Algorithms for computing parameters of graph-based extensions of BCH codes

A.V. Kelarev

School of Computing, University of Tasmania, Private Bag 100, Hobart, Tasmania 7001, Australia

Received 21 January 2006; received in revised form 1 August 2006; accepted 14 August 2006
Available online 27 October 2006

Abstract

This paper develops combinatorial algorithms for computing parameters of extensions of BCH codes based on directed graphs. One of our algorithms generalizes and strengthens a previous result obtained in the literature before.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Algorithms; Codes; Directed graphs

1. Introduction

BCH codes form a large class widely used in applications including CDs, DVDs, mobile phones and digital TV, as described in [35]. Extensions of BCH codes have also been actively investigated. Let us refer, for example, to [4–6,8,13,16,20,43] for some recent results on the subject.

Directed graphs have been used recently to define error-correcting codes extending the BCH codes and investigate their properties. This paper is devoted to a larger class of codes and develops combinatorial algorithms for computing their weights, information rates, and testing if there have been errors in transmission of a codeword. The algorithms are combinatorial in nature. They utilize only the structure of the graphs and involve minimal prerequisites on BCH codes.

The motivation for considering this class is twofold. First, our construction is inspired by analogy with the way all classical cyclic codes are defined by their generator polynomials: every cyclic code \( C \) has a generator polynomial \( g \) such that \( C \) coincides with the set of all multiples of \( g \) in a certain quotient ring \( R \). Second, the new class is substantially larger than the set of extensions of BCH codes proposed in [19], where only one formula for the smaller category was given. Our new results handle a much larger class and consist in developing combinatorial algorithms computing essential parameters for these codes. Downey, Fellows, Whittle and Vardy have proved that a number of fundamental problems in coding theory are NP-complete and W[1]-hard, see [10] and also [9]. A number of related problems in computational group theory are also hard (see, for example, [15]). This is why it is nice to see how in the...
special case of extensions of BCH codes efficient algorithms based on directed graphs can handle problems which are in general very difficult.

All the required information on coding theory is included in Section 2 for convenience of the readers. Sections 3 contains main algorithms and theorems. Open questions are recorded in Section 4.

2. Preliminaries

We use standard concepts concerning algorithms, graphs, and codes, following [7,12,14,17,18,21–24,27,31–34, 36]. Throughout the word “graph” means a directed graph without multiple edges but possibly with loops. An isolated vertex is a vertex that has no edges incident to it. For the purposes of our algorithms, all maximal complete subgraphs of $D$ as well as all isolated vertices are called the cliques of $D$. A vertex $v$ of the graph $D$ is called a source if there are no edges $(v', v)$ in $E$. For any $v \in V$, let

$$\text{outdeg}(v) = |\{v' \in V \mid (v, v') \in E\}|.$$ 

Let $c, d, \ell, q$ be positive integers, where $q$ is a power of a prime number, $2 \leq d \leq \ell$ and $\gcd(q, \ell) = 1$. Denote by $m$ the order of $q$ modulo $\ell$, i.e., the smallest positive integer such that $q^m \equiv 1 \pmod{\ell}$. Choose a primitive \(\ell \)th root $z$ of 1 in $\mathbb{F}_{q^m}$, i.e., an element $z$ such that $\ell$ is the least positive integer satisfying $z^\ell = 1$ in $\mathbb{F}_{q^m}$. For example, it is known that the set $\mathbb{F}_{q^m}^\ast$ of nonzero elements of the field $\mathbb{F}_{q^m}$ is cyclic, that is there exists an element $\xi$ in $\mathbb{F}_{q^m}$ such that $\mathbb{F}_{q^m}^\ast$ is equal to the set of all powers of $\xi$. Every element with this property is called a primitive element of $\mathbb{F}_{q^m}$. Since the identity $x^{q^m} = x$ holds for all $x \in \mathbb{F}_{q^m}$, it follows that $\xi^{((q^m-1)/\ell)}$ is a primitive $\ell$th root of 1 in $\mathbb{F}_{q^m}$. Denote by $m_{\ell}$, the minimal polynomial of $z^\ell$, i.e., the monic polynomial of smallest degree such that $m_{\ell}(z^\ell) = 0$. The BCH code $C = C_d$ of designed distance $d$ is a cyclic code over $\mathbb{F}_q$ with length $\ell$ and generator polynomial

$$g_d = \text{lcm}\{m_{\ell} \mid c \leq i \leq c + d - 2\}.$$ 

This means that $C$ consists of all multiples of the generator polynomial $g_d$ in the polynomial quotient ring $R = \mathbb{F}_q[x]/(1-x^\ell)$. Recall that $R$ is the set of all univariate polynomials of degree $\leq \ell$ with coefficients in $\mathbb{F}_q$. Suppose the usual addition and the multiplication defined modulo $1 - x^\ell$, so that the product of two polynomials in $R$ is equal to the remainder of their ordinary product upon division by $1 - x^\ell$. A codeword $v$ lies in $C$ if and only if $v(z^\ell) = 0$ for all $c \leq i \leq c + d - 2$. The weight of a code $C$ is the minimum weight of a nonzero element in the code, i.e., the number of nonzero coordinates of the element in a standard basis. For a linear code its weight determines the number of errors the code can correct or detect. The information rate of an $(n, m)$ code is the ratio $m/n$ of the number of message digits, which form the information to be transmitted, to the number of all digits. There exists a polynomial $f_d \in R$ such that $\text{wt}(f_d g_d) = \text{wt}(C) \geq d$. The information rate of $C$ is $\geq 1 - m(d-1)/\ell$. Further $R = \mathbb{F}_q[x]/(1-x^n)$ is always the quotient ring containing the BCH code of designed distance $d$.

Next, we define a graph-based construction well known in the literature. Let $D = (V, E)$ be a graph with the set $V = V_n = \{1, 2, \ldots, n\}$ of vertices. Denote by $e_{i,j} = e_{(i,j)}$ the standard elementary $n \times n$ matrix with 1 in the intersection of $i$th row and $j$th column and zeros in all other cells or entries. The edges of $D$ correspond to the standard elementary matrices in the set $M_n(R)$ of all $(n \times n)$-matrices over $R$. To simplify notation we may identify the edges of $D$ and their standard elementary matrices by putting $(i, j) = e_{i,j}$. Denote by

$$M_D(R) = \bigoplus_{w \in E} Re_w \subseteq M_n(R)$$

the set of all matrices with nonzero entries corresponding to the edges of the graph $D$, and zeros in all entries for which there are no edges in $D$. In other words, $M_D(R)$ can be defined as the set of all matrices which are obtained by replacing nonzero entries in the adjacency matrix of the graph $D$ with arbitrary elements of $R$. Throughout we assume that $E \neq \emptyset$, since otherwise $M_D(R) = 0$.

The following properties and notation will be used in the proofs. Every element $r$ in $M_D(R)$ has a unique representation of the form

$$r = \sum_{i,j=1}^n r_{i,j} e_{i,j} = \sum_{w \in E} r_w e_w \in M_D(R) \tag{1}$$
where $r_{i,j} \in R$. The standard addition and multiplication are defined on the set $M_D(R)$ by setting, for all $r, s \in M_D(R)$,

$$
\begin{align*}
r + s & = \sum_{i,j=1}^{n} r_{i,j}e_{i,j} + \sum_{i,j=1}^{n} s_{i,j}e_{i,j} = \sum_{i,j=1}^{n} (r_{i,j} + s_{i,j})e_{i,j}, \\
r \cdot s & = \sum_{i,j=1}^{n} r_{i,j}e_{i,j} \cdot \sum_{i,j=1}^{n} s_{i,j}e_{i,j} = \sum_{i,j,k=1}^{n} (r_{i,j}s_{j,k})e_{i,k}. 
\end{align*}
$$

It is well known and easy to verify that these rules correctly define $+$ and $\cdot$ as operations on the set $M_D(R)$ if and only if $D$ is transitive, i.e., the following property

$$
(x, y), (y, z) \in E \Rightarrow (x, z) \in E,
$$

holds for all $x, y, z \in V$. In this case the set $M_D(R)$ is called the matrix ring of the graph $D$ over $R$. The class of transitive graphs is important and plays essential roles in graph theory, see for example [28] and [18]. There are several well-known algorithms for testing whether a given graph is transitive, for example, with running times $O(|V|^3)$ and $O(|V|(|V| + |E|))$ (see [12, §12.4.2]. Many interesting results on matrix rings of graphs have been obtained by several authors, see [17] for references.

For every $r \in M_D(R)$ and $1 \leq i, j \leq n$, the entry $r_{i,j} = r(i,j)$ of $r$ can be expressed as

$$
\begin{align*}
r_{i,j} & = e_{i,i} \cdot r \cdot e_{j,j} 
\end{align*}
$$

where the product on the right-hand side is defined with (3). For any $r \in M_D(R)$, denote by $E(r)$ the set of edges $w \in E$ such that $r_w \neq 0$. We regard all elements in the standard basis. Hence in this case the weight $\text{wt}(r)$ of an element $r \in M_n(F)$, is equal to the number $|E(r)|$ of nonzero cells $r_{i,j}$.

### 3. Main results

By analogy with cyclic codes, we say that a code $C \subseteq M_D(R)$ is generated by the elements $h_1, \ldots, h_k$ in $M_D(R)$ if it consists of all codewords $c$ which can be represented as a finite sum of multiples of $g$, that is a finite sum of the form

$$
c = \sum_{i=1}^{k} k_i h_i + r_i h_i,
$$

where $k_i \in \mathbb{Z}$, $r_i \in M_D(R)$, and $\mathbb{Z}$ stands for the set of integers. In this case the elements $h_1, \ldots, h_k$ are called the generators of $C$. In the case of polynomial quotient ring $R$, each cyclic code can be generated by one generator polynomial. This is precisely the reason which makes cyclic codes convenient.

Denote by $M_D([0, 1])$ the set of all elements in $M_D(R)$ with all coefficients in the set $[0, 1]$. For clarity, further we consider the case where all generators $h_1, \ldots, h_k$ belong to $M_D([0, 1])$. A code $C$ is said to be a graph-based extension of the BCH code in $M_D(R)$ if it has got a set of generators of the form

$$
g_d h_1, \ldots, g_d h_k, \quad \text{where } h_1, \ldots, h_k \in M_D([0, 1]).
$$

In this case we write

$$
C = \text{BCH}_\ell(h_1, \ldots, h_k).
$$

Of course, our definition means that the code $\text{BCH}_\ell(h_1, \ldots, h_k)$ contains all elements generated by its generating set of the form (7). Obviously, for every loop $e \in E$ the code $\text{BCH}(e)$ coincides with the standard BCH code, and therefore all BCH codes are a special case of this class.

First, we find the weight of each code

$$
\text{BCH}_\ell(h_1, \ldots, h_k).
$$

As mentioned above, the Hamming weight of a code $C$ is the minimum number of nonzero coordinates a nonzero element in the code may have. The weight $\text{wt}(r)$ of an element $r \in M_n(F)$, is equal to the number $|E(r)|$ of nonzero cells $r_{i,j}$.
Theorem 1. For each transitive graph $D = (V, E)$ and every $h_1, \ldots, h_k \in M_D([0, 1])$, Algorithm 1 computes the weight of the code $\text{BCH}_k(h_1, \ldots, h_k)$. The running time of the algorithm is $O(kn^2)$.

A special element of $M_D(R)$ is denoted by $1$ and is called an identity if $1x = x1 = x$ for all $x \in M_D(R)$. Since $e_{i,i} \in M_D(R)$ if and only if the loop $(i, i)$ is in $E$, the following well-known fact is obvious, see [17].

Lemma 1. For every transitive graph $D$, the following conditions are equivalent:

(i) There exists an identity element in $M_D(R)$.
(ii) The graph $D$ contains all loops.

Proof of Theorem 1. Take any codeword $c$ in $C = \text{BCH}_k(h_1, \ldots, h_n)$. It follows from (6) that $c = \sum_{i=1}^{k} c_i g_d h_i + r_i g_d h_j$. Given that $D$ contains all loops, Lemma 1 shows that $M_D(R)$ possesses an identity element $1$. Hence $k_i h_i = (k_i 1) h_i$, where $1 \in M_D(R)$ and $(k_i 1) \in M_D(R)$. Therefore $c$ can be represented as

$$c = \sum_{i=1}^{k} c_i g_d h_i,$$

where $c_i \in M_D(R)$.

Algorithm 1 uses the following notation. The set of all sources of $D$ is denoted by $S(D)$, and the set of all vertices of $D$ which have loops is designated by $L(D)$. The list of all vertices, which begin edges of $h_m$ and have no loops, is denoted by $S_m$. For $1 \leq i \leq n$, the symbol $V_i(h_m)$ stands for the set of all vertices $j$ such that $h_m$ has an edge $(i, j)$. The set of all sources $s$ with $(s, i) \in E$ is denoted by $S(i)$. The list $C(s)$ stores all vertices $i$ for which $s$ has been chosen in $S(i)$ during execution of the algorithm. The set $E_{s,m,i}$ collects all edges which begin in the source $s$ and end in vertices of $V_i(h_m)$, where $s$ has been chosen in $S(i)$ by the algorithm. All sources which we can use to start edges of elements in $C$ are collected in $L$. Since $D$ is transitive, it follows that $A$ is acyclic. Hence $S(A)$ in line 4 is the set of all sources of $A$. Line 15 uses the fact that all nonzero coefficients of each $h_m$ are equal to 1.

---

**Algorithm 1.** Computes the weight of the code $\text{BCH}_k(h_1, \ldots, h_k)$ for given $h_1, \ldots, h_k \in M_D([0, 1])$.

1. Find the set $S(D)$ of all sources of $D$, and $L(D) = \{v \in V \mid (v, v) \in E\}$.
2. Find all cliques of $D$. Choose one vertex in each clique.
3. Find the subgraph $A$ of $D$ induced by the chosen vertices.
4. Find $S(A) = V(A) \cap S(D)$.
5. For each vertex $v$ of $D$, find $S(v) = \{s \in S(A) \mid (s, v) \in E\}$.
6. for ( $m = 1; m <= k; m++$ ) { /* The generator $h_m$ */
7. $S_m = \emptyset$; $I_m = \emptyset$; hasSources = false;
8. for ( $i = 1; i <= n; i++$ ) { /* Vertex $i$ */
9. Find $E_l(h_m) = \{(i, j) \in E(h_m), V_l(h_m) = \{j \mid (i, j) \in E_l(h_m)\}\}$.
10. if ( $V_l(h_m) = \emptyset$ ) continue;
11. if ( $i \not\in L(D)$ )
12. $S_m = S_m \cup E_i(h_m)$; $I_m = I_m \cup \{i\}$;
13. if ( $i \in S(D)$ ) { hasSources = true; continue; }
14. }
15. for $s \in S(i)$, put $s$ in $L, i$ in $C(s)$, $\{(s, j) \mid j \in V_l(h_m)\}$ in $E_{s,m,i}$.
16. }
17. if ( hasSources == false ) $S_m = \emptyset$;
18. }
19. for ( $s \in \bigcup_{m} I(m) \cup L$ )
20. Find $\sum_{m=1}^{k} \sum_{i=1}^{n} \sum_{(s,j) \in E_{s,m,i}} f_m, i, j(s, j) + \sum_{(m, s) \in I(m)} \sum_{g \in S_m} l_m, g \sum_{(f, g) \in L}$
21. with minimal weight $w_s$.
22. }
23. return $\text{wt}(f_d, g_d) \min_{s \in L} w_s$;

Fig. 1. Computing the weight of $\text{BCH}_k(h_1, \ldots, h_k)$. 
Now, let us prove correctness of the algorithm. Denote by RetVal the value returned by Algorithm 1. Let wt(C) be the weight of the code C. We need to prove that wt(C) = RetVal. First, we are going to prove that wt(C) ≤ RetVal. To this end it suffices to show that C always contains a nonzero element with weight RetVal. We consider two parts of the expression in line 20 separately, and denote them by

\[ P_1 = \sum_{m=1}^{k} \sum_{i=1}^{n} \sum_{(s,j) \in E_{s,m,i}} r_{m,i,j}(s, j), \]

\[ P_2 = \sum_{g \in S_m} \sum_{i \in \ell(h_m)} t_{m,g} g. \]

Let s be the vertex that achieves the minimum value of wt(P1 + P2), and suppose that P1 and P2 are recorded for this s. We claim that both \( f_{d,g_d} P_1 \) and \( f_{d,g_d} P_2 \) belong to \( C \). This will imply that \( f_{d,g_d} (P_1 + P_2) \) is in \( C \), and it will follow that RetVal ≥ wt(\( f_{d,g_d} (P_1 + P_2) \)) = wt(\( f_{d,g_d} (P_1 + P_2) \)) ≥ wt(C), as desired.

First, we look at \( P_2 \). It follows from lines 9 and 12 that each \( S_m \) contributes edges without loops to \( P_2 \), i.e., \( \sum_{g \in S_m} t_{m,g} g = \sum_{i \in \ell(D)} \sum_{g \in E_i(h_m)} t_{m,g} g \). Keeping in mind that all the nonzero coefficients of \( h_m \) are equal to 1 and the equality (3), we can re-write the last expression above as

\[ \sum_{g \in S_m} t_{m,g} g = h_m - \sum_{i \in \ell(D)} \sum_{g \in E_i(h_m)} t_{m,g} g. \]

The definition of \( M_D(R) \) yields that if \( g \in E_i(h_m) \), then \( (i,i)g = g \). On the other hand, if \( g \in E(h_m) \setminus E_i(h_m) \), then \( (i,i)g = 0 \). Therefore

\[ \sum_{i \in \ell(D)} \sum_{g \in E_i(h_m)} t_{m,g} g = \sum_{i \in \ell(D)} \sum_{g \in E_i(h_m)} t_{m,g}(i,i)g = \sum_{i \in \ell(D)} t_{m,g}(i,i)h_m \]

belongs to BCH(\( h_1, \ldots, h_k \)). Hence \( \sum_{g \in S_m} t_{m,g} g \) is in \( C \) too. Thus \( f_{d,g_d} P_2 \) belongs to \( C \).

Second, we consider \( P_1 \). Here \( s \) is the vertex which achieves the minimum value. When the algorithm executes, the list of all vertices \( i \) which have contributed to \( s \) is stored in \( C(s) \). Since \( (s,i)h_m = \sum_{j \in \ell(h_m)} (s,j) \) for each \( i \), the sum \( P_1 \) turns into

\[ P_1 = \sum_{m=1}^{k} \sum_{i=1}^{n} (s,i) \sum_{(s,j) \in E_{s,m,i}} r_{m,i,j} h_m. \]

Hence \( f_{d,g_d} P_1 \) belongs to BCH(\( h_1, \ldots, h_k \)), as required. This establishes the inequality wt(C) ≤ RetVal.

To prove the reversed inequality wt(C) ≥ RetVal let us choose an arbitrary nonzero element y in C. We have to show that wt(y) ≥ RetVal. Obviously, we may assume that y has been chosen so that wt(y) = \( |E(y)| \) achieves the minimum value. By (6) and (7), we get \( y = f_{d,g_d} x \) where

\[ x = \sum_{i=1}^{k} k_i h_i + \sum_{i=1}^{k} r_i h_i, \]

\( k_i \in \mathbb{Z}, r_i \in M_D(R) \). Since \( y \neq 0 \), there exist \( 1 \leq b, c \leq n \) such that \( x_{(b,c)} \neq 0 \). Consider two possible cases.

Case 1. \( b \in S(D) \). Then

\[ x = \sum_{i=1}^{k} k_i h_i + \sum_{i=1}^{k} r_i h_i \]

\[ = \sum_{i=1}^{k} \sum_{j=1}^{n} k_i (h_i)_{b,j} + \sum_{i=1}^{k} \sum_{j=1}^{n} r_{i,j} (b,j) h_i. \]

Obviously, all \( j \) with nonzero summands in the last sum above belong to the set \( C(s) \). By the definition of \( S(b) \), we get \( (s,i) \in E \). Therefore \( (s,i)x \) is recorded. It follows that \( w_x \leq wt(x) \). Hence \( m_1 \leq wt(x) \), and so RetVal ≤ wt(\( f_{d,g_d} x \)).
Case 2. \( b \notin S(D) \). Then there exists a vertex \( s \) that has been chosen in \( S(b) \) during execution of the algorithm. Clearly, \( (s, b)x_{(b, c)} \neq 0 \) implies \( (s, b)x \neq 0 \). Besides, \( f_d g_d(s, b)x_{(b, c)} \) belongs to \( C \), and \( |E((s, b)x)| \leq |E(x)| \). By the minimality assumption, we see that \( E(x) = E_b(x) \), that is all edges of \( x \) begin in \( b \). Replacing \( x \) with \( (s, b)x \) we reduce Case 2 to Case 1.

Thus, in both the cases \( wt(y) \geq RetVal \). This shows that \( wt(y) = RetVal \) and completes the proof of correctness.

In evaluating the running times we may assume that addition and multiplication in the finite field take \( O(n) \) time. The running time of line 1 is of the order of \( O(n) \). Thus, the total running time of the algorithm is \( O(kn) \). Line 6 to 18 require \( O(n^2) \) time. It is straightforward to verify that lines 20 and 21 can be accomplished in \( O(kn) \) time by standard Gaussian elimination. Thus, the total running time of the algorithm is \( O(kn^2) \). \( \square \)

Our next algorithm finds the largest weight of the code \( BCH_\ell(h_1, \ldots, h_k) \) that can be achieved for all \( h_1, \ldots, h_k \in M_D([0, 1]) \). This task turns out to be much easier than a brute force search through all sets of generators \( h_1, \ldots, h_k \) combined with a direct application of Algorithm 1. Surprisingly, it turns out that there is always an optimal code of this sort which is generated by one element and has the form \( BCH_\ell(h) \).

**Theorem 2.** For every transitive graph \( D = (V, E) \), Algorithm 2 computes the largest weight among all weights of the codes \( BCH_\ell(h_1, \ldots, h_k) \), for all \( h_1, \ldots, h_k \in M_D([0, 1]) \). The running time of the algorithm is \( O(n^3) \).

**Proof.** Let us prove correctness of the algorithm. Denote by Largest(D) the largest weight of the code \( BCH_\ell(h_1, \ldots, h_k) \) among all codes of this sort for \( h_1, \ldots, h_k \in M_D([0, 1]) \). Let RetVal be the value returned by Algorithm 2. It is the largest weight that can be achieved for all \( h_1, \ldots, h_k \) in \( M_D([0, 1]) \).

We begin by proving that \( \text{Largest}(D) \geq \text{RetVal} \). To this end it suffices to show that \( M_D(R) \) always contains a code of the form \( BCH_\ell(h_1, \ldots, h_k) \) with weight \( \text{RetVal} \). Consider two possible cases depending on which value the algorithm returns in line 14.

**Case 1.** \( c \geq b \). Then \( \text{RetVal} = wt(f_d g_d)c, \) where \( c = \sum_{v \in S(D)} \text{outdeg}(v) \). Since we have assumed that \( M_D(R) \neq 0 \) and \( E \neq \emptyset \), it is clear that \( c > 0 \) and \( D \) has sources. Denote by \( E_S \) the set of edges that begin in sources, i.e., put

\[
E_S = E \cap (S(D) \times V).
\]

Let \( h = \sum_{(u, v) \in E_S} e_{u, v} \) and let \( C = BCH_\ell(h) \). We claim that \( C \) is the desired code with weight \( \text{RetVal} \). Obviously, \( wt(h) = |E_S| = \sum_{v \in S(D)} \text{outdeg}(v) \). For any \( (i, j) \in E \) we have \( e_{i, j}h = 0 \), as \( j \) is not a source. Hence \( C = R g_d h \) and \( wt(C) = wt(f_d g_d h) = wt(f_d g_d) \sum_{v \in S(D)} \text{outdeg}(v) = wt(f_d g_d)c = \text{RetVal} \).

**Case 2.** \( c < b \). Then \( \text{RetVal} = wt(f_d g_d)b, \) where

\[
b = \max \{ \text{outdeg}(v) \mid v \in V, S(v) = \emptyset \},
\]

```
Algorithm 2. Computes the largest weight \( w_D \) among the weights of all codes \( BCH_\ell(h_1, \ldots, h_k) \) for all \( h_1, \ldots, h_k \in M_D([0, 1]) \).

1. int b = 0, c = 0;
2. Find \( S(D) = \{ v \in V \mid (V \times \{ v \}) \cap E = \emptyset \} \).
3. for ( v = 1; v <= n; v++ ) {
4.    Find outdeg(v) = \{ (v, v') \in E \}.
5.    Find S(v) = \{ s \in S(A) \mid (s, v) \in E \}.
6.    if ( v \in S(D) ) {
7.        /* Find the sum c of outdegrees for all sources. */
8.        c = c + outdeg(v);
9.        continue;
10.    }
11.   /* Find b = \max \{ \text{outdeg}(v) \mid v \notin S(D), S(v) = \emptyset \}. */
12.   if ( S(v) = \emptyset ) b = Math.max(b, outdeg(v));
13. }
14. return w(f_d g_d)+Math.max(b, c);
```

Fig. 2. Computing the largest weight.
and RetVal \( > \) \( \text{wt}(fdgd) \cdot c \). Choose \( u \in V \) such that
\[
\text{outdeg}(u) = \max_{v \in V} \text{outdeg}(v).
\]
The definition of \( c \) and the strict inequality \( c < b \) imply that \( u \) is not a source. Similarly, if there exists \( v \in S(v) \), then \( (v, u) \in E \) yields \( \text{outdeg}(v) \geq \text{outdeg}(u) \) by the transitivity of \( D \), and the source \( v \) contributes the summand \( \text{outdeg}(v) \) to \( c \) contradicting to \( c < b \). Therefore \( S(u) = \emptyset \). Thus, we see that \( \text{outdeg}(u) = b \).

Putting \( h = fdgd \sum_{(u,v) \in E} e_{u,v} \) we claim that now \( C = \text{BCH}_\ell(h) \) is the desired code with weight RetVal. Obviously,
\[
\text{wt}(h) = \text{wt}(fdgd) \cdot \text{outdeg}(u) = \text{wt}(fdgd) \cdot b = \text{RetVal}.
\]
Therefore \( \text{wt}(C) \leq \text{RetVal} \). Consider an arbitrary nonzero element \( y \in C \). It remains to verify that \( \text{wt}(y) \geq \text{wt}(h) \).

Since \( C = Rgdh + M_D(R)gdh \), we see that \( y \) can be expressed in the form \( y = rgdh + tdgh \) with \( r \in R \) and \( t \in M_D(R) \). Writing \( t = \sum_{(i,j) \in E} t_{i,j} e_{i,j} \), where \( t_{i,j} \in R \), we get
\[
y = rgh + \sum_{(i,j) \in E} t_{i,j} e_{i,j}.
\]
(14)

We may assume that (14) has been simplified by combining similar terms, i.e., terms corresponding to equal edges. If \( (u,v) \in E \), then \( t_{u,v} e_{u,v} = t_{u,v} e_{u,v} \) and this product can be combined with \( rgdh \). Therefore we may assume that \( t_{u,v} = 0 \). The remaining summands in \( tdgh \) do not result in edges beginning at \( u \). It follows that if \( r \neq 0 \), then \( \text{wt}(y) \geq \text{wt}(gdh) = \text{wt}(gdh) \), as required. Assume now \( r = 0 \). Since \( y = tdgh \neq 0 \), clearly there exists \( j \in V \) such that \( (j, u) \in E \) and \( t_{j,u} \neq 0 \). Therefore \( \text{wt}(y) = \text{wt}(\sum_{(i,j) \in E} t_{i,j} e_{i,j}) \geq \text{wt}(t_{j,u} \sum_{(u,v) \in E} e_{j,v}) = \text{wt}(t_{j,u} e_{j,u} dgh) = \text{wt}(gdh) \). Hence we see that
\[
\text{wt}(C) = \text{wt}(fdgd) \cdot \text{wt}(h) = \text{wt}(fdgd) \cdot \text{max}_{v \in V} \text{outdeg} v = \text{RetVal}.
\]
Thus, we have established that \( M_D(R) \) always contains a code of the form \( \text{BCH}_\ell(h) \) with weight \( \text{RetVal} \), and therefore
\[
\text{Largest}(D) \geq \text{RetVal}.
\]

Next, let us take any code \( C = \text{BCH}_\ell(h_1, \ldots, h_k) \) in \( M_D(R) \) and prove that its weight does not exceed \( \text{RetVal} \).
Choose \( x \in C \) such that \( \text{wt}(x) = \text{wt}(C) \). We can write it down as \( x = gd \sum_{(i,j) \in E} x_{i,j} e_{i,j} \).

First, consider the case where all vertices \( i \) with \( x_{i,j} \neq 0 \) are sources. Then
\[
\{(i, j) \mid x_{i,j} \neq 0\} \subseteq S,
\]
and so \( \text{wt}(x) \leq \text{wt}(fdgd) \cdot c \leq \text{RetVal} \), indeed.

Second, consider the case when there is at least one edge \( (u, v) \) such that \( x_{(u,v)} \neq 0 \) and \( u \) is not a source. Since \( u \notin S(D) \), there exists \( z \in V \) with \( (z, u) \in E \). Putting \( y = ezux \in C \), we get
\[
y = ezux = gd \sum_{(u,j) \in E} x_{u,j} e_{z,j}.
\]
Hence \( 0 \neq y \in C \) and \( \text{wt}(y) \leq \text{wt}(x) \). By the minimality of \( \text{wt}(x) \) we derive \( \text{wt}(x) = \text{wt}(y) \). On the other hand, \( \text{wt}(y) \leq \text{wt}(fdgd) \cdot c \leq \text{RetVal} \). This completes our proof of correctness of the algorithm.

In evaluating the running times we may assume that addition and multiplication in the finite field take \( O(n) \) time. Each of the lines 2, 4, and 5 can be executed in \( O(n^2) \) time. Evidently, the running time of lines 3 to 13 is \( O(n) \). Therefore the running time of the algorithm is \( O(n^3) \). \( \square \)

**Theorem 3.** For every transitive graph \( D = (V, E) \) with all loops and any \( h_1, \ldots, h_k \in M_D(D, \{0, 1\}) \), Algorithm 3 verifies whether an element \( x \in M_D(R) \) is a correct codeword of \( \text{BCH}_\ell(h_1, \ldots, h_k) \) without any errors. The running time of the algorithm is \( O(n^3k) \).
Algorithm 3. Returns true if $x = \sum_{w \in E} x_w e_w \in M_D(R)$ is a correct codeword of code $\text{BCH}_\ell(h_1, \ldots, h_k)$; otherwise returns false.

1. int[] r = new int[n];
2. for (int i = 1; i <= n; i++) {
3.     for (int j = 1; j <= n; j++) {
4.         if ( x(i,j) != 0 ) r[i] = r[i]+ x(i,j);
5.     }
6.     for( i = 1; m <= n; m++) {
7.         if ( (h_m)(i,j) !=0 ) { h[m][i] = h[m][i]+(h_m)(i,j)
8.     }
9. }
10. }
11. for (int i = 1; i <= n; i++) {
12.     if ( r[i] /\in \sum_{m,\ell,(i,\ell)} \in M_D(R) R h[m][\ell] ) { return false;
13.     }
14. }
15. return true;

Fig. 3. Testing for errors in a message.

Proof. Let us prove correctness of the algorithm. Notice that the sums $r[i] = \sum_{j=1}^{n} x(i,j)$ and $h[m][i] = \sum_{j=1}^{n} (h_m)(i,j)$ are found by the lines 4 and 7 of Algorithm 3. By Lemma 1, (6) simplifies as in (8), because $D$ contains all loops. Hence we see that

$$x = \sum_{(i,j)\in E} x_{i,j} e_{i,j} \in M_D(R)$$

belongs to $C = \text{BCH}_\ell(h_1, \ldots, h_k)$ if and only if the following inclusion is valid

$$x \in \sum_{m=1}^{k} M_D(R) h_m.$$  \hspace{1cm} (15)

It follows from (3) that (15) is satisfied if and only if the following equality holds for all $i$:

$$r[i] = \sum_{j=1}^{n} e_{i,j} \left( M_D(R) g_d \sum_{m=1}^{k} h_m \right) e_{j,j}.$$  \hspace{1cm} (16)

Since $g_d \in R$, we get $P g_d = g_d P$ for each matrix $P \in M_D(R)$. Using this, Lemma 1 and (8), we can rewrite the right-hand side of (16) as

$$\text{RHS} = g_d \sum_{\ell=1}^{n} R e_{i,\ell} \sum_{m=1}^{k} h[m][\ell].$$

Thus $x$ belongs to $C$ if and only if the following is satisfied for all $i$

$$r[i] \in g_d \sum_{j,\ell=1}^{n} R e_{i,\ell} \sum_{m=1}^{k} h[m][\ell].$$  \hspace{1cm} (17)

This is precisely the condition verified by the algorithm.

In evaluating the running times we may assume that addition and multiplication in the finite field take $O(n)$ time. Evidently, lines 3 to 10 run in $O(n^2)$ time. Line 12 can be executed in $O(n^3k)$ time using Gaussian elimination. Since all of these steps are inside a loop that repeats their execution $n$ times, we see that the total running time of the algorithm is $O(n^4k)$. $\square$
Algorithm 4. Computes the information rate of $\text{BCH}_\ell(h_1, \ldots, h_k)$ in $M_D(R)$.

1. int $u = 0$, $d = 0$;
2. for (int $i = 1$; $i <= n$; $i++)$
   3.     for (int $j = 1$; $j <= n$; $j++)$
       4.         if ($i, j$) $\in E$ $d++$;
       5.         for (int $m = 1$; $m <= n$; $m++$)
          6.             if ($h_m(i, j)$ $!=$ 0) $h[m][i] = h[m][i] + (h_m(i, j)$;
     7. }
3. }
4. for (int $i = 1$; $i <= n$; $i++)$
5. Calculate $s = \dim q \sum_{m, \ell, (i, \ell) \in E} \mathbb{R}_d h[m][\ell]$;
6. $u = u + s$;
7. }
8. }
9. return $u$ / (float) $d$;

Fig. 4. Computing the information rate.

**Theorem 4.** For every transitive graph $D = (V, E)$ with all loops, and any $h_1, \ldots, h_k \in M_D([0, 1])$, Algorithm 4 computes the information rate of the code $\text{BCH}_\ell(h_1, \ldots, h_k)$. The running time of the algorithm is $O(n^4k)$.

**Proof.** It is similar to the proof of Theorem 3 and is omitted. Line 11 is accomplished by the standard Gaussian elimination algorithm over the finite field.

4. Open questions

A graph $D$ is said to be balanced if, for all $x_1, x_2, x_3, x_4 \in V$ with $(x_1, x_2), (x_2, x_3), (x_3, x_4), (x_1, x_4) \in E$, the following conditions are equivalent

$$(x_1, x_3) \in E \iff (x_2, x_4) \in E.$$  

Clearly, every transitive graph is balanced. The following problem appears to be very difficult.

**Problem 1.** Develop algorithms analogous to Algorithms 3 and 4 for all balanced graphs and without the assumption that the graph contains all loops.

From the point of view of generating classes of codes, linear combinations of products are more general than sums of multiples in the following sense. If a code is the set of sums of multiples of some generators, then it is also equal to the set of all linear combinations of products of some generators. Therefore more codes can be represented as linear combinations of products of their generators. This makes the following problem interesting.

**Problem 2.** Develop algorithms answering analogous questions for codes equal to the set of all linear combinations of products of generating elements in $M_D(R)$.

**Problem 3.** For every given value of weight, develop an efficient combinatorial algorithm to find a code $\text{BCH}_d(h_1, \ldots, h_k)$ with this weight and the largest possible information rate.

Graph labelings provide valuable information used in several application areas, see the Dynamic Survey [11] published in the Electronic Journal of Combinatorics. It would be interesting to consider labeling the edges of the graph by the elements of a finite field. Properties of finite fields make it possible to consider combinatorial conditions for the labeling. We refer to [1–3,25,26,29,30,37–42], for recent results on combinatorial properties of labelings. This motivates the following open question.
Acknowledgements

The author is grateful to the Organisers of AWOCA 2006 for providing this excellent opportunity of discussing research results during the conference. The author would also like to express sincere appreciation to two referees for comments and corrections which have helped to seriously improve the text.

References