Searching for (near) optimal codes[⋆]

Xueliang Li[a], Yaping Mao[b], Meiqin Wei[a], and Ruihu Li[c]

[a]Center for Combinatorics and LPMC-TJKLC
Nankai University, Tianjin 300071, China

[b]Department of Mathematics, Qinghai Normal
University, Xining, Qinghai 810008, China

[c]The air force engineering University
Institute of science, Xi'an 710051, China

lxl@nankai.edu.cn; maoyaping@ymail.com;
weimeiqin8912@163.com; liruihu@aliyun.com

Abstract. Formally self-dual (FSD) codes are interesting codes and have received an enormous research effort due to their importance in mathematics and computer science. Danielsen and Parker proved that every self-dual additive code over $GF(4)$ is equivalent to a graph codes in 2006, and hence graph is an important tool for searching (near) optimal codes. In this paper, we introduce a new method of searching (near) optimal binary (formally self-dual) linear codes and additive codes from circulant graphs.

Keywords: graph code, FSD code, additive code, optimal code, circulant graph

1 Introduction

Let F_2 be the binary field, and let F_2^n denote the n-dimensional binary vector space. A k-dimensional linear subspace $\mathcal{C}$ of F_2^n is called an $[n,k]$ *linear code* and vectors in $\mathcal{C}$ are called *codewords*. Define $GF(4)=\{0,1,\omega,\omega^2\}$, where $\omega^2 = 1+\omega$. An additive code $\mathcal{C}$ over $GF(4)$ of length n is an additive subgroup of $GF(4)$. It is clear that $\mathcal{C}$ contains codewords for some $0 \leq k \leq 2n$, and can be defined by a $k \times n$ generator matrix, with entries from $GF(4)$, whose rows span $\mathcal{C}$ additively. We call $\mathcal{C}$ an $(n,2^k)$ *additive code*. The *Hamming weight* of a vector $x=(x_1,\cdots,x_n)$, denoted by $wt(x)$, is the number of its nonzero coordinates, the *Hamming distance* between two vectors x,y is equal to the Hamming weight $wt(x-y)$. The *minimum distance* d of a code is defined as the smallest possible distance between pairs of distinct codewords. An $[n,k]$ linear code with minimum distance d is denoted as an $[n,k,d]$ code and an $(n,2^k)$ additive code with minimum distance d is denoted as an $(n,2^k,d)$ code. The *weight distribution* of a code $\mathcal{C}$ is the sequence $(A_0,A_1,\cdots,A_n)$, where A_i is

[⋆] Supported by "973" program No.2013CB834204.

the number of codewords of weight i in $\mathcal{C}$. The *weight enumerator* of the code is the polynomial $W(z) = \sum_{i=0}^{n} A_i z^i$. The *inner product* of two vectors $x, y \in F_2^n$ is defined as $(x, y) = \sum_{i=1}^{n} x_i y_i$. The *dual code* of an $[n, k, d]$ code $\mathcal{C}$ is defined as $\mathcal{C}^{\perp} = \{x \in F_2^n \mid (x, y) = 0, \text{for all } y \in \mathcal{C}\}$. A binary code with the same weight distribution as its dual code is called *formally self-dual* (FSD). The *conjugation* of $x \in GF(4)$ is defined by $\bar{x} = x^2$, and the *trace map* is defined by $Tr(x) = x + \bar{x}$. The *Hermitian trace inner product* of $u = (u_1, \cdots, u_n)$ and $v = (v_1, \cdots, v_n)$. where $u, v \in GF(4)$, is defined as $u * v = Tr(u \cdot \bar{v}) = \sum_{i=1}^{n} Tr(u_i \bar{v_i}) = \sum_{i=1}^{n} (u_i v_i^2 + u_i^2 v_i)$. We define the *dual of the additive code* $\mathcal{C}$ with respect to the Hermitian trace inner product as $\mathcal{C}^{\perp} = \{u \in GF(4) \mid u * c = 0 \ for \ all \ c \in \mathcal{C}\}$. Then $\mathcal{C}$ is *self-orthogonal* if $\mathcal{C} \subseteq \mathcal{C}^{\perp}$, and $\mathcal{C}$ is *self-dual* if $\mathcal{C} = \mathcal{C}^{\perp}$.

In 2002, Tonchev [21] set up a relationship between an undirected graph and a binary linear code. Given a graph Γ_n on n vertices with adjacency matrix A_n, one can define a binary linear code with generator matrix $G = (I; A_n)$, where I is the identity matrix. Such a code is called the *linear graph code* of Γ_n, and a linear graph code is a $[2n, n]$ FSD code. In 2006, Danielsen and Parker [10] proved that every self-dual additive code over $GF(4)$ is equivalent to a graph code. Among additive codes over $GF(4)$, a graph code is an additive code over $GF(4)$ that has a generator matrix of form $\mathcal{C} = \Gamma + \omega I$, where Γ is the adjacency matrix of a simple undirected graph. An $[n, k, d]$ (or $(n, 2^k, d)$) code is *optimal* if there is no $[n, k, d+1]$ (or $(n, 2^k, d+1)$) code, and *near optimal* if there is no $[n, k, d+2]$ (or $(n, 2^k, d+2)$) code. An $[n, k, d]$ (or $(n, 2^k, d)$) code is *known best* if d attains the highest known minimum distance for $[n, k]$ (or $(n, 2^k)$) codes. For parameters of optimal codes, or lower and upper bounds on minimum distances of optimal codes, see [12].

Formally self-dual codes are important class of codes, they have connections to other mathematical structures such as block designs, lattices, modular forms, and sphere packing. Many works have been done on the FSD codes, and optimal FSD codes of length $n \leq 28$ have been classified, see [1, 2, 13, 14]. In 2002, Tonchev [21] set up a relationship between an FSD code and the adjacency matrix of an undirected graph, and showed that some interesting codes can be obtained from graphs with high degree of symmetry, such as strongly regular graphs. In 2006, Danielsen and Parker [10] proved that every self-dual additive code over $GF(4)$ is equivalent to a graph code. In 2012, Danielsen [6] focused his attention on additive codes over $GF(9)$ and transformed the problem of code equivalence into a problem of graph isomorphism. By an extension technique, they classified all optimal codes of lengths 11 and 12. In fact, computer searching reveals that circulant graph codes usually contain many strong codes, and some of these codes have highly regular graph representations, see [22]. In [6], Danielsen obtained some optimal additive codes from circulant graphs in 2005. Later, Varbanov investigated additive circulant graph codes over $GF(4)$, see [22]. Recently, finding optimal codes from graphs has received a wide attention of many researchers, see [6–11, 15, 21, 22]. Inspired by these works, we discuss the construction of (near) optimal FSD codes and additive codes from undirected circulant graphs in this paper.

The paper is organized as follows. Section 2 recalls some concepts in graph theory. In Section 3, we propose a new method to find (near) optimal binary linear codes from circulant graphs, and construct some (near) optimal or known best binary linear codes by using this method. In Section 4, we propose a new method to find additive optimal codes from circulant graphs.

2 Preliminaries

We introduce some concepts of graph theory for latter use in this paper, for more details please see [5]. An *undirected graph* $\Gamma = (V, E)$ is a set $V(\Gamma) = \{v_1, v_2, \cdots, v_n\}$ of vertices together with a collection $E(\Gamma)$ of edges, where each edge is an unordered pair of vertices. The vertices v_i and v_j are *adjacent* if $\{v_i, v_j\}$ is an edge. Then v_j is a *neighbour* of v_i. All the neighbours of a vertex v_i in Γ form the *neighbourhood* of v_i, and it is denoted by $N_\Gamma(v_i)$. The *degree of a vertex* v is the number of vertices adjacent to v. A graph is *regular* of degree k if all vertices have the same degree k. For a graph $\Gamma = (V, E)$, suppose that V' is a nonempty subset of V. The subgraph of Γ whose vertex set is V' and whose edge set is the set of those edges of Γ that have both ends in V' is called the *subgraph of* Γ *induced by* V', denoted by $\Gamma[V']$. We say that $\Gamma[V']$ is an induced subgraph of Γ. The *adjacency matrix* $A = (a_{ij})$ of $\Gamma = (V, E)$ is a symmetric $(0, 1)$-matrix defined as follows: $a_{i,j} = 1$ if the i-th and j-th vertices are adjacent, and $a_{i,j} = 0$ otherwise.

Circulant graphs and their various applications are the objects of intensive study in computer science and discrete mathematics, see [3, 4, 16, 19]. Recently, Monakhova published a survey paper on this subject, see [18]. Let $S = \{a_1, a_2, \cdots, a_k\}$ be a set of integers such that $0 < a_1 < \cdots < a_k < \frac{n+1}{2}$, and let the vertices of an n-vertex graph be labelled as $0, 1, 2, \cdots, n-1$. Then the *ciculant graph* $C(n, S)$ has $i \pm a_1, i \pm a_2, \cdots, i \pm a_k$ $(mod\ n)$ adjacent to each vertex i. A *circulant matrix* is obtained by taking an arbitrary first row, and shifting it cyclically one position to the right in order to obtain successive rows. We say that a circulant matrix is generated by its first row. Formally, if the first row of an n-by-n circulant matrix is $a_0, a_1, \cdots, a_{n-1}$, then the $(i, j)^{th}$ element is a_{j-i}, where subscripts are taken modulo n. The term circulant graph arises from the fact that the adjacency matrix for such a graph is a circulant matrix.

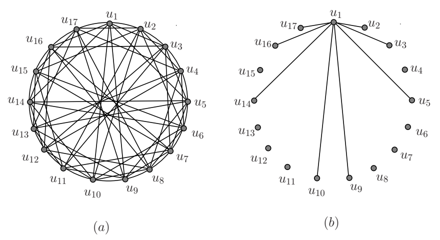

Figure 1: (a) The $(4, 4)$-Ramsey graph Γ; (b) the edge-induced graph $\Gamma(E_1)$.

For example, the $(4,4)$-*Ramsey graph* $\mathit{\Gamma}$ (see Figure 1) is a famous circulant graph, which can be obtained by regarding the vertices as elements of the field of integers modulo 17, and joining two vertices if their difference is a quadratic residue of 17 (either 1, 2, 4, 8, 9, 13, 15 or 16). For the vertex u_1, we have $E_1 = \{u_1u_2, u_1u_3, u_1u_5, u_1u_9, u_1u_{10}, u_1u_{14}, u_1u_{16}, u_1u_{17}\} \subseteq E(\mathit{\Gamma})$ and a vector $\alpha_{17} = (0,1,1,0,1,0,0,0,1,1,0,0,0,1,0,1,1)$. It is clear that the adjacency matrix A_{17} of the $(4,4)$-Ramsey graph is generated by α_{17}, where

$$A_{17} = \begin{pmatrix} 01101000110001011 \\ 10110100011000101 \\ 11011010001100010 \\ \dots\dots\dots\dots \end{pmatrix}.$$

3 New binary linear codes searching from circulant graphs

In this section, we discuss the construction of binary linear codes from circulant graphs. Since a circulant graph $\mathit{\Gamma}_n$ can be uniquely determined by its adjacency matrix A_n, or by the vector α_n correponding to E_1 ($E_1 \subseteq E(\mathit{\Gamma}_n)$), whose elements are incident with the vertex $u_1 \in V(\mathit{\Gamma}_n)$. We will make no difference of α_n, A_n or a circulant graph $\mathit{\Gamma}_n$, and simply say a circulant graph $\mathit{\Gamma}_n$ with vector α_n or a vector α_n of a circulant graph $\mathit{\Gamma}_n$. And in this section, we denote the binary linear code with generator matrix $G = (I, A_n)$ by $\mathcal{C}_n$. According to the relation between a graph code and the adjacency matrix of an undirected graph introduced by [21], we can get a $[34,17,8]$ optimal FSD code from the matrix $(I; A_{17})$, where A_{17} is the adjacency matrix of the $(4,4)$-Ramsey graph.

In [6], Danielsen got some optimal additive codes. One of them is the optimal additive code $(30, 2^{30}, 12)$ obtained from the vector
$\beta_{30} = (\omega,0,1,1,0,0,0,0,1,1,0,1,1,1,1,1,1,1,1,1,0,1,1,0,0,0,0,1,1,0)$,
which corresponds to a circulant graph of order 30 with vector
$\alpha_{30} = (0,0,1,1,0,0,0,0,1,1,0,1,1,1,1,1,1,1,1,1,0,1,1,0,0,0,0,1,1,0)$.
The graph code $\mathcal{C}_{30}$ with generator matrix $G = (I; A_{30})$ is a $[60,30,12]$ code, where A_{30} is the circulant matrix generated by α_{30}. The code $\mathcal{C}_{30}$ is a known best FSD code. The weight enumerator of $\mathcal{C}_{30}$ is

$$\begin{aligned} W_{\mathcal{C}_{30}}(z) = 1 &+ 4060z^{12} + 24360z^{14} + 294930z^{16} + 1728400z^{18} + 7758400z^{20} \\ &+26336640z^{22} + 67403540z^{24} + 129936240z^{26} + 192974265z^{28} \\ &+220819632z^{30} + 192974265z^{32} + 129936240z^{34} + 67403540z^{36} \\ &+26336640z^{38} + 7758660z^{40} + 1728400z^{42} + 294930z^{44} + 24360z^{46} \\ &+4060z^{48} + z^{60}. \end{aligned}$$

However, using known circulant graphs in the literature, we only get a few binary linear codes that are optimal, near optimal or known best. So, we need to study a new method for designing good binary linear codes from circulant graphs. To achieve this goal, we give some notations first.

Denote by L_n the highest known minimum distance of all $[2n, n]$ codes. For a graph Γ_n with vector α_n, we let $\mathcal{C}_n$ be its graph code and d_n be the distance of $\mathcal{C}_n$. If $d_n \geq L_n$, then $\mathcal{C}_n$ is a good FSD code. If $d_n < L_n$, then $\mathcal{C}_n$ is a "poor" code and α_n is a "poor" vector. For a graph Γ_n with "poor" vector α_n, we manage to find a new vector α'_n that gives a binary linear code $\mathcal{C}'_n = [2n, n, d'_n]$ such that $\mathcal{C}'_n$ is better than the code $\mathcal{C}_n$. The idea of finding a new vector α'_n from a "poor" vector $\alpha_n = (b_1, b_2, \cdots, b_n)$ is introduced as follows:

1) Let A_n be the circulant matrix generated by α_n and denote $G_n = (I; A_n)$. Then the j-th row of G_n is

$$\begin{aligned} g_j &= (e_j \mid a_j) \\ &= (0, \cdots, 0, 1, 0, \cdots, 0 \,|\, a_{j,1}, a_{j,2}, \cdots, a_{j,j-1}, a_{j,j}, a_{j,j+1}, \cdots, a_{j,n}) \\ &= (0, \cdots, 0, 1, 0, \cdots, 0 \,|\, b_{n-j+2}, b_{n-j+3}, \cdots, b_n, b_1, b_2, \cdots, b_{n-j+1}). \end{aligned}$$

2) We call codeword $\beta \in \mathcal{C}_n$ "bad" if $wt(\beta) < L_n$. Find "bad" codewords $\beta_1, \beta_2, \cdots, \beta_m$ (if exist) such that their weights are $d_n, d_n+1, \cdots, d_n+(m-1)$, where $m = L_n - d_n$. If there is no codeword with weight $d_n + i$ $(0 \leq i \leq m-1)$, then β_{i-1} is not under consideration.

3) Let $\beta_i = g_{j_1} + g_{j_2} + \cdots + g_{j_r} = (u_i \mid v_i)$, where $u_i, v_i \in F_2^n$ and $j_1, j_2, \cdots, j_r \in \{1, 2, \cdots, n\}$.

4) From $g_{j_l} = (e_{j_l} \mid a_{j_l})$, one can define a column vector set $\{\Lambda_i : i = 1, 2, \cdots, n\}$, where $\Lambda_i = (\lambda_{i,j_1}, \lambda_{i,j_2}, \cdots, \lambda_{i,j_r})$ and $\lambda_{i,j_1} = \sum_{l=1}^{r} a_{j_\ell, j_1+(i-1)}$, $\lambda_{i,j_2} = \sum_{l=1}^{r} a_{j_\ell, j_2+(i-1)}$, $\cdots$, $\lambda_{i,j_r} = \sum_{l=1}^{r} a_{j_\ell, j_r+(i-1)}$. Note that all the additions are proceeding modular n.

5) Using these Λ_i to set up a standard for adjusting a "poor" vector $\alpha_n = (b_1, b_2, \cdots, b_n)$ to α'_n.

The above method can be realized by the following *Algorithm* 1:

Step 1. Give a circulant graph Γ_n with vector $\alpha_n = (b_1, b_2, \cdots, b_n)$, and generate $G_n = (I; A_n)$.

Step 2. Calculate the distance d_n of binary linear code $\mathcal{C}_n$ (*Algorithm 1-1*). If $d_n \geq L_n$, then $\mathcal{C}_n$ is a good binary linear code, and stop. Or else, go to Step 3.

Step 3. Do adjustments of the elements of the "poor" vector α_n as follows.

Step 3.1. Find "bad" codewords $\beta_1, \beta_2, \cdots, \beta_m$ such that their weights are $d_n, d_n+1, \cdots, d_n+(m-1)$ by *Algorithm 1-2*, where $m = L_n - d_n$. If there is no codeword with weight $d_n + i$ $(0 \leq i \leq m-1)$, then β_{i-1} is not under consideration.

Step 3.2. For each β_i $(0 \leq i \leq m-1)$, we can find a combination of β_i by *Algorithm 1-2*. Suppose $\beta_i = g_{j_1} + g_{j_2} + \cdots + g_{j_r} = (u_i \mid v_i)$, where $u_i, v_i \in F_2^n$ and $j_1, j_2, \cdots, j_r \in \{1, 2, \cdots, n\}$ and

$$\begin{aligned} g_{j_k} &= (e_{j_k} \mid a_{j_k}) \\ &= (0, \cdots, 0, 1, 0, \cdots, 0 \,|\, a_{j_k,1}, a_{j_k,2}, \cdots, a_{j_k,j_k-1}, a_{j_k,j_k}, a_{j_k,j_k+1}, \cdots, a_{j_k,n}) \\ &= (0, \cdots, 0, 1, 0, \cdots, 0 \,|\, b_{n-j_k+2}, b_{n-j_k+3}, \cdots, b_n, b_1, b_2, \cdots, b_{n-j_k+1}). \end{aligned}$$

Step 3.3. Determine whether each element 1 of the generator vertex α_n is a "bad" element in the following way (since $b_1 = 0$, we begin with element b_2). If $b_2 = 1$, then $a_{j_1,j_1+1} = a_{j_2,j_2+1} = \cdots = a_{j_r,j_r+1} = b_2 = 1$. We calculate the exact value $\lambda_{2,j_1} = \sum_{\ell=1}^{r} a_{j_\ell,j_1+1}, \lambda_{2,j_2} = \sum_{\ell=1}^{r} a_{j_\ell,j_2+1}, \cdots, \lambda_{2,j_r} = \sum_{\ell=1}^{r} a_{j_\ell,j_r+1}$.

Note that $\lambda_{2,j_k} = 0$ or $\lambda_{2,j_k} = 1$ $(1 \le k \le r)$. Consider the set $\varLambda_2 = \{\lambda_{2,j_1}, \lambda_{2,j_2}, \cdots, \lambda_{2,j_r}\}$. If the number of elements with value "0" in $\varLambda_2$ is larger than the number of elements with value "1", then the element b_2 is called a *"bad" element* of the generator vector α_n. If b_2 is a "bad" element, then we change $b_2 = 1$ into $b'_2 = 0$ and obtain a new vector $\alpha'_n = (b_1, b'_2, \cdots, b_n)$. Then we return to Step 1. If b_2 is not a "bad" element or $b_2 = 0$, then we consider b_3 and continue to determining whether b_3 is a "bad" element. The procedure terminates till b_n has been considered.

Algorithm 1-1: Minimum distance of a binary linear code

Input: The value of n, the generator vector α_n of a binary linear code $\mathcal{C}_n$
Objective: The minimum distance of binary linear code $\mathcal{C}_n$
1. Input the value of n, the generator vector $\alpha_n = (b_1, b_2, \cdots, b_n)$;
2. Obtain the generator matrix $G = (I; A_n)$ of the binary linear code $\mathcal{C}_n$;
3. Get the minimum distance of the binary linear code $\mathcal{C}_n$.

For example, let $n = 19$ and $\alpha_n = (0,1,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1,1)$. The algorithm details are stated as follows:

Program:
```
    $n = 19$;
    $a = [0,1,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1,1]$
    $m = matrix(GF(2), [[a[(i-k)\%n]$ for $i$ in $[0..(n-1)]]$ for $k$ in $[0..n-1]])$;
    $f = lambda\ s : sum(map(lambda\ x : m[x], s))$;
    $s = []$;
    for $k$ in $[1..8]$:
        $t = min([list(i).count(1)$ for $i$ in Subsets$(range(n), k).map(f)])$;
        $s+ = [t]$;
        print $k, t$;
```

Output:
```
$s_i$ :  1 2 3 4 5 6 7 8
$s'_i$ : 8 6 4 2 2 4 2 2
```
Result: The elements of the first row $(s_1, s_2, \cdots, s_8)$ are the contribution of the matrix I for the weight of a codeword. The elements of the second row $(s'_1, s'_2, \cdots, s'_8)$ are the contribution of the matrix A_{19} for the weight of a codeword. The value of $\min\{s_i + s'_i \mid 1 \le i \le 8\} = 6$ is the minimum weight of the code $\mathcal{C}_{19}$ and then the minimum distance of the code $\mathcal{C}_{19}$ is also 6.

Algorithm 1-2: "Bad" codewords and their combinations

Input: The value of n, the generator vector α_n of a binary linear code $\mathcal{C}_n$
Objective: "Bad" codewords and their combinations
1. Input the value of n, the generator vector $\alpha_n = (b_1, b_2, \cdots, b_n)$;
2. Obtain the generator matrix $G = (I; A_n)$ of the binary linear code $\mathcal{C}_n$;
3. Get "bad" codewords $\beta_1, \beta_2, \cdots, \beta_m$ and a combination of each β_i $(1 \le i \le m)$.

For example, let $n = 19$ and $\alpha_n = (0,1,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1,1)$.

The algorithm details are stated as follows:

```
Program:
    $n = 19;$
    $a = [0,1,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1,1]$
    $m = matrix(GF(2), [[a[(i-k)\%n]$ for $i$ in $[0..(n-1)]]$ for $k$ in $[0..n-1]]);$
    $f = lambda\ s : sum(map(lambda\ x : m[x], s));$
    $g = lambda\ s : str(sorted(map(lambda\ x : x+1, s))).replace('[','').replace(']','');$
    $s = [];$
    for $k$ in $[1..8]$:
        $t = min([(i, list(f(i)).count(1))$ for $i$ in Subsets$(range(n), k)]$,
        $key = lambda\ x : x[-1]);$
        $s+ = [t];$
        print $k, t[1], g(t[0]);$
```

```
Output: $s_i$  $s'_i$  $\{j_1, j_2, \cdots, j_r\}$  (as defined in Step 3.2)
        1   8    {1}
        2   6    {1, 9}
        3   4    {1, 4, 12}
        4   2    {1, 2, 6, 16}
        5   2    {1, 2, 8, 11, 14}
        6   4    {1, 2, 3, 4, 6, 13}
        7   2    {1, 2, 4, 6, 7, 10, 17}
        8   2    {1, 2, 3, 6, 7, 8, 12, 16}
```

Now we show how to use our *Algorithm 1* for searching $[38, 19, 8]$ codes.

Step 1. Among all graphs with 19 vertices, we consider the graph Γ_{19}, which can be generated by the edge set $E_1 = \{u_1u_2, u_1u_3, u_1u_5, u_1u_{10}, u_1u_{11}, u_1u_{16}, u_1u_{18}, u_1u_{19}\}$, and then $\alpha_{19} = (0,1,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1,1)$. Obviously, $b_2 = b_3 = b_5 = b_{10} = b_{11} = b_{16} = b_{18} = b_{19} = 1$.

Step 2. From the Code Tables, we know that the lower bound of the minimum distance of linear code $[38, 19]$ over $GF(2)$ is 8, that is, $L_{19} = 8$. By *Algorithm 1-1*, we obtain that the minimum distance d_{19} of the code $(I; A_{19})$ is just 6, that is, $d_{19} = 6$. Clearly, $6 = d_{19} < L_{19} = 8$.

Step 3. Obviously, $m = L_{19} - d_{19} = 2$.

Step 3.1. From *Algorithm 1-2*, we find two "bad" codewords

$$\begin{aligned}\beta_1 = (&1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0\\ &\mid 0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0),\\ \beta_2 = (&1,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0\\ &\mid 0,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0).\end{aligned}$$

such that their weights are 6 and 7, that is, $wt(\beta_1) = 6$ and $wt(\beta_2) = 7$.

Step 3.2. For β_1, we can find a combination of $\beta_1 = \alpha_{19,1} + \alpha_{19,2} + \alpha_{19,6} + \alpha_{19,16}$ by *Algorithm 2-2*, where

$$\begin{aligned}
\alpha_{19,1} &= (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
&\qquad\qquad\qquad\qquad |\,0,1,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1,1), \\
\alpha_{19,2} &= (0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
&\qquad\qquad\qquad\qquad |\,1,0,1,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1), \\
\alpha_{19,6} &= (0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
&\qquad\qquad\qquad\qquad |\,0,1,0,1,1,0,1,1,0,1,0,0,0,0,1,1,0,0,0), \\
\alpha_{19,16} &= (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0 \\
&\qquad\qquad\qquad\qquad |\,1,0,0,0,0,1,1,0,0,0,0,1,0,1,1,0,1,1,0).
\end{aligned}$$

Note that $r = 4$, $j_1 = 1$, $j_2 = 2$, $j_3 = 6$ and $j_4 = 16$.

For β_2, we can find a combination of $\beta_2 = \alpha_{19,1} + \alpha_{19,4} + \alpha_{19,12}$ by *Algorithm 2-2*, where

$$\begin{aligned}
\alpha_{19,1} &= (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
&\qquad\qquad\qquad\qquad |\,0,1,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1,1), \\
\alpha_{19,4} &= (0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
&\qquad\qquad\qquad\qquad |\,0,1,1,0,1,1,0,1,0,0,0,0,1,1,0,0,0,0,1), \\
\alpha_{19,12} &= (0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0 \\
&\qquad\qquad\qquad\qquad |\,0,1,1,0,0,0,0,1,0,1,1,0,1,1,0,1,0,0,0).
\end{aligned}$$

Note that $r = 3$, $j_1 = 1$, $j_2 = 4$ and $j_3 = 12$.

Step 3.3. Recall that $\alpha_{19} = (0,1,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1,1)$ and $b_2 = b_3 = b_5 = b_{10} = b_{11} = b_{16} = b_{18} = b_{19} = 1$. Since $b_2 = 1$, we consider whether b_2 is a "bad" element in α_{19}.

For β_1, since $r = 4$, $j_1 = 1$, $j_2 = 2$, $j_3 = 6$ and $j_4 = 16$, we have $a_{1,2} = a_{2,3} = a_{6,7} = a_{16,17} = b_2 = 1$, and

$$\begin{aligned}
\lambda_{2,j_1} &= \textstyle\sum_{\ell=1}^{r} a_{j_\ell,j_1+1} = a_{1,2} + a_{2,2} + a_{6,2} + a_{16,2} = 0, \\
\lambda_{2,j_2} &= \textstyle\sum_{\ell=1}^{r} a_{j_\ell,j_2+1} = a_{1,3} + a_{2,3} + a_{6,3} + a_{16,3} = 0, \\
\lambda_{2,j_3} &= \textstyle\sum_{\ell=1}^{r} a_{j_\ell,j_3+1} = a_{1,7} + a_{2,7} + a_{6,7} + a_{16,7} = 0, \\
\lambda_{2,j_4} &= \textstyle\sum_{\ell=1}^{r} a_{j_\ell,j_4+1} = a_{1,17} + a_{2,17} + a_{6,17} + a_{16,17} = 0.
\end{aligned}$$

For β_2, since $r = 3$, $j_1 = 1$, $j_2 = 4$ and $j_3 = 12$, we have $a_{1,2} = a_{4,5} = a_{12,13} = b_2 = 1$. Then

$$\begin{aligned}
\lambda_{2,j_1} &= \textstyle\sum_{\ell=1}^{r} a_{j_\ell,j_1+1} = a_{1,2} + a_{4,2} + a_{12,2} = 1, \\
\lambda_{2,j_2} &= \textstyle\sum_{\ell=1}^{r} a_{j_\ell,j_2+1} = a_{1,5} + a_{4,5} + a_{12,5} = 0, \\
\lambda_{2,j_3} &= \textstyle\sum_{\ell=1}^{r} a_{j_\ell,j_3+1} = a_{1,13} + a_{4,13} + a_{12,13} = 0.
\end{aligned}$$

It is clear that the number of elements with value "0" in Λ_2's is larger than the number of elements with value "1", then the element b_2 is called a *"bad" element* of the generator vector α_n. We change $b_2 = 1$ into $b_2' = 0$ and obtain a new vector $\alpha_{19}' = (0,0,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1,1)$. Then we return to Step 1.

Let us now investigate the linear code $\mathcal{C}_{19}'$ with generator matrix $G = (I; A_{19}')$. By *Algorithm 1-1*, we get that the minimum distance d_{19}' of the linear code $\mathcal{C}_{19}'$

is 8. The code $\mathcal{C}'_{19}$ is a near optimal and also known best binary linear code over $GF(2)$. The weight enumerator of the code $\mathcal{C}'_{19}$ is

$$\begin{aligned}W_{\mathcal{C}'_{19}}(z) = 1 &+ 133z^8 + 2052z^{10} + 10108z^{12} + 36575z^{14} + 85595z^{16}\\ &+127680z^{18} + 127680z^{20} + 85595z^{22} + 36575z^{24} + 10108z^{26}\\ &+2052z^{28} + 133z^{30} + z^{38}.\end{aligned}$$

With the above approach and algorithms, we can also find three other near optimal binary linear $[38, 19, 8]$ codes by the generator matrices $G = (I; A''_{19})$, $G = (I; A'''_{19})$ and $G = (I; A''''_{19})$. The circulant matrices A''_{19}, A'''_{19} and A''''_{19} are separately generated by

$\alpha''_{19} = (0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1)$,
$\alpha'''_{19} = (0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1)$,
$\alpha''''_{19} = (0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0)$.

The weight enumerator of the code $\mathcal{C}''_{19}$ is

$$\begin{aligned}W_{\mathcal{C}''_{19}}(z) = 1 &+ 190z^8 + 1767z^{10} + 10507z^{12} + 36860z^{14} + 84341z^{16}\\ &+128478z^{18} + 128478z^{20} + 84341z^{22} + 36860z^{24} + 10507z^{26}\\ &+1767z^{28} + 190z^{30} + z^{38}.\end{aligned}$$

One can also check that the weight enumerators of the codes $\mathcal{C}'''_{19}$ and $\mathcal{C}''''_{19}$ are equal to the ones of $\mathcal{C}''_{19}$ and $\mathcal{C}'_{19}$, respectively.

At the end of this section, we list some more binary linear codes constructed from known circulant graphs by applying *Algorithm 1.*

1. In [6], Danielsen got a $(15, 2^{15})$ additive code from the vector $(\omega, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0)$, which corresponds to a circulant graph of order 15 with vector $\alpha_{15} = (0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0)$. Its graph code $\mathcal{C}_{15} = [30, 15, 6]$ is a poor code. Applying *Algorithm* 1, we obtain a new vector $\alpha'_{15} = (0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0)$. The code $\mathcal{C}'_{15} = [30, 15, 8]$ generated by α'_{15} is an optimal binary linear code. The weight enumerator of this code $\mathcal{C}'_{15}$ is

$$\begin{aligned}W_{\mathcal{C}'_{15}}(z) = 1 &+ 450z^8 + 1848z^{10} + 5040z^{12} + 9045z^{14} + 9045z^{16}\\ &+5040z^{18} + 1848z^{20} + 450z^{22} + z^{30}.\end{aligned}$$

2. Extending vector $\alpha_{17} = (0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1)$ of the $(4, 4)$-Ramsey graph, one can obtain a vector of length 19. For example, let $\alpha_{19} = (0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1)$, which corresponds to a circulant graph of order 19. It corresponds to the linear graph code $\mathcal{C}_{19} = [38, 19, 6]$, which is a poor code. By *Algorithm* 1, we obtain four new vectors as follows:

$\alpha'_{19} = (0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1)$,
$\alpha''_{19} = (0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0)$,
$\alpha'''_{19} = (0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1)$,
$\alpha''''_{19} = (0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1)$.

The codes $\mathcal{C}'_{19}$ and $\mathcal{C}''_{19}$ are $[38, 19, 8]$ codes with the same weight enumerators. The weight enumerator of the code $\mathcal{C}'_{19}$ is

$$\begin{aligned}W_{\mathcal{C}'_{19}}(z) = {} & 1+133z^{8}+2052z^{10}+10108z^{12}+36575z^{14}+85595z^{16}\\ & +127680z^{18}+127680z^{20}+85595z^{22}+36575z^{24}+10108z^{26}\\ & +2052z^{28}+133z^{30}+z^{38}.\end{aligned}$$

The codes $\mathcal{C}'''_{19}$ and $\mathcal{C}''''_{19}$ are $[38, 19, 8]$ codes with the same weight enumerators. The weight enumerator of the code $\mathcal{C}'''_{19}$ is

$$\begin{aligned}W_{\mathcal{C}'''_{19}}(z) = {} & 1+190z^{8}+1767z^{10}+10507z^{12}+36860z^{14}+84341z^{16}\\ & +128478z^{18}+128478z^{20}+84341z^{22}+36860z^{24}+10507z^{26}\\ & +1767z^{28}+190z^{30}+z^{38}.\end{aligned}$$

These four binary linear codes are all near optimal and known best.

3. In addition, we consider graphs with large number of vertices by similar approach. Let $\mathcal{C}_{25,1}$ and $\mathcal{C}_{25,2}$ be the two codes generated by vectors
$\alpha^1_{25} = (0,1,1,0,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1,0,1,0,1,1)$ and
$\alpha^2_{25} = (0,1,1,0,1,0,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1,0,1,1)$, respectively. These two codes are both poor codes. Then by *Algorithm* 1, we find
$\alpha'_{25} = (0,1,0,0,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1,0,1,0,1,1)$, and
$\alpha''_{25} = (0,1,1,0,1,0,1,0,1,0,0,0,0,1,1,0,0,0,0,1,0,1,0,0,1)$. Both α'_{25} and α''_{25} give binary linear codes known best. Let $\mathcal{C}'_{25}$ and $\mathcal{C}''_{25}$ be codes of α'_{25} and α''_{25}, respectively. It is easy to check that $\mathcal{C}'_{25}$ and $\mathcal{C}''_{25}$ are $[50, 25, 10]$ codes, and the weight enumerator of the code $\mathcal{C}'_{25}$ and $\mathcal{C}''_{25}$ is

$$\begin{aligned}W_{\mathcal{C}'_{25}}(z) = {} & 1+225z^{10}+1250z^{11}+3825z^{12}+11525z^{13}+28050z^{14}+64005z^{15}\\ & +147075z^{16}+294975z^{17}+535075z^{18}+9111100z^{19}+1409205z^{20}\\ & +1999925z^{21}+2642200z^{22}+3219675z^{23}+3623325z^{24}+377243z^{25}\\ & +3621975z^{26}+3216050z^{27}+2643475z^{28}+2009175z^{29}+1408010z^{30}\\ & +904475z^{31}+535400z^{32}+292725z^{33}+147525z^{34}+68880z^{35}\\ & +27975z^{36}+9775z^{37}+3500z^{38}+1125z^{39}+375z^{40}+125z^{41}\\ = {} & W_{\mathcal{C}''_{25}}(z).\end{aligned}$$

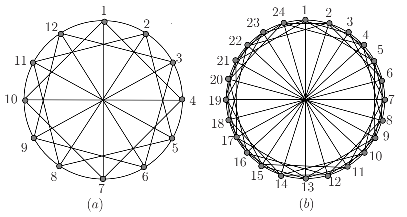

Figure 2: (a) A 5-valent graph $\varGamma_{12}$; (b) Circulant graph $\varGamma_{24}$.

4 New additive codes searching from circulant graphs

In fact, Glynn et al. [11] obtained an optimal code from a circulant graph, called a 5-valent graph. Recall that $\mathit{\Gamma}_{12}$ is a circulant graph of order 12; see Figure 2 (a). Let $V(\mathit{\Gamma}_{12}) = \{u_1, u_2, \cdots, u_{12}\}$. For the vertex u_1, we let $E_1 = \{v_1v_2, v_1v_4, v_1v_7, v_1v_{10}, v_1v_{12}\} \subseteq E(\mathit{\Gamma}_{12})$. For the vertex u_2, we just rotate the above vertices and edges, that is, we only permit the existence of the edge set $E_2 = \{v_2v_3, v_2v_4, v_2v_6, v_2v_{10}, v_2u_{12}\} \subseteq E(G)$. For each vertex $u_i \in V(\mathit{\Gamma}) \setminus \{u_1, u_2\} = \{u_3, u_4, \cdots, u_{12}\}$, we can also obtain an edge set E_i $(3 \leq i \leq 17)$. Observe that $E(\mathit{\Gamma}_{12}) = \bigcup_{i=1}^{12} E_i$. The adjacency matrix of the graph $\mathit{\Gamma}_{12}$ is the following circulant matrix

$$A_{12} = \begin{pmatrix} 010100100101 \\ 101010010010 \\ 010101001001 \\ \dots\dots\dots\dots\dots \end{pmatrix}.$$

The above matrix can also be obtained by vector $\alpha_{12} = (0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1)$. Observe that this vector just corresponds to the set E_1 of edges, which is an expression of the adjacency relation for the vertex u_1. We conclude that a 5-valent graph can be determined by the edge set E_1, and the adjacency matrix of this graph is determined by the above vector α_{12}. Furthermore, the matrix $A'_{12} = A_{12} + \omega I$ is also a circulant matrix, which can be obtained by vector $\alpha'_{12} = (\omega, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1)$. From the matrix $A_{12}+\omega I$, we can get a graph code $\mathcal{C}_{12}$. From the Code Tables, we know that $\mathcal{C}_{12}$ is an optimal $(12, 2^{12}, 6)$ additive code over $GF(4)$. The above statement suggests the following method for finding (near) optimal additive codes.

Step 1. Given an even integer n. Denote by L_n the lower bound of the additive code $(n,2^n)$ over $GF(4)$. From the Code Tables, we find the exact value of L_n for given n. We now construct a circulant graph $\mathit{\Gamma}_n$ by a set E_1 of edges as follows.

Step 1.1. Arrange the vertices from $V(\mathit{\Gamma}_n) = \{u_1, u_2, \cdots, u_n\}$ in a circular order.

Step 1.2. Determine the set E_1 of edges satisfying $|E_1| = L_n - 1$ or $|E_1| = L_n + 1$, where $E_1 = \{u_iu_1 \mid u_i \in N_{\mathit{\Gamma}_n}(u_1)\}$. If $|E_1| = L_n + 1$, then

$$\begin{aligned} E_1 &= \{u_1u_2, u_1u_3\} \cup \{u_1u_{3+2\cdot 1}, u_1u_{3+2\cdot 2}, \cdots, u_1u_{3+2\cdot\frac{L_n-4}{2}},\} \cup \{u_1u_{\frac{n}{2}+1}\} \\ &\quad \cup \{u_1u_{n-1-2\cdot\frac{L_n-4}{2}}, \cdots, u_1u_{n-1-2\cdot 2}, u_1u_{n-1-2\cdot 1}\} \cup \{u_1u_n, u_1u_{n-1}\} \\ &= \{u_1u_2, u_1u_3\} \cup \{u_1u_5, u_1u_7, \cdots, u_1u_{L_n-1}\} \cup \{u_1u_{\frac{n}{2}+1}\} \\ &\quad \cup \{u_1u_{n-L_n+3}, \cdots, u_1u_{n-5}, u_1u_{n-3}\} \cup \{u_1u_n, u_1u_{n-1}\}. \end{aligned}$$

If $|E_1| = L_n - 1$, then

$$\begin{aligned} E_1 &= \{u_1u_2, u_1u_3\} \cup \{u_1u_{3+2\cdot 1}, u_1u_{3+2\cdot 2}, \cdots, u_1u_{3+2\cdot\frac{L_n-6}{2}},\} \cup \{u_1u_{\frac{n}{2}+1}\} \\ &\quad \cup \{u_1u_{n-1-2\cdot\frac{L_n-6}{2}}, \cdots, u_1u_{n-1-2\cdot 2}, u_1u_{n-1-2\cdot 1}\} \cup \{u_1u_n, u_1u_{n-1}\} \\ &= \{u_1u_2, u_1u_3\} \cup \{u_1u_5, u_1u_7, \cdots, u_1u_{L_n-3}\} \cup \{u_1u_{\frac{n}{2}+1}\} \\ &\quad \cup \{u_1u_{n-L_n+5}, \cdots, u_1u_{n-5}, u_1u_{n-3}\} \cup \{u_1u_n, u_1u_{n-1}\}. \end{aligned}$$

Step 2. By the edge set E_1, we write the vector α_n corresponding to E_1. If $|E_1| = L_n + 1$, then

$$\begin{array}{lcccccccccccccccccccccccc} & u_2 & u_3 & & u_5 & & & u_{L_n-1} & & & & & u_{\frac{n}{2}+1} & & & & & u_{n-L_n+3} & & & u_{n-3} & & u_{n-1} & u_n \\ \alpha_n = (0 & 1 & 1 & 0 & 1 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 1 & 0 & 1 & 1). \end{array}$$

If $|E_1| = L_n - 1$, then

$$\begin{array}{lcccccccccccccccccccccccc} & u_2 & u_3 & & u_5 & & & u_{L_n-3} & & & & & u_{\frac{n}{2}+1} & & & & & u_{n-L_n+5} & & & u_{n-3} & & u_{n-1} & u_n \\ \alpha_n = (0 & 1 & 1 & 0 & 1 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 1 & 0 & 1 & 1). \end{array}$$

Step 3. Change the first component of the vector α_n into ω. Denote by α'_n the new vector. We generate a circulant matrix A'_n from α'_n,
$\alpha'_n = (\omega, 1, 1, 0, 1, 0, 1, \cdots, 0, 1, 0, 0, \cdots, 0, 1, 0, 0, \cdots, 0, 1, 0, 1, 0, \cdots, 1, 0, 1, 1)$.

Step 4. By *Algorithm 2*, we obtain the minimum distance d_n of the additive code $\mathcal{C}_n$ and determine whether $d_n = L_n$. If so, the code $\mathcal{C}_n$ is a (near) optimal or at least known best additive code.

Below is an algorithm (running in SAGE). For more details, we refer to [20].

Algorithm 2: Minimum distance of a circulant graph code

Input: The value of n, the generator vector α_n of a circulant graph code $\mathcal{C}_n$
Objective: The minimum distance of the circulant graph code $\mathcal{C}_n$
1. Input the value of n, the generator vector $\alpha_n = (b_1, b_2, \cdots, b_n)$;
2. Obtain the generator matrix G of the circulant graph code $\mathcal{C}_n$;
3. Get the minimum distance of the circulant graph code $\mathcal{C}_n$.

For example, let $n = 24$ and $\alpha_n = (\omega, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,$
$0, 0, 0, 1, 0, 1, 1)$. The algorithm details are stated as follows:
Program:

```
    F.<x> = GF(4,'x')
    n = 24;
    a = [x,1,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,1]
    m = matrix(F,n,n,[[a[(i-k)%n] for i in [0..(n-1)]] for k in [0..n-1]]);
    f = lambda s : sum(map(lambda x : m[x],s));
    s = [ ];
    for k in [1..8]:
        t = min([n - list(i).count(0) for i in Subsets (range(n),k).map(f)]);
        s+ = [t];
        print s;
```

Output:

```
[8]
[8, 8]
[8, 8, 10]
[8, 8, 10, 8]
[8, 8, 10, 8, 8]
[8, 8, 10, 8, 8, 8]
[8, 8, 10, 8, 8, 8, 8]
[8, 8, 10, 8, 8, 8, 8, 10]
```

Result: The minimum element of the last array is the minimum distance d_{24} of the code $\mathcal{C}_{24}$, that is, $d_{24} = 8$.

Inspired by the graph code $\mathcal{C}_{12}$ which corresponds to the 5-valent graph, we hope to find out some other optimal additive codes for $n = 24$.

Step 1. Recall that L_{24} is the lower bound of the additive code $(24, 2^{24})$ over $GF(4)$. From the Code Tables, we find that the exact value of L_{24} is 8, that is, $L_{24} = 8$. We now construct a circulant graph Γ_{24} by a set E_1 of edges as follows.

Step 1.1. Arrange the vertices from $V(\Gamma_n) = \{u_1, u_2, \cdots, u_{24}\}$ in a circular order.

Step 1.2. Determine the set E_1 of edges satisfying $|E_1| = L_{24} - 1 = 7$ or $|E_1| = L_{24} + 1 = 9$, where $E_1 = \{u_i u_1 \mid u_i \in N_{\Gamma_n}(u_1)\}$. If $|E_1| = 9$, then $E_1 = \{u_1u_2, u_1u_3, u_1u_5, u_1u_7, u_1u_{13}, u_1u_{19}, u_1u_{21}, u_1u_{23}, u_1u_{24}\}$. If $|E_1| = 7$, then $E_1 = \{u_1u_2, u_1u_3, u_1u_5, u_1u_{13}, u_1u_{21}, u_1u_{23}, u_1u_{24}\}$. In this case, the circulant graph Γ_{24} can be found out; see Figure 2 (b).

Step 2. By the edge set E_1, we write the vector α_{24} corresponding to E_1. If $|E_1| = 9$, then $\alpha_{24} = (0,1,1,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,1,0,1,1)$. If $|E_1| = 7$, then $\alpha_{24} = (0,1,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,1)$.

Step 3. Change the first component of the vector α_{24} into ω. Denote by α'_{24} the new vector. We generate a circulant matrix A'_{24} (A''_{24}) from α'_{24} (α''_{24}):

$\alpha'_{24} = (\omega,1,1,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,1,0,1,1)$,
$\alpha''_{24} = (\omega,1,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,1)$.

Step 4. By *Algorithm 2*, we obtain the minimum distance $d'_{24} = d''_{24} = 8$. Therefore, both $\mathcal{C}'_{24}$ and $\mathcal{C}''_{24}$ are known best additive codes over $GF(4)$. The weight enumerators of the codes $\mathcal{C}'_{24}$ and $\mathcal{C}''_{24}$ are

$$\begin{aligned} W_{\mathcal{C}'_{24}}(z) &= 1 + 528z^8 + 13992z^{10} + 171276z^{12} + 1118040z^{14} + 3773517z^{16} \\ &\quad +6218520z^{18} + 4413948z^{20} + 1034088z^{22} + 33306z^{24}, \\ W_{\mathcal{C}''_{24}}(z) &= 1 + 648z^8 + 13032z^{10} + 174636z^{12} + 1111320z^{14} + 3781917z^{16} \\ &\quad +6211800z^{18} + 4417308z^{20} + 1033128z^{22} + 33426z^{24}. \end{aligned}$$

Applying the above method, we can obtain (near) optimal additive codes over $GF(4)$ from the first two generator vectors.

n	d	L_n	First row of generator matrix	about the code
8	4	4	$(\omega,1,1,0,1,0,1,1)$	optimal
8	4	4	$(\omega,1,0,0,1,0,0,1)$	optimal
10	4	4	$(\omega,1,1,0,0,1,0,0,1,1)$	optimal
10	4	4	$(\omega,1,0,0,0,1,0,0,0,1)$	optimal
16	6	6	$(\omega,1,1,0,1,0,0,0,1,0,0,0,1,0,1,1)$	optimal
22	8	8	$(\omega,1,1,0,1,0,1,0,0,0,0,1,0,0,0,0,1,0,1,0,1,1)$	optimal
24	8	8	$(\omega,1,1,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,1,0,1,1)$	known best
24	8	8	$(\omega,1,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,1)$	known best

From the above analysis, we see that the circulant graphs under our consideration are all relatively sparse. So one may think that only sparse circulant graphs

produce optimal graph codes. However, the following fact gives it a negative answer. Danielsen [6] obtained an optimal additive code $(30, 2^{30}, 12)$ from the vector $\alpha_{30} = (\omega, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0)$, which corresponds to a circulant graph of order 30 such that its adjacency matrix A_{30} is generated by the first row
$\alpha_{30} = (0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0)$.
It is clear that the circulant graph is 17-regular. Since the order of this graph is 30, it follows that the degree of each vertex is relatively large, i.e., it is a relatively dense graph. Let
$\alpha_n = (\omega, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, \overbrace{1, 1, \ldots, 1, 1}^{n-21}, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0)$. For $n = 32$ and $n = 34$, we have the following vectors:
$\alpha_{32} = (\omega, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0)$,
$\alpha_{34} = (\omega, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0)$.
By *Algorithm 2*, we know that $\mathcal{C}_{32}$ and $\mathcal{C}_{34}$ are $(32, 2^{32}, 10)$ and $(34, 2^{34}, 10)$ additive codes, respectively, and both are known best additive codes over $GF(4)$.

The weight enumerators of the codes $\mathcal{C}_{32}$ and $\mathcal{C}_{34}$ are

$$\begin{aligned} W_{\mathcal{C}_{32}}(z) = {} & 1 + 1325z^{10} + 41973z^{12} + 745155z^{14} + 8030541z^{16} + 53150370z^{18} \\ & + 213875634z^{20} + 510617670z^{22} + 691665390z^{24} \\ & + 491629473z^{26} + 159600905z^{28} + 17838471z^{30} + 286740z^{32}, \\ W_{\mathcal{C}_{34}}(z) = {} & 1 + 492z^{10} + 14373z^{12} + 291849z^{14} + 3494061z^{16} + 26279603z^{18} \\ & + 123536402z^{20} + 357928154z^{22} + 620714798z^{24} + 614055698z^{26} \\ & + 319190777z^{28} + 75747789z^{30} + 6157556z^{32} + 72095z^{34}. \end{aligned}$$

5 Concluding remarks

In recent years, graphs have been used to construct codes. But, usually they cannot produce good codes. Proper graphs have to be chosen in order to construct good codes. Because the structure of a circulant graph is very symmetric, the row vectors of its adjacency matrix may span a subspace with the property that the minimum Hamming distance among the vectors of the subspace is comparatively large. Our paper uses this possibility to develop two algorithms for searching for good binary linear codes and additive codes. Starting from a circulant graph, the algorithms modify the generator vector of the circulant graph successively, and get a good code at some step, or fail when all "1"'s of the generator vector have been checked. Therefore, sometimes our algorithms can find good codes, but sometimes they would fail to produce any good code. In general, the running time of our algorithms is exponential in the order n of the circulant graph, since we have to generate all the vectors of the subspace spanned by the row vectors of an $n \times n$ or $n \times 2n$ generator matrix, and this complexity cannot be lower down generally. However, as is seen in the above sections, the row vectors of a circulant are generated by a single vector–its first row vector. This gives us some reasonable hope to use the property to lower down the complexity of generating the subspace spanned by the row vectors of a circulant, which will be left for us to further study.

References

1. K. Betsumiya, M. Harada,*Binary optimal odd formally self-dual codes*, Des. Codes Cryptogr. 23(2001), 11-21.
2. K. Betsumiya, M. Harada, *Classification of formally self-dual even codes of lengths up to 16*, Des. Codes Cryptogr. 23(2001), 325-332.
3. J.C. Bermond, F. Comellas, D.F. Hsu, *Distributed loop computer networks: A survey*, J. Parallel Distributed Comput. 24(1995), 2-10.
4. F.T. Boesch, J.F. Wang, *Reliable circulant networks with minimum transmission delay*, IEEE Trans. Circuits Syst. 32(1985), 1286-1291.
5. J.A. Bondy, U.S.R. Murty, *Graph Theory*, GTM 244, Springer, 2008.
6. L.E. Danielsen, *On self-dual quantum codes*, In: Graphs and Boolean Functions, 2005.
7. L.E. Danielsen, *Graph-based classification of self-dual additive codes over finite field*, Adv. Math. Commun. 3(4)(2009), 329-348.
8. L.E. Danielsen, *On the classification of Hermitian self-dual additive codes over GF(9)*, IEEE Trans. Inform. Theory 58(8)(2012), 5500-5511.
9. L.E. Danielsen, M.G. Parker, *Directed graph representation of half-rate additive codes over GF(4)*, Des. Codes Cryptogr. 59(2011), 119-130.
10. L.E. Danielsen, M.G. Parker, *On the classification of all self-dual additive codes over* $GF(4)$ *of length up to* 12, J. Combin. Theory, Series A 113(2006), 1351-1367.
11. D.G. Glynn, T.A. Gulliver, J.G. Marks, M.K. Gupta, *The Geometry of Additive Quantum Codes*, Preface, Springer, 2006.
12. M. Grassl, *Bounds on the minimum distance of linear codes*, http://www.codetables.de. Accessed 31 Oct 2013
13. T.A. Gulliver, P.R.J. Östergård, *Binary optimal linear rate* 1/2 *codesraphs*, Discrete Math. 283(2004), 255-261.
14. Sunghyu Han, Heisook Lee, Yoonjin Lee, *Binary formally self-dual odd codes*, Des. Codes Cryptogr. 61(2011), 141-150.
15. W.C. Huffman, V. Pless, *Fundamentals of Error-Correcting Codes*, Cambridge University, 2003.
16. B. Mans, F. Pappalardi, I. Shparlinski, *On the spectral Adam property for circulant graphs*, Discrete Math. 254(1-3)(2002), 309-329.
17. P.T. Meijer, *Connectivities and Diameters of Circulant Graps*, B. Sc. (Honors), Simon Fraser University, 1987.
18. E.A. Monakhova, *A survey on undirected circulant graphs*, Discrete Math., Algor. Appl. 4(1)(2012), 1250002[30pages].
19. M.E. Muzychuk, G. Tinhofer, *Recognizing circulant graphs of prime order in polynomial time*, Electron. J. Combin. 5(1)(1998), 501-528.
20. W.A. Stein et al., *Sage Mathematics Software (Version 6.1.1)*, The Sage Development Team, 2014, http://www.sagemath.org.
21. V. Tonchev, *Error-correcting codes from graphs*, Disrete Math. (257)(2002), 549-557.
22. Z. Varbanov, *Additive circulent graph codes over GF(4)*, Math. Maced. 6(2008), 73-79.