)\geq 0\\ -1, & \text{otherwise}\end{cases} \tag{1}$$

where

$$H_i(t)=\sum_{j=1}^{n} W_{ji}V_j(t)-T_i.$$

The next state of the network, i.e., $V(t+1)$, is computed from the current state by performing the evaluation (1) at a subset of the nodes of the network, to be denoted by S. The modes of operation are determined by the method by which the set S is selected in each time interval.

If the computation is performed at a single node in any time interval, i.e., $|S|=1$, then we will say that the network is operating in a serial mode, and if $|S|=n$, then we will say that the network is operating in a *fully parallel* mode. All the other cases, i.e., $1<|S|<n$, will be called parallel modes of operation. The set S can be chosen at random or according to some deterministic rule. A state $V(t)$ is called *stable* if and only if $V(t)=\operatorname{sgn}(WV(t)-T)$, i.e., there is no change in the state of the network no matter what the mode of operation is.

Example: Consider the network in Fig. 1 with $W_{1,2}=-1$ and T the 0 vector. It can be verified that the stable states of this simple network are $(-1,1)$ and $(1,-1)$.

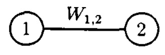

Fig. 1. Neural network with two nodes.

One of the most important properties of the model is its convergence property, as summarized by the following proposition.

Proposition 1 [5], [8], [14]: Let $N=(W,T)$ be a neural network, with W a symmetric matrix; then the network N always converges to a stable state while operating in a serial mode and to a cycle of length at most two while operating in a fully parallel mode.

The main idea in the proof of the convergence property is to define a so-called energy *function* and to show that this energy function is nondecreasing when the state of the network changes as a result of computation. The energy function used in the proof of Proposition 1 is

$$E(t)=V^T(t)WV(t)-2V^T(t)T. \tag{2}$$

A neural network when operating in a serial mode will always get to a stable state that corresponds to a local maximum of the energy function. This suggests the use of the network as a device for performing a local search algorithm for finding a maximal value of the energy function [4], [5], [15]. Clearly, every optimization problem that can be defined in a form of a quadratic function over $\{-1,1\}^n$ as in (2) can be mapped to a neural network which will perform a search for its optimum. One of the optimization problems that is not only representable by a quadratic function but is actually equivalent to it is the problem of finding the minimum cut in a graph [5], [19]. To clarify the above statement, let us start by defining the term "cut" in a graph.

Definition: Let $\hat{G}=(\hat{V},\hat{E})$ be a weighted and undirected graph, where $\hat{V}$ denotes the set of nodes of $\hat{G}$ and $\hat{E}$ denotes the set of edges of $\hat{G}$. Let W be the symmetric matrix that corresponds to the weights of the edges of $\hat{G}$. For example, for the network in Fig. 1 we have

$$W=\begin{pmatrix}0 & -1\\ -1 & 0\end{pmatrix}.$$

Let $\hat{V}_1$ be a subset of $\hat{V}$, and let $\hat{V}_{-1}=\hat{V}-\hat{V}_1$. The set of edges each of which is incident at a node in $\hat{V}_1$ and at a node in $\hat{V}_{-1}$ is called a *cut* in $\hat{G}$.

Definition: The weight of a cut is the sum of its edge weights. A *minimum cut* (MC) of a graph is a cut with minimum weight.

The equivalence between the MC problem and the problem of maximizing the energy function of a neural network is summarized by the following theorem (generalizations of this equivalence can be found in [4], [5]). We include the proof to exhibit a principle that will be useful later.

Proposition 2 [5], [19]: Let $N=(W,T)$ be a neural network with all thresholds being zero; i.e., $T\equiv 0$. The problem of finding a state V for which the energy E is maximum is equivalent to finding a minimum cut in the graph corresponding to N.

Proof: Since $T\equiv 0$, the energy function is

$$E=\sum_{i=1}^{n}\sum_{j=1}^{n} W_{i,j}V_iV_j. \tag{3}$$

Let W^{++} denote the sum of weights of edges in N with both endpoints equal to 1, and let W^{--} and W^{+-} denote the corresponding sums of the other two cases. It follows that

$$E=2(W^{++}+W^{--}-W^{+-}) \tag{4}$$

which also can be written as

$$E=2(W^{++}+W^{--}+W^{+-})-4W^{+-}. \tag{5}$$

Since the first term in the above equation is constant (it is the sum of the weights of the edges), it follows that the maximization of E is equivalent to the minimization of W^{+-}. Clearly, W^{+-} is the weight of the cut in N with $\hat{V}_1$ being the nodes of N with a state equal to 1. Q.E.D.

Hence a neural network operating in a serial mode is equivalent to performing a local search algorithm for finding a minimum cut in the network. Changing the state of a node in the network is equivalent to moving it from one side of the cut to the other in the local search algorithm.

The foregoing definition of the model results in an energy function that is quadratic. The definition of the model can be generalized to energy functions of a higher degree [1]. In the general case, every neuron computes an algebraic threshold function that is equivalent to checking which state (either 1 or -1) will result in a higher value of the energy function.

Example: Consider the energy function

$$E=W_{1,2,3}V_1V_2V_3+W_{1,2}V_1V_2+W_{2,3}V_2V_3+W_1V_1.$$

For example, the generalization of (1) for node 1 is

$$V_1(t+1)=\operatorname{sgn}(H_1(t))$$

where

$$H_1(t)=W_{1,2,3}V_2V_3+W_{1,2}V_2+W_1.$$

We will start by investigating the connections between quadratic energy function and error-correcting codes and then continue by looking at general energy functions.

III. Neural Networks and Graph Theoretic Codes

The main goal of this section is to establish the relations between neural networks and graph theoretic error-correcting codes. Let us start by defining the family of graph theoretic codes (for more details see [9], [18]).

Let $\hat{G}=(\hat{V},\hat{E})$ be an undirected graph. A subset of the set of edges of $\hat{G}$ can be represented by a characteristic vector of length $|\hat{E}|$, with edge e_i corresponding to the ith entry of the characteristic vector. That is, every $S\subseteq\hat{E}$ can be represented by a vector denoted by 1_s such that

$$1_s(i)=\begin{cases}1, & \text{if } e_i\in S\\ 0, & \text{otherwise}\end{cases}. \tag{6}$$

Definition: The *incidence matrix* of a graph $\hat{G}=(\hat{V},\hat{E})$, denoted by $D_{\hat{G}}$, is a $|\hat{V}|\times|\hat{E}|$ matrix in which row i is the characteristic vector of the set of edges incident upon node $i\in\hat{V}$.

The following facts from graph theory [3] are the basis for the definition of the family of graph theoretic codes.

Fact 1: The set of characteristic vectors that corresponds to the cuts in a connected graph $\hat{G}=(\hat{V},\hat{E})$ forms a linear vector space over GF(2) of dimension $(|\hat{V}|-1)$. The linear vector space that corresponds to the cuts of a graph $\hat{G}$ will be called the *cut space* of $\hat{G}$. It is interesting to note that the circuits in a graph also constitute a linear vector space.

Fact 2: Given a connected graph $\hat{G}=(\hat{V},\hat{E})$, the incidence matrix of $\hat{G}$ has rank $|\hat{V}|-1$. Every row in $D_{\hat{G}}$ is a characteristic vector of a cut, and every $|\hat{V}|-1$ rows of $D_{\hat{G}}$ form a basis for the cut space of $\hat{G}$.

Hence given a connected graph $\hat{G}$, the cut space of the graph is a *linear block code* [16], [18] of dimension $|\hat{V}|-1$; thus every connected graph has an $[|\hat{E}|,|\hat{V}|-1]$ code associated with its cuts. The code associated with the cuts of a graph $\hat{G}$ will be denoted by $C_{\hat{G}}$.

The codes associated with graphs, that is, cut codes and circuit codes, are called *graph theoretic codes*. In this paper only cut codes will be discussed.

Example: Let $\hat{G}$ be the graph in Fig. 2; $\hat{G}$ has five nodes and eight edges. The incidence matrix of the graph $\hat{G}$ is

$$D_{\hat{G}}=\begin{pmatrix}1&1&1&0&0&0&0&0\\0&0&1&1&1&0&0&0\\0&1&0&1&0&1&1&0\\1&0&0&0&0&1&0&1\\0&0&0&0&1&0&1&1\end{pmatrix}. \tag{7}$$

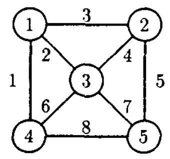

Fig. 2. Graph corresponding to [8,4] code.

Any four rows of $D_{\hat{G}}$ form a basis of the cut space of $\hat{G}$. For example, the matrix consisting of the first four rows of $D_{\hat{G}}$ is a generator matrix of the error-correcting linear block code associated with $\hat{G}$. As is easily observed, $\hat{G}$ does not contain a cut with fewer than three edges (apart from the empty cut); thus the code $C_{\hat{G}}$ has minimum Hamming distance 3 and can correct one error.

Given a graph $\hat{G}$, an interesting question is how to formulate the maximum likelihood decoding (MLD) problem of the code $C_{\hat{G}}$ in a graph theoretic language. That is, given a graph $\hat{G}=(\hat{V},\hat{E})$ and a vector Y in $\{0,1\}^{|\hat{E}|}$, what is the codeword in $C_{\hat{G}}$ closest to Y in Hamming distance? The following lemma will answer this question.

Lemma 1: Let $\hat{G}=(\hat{V},\hat{E})$ be a connected graph. Let $C_{\hat{G}}$ be the code associated with $\hat{G}$. Let Y be a vector over $\{0,1\}^{|\hat{E}|}$. Construct a new graph, to be denoted by $\hat{G}_Y$, by assigning weights to the edges of $\hat{G}$ as follows:

$$W_i=(-1)^{Y_i}. \tag{8}$$

W_i is the weight associated with edge i in $\hat{G}$. Then MLD of Y with respect to $C_{\hat{G}}$ is equivalent to finding the minimum cut in $\hat{G}_Y$.

Proof: Assume the number of ones in Y is a. Let M be an arbitrary codeword in $C_{\hat{G}}$. Let $N^{i,j}$ denote the number of positions in which M contains an $i\in\{0,1\}$ and Y contains a $j\in\{0,1\}$. Clearly,

$$a=N^{0,1}+N^{1,1}.$$

Thus

$$-N^{1,1}+N^{1,0}=N^{0,1}-a+N^{1,0}. \tag{9}$$

Minimizing the right side in (9) over all $M\in C_{\hat{G}}$ is equivalent to finding a codeword which is the closest to Y. On the other hand, minimizing the left side is equivalent to finding the minimum cut in $\hat{G}_Y$. Q.E.D.

From Lemma 1 we have the following.

Theorem 1: Let $\hat{G}=(\hat{V},\hat{E})$ be a connected graph. Then MLD of a word Y with respect to $C_{\hat{G}}$ is equivalent to finding the maximum of the energy function E of the neural network defined by the graph $\hat{G}_Y$ with all its threshold values equal to zero.

Proof: By Lemma 1 MLD of Y with respect to $C_{\hat{G}}$ is equivalent to finding the minimum cut in $\hat{G}_Y$. By Proposition 2 finding the minimum cut in a graph is equivalent to finding the maximum of the energy function of a neural network defined by a graph with all thresholds being zero. Q.E.D.

Graph theoretic error-correcting codes are limited in the sense that [9]

$$d^*\le\frac{2|\hat{E}|}{|\hat{V}|} \tag{10}$$

where d^* is the minimum distance of the code. For example, a [7,4] Hamming code is not a graph theoretic code because it has minimum distance 3, and $14/5<3$. Hence an interesting question is whether the equivalence stated by Theorem 1 can be generalized to all linear block codes. The energy function associated with the MLD of graph theoretic codes is quadratic. The energy function associated with the MLD of a general linear block code turns out to be a polynomial over the n-cube. The discussion regarding the generalization is the subject of Section IV.

IV. Error-Correcting Codes and Energy Functions

In this section we will extend the results of Section III and show that the MLD problem of linear block codes is equivalent to the maximization of polynomials over the binary n-cube. This result is generalized to nonbinary codes in Appendix I and to nonlinear codes in Appendix II.

Consider a binary linear $[n,k]$ error-correcting block code, to be denoted by $\mathscr{C}$ [16], [18]. The code $\mathscr{C}$ is defined by a $k\times n$ generator matrix G. An information vector $b=(b_1,b_2,\cdots,b_k)$ is encoded into the codeword $v=(v_1,v_2,\cdots,v_n)$ by

$$v_j=\bigoplus_{i=1}^{k}b_ig_{i,j},\qquad 1\le j\le n$$

where $\oplus$ denotes exclusive OR.

The key idea in the derivation is to represent the symbols of the additive group Z_2 as symbols in the multiplicative group $\{1,-1\}$ through the transformation

$$a\to(-1)^a$$

that is,

$$0\to1,\qquad 1\to-1.$$

We will use a different notation for the $\{1,-1\}$ representation. The information vector $b=(b_1,b_2,\cdots,b_k)$ is represented by $x=(x_1,x_2,\cdots,x_k)$, where $x_i=(-1)^{b_i}$, and the encoded codeword $v=(v_1,v_2,\cdots,v_n)$ is represented by $y=(y_1,y_2,\cdots,y_n)$. Hence

$$y_j=(-1)^{v_j}=(-1)^{\oplus_{i=1}^{k}b_ig_{ij}}=\prod_{i=1}^{k}(-1)^{b_ig_{ij}}=\prod_{i=1}^{k}x_i^{g_{ij}}. \tag{11}$$

Example: Consider the [7,4] systematic Hamming code whose generator matrix is given by

$$G=\begin{pmatrix}1&0&0&0&0&1&1\\0&1&0&0&1&0&1\\0&0&1&0&1&1&0\\0&0&0&1&1&1&1\end{pmatrix}.$$

Given the four information symbols (b_1,b_2,b_3,b_4), the corresponding codeword is

$$v=(b_1,b_2,b_3,b_4,b_2\oplus b_3\oplus b_4,b_1\oplus b_3\oplus b_4,b_1\oplus b_2\oplus b_4).$$

In the $\{1,-1\}$ representation, this looks like

$$y=(x_1,x_2,x_3,x_4,x_2x_3x_4,x_1x_3x_4,x_1x_2x_4)$$

where $x_j=(-1)^{b_j}$.

Definition: In the $\{1,-1\}$ representation of a code instead of a generator matrix, given an information vector $x=(x_1,x_2,\cdots,x_k)$, we will use an *encoding procedure* $x\to y$, where $y=(y_1,y_2,\cdots,y_n)$ and every y_j is a monomial that consists of a subset of the x_i. An encoding procedure is *systematic* if and only if $y_j=x_j$ for $1\le j\le k$.

In the example, the [7,4] Hamming code is described by the systematic encoding procedure

$$(x_1,x_2,x_3,x_4)\rightarrow(x_1,x_2,x_3,x_4,x_2x_3x_4,\\ x_1x_3x_4,x_1x_2x_4). \quad (12)$$

Another example, the first-order (shortened) Reed–Muller code $R(1,3)$ [16], is described by the systematic encoding procedure

$$(x_1,x_2,x_3)\rightarrow(x_1,x_2,x_3,x_1x_2,x_1x_3,x_2x_3,x_1x_2x_3)$$

while the first-order Reed–Muller code $R(1,3)$ is described by the encoding procedure

$$(x_0,x_1,x_2,x_3)\rightarrow(x_0,x_0x_1,x_0x_2,x_0x_1x_2,x_0x_3,\\ x_0x_1x_3,x_0x_2x_3,x_0x_1x_2x_3). \quad (13)$$

The generalization to any $R(1,m)$ first-order Reed–Muller code is obvious.

Definition: Let G be a $k\times n$ matrix of ones and zeros. The *polynomial representation* of G with respect to a vector $\omega\in\{1,-1\}^n$, denoted by E_ω, is

$$E_\omega(x)=\sum_{j=1}^{n}\omega_j\prod_{i=1}^{k}x_i^{g_{i,j}}. \quad (14)$$

Consider the linear block code defined by the generator matrix G (or, equivalently, by the encoding procedure associated with G). The polynomial representation of G, i.e., $E_\omega(x)$, will be called the energy function of ω with respect to the encoding procedure $x\rightarrow y$. Note that $E_\omega(x)=\omega\cdot y(x)$, where the centered dot denotes inner product. To establish the connection between energy functions and linear block codes, we will prove that finding the global maximum of $E_\omega(x)$ is equivalent to MLD of a vector ω with respect to the code C.

Theorem 2: Given an $[n,k]$ code C defined by an encoding procedure $x\rightarrow y$, and a vector $\omega\in\{1,-1\}^n$, the closest codeword (in Hamming distance) to ω in C corresponds to an information vector $b=(b_1,b_2,\cdots,b_k)$ if and only if

$$E_\omega(b)=\max_{x\in\{1,-1\}^k}E_\omega(x).$$

Proof: Notice that, for any information vector $x\in\{1,-1\}^k$,

$$\begin{aligned}E_\omega(x)&=\sum_{j=1}^{n}\omega_jy_j(x)\\ &=|\{j:\ \omega_j=y_j(x)\}|-|\{j:\ \omega_j\neq y_j(x)\}|\\ &=n-2|\{j:\ \omega_j\neq y_j(x)\}|\\ &=n-2d_H(\omega,y)\end{aligned}$$

where d_H denotes Hamming distance. This expression implies that $E_\omega(b_1,b_2,\cdots,b_k)$ will achieve a maximum if and only if $d_H(\omega,y)$ achieves a minimum. Q.E.D.

Example: Consider the [7,4] Hamming code, defined by the encoding procedure in (12). Assume that we want to perform MLD of the received word

$$\omega=(1,-1,-1,1,1,-1,1).$$

Then,

$$E_\omega(x_1,x_2,x_3,x_4)=x_1-x_2-x_3+x_4+x_2x_3x_4\\ -x_1x_3x_4+x_1x_2x_4.$$

The maximum of this polynomial occurs at $E_\omega(1,-1,-1,1)=5$. Thus, the received word is decoded as $(1,-1,-1,1)$.

Example: Consider the $R(1,3)$ first-order Reed–Muller code, defined by the encoding procedure in (13). Given the received word $\omega=(\omega_0,\omega_1,\cdots,\omega_7)$, the energy function is

$$\begin{aligned}E_\omega(x_0,x_1,x_2,x_3)&=x_0(\omega_0+\omega_1x_1+\omega_2x_2+\omega_3x_1x_2+\omega_4x_3\\ &\quad+\omega_5x_1x_3+\omega_6x_2x_3+\omega_7x_1x_2x_3)\\ &=x_0(\omega_0+E_\omega(x_1,x_2,x_3))\end{aligned}$$

where

$$E_\omega(x_1,x_2,x_3)=\omega_1x_1+\omega_2x_2+\omega_3x_1x_2+\omega_4x_3+\omega_5x_1x_3\\ +\omega_6x_2x_3+\omega_7x_1x_2x_3.$$

Hence it is enough to find

$$\max_{x_1,x_2,x_3\in\{1,-1\}}|E_\omega(x_1,x_2,x_3)|.$$

If the energy that corresponds to the maximum is positive, then $x_0=1$; otherwise, $x_0=-1$. Assume that we receive $\omega=(-1,1,1,-1,1,1,1,-1)$. We have,

$$E_\omega(x_1,x_2,x_3)=x_1+x_2-x_1x_2+x_3+x_1x_3\\ +x_2x_3-x_1x_2x_3;$$

then

$$\max_{x_1,x_2,x_3\in\{1,-1\}}|E_\omega(x_1,x_2,x_3)|=E_\omega(1,1,1)=3.$$

Since the energy is positive, the received word is decoded as $(1,1,1,1)$. In this case the decoding is not unique, since the maximum is achieved at more than one point.

Given an encoding procedure, we can use the same argument as in Theorem 2 to express the minimum distance of the code. Consider the encoding procedure

$$x=(x_1,x_2,\cdots,x_k)\rightarrow y=(y_1,y_2,\cdots,y_n)$$

and the energy function with $\omega=(1,1,\cdots,1)$

$$E(x_1,x_2,\cdots,x_k)=y_1+y_2+\cdots+y_n.$$

As before,

$$E(x_1,x_2,\cdots,x_k)=n-2d_H((11\cdots1),(y_1,y_2,\cdots,y_n))$$

and

$$\min_{(x_1,x_2,\cdots,x_k)\neq(11\cdots1)}d_H((11\cdots1),(y_1,y_2,\cdots,y_n))$$

occurs at

$$M\overset{\text{def}}{=}\max_{(x_1,x_2,\cdots,x_k)\neq(11\cdots1)}E(x_1,x_2,\cdots,x_k).$$

The conclusion is therefore that d^* (the minimum distance of the code) is given by

$$d^* = \frac{n-M}{2}. \qquad (15)$$

Example: For the [7,4] Hamming code,

$$M = \max_{(x_1,x_2,x_3,x_4)\neq(1111)} x_1+x_2+x_3+x_4+x_2x_3x_4 + x_1x_3x_4+x_1x_2x_4 = 1.$$

Thus

$$d^* = \frac{n-M}{2} = \frac{7-1}{2} = 3.$$

Example: For the $R(1,3)$ first-order Reed–Muller code,

$$M = \max_{(x_0,x_1,x_2,x_3)\neq(1111)} x_0(1+x_1+x_2+x_1x_2+x_3+x_1x_3 + x_2x_3+x_1x_2x_3)$$

which can be written as

$$M = \max_{(x_0,x_1,x_2,x_3)\neq(1111)} x_0(1+x_1)(1+x_2)(1+x_3). \qquad (16)$$

The maximum in (16) is $M=0$, because at least one of the x_i (for $i>1$) must be equal to -1. Thus

$$d^* = \frac{n-M}{2} = \frac{8}{2} = 4.$$

The same argument can be used for any $R(1,m)$ code, giving

$$d^* = \frac{2^m}{2} = 2^{m-1}.$$

V. Representing Linear Codes as Stable States of Energy Functions

Let $\mathscr{C}$ be a linear block code (over GF(2)) defined by the generator matrix G. Let E_C be a polynomial over $\{1,-1\}$ (energy function) with the property that every local maximum in E_C corresponds to a codeword in $\mathscr{C}$ and every codeword in $\mathscr{C}$ corresponds to a local maximum in E_C.

Consider the following question: given a code $\mathscr{C}$ defined by G, is there an efficient algorithm to construct E_C? This section describes the development of such an algorithm.

Consider the $[n,k]$ linear block code $\mathscr{C}$. Without loss of generality, let us assume that the generator matrix G is given in a systematic form; that is,

$$G = [I_k : P] \qquad (17)$$

where I_k is a $k\times k$ identity matrix, and P is a $k\times(n-k)$ matrix. The parity check matrix of $\mathscr{C}$ is

$$H^T = \begin{bmatrix} P \\ \cdots \\ I_{n-k} \end{bmatrix}. \qquad (18)$$

By the definition of H, for all $v\in\mathscr{C}$,

$$vH^T = \mathbf{0} \qquad (19)$$

where $\mathbf{0}$ in (19) is an all zero vector of length $(n-k)$. Equation (19) can be written using the polynomial representation devised in (14), with the vector of coefficients being the all-ones vector.

Lemma 2: Let $\tilde{E}(x)$ be the polynomial representation of H^T with respect to the all-ones vector. Then $x\in\mathscr{C}$ if and only if $\tilde{E}(x) = n-k$.

Proof: $\tilde{E}$ has $(n-k)$ terms, and all the coefficients are equal to 1. Hence $\tilde{E} = n-k$ if and only if all the terms are equal to 1. Q.E.D.

The lemma ensures that in the polynomial $\tilde{E}$ every codeword corresponds to a global maximum (stable state). Does every local maximum correspond to a codeword?

Theorem 3: Let $\mathscr{C}$ be a linear block code, with G, H, E_C, and $\tilde{E}$ as defined. Then $\tilde{E}$ is a polynomial with the properties of E_C. That is, x corresponds to a local maximum in $\tilde{E}$ if and only if $x\in\mathscr{C}$.

Proof: From Lemma 2, the global maximum of $\tilde{E}$ is $n-k$; thus every codeword is a global (and a local) maximum. The converse follows from the fact that H has a systematic form. Specifically, the last $n-k$ variables in $\tilde{E}$, that is, $x_{k+1},\cdots,x_n$, each appear only in one term. That is, x_{k+1} appears only in the first term, x_{k+2} appears only in the second term, and so on. Assume that a vector V exists that corresponds to a local maximum (which is not global). That is, $\tilde{E}(V) = L$, where $L < n-k$. Hence at least one term exists in $\tilde{E}(V)$ that is not 1. However, this term can be made 1 by flipping the value of the variable that appears only in this term. This contradicts the fact that V is a local maximum. Q.E.D.

Examples:

1) Consider the single parity check code; it is an $[n,n-1]$ code and

$$G = [I_{n-1} : 1_{n-1}]$$
$$H^T = 1_n$$

where 1_n is the all-ones vector of length n. Hence

$$\tilde{E}(x) = x_1x_2\cdots x_n.$$

Clearly $\tilde{E}(x)=1$ if and only if $x\in\mathscr{C}$. Also, $\tilde{E}(x)=-1$ for all $x\notin\mathscr{C}$. Thus local maxima in $\tilde{E}$ have a one-to-one correspondence with codewords in $\mathscr{C}$.

2) Consider the simple repetition code; it is an $[n,1]$ code and

$$G = [1,1,\cdots,1]$$
$$H = [1_{n-1} : I_{n-1}].$$

Then

$$\tilde{E}(x) = x_1(x_2+x_3+\cdots+x_n).$$

Clearly two stable states exist in $\tilde{E}$, the all-ones and the all-(-1's) vectors.

3) Consider the [7,4] Hamming code (see also Section IV),

$$H^T = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{20}$$

$$\tilde{E}(x) = x_1x_2x_4x_5 + x_1x_3x_4x_6 + x_2x_3x_4x_7. \tag{21}$$

Again, the polynomial in (21) has the [7,4] Hamming code as the set of its local maxima.

To summarize, given a linear code $\mathscr{C}$, the algorithm for constructing a polynomial E_C is as follows.

1) Construct the systematic generator matrix of $\mathscr{C}$ by performing row operations on the generator matrix G.
2) Construct the systematic parity check matrix of $\mathscr{C}$, in accordance with (18).
3) Construct $\tilde{E}$, which is the polynomial representation of H^T with respect to the all-ones vector. By Theorem 3, let $E_C = \tilde{E}$.

A few remarks and generalizations regarding the above development are now in order.

1) The construction just described also works for cosets of linear codes. Let ω be the vector of length $n-k$ of the coefficients of $\tilde{E}$. In the foregoing construction we chose ω to be the all-ones vector and concluded that $E_C = \tilde{E}$. Let $\hat{\mathscr{C}}$ be a coset of $\mathscr{C}$, and let S be the syndrome which corresponds to $\hat{\mathscr{C}}$. It can be proven (basically as in the proof of Theorem 3) that a one-to-one correspondence exists between local maxima of the polynomial representation of H^T with $\omega = S$ and the vectors in the coset $\hat{\mathscr{C}}$. Clearly, the syndrome that corresponds to the code $\mathscr{C}$ is the all-ones vector (remembering that, in the transformation in Section IV, 0 goes to 1).

2) The foregoing construction (with respect to the one suggested in Section IV) is a dual way of defining the MLD problem. Consider a linear block code $\mathscr{C}$ defined by its parity check matrix H. Given a vector V, the MLD problem can be defined as finding the local maximum in E_C closest to V or, equivalently, finding a local maximum of the energy function associated with the syndrome (corresponding to V) that is achieved by a vector of minimum weight.

VI. Solving 0–1 Nonlinear Programming Problems Using Decoding Techniques

An unconstrained *nonlinear* 0–1 *program* [12] is a problem of the form

$$\max\left(\sum_{i=1}^{n} w_i \prod_{j\in S_i} x_j\right) \tag{22}$$

where S_i is a subset of $\{1,\cdots,n\}$, and $x_j \in \{0,1\}$. Basically, the problem in (22) is a problem of finding a maximum of a 0–1 polynomial. A special case of (22) is the quadratic polynomial over $\{1,-1\}$ that was presented in Section II. Clearly, every polynomial over $\{1,-1\}$ can be transformed to an equivalent one over $\{0,1\}$ by a change of variable as discussed in Appendix II. The maximization of a quadratic polynomial over $\{0,1\}$ is known to be NP-hard [7]. One way to prove it is by showing that the maximum cut in a graph problem can be reduced to it. The reduction is based on the same technique used in Section II to show the equivalence between quadratic energy functions and cuts in a graph.

The problem in (22) was studied extensively in [10], [12]. The main effort concentrated in identifying special cases which are solvable in polynomial time [13] and in devising approximation techniques [11]. The most common technique for solving unconstrained 0–1 programs is by transforming them to the problem of finding the maximum weight independent set in a graph [2], [20]. Finding the maximum weight independent set in a graph is NP-hard, but some solvable cases exist. For example, the problem is solvable in polynomial time (by "min cut–max flow" techniques) if the graph is bipartite [2]. A known class of problems, like (22), that are solvable in polynomial time includes those problems that correspond to finding the maximum weight independent set in a bipartite graph.

Definition: Let $\hat{G} = (\hat{V}, \hat{E})$ be a graph; S is an *independent set* of nodes in the graph if and only if $S \subseteq \hat{V}$ and no two nodes of S are connected by an edge. Suppose every node in $\hat{V}$ is assigned a positive integer called the *weight* of a node. The problem of finding an independent set of nodes such that the sum of its weights is maximal over all possible independent sets is known as the *maximum weight independent set* problem.

The problem in (22) is transformed to the problem of finding the maximum weight independent set by using the concept of a *conflict graph* of a 0–1 polynomial [2], [20]. The idea is illustrated in the following example.

Example: Consider the following 0–1 polynomial:

$$f(X) = -2x_1 - 2x_2 + 5x_1x_2 - 4x_1x_2x_3. \tag{23}$$

One can show that $f(X)$ can be transformed to an equivalent polynomial so that all the terms (except the constant one) have positive coefficients. The new polynomial involves both the variables and their complements. This is done by noticing that

$$x = 1 - \bar{x}.$$

Hence

$$f(X) = -4 + 2\bar{x}_1 + 2\bar{x}_2 + x_1x_2 + 4x_1x_2\bar{x}_3. \tag{24}$$

Clearly, the maximization of $f(X)$ is equivalent to the maximization of $f(X)$ without the constant term; thus the constant term can be neglected.

The conflict graph $\hat{G}(f)$ associated with a polynomial $f(X)$ has a node set which corresponds to the terms of $f(X)$, each node to a term (but the constant term). Two nodes in $\hat{G}(f)$ are connected by an edge if and only if one of the corresponding terms contains a variable and the other corresponding term contains the same variable complemented. The weight of a node in $\hat{G}(f)$ is the coefficient of the corresponding term in f. Fig. 3 shows the conflict graph associated with $f(X)$.

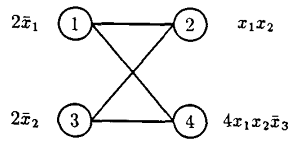

Fig. 3. Conflict graph associated with $f(x)$.

The maximum weight independent set of $\hat{G}(f)$ is $\{2,4\}$; that is, the nodes that correspond to x_1x_2 and to $x_1x_2\bar{x}_3$. The weight of the set is five, the assignment which achieves the maximum corresponds to $x_1=1$, $x_2=1$, and $x_3=0$. Thus the maximum of $f(X)$ is $-4+5=1$.

One can prove that this procedure works in the general case as follows.

1) Every maximum weight independent set in $\hat{G}(f)$ corresponds to a maximum in f (and vice versa), with the values of the terms associated with the nodes in the set equal to 1.
2) In general, the problem of finding the maximum weight independent set in a graph is solvable in polynomial time for bipartite graphs (the graph in Fig. 3 is bipartite).
3) The conflict graph associated with a polynomial is not unique, because a term can be made positive by complementing any odd number of its variables.

In the following we will show how decoding techniques can be used to maximize 0–1 nonlinear programs. Consider the 0–1 polynomials associated with Hamming codes (see Section IV). The family of these polynomials will be denoted by HP (Hamming polynomials). It will be shown by an example that HP is not contained in the family of polynomials related to bipartite conflict graphs. Thus HP is not a subset of the family of polynomials whose maximization is known to be easy.

Consider the following polynomial over $\{0,1\}$:

$$\begin{aligned} M(X) = 3 - 6x_1 - 2x_2 - 2x_3 - 6x_4 \\ + 4(x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4) \\ - 8x_2x_3x_4. \end{aligned} \tag{25}$$

The polynomial $M(X)$ is not associated with a bipartite conflict graph (see the following proposition).

Proposition 3: There does not exist $\hat{G}(M)$, a conflict graph associated with $M(X)$, which is bipartite.

Proof: The proof is straightforward; it follows from checking all the possible ways to convert the sign of the cubic term. Q.E.D.

A maximum of a polynomial that belongs to HP can be found by applying the decoding procedure for Hamming codes. Also, an efficient method exists for recognizing whether or not a given polynomial is in HP. We will describe both the recognition procedure and the maximization procedure by continuing with the previous example.

Proposition 4: The polynomial $M(X)$ is a Hamming polynomial.

Proof:

1) Transform $M(X)$ to an equivalent polynomial over $\{1,-1\}$ by a change of variable $x = 0.5(1-u)$:

$$M(U) = -u_2 - u_3 + u_4 + u_1u_2 + u_1u_3 + u_1u_4 + u_2u_3u_4. \tag{26}$$

2) By the derivation in Section IV, M is clearly equivalent to the MLD of

$$\omega = (1,1,0,0,0,0,0)$$

with regard to the code defined by the following generator matrix:

$$G = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}. \tag{27}$$

3) The matrix G can be brought to a systematic form, denoted by $\tilde{G}$, by row operations:

$$\tilde{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{28}$$

From $\tilde{G}$ we obtain $\tilde{H}$, the systematic parity check matrix (see Section V):

$$\tilde{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{29}$$

The polynomial $M(U)$ is a Hamming polynomial because its parity check matrix contains all the possible columns (but the all-zeros column).

4) To decode ω, we will use the syndrome; that is,

$$\omega\tilde{H}^T = (0,1,0).$$

The error is in the location corresponding to the row in $\tilde{H}^T$ that is equal to the syndrome. Hence the result of the decoding is $(1,1,0,0,0,1,0)$. The maximum is attained at $X^* = (1,1,1,0)$, and $M(X^*) = 5$.

By this procedure we proved that M is in HP and found its maximum. Q.E.D.

A few remarks with regard to the foregoing procedure are in order.

1) The procedure in the proof can be applied to a general 0–1 polynomial. Consider the polynomial representation over $\{1,-1\}$ (the one obtained after step 1 above). A necessary condition that a polynomial is in HP is that the absolute values of the coefficients in the $\{1,-1\}$ representation are equal (the constant is neglected).

2) The complexity of the recognition process is determined by the complexity of the transformation from the

$\{0,1\}$ representation to the $\{1,-1\}$ (step 1). This transformation is exponential in the degree of the polynomial over $\{0,1\}$.

By Section IV, maximization of polynomials over $\{1,-1\}$ with coefficients in $\{1,-1\}$ is equivalent to MLD problem of linear block codes. The generalization to polynomials that have integer (or rational) coefficients follows immediately by expressing a term with a coefficient being equal to a (a positive integer) as a identical terms with coefficients equal to one.

To summarize, we have established a technique for solving 0–1 nonlinear programs by decoding techniques. In particular, for the family of Hamming polynomials it was proven that this family of polynomials is not a subset of the family of polynomials associated with bipartite conflict graphs. Both a recognition procedure and a solution procedure were derived.

VII. Conclusion

The following three problems were considered:

1) maximum likelihood decoding of error-correcting codes,
2) finding the global maximum of a neural network,
3) solving 0–1 nonlinear programming problems.

It is well-known that all of these problems are NP-hard [7], i.e., the problems are believed to be intractable. We have shown that these three problems are related to one another in a very natural way and revealed the equivalence between them for the binary as well as for the nonbinary case.

Here are a few implications of the equivalence between the three problems.

1) Solvable cases in one problem can be used to identify new solvable cases in the equivalent problem. In particular we have shown how to use the decoding algorithm for the Hamming code to solve certain 0–1 nonlinear problems that are not solvable by known techniques. In the opposite direction, the design of codes that correspond to solvable cases of 0–1 nonlinear problems and hence have an efficient decoding algorithm, is a subject for future research.

2) The equivalence between problems can be used to prove new results. For example, we have found a simple proof for the minimum distance of Reed–Muller codes by considering the equivalent problem in polynomials.

3) Neural networks is an area that attracts new interest. The equivalence established in the paper reveals their rich graph theoretic structure as well as their algebraic structure. In particular, it enables us to solve the programming problem for neural networks. That is, given an error-correcting code, we can construct a neural network in which every local maximum is a codeword and vice versa. Since the MLD problem is NP-hard, we cannot expect to have the network converge to the closest local maximum/codeword. Finding families of codes for which there exist networks that perform MLD is another interesting research direction.

Acknowledgment

The authors would like to thank Dr. A. Dembo for his valuable comments on the earlier version of this manuscript and the referees for their suggestions.

Appendix I
Generalization to Nonbinary Codes

Consider now a linear $[n,k]$ error-correcting code over a field GF(p) with p a prime. Let G be the generator matrix of the code. Then k symbols in Z_p $(b_1,b_2,\cdots,b_k)$ are encoded into codeword $v=(v_1,v_2,\cdots,v_n)$ by

$$v_j=\sum_{m=1}^{k} b_m g_{mj} (\operatorname{mod} p), \qquad 1\le j\le n.$$

Again, the key idea is to use the multiplicative representation. Let u be the pth root of unity

$$u=e^{2\pi i/p}.$$

The additive group Z_p can be represented as a multiplicative group of p roots of unity through the transformation $a\to u^a$.

In the multiplicative representation, the k information symbols $(b_1,b_2,\cdots,b_k)$ are represented as

$$(x_1,x_2,\cdots,x_k)=(u^{b_1},u^{b_2},\cdots,u^{b_k}).$$

Thus the encoded codeword $v=(v_1,v_2,\cdots,v_n)$ is represented as $y=(y_1,y_2,\cdots,y_n)$ where

$$y_j=u^{v_j}=u^{\sum_{m=1}^{k} b_m g_{mj} (\operatorname{mod} p)}=\prod_{m=1}^{k} u^{b_m g_{mj}}=\prod_{m=1}^{k} x_m^{g_{mj}}.$$

Example: Consider the [4,2] ternary Hamming code whose generator matrix is

$$G=\begin{pmatrix}1&0&2&0\\0&1&2&1\end{pmatrix}.$$

Given the two information symbols (b_1,b_2), the corresponding codeword is

$$v=(b_1,b_2,2b_1+2b_2(\operatorname{mod} 3),b_2).$$

In the multiplicative representation, this becomes

$$y=(x_1,x_2,x_1^2x_2^2,x_2)$$

where $x_j=u^{b_j}$, $u=e^{2\pi i/3}$.

Hence as for the binary case, we can represent a code on a field with p elements (p a prime) by an encoding procedure. The elements are now pth roots of unity. Thus, given k information symbols, we have the 1–1 assignment

$$x=(x_1,x_2,\cdots,x_k)\to y=(y_1,y_2,\cdots,y_n)$$

where $y_j=y_j(x_1,x_2,\cdots,x_k)$ is a monomial.

We will present two generalizations. In the first generalization we consider solving the MLD problem with the metric being the Hamming distance while in the second we consider the Lee distance. The generalization for the case in which the MLD problem is defined using the Hamming distance is based on the following well-known lemma [16].

Lemma 3: Let p be a prime, and let $u=e^{2\pi i/p}$. Assume $k\in\{0,\cdots,(p-1)\}$; then

$$\frac{1}{p}\sum_{m=0}^{p-1} u^{km}=\begin{cases}1, & \text{if } k=0\\0, & \text{otherwise.}\end{cases}$$

The generalization is stated by the following theorem.

Theorem 4: Consider an $[n,k]$ block code over GF(p) with p a prime. Let $x \to y$ be the corresponding encoding procedure. Let E_ω^p be the following:

$$E_\omega^p(y) = \sum_{j=1}^{n} \sum_{m=0}^{p-1} (\bar{\omega}_j y_j)^m.$$

Then the MLD of ω is equivalent to finding the maximum of E_ω^p.

Next we consider MLD with respect to the Lee distance. Here the result is not general for every p as in the previous case. We will show that for $p = 3$ or 5, there are easy expressions of the energy function. We start by redefining the energy function. Given an encoding procedure

$$x = (x_1, x_2, \cdots, x_k) \to y = (y_1, y_2, \cdots, y_n)$$

and $\omega = (\omega_1, \omega_2, \cdots, \omega_n)$ a vector whose entries are pth roots of unity, we define the energy function as follows:

$$E_\omega(x) = \lfloor \mathrm{Re}(\bar{\omega}_1 y_1) \rfloor + \lfloor \mathrm{Re}(\bar{\omega}_2 y_2) \rfloor + \cdots + \lfloor \mathrm{Re}(\bar{\omega}_n y_n) \rfloor \quad (30)$$

where Re(x) denotes real part, $\lfloor x \rfloor$ integer part, and $\bar{x}$ complex conjugate of x.

Notice that this energy function coincides with the one for $p = 2$ (in that case, $u = -1$). Before proceeding further, let us recall the definition of the *Lee distance* [18].

Definition: the *Lee weight* of an n-tuple $(a_1, a_2, \cdots, a_n)$, $a_j \in Z_p$, p a prime, is defined as

$$w_L = \sum_{j=1}^{n} |a_j|$$

where

$$|a_j| = \begin{cases} a_j, & 0 \le a_j \le p/2 \\ p - a, & p/2 < a_j \le p-1. \end{cases}$$

The *Lee distance* between two n-tuples is defined as the Lee weight of their difference.

We study the cases $p = 3$ and $p = 5$. Henceforth, $x \to y$ denotes an encoding procedure that defines a code C, and x and y are vectors of length k and n, respectively, of third or fifth roots of unity.

We are going to prove two theorems. The first one is similar to Theorem 2. It states that MLD in a ternary code is equivalent to the maximization of the energy function in (30). The second theorem states something similar for codes on the fifth roots of unity but with respect to the Lee distance.

Theorem 5: Let $p = 3$, $a \to b$; then b is the closest codeword (in Hamming distance) to a word ω if and only if

$$E_\omega(a) = \max_x E_\omega(x).$$

Proof: The proof is similar to that of Theorem 2. Q.E.D.

Example: Consider again the [4,2] ternary Hamming code. Assume $\omega = (u, u^2, 1, u)$ is received ($u = e^{2\pi i/3}$); then

$$E_\omega(x_1, x_2) = \lfloor \mathrm{Re}(u^2 x_1) \rfloor + \lfloor \mathrm{Re}(u x_2) \rfloor + \lfloor \mathrm{Re}(x_1^2 x_2^2) \rfloor + \lfloor \mathrm{Re}(u^2 x_2) \rfloor.$$

It can be easily verified that $\max E_\omega(x_1, x_2) = E_\omega(u, u^2) = 2$; thus ω is decoded as (u, u^2).

Theorem 6: Let $p = 5$, $a \to b$; then b is the closest codeword (in Lee distance) to a word ω if and only if

$$E_\omega(a) = \max_x E_\omega(x).$$

Proof: Using the definition of the energy function,

$$\begin{aligned} E_\omega(a) &= |\{ j: \bar{\omega}_j b_j = 1\}| - |\{ j: \bar{\omega}_j b_j = u^2 \text{ or } u^3\}| \\ &= n - |\{ j: \bar{\omega}_j b_j = u \text{ or } u^4\}| - 2|\{ j: \bar{\omega}_j b_j = u^2 \text{ or } u^3\}| \\ &= n - d_L(\omega, b) \end{aligned}$$

where d_L denotes Lee distance. Hence $E_\omega(a)$ reaches a maximum if and only if $d_L(\omega, b)$ reaches a minimum. Q.E.D.

Example: Consider the [6,2] code on Z_5 generated by

$$G = \begin{pmatrix} 1 & 2 & 3 & 4 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

The corresponding encoding procedure, taking the symbols as fifth roots of unity, is given by

$$(x_1, x_2) \to (x_1 x_2, x_1^2 x_2, x_1^3 x_2, x_1^4 x_2, x_1, x_2).$$

Assume $\omega = (u^2, u^4, 1, u^3, u, 1)$ is received, where $u = e^{2\pi i/5}$. The energy function is then

$$E_\omega(x_1, x_2) = \lfloor \mathrm{Re}(u^3 x_1 x_2) \rfloor + \lfloor \mathrm{Re}(u x_1^2 x_2) \rfloor + \lfloor \mathrm{Re}(x_1^3 x_2) \rfloor + \lfloor \mathrm{Re}(u^2 x_1^4 x_2 \rfloor + \lfloor \mathrm{Re}(u^4 x_1) \rfloor + \lfloor \mathrm{Re}(x_2) \rfloor.$$

It can be verified that the maximum occurs at $E_\omega(u^2, 1) = 4$; thus ω is decoded as $(u^2, 1)$.

Appendix II
Generalization to Nonlinear Codes

We consider nonlinear block codes and generalize the result in Section IV. The key idea for getting this generalization is to consider the representation of Boolean functions as polynomials over the field of real numbers. Although part of this discussion is known (see, for example, [16], [17]), we include a detailed derivation as we believe that it is novel with regard to the mode of presentation. Let us start with some definitions and notation.

Definition: A Boolean function f on n variables, is a mapping,

$$f: \{0,1\}^n \to \{0,1\}.$$

As in Section IV, it is useful to define Boolean functions using the symbols 1 and -1 instead of the symbols 0 and 1, respectively.

Definition: A *Hadamard matrix* of order m, denoted by H_m, is an $m \times m$ matrix of $+1$'s and -1's such that

$$H_m H_m^T = m I_m \quad (31)$$

where I_m is the $m \times m$ identity matrix. The above definition is equivalent to saying that any two rows of H are orthogonal.

Hadamard matrices of order 2^k exist for all $k \ge 0$. The so-called Sylvester construction is as follows:

$$H_1 = [1]$$

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_{2^{n+1}} = \begin{bmatrix} H_{2^n} & H_{2^n} \\ H_{2^n} & -H_{2^n} \end{bmatrix}. \quad (32)$$

Definition: Given a Boolean function f of order n, P_f is a polynomial (with coefficients over the field of real numbers) equivalent to f if and only if for all $X \in \{1, -1\}^n$:

$$f(X) = P_f(X).$$

Problem: Given a Boolean function f of order n, compute P_f, a polynomial which is equivalent to f.

As an example, let $f = x_1 \oplus x_2$; that is, f is the XOR function of two variables. It is easy to check that in the $\{1,-1\}$ representation, $P_f = x_1 x_2$.

Notice that, for every Boolean function f, the polynomial P_f is linear in each of its variables because $x^2 = 1$ for $x \in \{-1,1\}$. It turns out that every Boolean function has a unique representation as a polynomial. This representation is derived by using the Hadamard matrix, as described by the following theorem.

Theorem 7: Let f be a Boolean function of order n. Let P_f be a polynomial equivalent to f. Let A denote the vector of coefficients of P_f. Let P denote the vector of the 2^n values of P_f (and f). Then

1) the polynomial P_f always exists and is unique;
2) the coefficients of P_f are computed as

$$A = \frac{1}{2^n} H_{2^n} P.$$

Proof: The proof is constructive. The idea is to compute A by solving a system of linear equations. Let us start by computing the coefficients of P_f for f a function of one variable:

$$P_f(x_1) = a_0 + a_1 x_1$$

and

$$P_f(1) = a_0 + a_1$$
$$P_f(-1) = a_0 - a_1.$$

Clearly,

$$P = H_2 A \tag{33}$$

and by (31),

$$A = \frac{1}{2} H_2 P. \tag{34}$$

Claim: The above result can be generalized to n variables as follows:

$$P = H_{2^n} A. \tag{35}$$

The proof is by induction. The case $n = 1$ has been proven. Assume (35) is true for n. Clearly, every polynomial of $n+1$ variables can be written as a combination of two polynomials each of n variables,

$$P_f(x_1,\cdots,x_{n+1}) = P_f^1(x_1,\cdots,x_n) + x_{n+1} P_f^2(x_1,\cdots,x_n). \tag{36}$$

There are two possibilities, either $x_{n+1} = 1$ or $x_{n+1} = -1$. Hence, by the induction hypothesis (35), the system of linear equations for $n+1$ variables becomes

$$P = \begin{bmatrix} H_{2^n} & H_{2^n} \\ H_{2^n} & -H_{2^n} \end{bmatrix} A. \tag{37}$$

From the recursive definition of Hadamard matrices (32),

$$P = H_{2^{n+1}} A. \tag{38}$$

Hadamard matrices are nonsingular; thus, for any given f a unique P_f (defined by the vector of coefficients A) always exists. Q.E.D.

The representation theory developed above holds also if one is interested in the question of finding an equivalent polynomial, over $\{0,1\}$, of a Boolean function. To see this, simply observe that any monomial over $\{1,-1\}$ can be written as a polynomial over $\{0,1\}$ by the change of variable $x = 1 - 2u$, as follows:

$$\prod_{i=1}^{k} x_i = 1 + \sum_{i=1}^{k} (-2)^i \sum_{S_i} \prod_{j \in S_i} u_j \tag{39}$$

with S_i a subset of $\{1,\cdots,k\}$ with i elements.

For example,

$$x_1 x_2 x_3 = 1 - 2(u_1 + u_2 + u_3) + 4(u_1 u_2 + u_1 u_3 + u_2 u_3) - 8 u_1 u_2 u_3.$$

It is also possible to derive the recursive definition of the transformation from a Boolean function to an equivalent polynomial over $\{1,-1\}$ by using the recursive definition of the Hadamard matrix (for more details see [6]).

The representation theory developed above can be used for representing error-correcting codes in a way that generalizes the representation described in Section IV. Consider the linear $[n,k]$ block code $\mathscr{C}$. The code $\mathscr{C}$ can be represented by viewing each coordinate of the code as a Boolean function of k variables. A vector $V \in \mathscr{C}$ if and only if there exists a vector $X \in \{1,-1\}^k$ such that

$$V = (f_1(X), f_2(X), \cdots, f_n(X)).$$

Clearly, the Boolean functions associated with the coordinates of a linear block code are determined by the basis by which the code is represented. For linear block codes, every coordinate f_i corresponds to an XOR operation of some variables (according to the basis of the code). Thus, for every i, the Boolean function f_i can be transformed by the method described in Theorem 7 to an equivalent polynomial over $\{1,-1\}^k$ which consists of one monomial only. By the same argument as in Theorem 2, the MLD of a given word W is equivalent to solving the following maximization problem, with $X \in \{1,-1\}^k$,

$$\max\left(\sum_{i=1}^{n} W_i f_i(X) \right). \tag{40}$$

The MLD problem as defined by (40) holds also for nonlinear codes. For nonlinear codes, a coordinate f_i can consist of more than one monomial. For example, consider the following nonlinear code of four codewords:

$$\mathscr{C} = [(00100),(11111),(10101),(01011)].$$

Then,

$$f_1 = x_1 x_2$$
$$f_2 = x_1$$
$$f_3 = 0.5(-1 - x_1 - x_2 + x_1 x_2)$$
$$f_4 = x_1$$
$$f_5 = 0.5(-1 + x_1 + x_2 + x_1 x_2).$$

From this generalization, it follows that, for both linear and nonlinear codes, the MLD problem is equivalent to a maximization of a polynomial over $\{1,-1\}$. Hence a rather surprising theorem follows.

Theorem 8: The following three problems are equivalent:
1) maximization of polynomials with rational coefficients over the k-cube;
2) the MLD problem of an $[n,k]$ linear block code;

3) the MLD problem of a not necessarily linear block code that consists of 2^k codewords.

References

[1] P. Baldi, Ph.D. dissertation, California Inst. Technol., Pasadena, 1986.

[2] M. L. Balinski, "On a selection problem," *Management Sci.*, vol. 17, no. 3, pp. 230–231, Nov. 1970.

[3] N. Biggs, *Algebraic Graph Theory*. Cambridge, England: Cambridge Univ. Press, 1975.

[4] J. Bruck and J. Sanz, "A study on neural networks," *Int. J. Intelligent Systems*, vol. 3, pp. 59–75, 1988.

[5] J. Bruck and J. W. Goodman, "A generalized convergence theorem for neural networks," *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 1089–1092, Sept. 1988.

[6] J. Bruck and M. Blaum, "Neural networks, error-correcting codes and polynomials over the binary *n*-cube," IBM Almaden Tech. Rep. RJ 6003, 1987.

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability, a Guide to the Theory of NP-Completeness.* San Francisco, CA: Freeman, 1979.

[8] E. Goles, F. Fogelman, and D. Pellegrin, "Decreasing energy functions as a tool for studying thresholds networks," *Disc. Appl. Math.*, vol. 12, pp. 261–277, 1985.

[9] S. L. Hakimi and H. Frank, "Cut-set matrices and linear codes," *IEEE Trans. Inform. Theory*, vol. IT-11, pp. 457–458, July 1965.

[10] P. L. Hammer and S. Rudeanu, *Boolean Methods in Operation Research*. New York: Springer-Verlag, 1968.

[11] P. L. Hammer, P. Hansen, and B. Simeone, "Roof duality, complementation and persistency in quadratic 0-1 optimization," *Math Prog.*, vol. 28, pp. 121–155, 1984.

[12] P. Hansen, "Methods of nonlinear 0-1 programming," *Ann. Discrete Math.*, vol. 5, pp. 53–70, 1979.

[13] P. Hansen and B. Simeone, "Unimodular functions," *Disc. Appl. Math.*, vol. 14, pp. 269–281, 1986.

[14] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci.* (USA) vol. 79, pp. 2554–2558, 1982.

[15] J. J. Hopfield and D. W. Tank, "Neural computations of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

[16] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. New York: North-Holland, 1977.

[17] A. Mukhopadhyay, *Recent Development in Switching Theory*. New York: Academic Press, 1971.

[18] W. W. Peterson and W. J. Weldon, *Error-Correcting Codes*. Cambridge, MA: MIT Press, 1971.

[19] J. C. Picard and H. D. Ratliff, "Minimum cuts and related problems," *Networks*, vol. 5, pp. 357–370, 1974.

[20] J. M. W. Rhys, "A selection problem of shared fixed costs and network flows," *Management Sci.*, vol. 17, no. 3, pp. 200–207, Nov. 1970.