Is Code Equivalence Easy to Decide?

Erez Petrank[*] Ron M. Roth[†]

Abstract

We study the computational difficulty of deciding whether two matrices generate equivalent linear codes, i.e., codes that consist of the same codewords up to a fixed permutation on the codeword coordinates. We call this problem Code Equivalence. Using techniques from the area of interactive proofs, we show on the one hand, that under the assumption that the polynomial-time hierarchy does not collapse, Code Equivalence is not NP-complete. On the other hand, we present a polynomial-time reduction from the Graph Isomorphism problem to Code Equivalence. Thus, if one could find an efficient (i.e., polynomial-time) algorithm for Code Equivalence, then one could settle the long-standing problem of determining whether there is an efficient algorithm for solving Graph Isomorphism.

Keywords: Code equivalence, Graph isomorphism, Interactive proofs, Polynomial hierarchy.

[*]DIMACS Center, P.O.Box 1179, Piscataway, NJ 08855. e-mail: `erez@dimacs.rutgers.edu`.

[†]Computer Science Department, Technion, Haifa 32000, Israel. e-mail: `ronny@cs.technion.ac.il`. Currently on sabbatical leave at Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304, USA.

1 Introduction

Let F be a finite field and let G_1 and G_2 be generator matrices of two linear codes C_1 and C_2 over F. We say that G_1 and G_2 are *code-equivalent*, denoted $G_1 \sim G_2$, if the sets C_1 and C_2 are the same, up to a fixed permutation on the coordinates of the codewords of C_1. In other words, $G_1 \sim G_2$ if and only if both matrices have the same order $k \times n$, and there exist an $n \times n$ permutation matrix P and a nonsingular $k \times k$ matrix S over F such that $G_1 = SG_2P$. The problem of deciding whether two generator matrices are code-equivalent will be referred to as the Code Equivalence problem.

The purpose of this note is to study the computational difficulty of the Code Equivalence problem. As one application of a related problem, we mention the public-key cryptosystems due to McEliece [9] and Niederreiter [11]. Recall that an *alternant code* over $GF(q)$ is defined by a parity-check matrix of the form $[\,y_j\alpha_j^i\,]_{i=0,j=0}^{r-1,n-1}$, where the α_j's are distinct elements in $GF(q^m)$ and the y_j's are nonzero elements in $GF(q^m)$ [8, Ch. 12]. Goppa codes are special cases of alternant codes where certain restrictions are imposed on the values y_j's, and generalized Reed-Solomon codes are special cases of alternant codes where $m = 1$. The mentioned cryptosystems are based on the assumption that it is difficult to identify the values α_j and y_j out of an arbitrary generator matrix (or parity-check matrix) of an alternant code. Namely, it is difficult to obtain a code-equivalent matrix of the form $[\,y_j\alpha_j^i\,]_{i=0,j=0}^{r-1,n-1}$. On the other hand, as shown in [12], it is easy to extract the values α_j and y_j from any systematic generator matrix of a generalized Reed-Solomon code; hence, cryptosystems based on such a code are breakable. This was pointed out explicitly by Sidelnikov and Shestakov in [13]. For related work, see also the references cited in [10, p. 317].

The significance of the Code Equivalence problem can also be exhibited through the results of Kasami, Lin, and Peterson [6], and Kolesnik and Mironchikov [7], who showed that Reed Muller codes are equivalent to subcodes of extended BCH codes. Thus, it should be interesting to design an efficient algorithm that decides whether two codes are indeed equivalent, and thus, infer from the properties that arise from one code representation to the other.

On the positive side, we first show that the Code Equivalence problem is unlikely to be NP-complete. The proof of this assertion relies on techniques developed in the field of *interactive proofs*, which we summarize in Section 2. In Section 3, we invoke results of Goldwasser, Micali, and Rackoff [4], Goldreich, Micali, and Wigderson [3], Goldwasser and Sipser [5], and Boppana, Håstad, and Zachos [2], to show that if Code Equivalence is NP-Complete, then the polynomial hierarchy collapses.

Yet, we do state also a negative result, namely, that Code Equivalence is also unlikely to be too easy. We do this by relating Code Equivalence to the Graph Isomorphism problem. Let $\mathcal{G}_1 = (V, E_1)$ and $\mathcal{G}_2 = (V, E_2)$ be two undirected graphs with the same set of vertices V, and with sets of edges E_1 and E_2, respectively. We say that $\mathcal{G}_1$ is isomorphic to

$\mathcal{G}_2$ if there exists a permutation (isomorphism) $\pi : V \to V$ such that $\{u, v\} \in E_1$ if and only if $\{\pi(u), \pi(v)\} \in E_2$ (we assume here that the graphs have no parallel edges; if they do, then E_1 and E_2 are multisets, in which case isomorphism requires equality of the multiplicities of $\{u, v\}$ and $\{\pi(u), \pi(v)\}$ in E_1 and E_2, respectively). The problem of deciding efficiently (i.e., in polynomial-time) whether two graphs are isomorphic is a notoriously open question in Computer Science. The problem has been studied extensively in recent decades, but the state of the art is that there is no known efficient algorithm for determining whether two given graphs are isomorphic.

In Section 4, we show a polynomial-time reduction from GRAPH ISOMORPHISM to CODE EQUIVALENCE. This implies that presenting an efficient algorithm to determine whether two given linear codes are equivalent immediately yields an efficient algorithm for GRAPH ISOMORPHISM.

2 Interactive proofs

Loosely speaking, an interactive proof for a problem consists of a pair $(\mathcal{P}, \mathcal{V})$ of a non-restricted Turing machine (prover) $\mathcal{P}$ and a polynomial-time Turing machine (verifier) $\mathcal{V}$. On a given input, the prover tries to convince the verifier that the input satisfies the conditions of the problem. For example, in our case, the problem is CODE EQUIVALENCE and an input is a pair of generator matrices (G_1, G_2).

The interactive proof is conducted through several rounds in which the verifier sends messages (intuitively, asks questions) and the prover responds with messages (intuitively, answers those questions). Both parties are allowed to toss coins in order to construct their messages. At the end of the interaction, the verifier decides whether the proof should be accepted or rejected.

The requirement of an interactive proof is that if the input indeed satisfies the conditions of the problem, then the verifier $\mathcal{V}$ will accept with probability almost 1; otherwise, if the input does not satisfy the condition of the problem, then $\mathcal{V}$ will accept with probability almost 0.

We give next a formal definition of an interactive proof.

Let Σ be an alphabet. The set of all finite words over Σ will be denoted by Σ^*. A *language* L is a subset of Σ^*.

As an example, we define the language of CODE EQUIVALENCE as follows. Let F be a finite field and assume a one-to-one and onto mapping from the set of all pairs of generator matrices (G_1, G_2) over F into words of Σ^*. Such a mapping provides a representation of the pairs of generator matrices over F as words over Σ, and hereafter we interchange between such pairs of matrices and their word representation. The language of CODE EQUIVALENCE is the set of all (words over Σ representing) generator matrices (G_1, G_2) over F such that $G_1 \sim G_2$.

An interactive proof for a language $L \subseteq \Sigma^*$ is defined by means of a pair of probabilistic Turing machines, a prover $\mathcal{P}$ and a verifier $\mathcal{V}$; namely, in addition to their working tapes, each machine has access to its own private random tape (or coin tosses). The verifier $\mathcal{V}$ is bounded to be polynomial-time; namely, the running time of $\mathcal{V}$ is at most polynomially large in the length of the input (regarded as a word over Σ). No complexity restrictions are imposed on the prover $\mathcal{P}$.

On a common input $x \in \Sigma^*$, the two machines exchange messages, one message per round: In each odd round, the verifier sends a message which may depend on the input x, the history of the interaction so far (which the verifier can record on its working tape), and its (private) random tape. In each even round, the prover sends a message, which may as well depend on x, the history so far, and its random tape. The sequence of rounds is referred to as a *protocol*, or a *proof*. After the protocol has ended, the verifier halts in one out of two states, "$x \in L$" or "$x \notin L$," corresponding, respectively, to *acceptance* or *rejection* of the proof. When $\mathcal{V}$ accepts the proof, we say that $\mathcal{P}$ *convinces* $\mathcal{V}$ that $x \in L$.

We say that the pair $(\mathcal{P}, \mathcal{V})$ is an interactive proof for the language L if for every word $x \in \Sigma^*$ of length m, the following two conditions hold:

1. **Completeness condition:** If $x \in L$, then $\mathcal{P}$ convinces $\mathcal{V}$ with probability at least $1 - 2^{-m}$.
2. **Soundness condition:** If $x \notin L$, then for any prover $\mathcal{P}^*$ (possibly an impostor that tries to deceive), the probability that $\mathcal{V}$ is convinced after interacting with $\mathcal{P}^*$ is at most 2^{-m}.

3 Code Equivalence is unlikely to be hard

We show that Code Equivalence is unlikely to be hard by constructing an interactive proof with a constant number of rounds for the problem of Code non-Equivalence, i.e., the complementary problem of Code Equivalence. To do this, we use techniques developed by Goldreich, Micali, and Wigderson in [3], who, in turn, used ideas developed by Goldwasser, Micali, and Rackoff in [4].

Having found a constant-round interactive proof for the Code non-Equivalence problem, we then invoke a result of Goldwasser and Sipser [5], stating that if a problem has a constant-round interactive proof, then it also has a constant-round Arthur–Merlin protocol (Arthur–Merlin protocols [1] are a special case of interactive proofs which we do not present here). At this point, we apply a result of Boppana, Håstad, and Zachos [2], stating that if the complement of a problem Π has an Arthur–Merlin protocol with a constant number of rounds, and if Π is NP-complete, then the polynomial-time hierarchy collapses.

It is believed that the polynomial-time hierarchy does not collapse, and thus we end up with the conclusion that CODE EQUIVALENCE is unlikely to be NP-complete.

Hence, it remains to show the constant-round interactive proof for the CODE NON-EQUIVALENCE problem. We present next a protocol for an interactive proof for CODE NON-EQUIVALENCE that follows along the ideas of the protocols for QUADRATIC NON-RESIDUOSITY in [4] and for GRAPH NON-ISOMORPHISM in [3].

The infinitely-powerful prover and the polynomial-time verifier have on their input tapes two $k \times n$ matrices, G_1 and G_2, over a given finite field F. The prover would like to convince the verifier that $G_1 \not\sim G_2$.

The outline of our interactive proof is as follows. The verifier first selects at random one of the matrices G_i, $i = 1, 2$. Next, it chooses uniformly at random a matrix H among all matrices over F which are code-equivalent to G_i, and sends H to the prover. Now, if $G_1 \not\sim G_2$, then the (all-powerful) prover can determine which one of the matrices, G_1 or G_2, was selected by the verifier. However, if $G_1 \sim G_2$, then the prover has no way of telling which of the two matrices were selected. The reply of the prover is an index $j \in \{1, 2\}$, and the verifier accepts if and only if $j = i$.

Clearly, the prover will always succeed in convincing the verifier when $G_1 \not\sim G_2$. However, when $G_1 \sim G_2$, the prover cannot do better than guessing and thus will fail with probability $1/2$. In order to reduce the error probability to 2^{-m} (without increasing the number of rounds) we repeat this procedure in parallel m times.

Following is a formal description of the protocol we have just outlined:

First round by the verifier: Given two $k \times n$ matrices G_1 and G_2 over F, the verifier selects uniformly at random $b_1, b_2, \ldots, b_m \in \{1, 2\}$. For each $\ell = 1, 2, \ldots, m$, the verifier selects a matrix H^ℓ over F which is code-equivalent to G_{b_ℓ}, by uniformly selecting a random nonsingular $k \times k$ matrix S_ℓ (change of base) and a random $n \times n$ permutation matrix P_ℓ; the matrix H^ℓ is then given by $S_\ell G_{b_\ell} P_\ell$. The verifier sends the resulting matrices $(H^1, H^2, \ldots, H^m)$ to the prover.

Second round by the prover: For each $\ell = 1, 2, \ldots, m$, the prover gives its estimate $b'_\ell \in \{1, 2\}$ for the value b_ℓ according to whether H^ℓ is code-equivalent to G_1 or to G_2. The prover sends $(b'_1, b'_2, \ldots, b'_m)$ to the verifier.

A final verification by the verifier: The verifier accepts the proof if and only if $b'_\ell = b_\ell$ for all $\ell = 1, 2, \ldots, m$.

We now analyze the protocol.

Completeness: Clearly, if $G_1 \not\sim G_2$, then the prover will never fail in convincing the verifier.

Soundness: We now assume that $G_1 \sim G_2$. For each $\ell = 1, 2, \ldots, m$, the verifier picks a random element H^ℓ in the code-equivalence class of G_{b_ℓ}, but this is also a random element in the code-equivalence class of the other matrix G_{3-b_ℓ}. In other words, the distribution of $(b_1, b_2, \ldots, b_m)$ given $(H^1, H^2, \ldots, H^m)$ is identical to the *a priori* distribution of $(b_1, b_2, \ldots, b_m)$ which, in turn, is a uniformly selected random vector in $\{1, 2\}^m$. The probability that the prover can find (or rather guess) this vector (no matter what its strategy is) is at most 2^{-m}. Therefore, the probability that the verifier is convinced that the matrices are not code-equivalent is at most 2^{-m}.

4 Code Equivalence is not too easy

In this section, we show a polynomial-time reduction from Graph Isomorphism to Code Equivalence. Therefore, if there is a polynomial-time algorithm that decides whether two generator matrices are code-equivalent, then we have a polynomial-time algorithm that decides whether two graphs are isomorphic.

We present a mapping ϕ from the set of all graphs to the set of generator matrices over $GF(2)$ such that two graphs are isomorphic if and only if their respective images under ϕ are code-equivalent. For a graph $\mathcal{G} = (V, E)$, the image $\phi(\mathcal{G})$ is a generator matrix G with the following properties:

(a) G is a generator matrix over $GF(2)$ of order $|E| \times (3|E| + |V|)$;

(b) each row in G has Hamming weight ≤ 5; and —

(c) any set of two rows or more in G sums up to a vector with Hamming weight ≥ 6.

Let G be a matrix that satisfies properties (a)—(c) and let C be the binary code which is generated by G. It follows from (b) and (c) that C has exactly $|E|$ nonzero codewords of Hamming weight ≤ 5, and those codewords are the rows of G. That is, up to permutation of the rows, there is a *unique* generator matrix G of C that satisfies properties (a)—(c).

The mapping ϕ is defined as follows. Given a graph $\mathcal{G} = (V, E)$, let D denote the $|E| \times |V|$ incidence matrix of $\mathcal{G}$; namely, the rows of D are indexed by the edges of $\mathcal{G}$, the columns of $\mathcal{G}$ are indexed by the vertices of $\mathcal{G}$, and, for every $e \in E$ and $u \in V$ we have

$$D_{e,u} = \begin{cases} 1 & \text{if } e = \{u, v\} \text{ for some } v \in V \\ 0 & \text{otherwise} \end{cases} .$$

(A rows in D has Hamming weight 1 if the respective edge is a self-loop, and rows can be equal in case of parallel edges.)

The matrix $\phi(\mathcal{G})$ is given by $[\, I \,|\, I \,|\, I \,|\, D \,]$, where I stands for the $|E| \times |E|$ identity matrix.

We now analyze the reduction. Clearly, the reduction can be easily computed in polynomial time. Now, suppose that $\mathcal{G}_1 = (V, E_1)$ is isomorphic to $\mathcal{G}_2 = (V, E_2)$ through a permutation $\pi : V \to V$. Let $G_1 = \phi(\mathcal{G}_1) = [\, I \,|\, I \,|\, I \,|\, D_1 \,]$ and $G_2 = \phi(\mathcal{G}_2) = [\, I \,|\, I \,|\, I \,|\, D_2 \,]$. We show that $G_1 \sim G_2$. Clearly, D_1 and D_2 have the same order, and D_2 can be obtained by applying the permutation π on the columns of D_1, followed by row permutation of the resulting matrix. Hence, G_2 can be obtained by permuting the rows and columns of G_1 and, thus, $G_1 \sim G_2$.

On the other hand, let $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_2 = (V_2, E_2)$ be graphs with $G_1 = \phi(\mathcal{G}_1)$ and $G_2 = \phi(\mathcal{G}_2)$ and suppose that $G_1 \sim G_2$. In particular, $|E_1| = |E_2|$ and $|V_1| = |V_2|$ and without loss of generality we can assume that $V_1 = V_2 = V$. We show that $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic.

Let $k = |E_1| = |E_2|$ and $n = 3k + |V|$, and recall that $G_1 \sim G_2$ means that there is an $n \times n$ permutation matrix P and a $k \times k$ nonsingular matrix S so that $G_2 = SG_1P$. Now, the matrices G_2 and G_1P both generate the same linear code and both satisfy properties (a)—(c). Hence, those matrices must be the same up to permutation of rows and, so, S must be a $k \times k$ permutation matrix.

Write $G_i = [\, I \,|\, I \,|\, I \,|\, D_i \,]$, $i = 1, 2$. We have

$$[\, I \,|\, I \,|\, I \,|\, D_2 \,] \;=\; G_2 \;=\; SG_1P \;=\; [\, S \,|\, S \,|\, S \,|\, SD_1 \,]P \,. \tag{1}$$

Since S is a permutation matrix, the first $3k$ columns of SG_1 consist of all unit vectors of length k, each appearing exactly three times. Hence, the first $3k$ columns of G_2 are obtained by permuting the first $3k$ columns of SG_1. Noting from (1) that the n columns of G_2 form a permutation of the n columns of SG_1, it follows that we also have $G_2 = SG_1\hat{P}$ for some block-diagonal $n \times n$ permutation matrix

$$\hat{P} \;=\; \begin{bmatrix} S^{-1} & & & \\ & S^{-1} & & \\ & & S^{-1} & \\ & & & T \end{bmatrix} ,$$

where T is a $|V| \times |V|$ permutation matrix. Hence, $D_2 = SD_1T$, which, in turn, implies that $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic. The isomorphism π is the permutation associated with the matrix T.

References

[1] L. Babai, S. Moran, *Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes*, *JCSS*, 36 (1988), 254–276.

[2] R. Boppana, J. Håstad, S. Zachos, *Does co-NP have short interactive proofs, Information Proc. Letters*, 25 (1987), 27–132.

[3] O. Goldreich, S. Micali, A. Wigderson, *Proofs that yield nothing but their validity and a methodology of cryptographic protocol design, J. of ACM,* 38 (1991), 691–729. See also *Proc. 27th IEEE Symp. Found. of Comp. Science* (1986).

[4] S. Goldwasser, S. Micali, C. Rackoff, *The knowledge complexity of interactive proofs, SIAM J. Comput.,* 18 (1989), 186–208. See also *Proc. 17th Annual ACM Symp. on the Theory of Computing* (1985).

[5] S. Goldwasser, M. Sipser, *Private coins vs. public coins in interactive proof systems, Advances in Computing Research,* S. Micali (Ed.), 5 (1989), 73–90.

[6] T. Kasami, S. Lin, W.W. Peterson, *New generalizations of the Reed-Muller codes, Part I: Primitive codes, IEEE Trans. Inform. Theory,* 14 (1968), 189–199.

[7] V.D. Kolesnik, E.T. Mironchikov, *Cyclic Reed-Muller codes and their decoding, Problems Infor. Trans.,* 4 (1968), 15–19.

[8] F.J. MacWilliams, N.J.A Sloane, *The Theory of Error-Correcting Codes,* North Holland, Amsterdam, 1977.

[9] R.J. McEliece, *A public-key cryptosystem based on algebraic coding theory,* DSN Progress Report 42-44, JPL, Caltech, Pasadena, CA (Jan–Feb 1978), 114–116.

[10] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of Applied Cryptography,* CRC Press, Boca Raton, Florida, 1997.

[11] H. Niederreiter, *Knapsack-type cryptosystems and algebraic coding theory, Prob. Contr. Inform. Theory,* 15 (1986), 157–166.

[12] R.M. Roth, G. Seroussi, *On generator matrices of MDS codes, IEEE Trans. Inform. Theory,* 31 (1986), 826–830.

[13] V.M. Sidelnikov, S.O. Shestakov, *On cryptosystems based on generalized Reed-Solomon codes, Discretnaya Math.,* 4 (1992), 57–63 (in Russian).