On the Inherent Intractability of Finding Good Codes

R. J. McEliece and H. C. A. van Tilborg[1]
Communications Systems Research Section

In this article we will show that the problem of computing the minimum distance of an arbitrary binary linear code is NP *complete. This strongly suggests, but does not imply, that it is impossible to design a computer algorithm for computing the minimum distance of an arbitrary code whose running time is bounded by a polynomial in the number of inputs.*

I. Introduction

One of the recurring problems in the design of coded communication systems for the DSN or other applications is the search for the best code for the job. For reasons of economy and simplicity, the search is usually restricted to the class of binary linear codes, a linear code being one that is defined as the rowspace of a certain binary matrix G called the code's *generator matrix*. If the matrix G has size $k \times n$ and row-rank k, the code is called an (n, k) *linear code*; its transmission rate is k/n bits per symbol. Among all linear codes C with a fixed n and k there is a wide range of performance possible; the best such code (on a memoryless channel) is, however, the one with the largest *minimum distance* d_{min}. Here

$$d_{min} = min\{d_H(\mathbf{x}, \mathbf{y}): x, y \in C, \mathbf{x} \neq y\}$$

where $d_H(\mathbf{x},\mathbf{y})$ is the Hamming distance between the distinct code vectors $\mathbf{x}$ and $\mathbf{y}$, i.e. the number of components in which $\mathbf{x}$ and $\mathbf{y}$ differ. Since the code is linear, d_{min} is equal to the *minimum* nonzero *weight* of the code w_{min}:

$$w_{min} = min\{w_H(\mathbf{x}): x \in C, x \neq 0\}$$

where $w_H(\mathbf{x})$ is the Hamming weight of $\mathbf{x}$, i.e. the number of nonzero components of $\mathbf{x}$.

Now given an arbitrary binary $k \times n$ matrix G, to find w_{min} for the corresponding code it is apparently necessary to compute each of the $2^k - 1$ nonzero linear combinations of the rows. This procedure is not feasible even by computer unless k is relatively small; we would like to find a more efficient procedure, if possible. However, in the remaining sections of this paper we will show that it is quite unlikely that such a procedure exists.

In the next section we shall describe a recently-developed technique that can sometimes be used to show the inherent intractability of a specific problem. In Section III we shall apply this technique to the w_{min} problem.

[1] Caltech visiting Assistant Professor of Mathematics.

II. *NP*-Complete Problems

The following discussion will describe in heuristic language a set of results that can be made quite precise. The interested reader should consult Ref. 1, Chapter 10, or Ref. 2 for details.

The class P is defined to be the set of computational problems that can be solved by an algorithm that is guaranteed to terminate in a number of steps bounded by a polynomial in the length of the input. (P is sometimes called the class of *polynomial-time* algorithms.) Problems in the class P are generally regarded to be tractable; conversely those not in P are considered intractable. The class P includes such problems as solving linear equations, finding the minimum cut in a flowgraph, certain scheduling problems, etc.

The class NP is defined to be the set of computational problems that can be solved by a backtrack-search-type algorithm, where the depth of search is bounded by a polynomial in the length of the input. Alternatively, NP is the set of problems solvable by a nondeterministic algorithm whose running time is bounded by a polynomial in the length of the input. A nondeterministic algorithm is one that when confronted with a choice between two alternatives, can create two copies of itself and simultaneously follow up the consequences of both courses. This repeated splitting may lead to an exponentially growing number of copies; the algorithm is said to solve the given problem if any of these copies produces the correct answer. For this reason the class NP is often called the class of *nondeterministic polynomial-time* algorithms.

The class NP is quite extensive; it contains such problems as the traveling salesman's problem, the 0-1 integer programming problem, the graph characteristic number problem, and many decoding problems.

The class NP clearly contains the class P as a subclass: $NP \supseteq P$. It is conversely intuitively evident that NP is "much larger" than P; however, no one has yet succeeded in proving this and the query $NP \neq P$? is currently one of the central problems of computer science. However, recently a circle of results has been developed that strongly suggests, but does not rigorously imply, that $NP \neq P$. We now describe these results.

In 1971 Cook (see Ref. 1, Theorem 10.3) proved that a certain problem in NP (called the *satisfiability* problem) has the following curious property. Any problem (p) in NP can be reduced to the satisfiability problem, in the sense that if a polynomial-time algorithm could be found for the satisfiability problem, then that algorithm could be modified to yield a polynomial-time algorithm for problem (p). Thus while it is possible that satisfiability *might* possess a polynomial-time algorithm, if it does, so would the traveling salesman's problem, integer programming, etc., indeed any problem in NP. But researchers have worked on these NP problems for many years without ever finding a polynomial-time algorithm for *any* of them. This is strong evidence that satisfiability does not possess a polynomial-time algorithm and that $NP \neq P$.

In a later paper Karp (Ref. 2) reversed things and showed that the satisfiability problem can itself be reduced in the sense described above to many other (21, to be exact) NP problems. Thus if any of these NP problems possesses a polynomial time algorithm, then so does every problem, and hence $NP = P$. NP problems with this property are now called *NP-complete* problems. At this writing there are dozens of problems known to be NP complete; and if the "obvious" assertion $NP \neq P$ is true, then no NP-complete problem can have a polynomial-time algorithm.

In the next section we shall show that certain problems related to assessing the performance of a binary linear code are NP complete. We shall do this by reducing a known NP-complete problem to the problems that interest us.

III. The *NP* completeness of Finding w_{min}

In this section we will show that the problem of finding w_{min} for a linear code from its generator matrix is NP complete. However, we must first state our problem in a suitable form. Here is our formally stated problem:

$$(G_k, w)$$

INPUT: A binary $k \times n$ matrix G_k of rank k, and a positive integer w.

PROPERTY: There is a linear combination of rows of G with Hamming weight w.

(In this subject, the problems are always stated in this way, since the theory can deal only directly with problems possessing a "yes" or "no" answer. The input must be encoded into a binary string of length N, say, and fed into a computing device, which outputs either 0 (yes) or 1 (no) after a certain length of time. If there is such a device for which the time is bounded by a polynomial function of N, the problem is in P; otherwise not.)

Although we have so far been describing codes by their generator matrices, it is also possible to use the *parity-check* matrices. If the code C has G_k as a generator matrix, then it will possess a $(n - k) \times n$ parity-check matrix H_{n-k}, where $\mathbf{c} \in C$ iff $H\mathbf{c}^T = \mathbf{0}$. Since one can compute H_{n-k} from G_k (and

vice versa) in polynomial time via elementary row operations, it follows that the problem (G_k, w) is *NP* complete iff the following problem is:

$$(H_{n-k}, w)$$

INPUT: A binary $(n - k) \times n$ matrix of rank k and a positive integer w.

PROPERTY: There is a vector $\mathbf{c}$ of weight w such that $H_{n-k}\mathbf{c}^T = 0$.

We show now that (G_k, w) (and hence also (H_{n-k}, w)) is *NP* complete by successively showing that the following problems are *NP* complete.

PARTITION INTO TRIANGLES

INPUT: The incidence matrix of an undirected graph Γ.

PROPERTY: The vertices of Γ can be covered by disjoint triangles.

$$(G, \ell, w)$$

INPUT: A binary matrix G and integers ℓ, w.

PROPERTY: There exists a linear combination of ℓ rows of G having weight w.

$$(G, w)$$

INPUT: A binary matrix G and an integer w.

PROPERTY: There exists a linear combination of rows of G having weight w.

$$(G_k, \ell, w)$$

INPUT: A binary $k \times n$ matrix G of rank k and integers ℓ, w.

PROPERTY: There exists a linear combination of ℓ rows of G_k having weight w.

It is easily verified that each of the 6 problems listed above is in *NP*. Furthermore, the problem PARTITION INTO TRIANGLES is known to be *NP* complete (Ref. 3).

We shall now produce a sequence of reductions (see Fig. 1) that will show that each of the other five is also *NP* complete.

A reduction from problem A to problem B will be denoted by the symbol $A \propto B$. Such a reduction will always be of the following general form: we show that problem A can be "encoded" in polynomial time into problem B in such a way that a solution to the B problem immediately yields a solution to the corresponding A problem. In the cases that the input variables of problems A and B are denoted by the same letters, we shall add a prime to the input symbols for problem B.

PARTITION INTO TRIANGLES $\propto (G, \ell, w)$

Let the columns of G correspond to the n points of Γ; the rows to the triangles of Γ. (Thus each row of G has exactly three nonzero entries.) Let $\ell = n/3$ and $w = n$. Then (G, ℓ, w) has an affirmative answer iff TRIANGLE does.

$$(G, \ell, w) \propto (G', w')$$

If G has n columns, let G' be the matrix formed by adjoining $n + 1$ copies of the identity matrix I to G, i.e. $G' = (G|I| \ldots |I|)$. Let $w' = \ell(n + 1) + w$. Since ℓ and w can be recovered from w' (they are the quotient and remainder when w' is divided by $n + 1$), it follows that (G', w') has an affirmative answer iff (G, ℓ, w) does.

$$(G, \ell, w) \propto (G_k, \ell', w')$$

Let $G_k = (G|I)$ and set $w' = w + \ell$, $\ell' = \ell$. Note that G_k has rank k even though G may have rank less than k.

$$(G_k, \ell, w) \propto (G_k', w')$$

If G_k has n columns, let G_k' be the matrix formed by adjoining to G_k $n + 1$ copies of the identity matrix I_k, i.e. $G_k' = (G|I_k| \ldots |I_k|)$. Let $w' = \ell(n + 1) + w$ as before.

References

1. Aho, Alfred; Hopcraft, John; and Ullman, Jeffrey, *The Analysis and Design of Computer Algorithms.* Addison-Wesley Publishing Co., Reading, Mass., 1974.

2. Karp, Richard, "Reducibility Among Combinatorial Problems," in *Complexity of Computer Computations,* edited by R. Miller and J. Thatcher, pp. 85-103, Plenum Press, New York, 1972.

3. Karp, Richard, "On the Computational Complexity of Combinatorial Problems," *Networks,* Vol. 5, pp. 45-68, 1975.

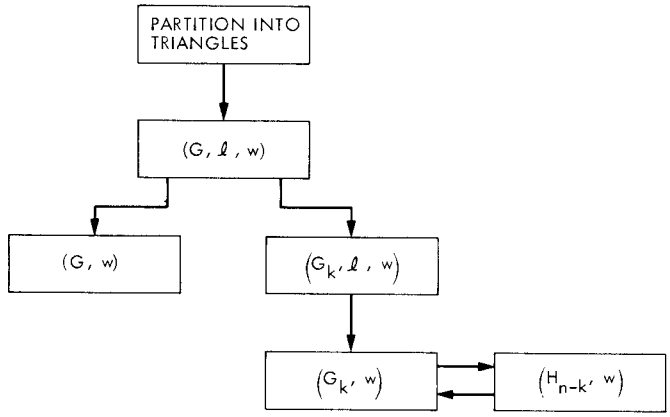

Fig. 1. The reductions showing the *NP* completeness of our problems