The Hardness of Decoding Linear Codes with Preprocessing

JEHOSHUA BRUCK AND MONI NAOR

Abstract—**The problem of maximum likelihood decoding of linear block codes is known to be hard [3]. It is shown that the problem remains hard even if the code is known in advance, and can be preprocessed for as long as desired in order to devise a decoding algorithm. The hardness is based on the fact that existence of a polynomial time algorithm implies that the polynomial hierarchy collapses. Namely, some linear block codes probably do not have an efficient decoder. The proof is based on results in complexity theory that relate uniform and nonuniform complexity classes.**

I. INTRODUCTION

CONSIDER an $[n,k]$ linear error-correcting block code. The code can be defined by a parity check matrix (basis of the null space) H. When we assume that the code is used on a binary symmetric channel, the maximum likelihood decoding (MLD) of a corrupted word y is equivalent to finding the closest (in Hamming distance sense) codeword to y [12].

The complexity of MLD was investigated by Berlekamp *et al.*, [3], who considered the following minimization problem.

Given an $(n-k)\times n$ parity check matrix H and a binary vector y of length n, find a solution of minimum weight to the equation $Hx = s$, where $s = Hy$. The associated decision problem follows.

A. Maximum Likelihood Decoding (MLD)

Input: A binary matrix H, a binary vector s and a nonnegative integer w.

Question: Is there a vector x of Hamming weight $\leq w$ such that $Hx = s$?

It is proved in [3] that MLD is NP-Complete. Namely, it is unlikely that there is a general MLD algorithm for linear block codes whose time complexity is a polynomial in the size of the input. The result in [3] seems discouraging to anyone who is looking for a general MLD algorithm for linear codes.

However, this formulation might not be the relevant one. Although H could define any linear code, it should not be treated as part of the input, since in practice the code to be used is known in advance. Namely, it can be assumed that there is an unbounded amount of time to preprocess the code and come up with an efficient decoding procedure.

There has been some progress in devising decoding algorithms using preprocessing. An example is the zero-neighbors (ZN) algorithm [11] that uses a precomputed set of codewords (set of zero-neighbors). In every iteration of the ZN algorithm, the distance between the received word and the set of ZN is computed. The algorithm performs at most n iterations of this kind until it converges to the right solution. It is indicated in [11] that this set is in general of exponential size. Hence, it is natural to ask whether for every linear code there exists a small (polynomial size) set of codewords such that the MLD problem can be solved by computing the distance of the received word to this set. We can ask a more general question: does every linear code have an efficient decoder? We can also consider circuits as a nonuniform model for computation and ask: does every linear code have a small circuit for maximum likelihood decoding? If an efficient decoder exists, then with sufficient preprocessing it would be found. To formulate the general question more precisely, consider the following search problem:

Given a binary vector y of length n, find a solution of minimum weight to the equation $Hx = s$, where $s = Hy$. The associated decision problem follows.

B. Maximum Likelihood Decoding with Preprocessing (MLDP)

Input: A binary vector s and a nonnegative integer w.

Question: Is there a vector x of Hamming weight $\leq w$ such that $Hx = s$?

Notice that MLDP is the same as MLD with H not being part of the input. The question is whether there exists a polynomial p such that, for any code defined by H, there exists a circuit of size bounded by $p(|H|)$ that answers MLDP.

The main contribution of this paper is showing that a positive answer to the foregoing question is unlikely to be true, since this would imply that the polynomial time hierarchy of [13], [14] collapses. One of the conclusions of this result is that decoding schemes like the ZN algorithm [11] are likely to be exponential.

Manuscript received October 10, 1988; revised July 12, 1989. The material in this paper was partially presented at the 1990 IEEE Information Theory Symposium, San Diego, CA, January 14–19.

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

IEEE Log Number 8933854.

0018-9448/90/0300-0381$01.00 ©1990 IEEE

The proof uses results from complexity theory that relate uniform and nonuniform complexity classes [10]. We reduce the MLD problem to the problem of finding the minimum cut (MC) in a graph with edge weights $\in\{-1,1\}$ by considering the so called graph-theoretic codes [9]. The result is then proved by showing that, for every size n, there exists a graph that can encode by the weights on its edges *any* simple maximum cut (SMC) problem [8] related to a graph with n nodes. Thus, the MC problem is intractable even when preprocessing is allowed.

The rest of the paper is organized as follows: In Section II we show the reduction of MLDP to a problem in graph theory by using the concept of graph-theoretic codes. In Section III we give some necessary background in complexity theory, and in Section IV we formulate and prove the main result.

II. Graph-Theoretic Codes

The main goal in this section is to formulate the MLD problem in graph-theoretic language. We do so by considering the family of graph-theoretic codes [9] and prove that the MLD problem of these codes can be reduced to the problem of finding the minimum cut in a graph. This result was established in [5]; we present it here for the sake of the completeness of presentation.

Definition: Let $G=(V,E)$ be an undirected graph. Let $V_1\subseteq V$, and let $V_2=V-V_1$. The set of edges each of which is incident at a node in V_1 and at a node in V_2 is called a cut in G. The weight of a cut is the sum of its edge weights. A minimum cut of a graph is a cut with minimum weight.

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

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

Definition: The incident matrix of a graph $G=(V,E)$, to be denoted by D_G, is a $|V|\times|E|$ matrix in which row i is the characteristic vector of the set of edges incident upon node $i\in V$.

The following facts from graph theory [4] are the basis for the definition of the family of graph theoretic codes. (For more details see [9].)

Fact 1: The set of characteristic vectors that correspond to the cuts in a connected graph $G=(V,E)$ forms a linear vector space over $GF(2)$, with dimension $(|V|-1)$. The linear vector space that corresponds to the cuts of a graph G is called the cut space of G.

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

Hence, given a connected graph G, the cut space of the graph is a linear block code of dimension $(|V|-1)$; thus, every graph has an $[|E|,|V|-1]$ code associated with its cuts. The code associated with the cuts of a graph G will be denoted by C_G.

Example: Let G be the graph in Fig. 1; G has five nodes and eight edges. The incident matrix of the graph G is

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

Any four rows of D_G form a basis of the cut space of G. For example, the matrix that consists of the first four rows of D_G is a generator matrix of the error correcting linear block code associated with G. It is easily observed that G does not contain a cut with less than three edges (besides the empty cut); thus, the code C_G has minimum distance 3 and can correct one error.

Given a graph G, we show how to formulate the MLD problem of the code C_G in a graph-theoretic language.

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

$$W_i=(-1)^{y_i}. \tag{3}$$

W_i is the weight associated with edge i in G.

Then MLD of y with respect to C_G is equivalent to finding a minimum cut in G_y.

Proof: Let us assume that y contains a 1's. Let M be an arbitrary codeword in C_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,

$$-N^{1,1}+N^{1,0}=N^{0,1}-a+N^{1,0}. \tag{4}$$

Minimizing the right-hand side in (4) over all $M\in C_G$ is equivalent to finding a codeword that is the closest to y. On the other hand, minimizing the left-hand side is equivalent to finding the minimum cut in G_y. □

Using the previous relation we present another proof that the decision problem MLD is NP-complete by trans-

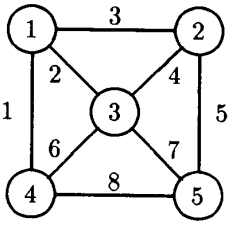

Fig. 1. Graph that corresponds to (8,4) code.

formation from the SMC problem [8]. (This result is proved in [3] by using a different transformation.)

A. Simple Max Cut (SMC)

Input: Graph $G=(V,E)$ and a nonnegative integer w.
Question: Is there a cut in G of weight at least w?
Proposition 1: MLD is NP-complete.

Proof: Observe that the SMC problem in a graph can be solved by MLD of the all-1 vector with respect to the code defined by the graph. □

This result is presented as a warm-up for the real result. Here, we are interested in showing that MLDP is also hard. Again we will show that it is hard for graph-theoretic codes by considering the following related min cut problem.

B. Min Cut with Preprocessing (MCP)

Input: A vector of length $|E|$ of 1's and -1's that defines the weights in a graph $G=(E,V)$ that is known in advance. A nonnegative integer w.
Question: Is there a cut in the graph G with weight no more than w?

Notice that the graph G is known in advance (it is not part of the input), and only the weights of the edges are part of the input. Clearly by Lemma 1, showing that MCP is hard will imply that MLDP is hard.

In Section IV we prove the main result by showing that it is unlikely that there exists a polynomial time algorithm for MCP. Before that, we give in the next section some necessary background in complexity theory.

III. Background from Complexity Theory

A. The Polynomial Time Hierarchy.

In [13], [14] Meyer and Stockmeyer introduced the polynomial-time hierarchy. The formulation we give here is from [6]. An alternating Turing machine is a nondeterministic Turing machine in which the states are labeled with $\wedge$, $\vee$, *accept* and *reject*. The execution of an alternating Turing machine on an input yields a computation tree (of all possible nondeterministic choices). This tree can be interpreted as a formula, where each state is considered as an $\wedge$ or as an $\vee$ according to its label, and a leaf is true or false depending on whether it is an accepting state or a rejecting state. We say that the alternating Turing machine accepts the input if the corresponding formula is true. An alternating Turing machine is a Σ_i machine if, along any path in the computation tree, the number of times the label alternates from an $\vee$ to $\wedge$ is at most i, assuming the initial state is an $\vee$. A language is in the class Σ_i^p if there is a Σ_i machine that accepts the language and whose run time is bounded by a polynomial in the input size. It should be clear from the foregoing definition that $NP=\Sigma_1^p$. The polynomial time hierarchy, PH, is defined to be $\bigcup_{i\geq 1}\Sigma_i^p$. It is a widely held conjecture that all the levels of the polynomial hierarchy are distinct, i.e., $\Sigma_i^p \neq \Sigma_{i+1}^p$. This conjecture is supported by the oracle separation results of [1], [2], [7], [15]. Note that if $\Sigma_i^p=\Sigma_{i+1}^p$ then $\Sigma_i^p=PH$.

Nonuniform Complexity: Turing machines that take advice were introduced by Karp and Lipton [10] in order to study nonuniform complexity. An advice is a function $h: N \mapsto \{0,1\}^*$ where N is the set of natural numbers. An advice is polynomial if $|h(l)|$ is bounded by some polynomial in l. Note that the polynomial refers only to the length of the advice h; there are no restrictions on the complexity of computing h.

Definition: A language L is in P/poly (P with polynomial advice) if there is a polynomial advice h and a polynomial time Turing machine M such that, if M is given an input x followed by $h(|x|)$, M accepts x if and only if $x\in L$.

Note that the advice depends only on the length of x. An important observation is that $L\in P/\text{poly}$ if and only if the size of the circuits that accept L grows polynomially. It is not known whether $NP\subseteq P/\text{poly}$, nor it is known whether containment implies $P=NP$. However, Karp and Lipton [10] were able to prove the following result that relates between uniform and nonuniform complexity classes:

Karp and Lipton (KL) Theorem: If $NP\subseteq P/\text{poly}$ then the Polynomial-Time Hierarchy collapses to the second level, i.e., $PH=\Sigma_2^p$. (Karp and Lipton showed it collapses to the 3rd level, and that was improved by Sipser who showed it collapses to the 2nd level.) Thus, it seems unlikely that polynomial size circuits exist for any NP-Complete problem.

IV. The Main Result

The next theorem states the main result of the paper.

Theorem: If polynomial size circuits for the MLDP problem exist, then the polynomial hierarchy collapses in the early stage: $\Sigma_2^p=PH$.

Proof: Clearly by Lemma 1 it is enough to prove the result for MCP. We show that the existence of polynomial size circuits for MCP implies that SMC is in P/poly (Lemma 2). Since SMC is NP-complete the result follows from the KL Theorem in Section III. □

Lemma 2: The existence of a polynomial size circuit for MCP implies that SMC $\in P/\text{poly}$.

Proof: The key in the proof is to construct for every n a graph $\hat{K}_n$ with the property that every SMC problem in an arbitrary graph with n nodes can be transformed to an instance (weights of the edges) of an MCP problem in $\hat{K}_n$. Hence, for every n, we have a circuit for SMC that under the hypothesis is of polynomial size and the result follows.

We use the complete graph of n nodes K_n and construct the graph $\hat{K}_n$ by adding a node and two edges in parallel to every edge in K_n. For example, Fig. 2 de-

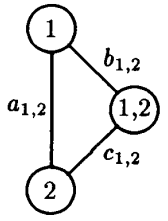

Fig. 2. Graph $\hat{K}_2$.

scribes $\hat{K}_2$. Hence, $\hat{K}_n$ has $n + \binom{n}{2}$ nodes and $3\binom{n}{2}$ edges.

- The set of nodes in $\hat{K}_n$ consists of the n nodes of K_n, labeled v_i, $1 \le i \le n$, and $\binom{n}{2}$ nodes each of which corresponds to an edge (i, j) in K_n and is labeled $v_{i,j}$.
- The set of edges in $\hat{K}_n$ consists of the $\binom{n}{2}$ edges of K_n labeled $a_{i,j}$, $1 \le i < j \le n$, and 2 additional edges for every edge in K_n: edge $b_{i,j}$ connects node v_i with node $v_{i,j}$ and edge $c_{i,j}$ connects node v_j with node $v_{i,j}$.

Now we want to show that every instance of the SMC problem can be transformed to an instance of MCP in $\hat{K}_n$. Let the graph $G = (V, E)$ with n nodes be an instance of the SMC problem. We transform it to an instance of the MCP problem in $\hat{K}_n$ as follows.

- If edge $(i, j) \in E$, then $a_{i,j} = -1$, $b_{i,j} = 1$ and $c_{i,j} = -1$ in $\hat{K}_n$.
- If edge $(i, j) \notin E$, then $a_{i,j} = -1$, $b_{i,j} = 1$ and $c_{i,j} = 1$ in $\hat{K}_n$.

We claim that finding the simple maximum cut in G can be done by finding the minimum cut in the corresponding $\hat{K}_n$. The first direction is to show that there exists a cut in $\hat{K}_n$ with weight -2 times the weight of the max cut in G. We do it by considering the edges of $G = (V, E)$.

- If edge $(i, j) \in E$, then
  1) If edge (i, j) is in the max cut in G we can include both edge $a_{i,j}$ and edge $c_{i,j}$ (and not $b_{i,j}$) in the min cut in $\hat{K}_n$; this adds -2 to the min cut.
  2) If edge (i, j) is not in the max cut in G, we can include both edge $b_{i,j}$ and edge $c_{i,j}$ (and not $a_{i,j}$) in the min cut in $\hat{K}_n$; this adds 0 to the min cut.
  In either case we get that the weight of the cut in $\hat{K}_n$ is -2 times the weight of the max cut in G.
- If edge $(i, j) \notin E$, then
  (1) If nodes v_i and v_j are on the same side of the max cut in G, we put node $v_{i,j}$ also to be on the same side of the cut in $\hat{K}_n$.
  (2) If nodes v_i and v_j are on opposite sides of the max cut in G, we put node $v_{i,j}$ on any of the sides in $\hat{K}_n$.
  In either case we get that there is no contribution to the weight of the cut in $\hat{K}_n$.

The other direction is to show that there is a cut in G with weight -0.5 times the weight of the minimum cut in the corresponding $\hat{K}_n$. The proof is similar to the first direction and is done again by considering the edges of $\hat{K}_n$. □

To summarize, we describe the idea in the proof of the main result. We have shown that for a particular family of linear block codes (graph theoretic codes obtained from $\hat{K}_n$): the problem of maximum likelihood decoding with preprocessing (MLDP) is hard (assuming the polynomial hierarchy does not collapse). If we could solve MLDP in $\hat{K}_n$ with a nonuniform (depending on n) family of polynomial size circuits, then we could solve minimum cut with preprocessing (MCP) on the same graph—$\hat{K}_n$—with the input being a vector of weights from $\{1, -1\}$, again using a nonuniform family of polynomial size circuits. Then we could solve simple max cut (SMC) on arbitrary n node graphs using a family of polynomial size circuits that depend only on n. Since the SMC problem is NP-complete, the Karp–Lipton theorem would imply the collapse of the polynomial-time hierarchy, which is considered doubtful by complexity theorists.

V. Conclusion

We proved that some linear block codes do not have a polynomial time algorithm for MLD (the existence of one implies that the polynomial hierarchy collapses) even when we allow preprocessing. Namely, knowledge of the code does not help in general in designing an efficient decoder simply because there exist codes that probably do not have an efficient decoder.

We know that some families of codes do have an efficient MLD algorithm in the uniform sense. Namely, the algorithm is the same for all codes in the family. On the other hand, the result in [3] indicates that there is no general MLD efficient algorithm for all linear block codes (unless $P = NP$). It is interesting to consider an intermediate case, namely, a family of codes that does not have a uniform decoding algorithm but has a nonuniform decoding algorithm. An open problem is to exhibit (or prove the existence of) such a family of codes.

Acknowledgment

The authors thank Dr. Mario Blaum for useful discussions and the Associate Editor, Dr. Don Coppersmith, for his comments.

References

[1] T. Baker, J. Gill, and R. Solovay, "Relativization of the $P = ?NP$ question," *Siam J. on Computing*, vol. 4, pp. 431–442, 1975.
[2] T. Baker and A. Selman, "A second step toward the polynomial hierarchy," *Theoretical Comput. Sci.*, vol. 8, pp. 177–187, 1979.
[3] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 384–386, May 1978.
[4] N. Biggs, *Algebraic Graph Theory*. New York: Cambridge, 1975.
[5] J. Bruck and M. Blaum, "Neural networks, error-correcting codes and polynomials over the binary n-cube," *IEEE Trans. Inform. Theory*, vol. 35, no. 5, pp. 976–987, Sept. 1989.
[6] A. Chandra, D. Kozen, and L. Stockmeyer, "Alternation," *Journal of the Association for Computing Machinery*, vol. 28, pp. 114–133, 1981.

[7] M. Furst, J. Saxe, and M. Sipser, "Parity circuit and the polynomial-time hierarchy," *Mathematical Systems Theory*, vol. 17, pp. 13–27, 1984.
[8] M. R. Garey and D. S. Johnson, *Computers and Intractability, a Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.
[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] R. M. Karp and R. J. Lipton, "Some connections between nonuniform and uniform complexity classes," in *Proc. 12th ACM Symp. on Theory of Computing*, 1980, pp. 302–309.
[11] L. B. Levitin and C. R. P. Hartmann, "A new approach to the general minimum distance decoding problem: The zero-neighbors algorithm," *IEEE Trans. Inform. Theory*, vol. IT-31, pp. 378–384, May 1985.
[12] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. New York: North-Holland, 1977.
[13] A. R. Meyer and L. J. Stockmeyer, "The equivalence problem for regular expression with squaring requires exponential space," in *Proc. 13th IEEE Symp. on Switching and Automata Theory*, 1972, pp. 25–29.
[14] L. J. Stockmeyer, "The polynomial-time hierarchy," *Theoretical Computer Science*, vol. 3, pp. 1–22, 1977.
[15] A. C. Yao, "Separating the polynomial-time hierarchy by oracles," in *Proc. 26th IEEE Symp. Foundation Comput. Sci.*, 1985, pp. 1–10.