第 49 卷 第 1 期
2022 年 1 月

湖南大学学报（自然科学版）
Journal of Hunan University（Natural Sciences）

Vol. 49 No. 1
January 2022

Open Access Article https://doi.org/10.55463/issn.1674-2974.49.1.26

On the Computation of the Automorphisms Group of Some Optimal Codes Using Genetic Algorithm

El Mehdi Bellfkih[1]*, **Said Nouh**[2], **Imrane Chems Eddine Idrissi**[2], **Khalid Louartiti**[1], **Jamal Mouline**[1]

[1] LAMS Lab, Faculty of Sciences Ben M'sick, Hassan II University, Casablanca, Morocco

[2] LTIM Lab, Faculty of Sciences Ben M'sik, Hassan II University, Casablanca, Morocco

Abstract: Many research papers in coding theory have recently focused on designing high-rate codes or improving codes that exist through a better understanding and then improving the coding and decoding algorithm. As a result, this paper aims to investigate the computation of the Automorphisms groups of some optimal codes (e.g., some linear circulant codes where their distance meets the lower bound and nonlinear Nordstrom-Robinson (24, 28, 6) code). These Automorphisms groups provide information about the structure of the code, which aids in both the design and enhancement and improvement of decoding algorithms. A new genetic algorithm-based method is proposed, with a detailed description of its components, the fitness function, selection, crossover, and mutation, and is used to find an important collection of Automorphisms; the results obtained have shown that the proposed method is effective in finding stabilizers set for some types of codes.

Keywords: automorphism group, optimal codes, genetic algorithm, crossover, mutation.

用遺傳算法計算一些最優碼的自同構群

摘要：編碼理論方面的許多研究論文最近都集中在設計高速碼或通過更好地理解改進現有的碼，然後改進編解碼算法。因此，本文旨在研究一些最優碼的自同構群的計算（例如，一些距離滿足下限的線性循環碼和非線性諾德斯特龍-羅賓遜(24，28，6) 碼)。這些自同構組提供有關代碼結構的信息，這有助於解碼算法的設計、增強和改進。提出了一種新的基於遺傳算法的方法，詳細描述了其組成部分、適應度函數、選擇、交叉和變異，並用於尋找重要的自同構集合；獲得的結果表明，所提出的方法可以有效地找到為某些類型的代碼設置的穩定器。

关键词：自同構群、最優碼、遺傳算法、交叉、變異。

1. Introduction and Preliminaries

There are several ways in coding theory that address its application, one of which is establishing the automorphism groups of codes, which allows us to determine the structure of the codes, classify them, and aid the decoding algorithm. This is still difficult because determining the whole automorphism group of codes is difficult, except for finite simple groups, which have been realized using sporadic groups [1] (e.g., the automorphism group of Golay Codes is Mathieu groups).

Recalling that the hamming distance between two codewords (vectors) c, c' in F_2^n is defined as the number of coordinates in which c and c' differ. A binary linear [n,k,d]-code Cover F_2 is a k-dimensional subspace of the vector space F_2^n,

$$\mathrm{d} = d(C) = \min_{c \neq c' \in C} d(c, c') = \min_{c \in C\{0\}} wt(c) \qquad (1)$$

and its generator matrix G is a k × n matrix whose rows are the basis of C.

Received: October 26, 2021 / Reviewed: November 24, 2021 / Accepted: December 23, 2021 / Published: January 28, 2022

About the authors: El Mehdi Bellfkih, LAMS Lab, Faculty of Sciences Ben M' sick, Hassan II University, Casablanca, Morocco; Said Nouh, Imrane Chems Eddine Idrissi, LTIM Lab, Faculty of Sciences Ben M' sik, Hassan II University, Casablanca, Morocco; Khalid Louartiti, Jamal Mouline, LAMS Lab, Faculty of Sciences Ben M' sick, Hassan II University, Casablanca, Morocco

Corresponding author El Mehdi Bellfkih, elmehdi.bellfkih@gmail.com

Need to be mentioned, some codes consider another distance which is Lee distance, and it is calculated as follows

$$\sum_{i=1}^{n} \min\left(\left|c_i - c_i^{'}\right|, 2 - \left|c_i - c_i^{'}\right|\right) \tag{2}$$

There is Singleton Bound which shows the links between the minimal distance d and the binary code of length n; this code satisfies

$$|C| \leq 2^{n-d+1} \tag{3}$$

The Gilbert-Varshamov bound in case of binary C(n,d) code can be presented as follows:

$$|C|\left(\sum_{0}^{d-1}\binom{n}{i}\right) \geq 2^{n} \tag{4}$$

This limit shows us that there is a code with a least a certain number of codewords, at least the following number of codewords:

$$2^{n}\left(\sum_{0}^{d-1}\binom{n}{i}\right)^{-1} \tag{5}$$

Let C be a binary linear [n,k,d] code and G its generator matrix, considering the action of the symmetric group Sn on the G columns. for all σ in Sn, denote by G.σ the matrix obtained from the permutation of the G columns. Let $\sigma \in S_n$, $c = (c_1, c_2, \ldots, c_n) \in C$

$$\sigma(c) = \sigma(c_1, c_2, \ldots, c_n) = \left(c_{\sigma(1)}, c_{\sigma(2)}, \ldots, c_{\sigma(n)}\right) \tag{6}$$

$$\sigma(C) = \sigma(c), c \in C \tag{7}$$

An automorphism of C is any permutation of the G columns that maps the rows of G into rows of the same matrix. The set of all automorphism permutations forms a subgroup of S_n, denoted by Aut(C),

$$Aut(C) = \sigma \in S_n, \sigma(C) = C \tag{8}$$

Let A be a group, and A is an automorphism group of C if A ⊆ Aut(C) and A is the automorphism group of C if A = Aut(C) [2].

1.1. Optimal Codes

We will focus on code where the difference between the smallest known upper bound (3) and the actual size of the code is 0; we choose some linear circulant codes and nonlinear Nordstrom- Robinson codes of length 24, 28 codewords, and hamming minimal distance 6.

1.2. Linear Circulant Codes

Linear circulant codes which are a class of linear code with a specific construction relay on the left shifting of header parts in parity section (8), it is characterized by the different rates, Because a higher rate code makes better use of redundancy than a lower rate code, codes with greater information rates are preferred in the communication domain. On the other hand, the larger the code rate, the smaller is the minimum distance, and vice-versa which means more correctable errors, need to be mentioned the lowest rate of our linear code, which is $\frac{1}{2}$.

The generator matrix of an optimal code of rate R = $\frac{\alpha}{\alpha+1}$ and redundancy r = n − k and header h where $h = h_1 h_2 \ldots h_{r-1} h_r h_{r+1} \ldots h_{\alpha r-1} h_{\alpha r}$ [3].

$$G = [I_k | P_{kxr}] = \begin{bmatrix} 1 & 0 & \cdots & 0 & h_1 & h_2 & \cdots & h_{r-1} & h_r \\ 0 & 1 & \cdots & \vdots & h_2 & h_3 & \cdots & h_r & h_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & h_r & h_1 & \vdots & h_{r-1} & h_{r-2} \\ \vdots & \vdots & \vdots & \vdots & h_{r+1} & h_{r+2} & \cdots & h_{2r-1} & h_{2r} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & h_{2r} & h_{r+1} & \cdots & h_{2r-2} & h_{2r-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & 1 & h_{\alpha r} & h_{(\alpha-1)r+1} & \cdots & h_{6r-2} & h_{6r-1} \end{bmatrix} \tag{9}$$

$$\mathrm{H} = \left[P_{rxk}^{T} | I_r\right] \tag{10}$$

Example 1: [28, 21, 4] – Code of rate R= $\frac{1}{2}$, redundancy r = 7 and header h = 111010101101110000111, we have the following generator matrix G_1 and check matrix H_1:

$$G_1 = \begin{bmatrix} 0100000000000000000001101011 \\ 0010000000000000000001010111 \\ 0001000000000000000000101111 \\ 0000100000000000000001011110 \\ 0000010000000000000000111101 \\ 0000001000000000000001111010 \\ 0000000100000000000000110111 \\ 0000000010000000000001101110 \\ 0000000001000000000001011101 \\ 0000000000100000000000111011 \\ 0000000000010000000001110110 \\ 0000000000001000000001101101 \\ 0000000000000100000001011011 \\ 0000000000000010000000000111 \\ 0000000000000001000000001110 \\ 0000000000000000100000011100 \\ 0000000000000000010000111000 \\ 0000000000000000001001110000 \\ 0000000000000000000101100001 \\ 0000000000000000000011000011 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} 1110101011011100001111000000 \\ 1101011110111000011100100000 \\ 1010111101110100111000010000 \\ 0101111011101101110000001000 \\ 1011110111011011100000000100 \\ 0111101110110111000010000010 \\ 1111010101101110000110000001 \end{bmatrix} \tag{11}$$

Example 2: [36, 24, 6] – Code of rate R = $\frac{3}{4}$, redundancy r = 2 and header h = 011100110000010111011010, we have the following generator matrix G_2 and check matrix H_2:

$$G_2 = \begin{bmatrix} 100000000000000000000000011100110000 \\ 010000000000000000000000111001100000 \\ 001000000000000000000000110011000001 \\ 000100000000000000000000100110000011 \\ 000010000000000000000000001100000111 \\ 000001000000000000000000011000001110 \\ 000000100000000000000000110000011100 \\ 000000010000000000000000100000111001 \\ 000000001000000000000000000001110011 \\ 000000000100000000000000000011100110 \\ 000000000010000000000000000111001100 \\ 000000000001000000000000001110011000 \\ 000000000000100000000000010111011010 \\ 000000000000010000000000101110110100 \\ 000000000000001000000000011101101001 \\ 000000000000000100000000111011010010 \\ 000000000000000010000000110110100101 \\ 000000000000000001000000101101001011 \\ 000000000000000000100000011010010111 \\ 000000000000000000010000110100101110 \\ 000000000000000000001000101001011101 \\ 000000000000000000000100010010111011 \\ 000000000000000000000010100101110110 \\ 000000000000000000000001001011101101 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} 011100110000010111011010100000000000 \\ 111001100000101110110100010000000000 \\ 110011000001011101101001001000000000 \\ 1001100000011111011010010000100000000 \\ 0011000000111110110100101000010000000 \\ 0110000001110101101001011000001000000 \\ 1100000011100011010010111000000100000 \\ 1000000111001110100101110000000010000 \\ 0000001110011101001011101000000001000 \\ 0000111001100100101110110000000000100 \\ 0001110011001001011101100000000000010 \\ 0011100110000001011101101000000000001 \end{bmatrix} \quad (12)$$

1.3. Nordstrom-Robinson Code (NR)

Although there are infinite families of excellent nonlinear codes, no binary code compares to the nonlinear NR code (16, 8, 6). Snover has shown that NR code is the unique code of length 16, minimal distance 6, and 256 words. Different approaches to the construction of NR code has been established one of them is based on Golay code [24, 12, 8] (G) over F_2, where the NR code consists of all vectors $y \epsilon Z_2^{16}$ such that xy ∈ G where
x ∈ {00000000, 11000000, 10100000, … , 10000001}.

Another approach is using Z/4Z linear codes, which is proven by Forney et al., the binary image of the octal code (O_8) is the NR code, where the generator matrix is as follows [5]:

$$\begin{bmatrix} 10002111 \\ 01001213 \\ 00101321 \\ 00011132 \end{bmatrix} \quad (13)$$

where the binary image φ is defined as follows:

$$ц: Z_4 \longrightarrow Z_2^2$$

$$ц(0) = 00, ц(1) = 01, ц(2) = 11, and\ ц(3) = 10 \quad (14)$$

The generator matrix (15) is an extended Hamming code [8, 4, 4] over F_2 when octal code generator matrix entries are modulo 2. The Lee weight distribution of the linear Z_4 code is identical to the Hamming weight distribution of the F_2 image.

Now, we can obtain the generator matrix GN R through the binary image φ of generator matrix vectors of O_8 code (14)

$$G_{NR} = \begin{bmatrix} 0100000011010101 \\ 0001000001110110 \\ 0000010001101101 \\ 0000000101011011 \end{bmatrix}$$

$$G_{NR} \xrightarrow{Standart\ form} G_{NR} = \begin{bmatrix} 1000000011010101 \\ 0100000001110110 \\ 0010000001101101 \\ 0001000001011011 \end{bmatrix} \quad (15)$$

1.4. Genetic Algorithm

The GA is a member of the evolutionary algorithm family. C. Darwin observed that species evolution is dependent on two components: selection and reproduction, and the population of a GA evolves through genetic operators influenced by evolutionary biology. The reproduction of the fittest and most vigorous individuals is provided by the selection, while reproduction is a phase in which evolution is carried out. Based on this evolution in nature, JH Holland invented the genetic algorithm in its first version, which was based on the programming of individuals in a binary system ؛which improved significantly later. The task in such permutation problems (e.g., travel salesman problem (TSP), job-shop scheduling problem (JSP), bandwidth-reduction problem (BRP), and linear ordering problem (LOP)), which is a class of combinatorial optimization problems, is to arrange some objects in chromosomes in a certain order, with no duplicates. It should be done to optimize the objective function, where the representation of the chromosomes depends on [4]. GA solves the permutation problem by rapidly searching the search space. It employs the selection, crossover, and mutation operators, resulting in better chromosomes at the lowest possible cost [5]. The algorithm's effectiveness has been represented by several research papers, such as the research of Rajappa and Elsayed et al. Also, it has been demonstrated that using evolutionary algorithms to solve combinatorial optimization problems is effective [6-8]. There are powerful, nature-inspired algorithms such as the Gaining-sharing knowledge-based algorithm (GSK) [9-11], which have demonstrated superior results in solving optimization problems. The GA has several advantages such as:

- Only evaluate the objective function, regardless of its nature (e.g., continuity, derivation, and others), giving it more flexibility and a broader range of applications;
- Generation takes a parallel form by working on multiple points at once (population of size N), rather than a single iteration in classical algorithms;
- The use of probabilistic transition rules (selection, crossover, and mutation probabilities) rather than deterministic trajectories

Many researchers have indicated that comprehension of the GA parameters' interaction process, notably crossover probability, mutation probability, and population size, is the most important factor in evaluating the process. These variables are linked somehow and have an impact on GA efficiency. The optimal condition to use GA [18] is when there is diversity in the original population with a high crossover probability and a low mutation probability.

It is worth noting that the usual crossover operator cannot be used to solve permutation problems because chromosome ordering is crucial, and no genes should be duplicated or missed [12]. In addition, compared to other scenarios, it is more computationally expensive since a legalization step is necessary after each substring exchange for offspring with duplicate numbers. In this case, the time required to complete a crossover operation increases fast as chromosome size increases, reducing the efficiency of permutation-based GAs [13]. In their research publication [14], Chun Liu and Andreas Kroll devised a genetic algorithm that did not employ the crossover operator. It is worth repeating that GA has been utilized to compute the minimum distance of linear block codes [15] and determine automorphisms set for some block codes like BCH and RQ codes of modest length.

2. Genetic Algorithm–Based Method

Our GA-based technique employs an encoding scheme that treats a permutation (chromosome) as a series of numbers ranging from 0 to the code length minus one. The following is how our method parameters work:

2.1. The Search Space and Fitness Function

The search space in which the GA-based method will search for permutations has n! permutations (n is the code length). Each permutation matrix P_σ will be paired with its corresponding permutation for all permutations $\sigma \in S_n$.

$$P_\sigma = \sigma(I_n) \qquad (16)$$

$S_c \subset C$ is a codewords set, such that, $\forall c_i \in S_c$,

$$c_i H^T = 0 \qquad (17)$$

S_c is a matrix where its rows are formed by codewords.

$$S_c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_l \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ c_{l1} & c_{l2} & \cdots & c_{ln} \end{bmatrix} \qquad (18)$$

Applying the action of S_n on S_c, $\forall \sigma \in S_n$ s.t,

$$y(S_c) = S_c P_y = \begin{bmatrix} c_{y(11)} & c_{y(12)} & \cdots & c_{y(1n)} \\ c_{y(21)} & c_{y(22)} & \cdots & c_{y(2n)} \\ \vdots & \vdots & \vdots & \vdots \\ c_{y(l1)} & c_{y(l2)} & \cdots & c_{y(ln)} \end{bmatrix}$$

(19)

$$y(S_c)H^T = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_l \end{bmatrix} \qquad (20)$$

where H is the sparse parity check matrix

The permutation of S_c columns will generate another matrix of codewords if

$$\sigma(S_c) H^T = 0_{l\times(n-k)} \qquad (5)$$

The fitness values of permutations will be used to choose the best permutations (individuals) using the fitness function, which is defined as follows:

$$f_\sigma = |\{C \cap \sigma(S_c)\}| = |\{s_i \in \{s_1, s_2, \ldots, s_l\} | s_i = 0_{1\times(n-k)}\}| \qquad (21)$$

The values of each permutation are computed using the fitness function (21) (a maximization function), with the best fitness value being equivalent to the best permutation that stabilizes the maximum of codewords in S_c [16].

Let ρ be a function defined on $S = \{0,1,\ldots,n-1\}$ such that:

$$\rho = \text{ChooseRandomly}(x, y), \text{ where } x, y \in S \qquad (22)$$

$$\text{Offspring}[i] = \begin{cases} \rho(P_1[i], P_2[i]) \; when \; i = 1 \\ \{(\end{cases} \qquad (23)$$

where $P_i = Gene_{i1}, Gene_{i2}, \ldots Gene_{in-1}$

The crossover operator that was shown above (23) is based on a random choice (22) with a probability of 0.5 for genes that do not exist in the offspring genes, otherwise choose the other, chosen to ensure that all produced individuals are within the search space and elements of S_n without relying on mutation due to the mutation operator's low probability.

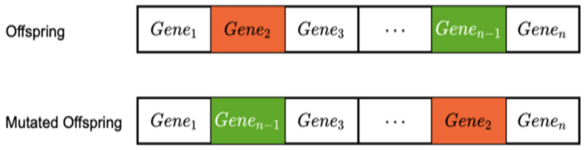

Fig. 1 Mutation operator

2.2. The Inputs and Outputs Method

The GA-based method works as follows:

Inputs:

- Codewords set S_c
- Initial population size N_i
- Number of generations N_{max}
- Crossover probability p_c
- Mutation probability p_m

Outputs:

- Set of Automorphisms permutations S_{Aut}

```
Data: Set of codewords $S_c$
Result: Set of Automorphisms $S_{Aut}$
$N_i \leftarrow$ Initial population of permutations $\sigma_i$ randomly generated;
$N_g \leftarrow 1$;
while $N_g < N_{max}$ do
    Evaluate the permutation fitness $f_{\sigma_i}$ ;
    Choose the $N_e$ elite individuals from the sorted current population into next population $f_{\sigma_i}$ ;
    for $i=N_e$ to $N_i$, do
        Select pair of elite individuals from $N_e$ individuals;
        Apply the crossover and mutation operators;
    end
    $N_g \leftarrow N_g + 1$
end
```

Algorithm 1: GA-based method algorithm.

3. Results and Discussion

The default parameters of the GA-based method are given in Table 1, and the permutation is presented as a list of integers ranging from 0 to the length of the code minus one (each number representing a gene).

Table 1 Parameters of GA-based method

Parameter	Value
Initial population size	100
Selection	Elitism
Crossover probability	1
Mutation probability	0.07
Number of generations	200

Because every error-correcting code has an Automorphisms group, a set of Automorphisms permutations exists. We obtained a large number of stabilizers after running the algorithm. Tables 2 and 3, which include 15 Automorphisms permutations produced by our GA-based method for linear circulant [21, 14, 4] code and linear circulant [24, 16, 4] code, are cited as the example set of stabilizers for each code mentioned below.

Table 2 Automorphisms set of linear circulant [21, 14, 4] code of header 10010011101101

[4,9,16,3,7,12,8,14,2,10,20,5,15,6,19,18,13,1,11,0,17]
[14,7,2,15,10,12,20,13,0,8,19,11,3,17,9,5,4,1,18,16,6]
[2,1,0,3,13,11,6,19,8,9,20,5,12,4,14,15,16,17,18,7,10]
[13,3,0,11,8,4,1,17,20,5,18,2,19,6,7,10,14,15,16,9,12]
[19,20,15,13,18,4,9,10,1,7,5,12,14,2,8,17,11,0,6,3,16]
[7,4,0,12,9,15,10,19,16,13,14,18,3,1,20,11,8,17,5,2,6]
[17,18,13,11,7,10,8,9,14,12,5,16,19,1,6,2,20,15,4,0,3]
[7,3,0,12,2,6,10,1,20,5,18,14,4,19,16,17,8,11,13,9,15]
[13,3,0,10,8,4,15,12,5,20,9,2,19,6,7,11,14,1,16,18,17]
[3,0,11,7,4,1,14,9,18,5,16,19,6,17,12,8,10,13,15,2,20]
[18,7,12,16,17,2,1,8,15,5,13,4,10,9,3,19,14,20,11,6,0]
[13,9,16,18,19,12,1,17,20,0,2,11,15,6,7,3,4,8,5,10,14]
[9,11,1,16,6,0,12,18,3,10,19,7,13,14,17,5,8,4,20,15,2]
[15,13,8,4,9,11,20,5,7,6,14,2,10,3,1,16,18,17,19,0,12]
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

Table 2 Automorphisms set of linear circulant [24, 16, 4] code of header 1001011011011111

[5,4,3,6,1,0,7,2,10,17,8,15,20,21,12,23,18,9,16,11,22,13,14,19]
[0,5,6,3,4,1,2,7,18,9,10,23,22,21,14,15,16,17,8,19,20,13,12,11]
[6,1,0,3,2,5,4,7,15,20,21,12,11,10,17,18,13,14,19,8,9,16,23,22]
[0,1,2,3,4,5,6,7,8,17,10,23,12,21,14,19,16,9,18,15,20,13,22,11]
[6,3,0,1,2,7,4,5,20,19,8,17,16,11,22,21,18,15,14,13,12,23,10,9]
[0,1,2,3,4,5,6,7,8,9,16,23,12,13,20,19,10,17,18,15,14,21,22,11]
[4,5,2,7,0,1,6,3,22,21,20,15,18,17,10,23,14,13,12,11,16,9,8,19]
[2,7,4,1,6,3,0,5,16,11,12,13,14,19,18,17,22,23,10,9,8,15,20,21]
[3,4,1,2,7,0,5,6,23,22,13,20,15,18,17,16,21,12,11,10,9,8,19,14]
[1,0,3,2,5,4,7,6,16,17,18,15,14,21,12,11,8,9,10,23,22,13,20,19]
[1,2,7,0,5,6,3,4,13,14,15,8,17,10,23,12,19,20,21,22,11,16,9,18]
[3,2,5,0,7,6,1,4,18,19,20,21,22,23,10,9,14,15,8,17,16,11,12,13]
[7,2,1,0,3,6,5,4,18,15,20,13,12,23,16,9,14,19,8,17,10,11,22,21]
[6,7,4,5,2,3,0,1,16,23,12,13,14,19,18,9,22,11,10,17,8,15,20,21]
[1,0,3,6,5,4,7,2,16,17,8,19,20,21,22,11,18,9,10,23,12,13,14,15]

Table 4 contains, as an example, a set of 15 Automorphisms permutations produced by our GA-based method for NR (24,28,6) code. To be mentioned, any combination of two Automorphisms permutations is an Automorphism permutation for the abovementioned codes. This implies that if SAut includes all of the automorphism group's generators, then all of the others can be obtained.

Remarque, the octal code O_8 cannot supersedes the NR code due to their Automorphisms groups orders (|Aut(NR)|=16 × 7! and |Auth|=| GL (4,2)|=1344).

Table 3 Automorphisms set of NR (16,256,6) code

[8,3,9,2,5,4,6,7,0,1,13,15,11,10,12,14]
[8,9,1,2,4,7,6,5,0,3,15,14,10,12,11,13]
[8,1,2,9,5,7,4,6,0,3,10,12,11,13,15,14]
[8,2,9,3,7,4,6,5,0,1,11,15,13,14,12,10]
[0,2,3,1,6,4,7,5,8,9,12,13,14,15,10,11]
[0,9,1,2,4,6,7,5,8,3,15,14,10,12,11,13]
[8,2,3,9,7,5,6,4,0,1,12,14,13,15,11,10]
[0,3,2,1,4,6,5,7,8,9,12,11,10,15,14,13]
[8,1,2,3,5,4,6,7,0,9,10,11,12,13,14,15]
[0,9,3,1,6,4,5,7,8,2,13,12,14,10,15,11]
[0,3,2,1,7,4,6,5,8,9,12,11,10,15,14,13]
[0,1,9,2,6,4,7,5,8,3,15,13,11,12,10,14]
[0,9,2,1,4,5,6,7,8,3,11,12,10,14,15,13]
[8,1,9,3,6,5,7,4,0,2,15,11,13,12,14,10]
[0,2,1,3,4,6,7,5,8,9,10,15,14,13,12,11]

4. Conclusion

In this research, a genetic algorithm-based method for determining an important Automorphisms set of certain provided linear circulant codes, as well as the Nordstrom-Robinson (24, 28, 6) code, is suggested, which may be utilized to improve their decoding algorithms (the hard decision algorithm and the soft decision algorithm). The abovementioned codes yielded significant results, and we did not need a requirement on either the level of the search space or the level of the fitness function. This genetic algorithm-based method can be used to find a set of automorphisms groups of codes, and it can be completely relied on in decoding algorithms based on the set of automorphism permutations, especially in small and medium-length it can be used in a variety of codes. Our future work will improve the algorithm's efficiency and accelerate its function in finding automorphism groups for various types and lengths of codes by searching for a more efficient fitness function, adjusting the selection operator, the stochastic crossover operator, and the stochastic crossover operator.

References

[1] HAILY A., and HARZALLA D. On the automorphism group of some classes of systematic codes. *Asian Journal of Mathematics and Computer Research*, 2016, 13(2).

[2] 4 Finite fields. In: *The Theory of Error-Correcting Codes*. F.J. MACWILLIAMS and N.J.A. SLOANE (eds.). North-Holland Mathematical Library, 1977, 16: 93-124. https://doi.org/10.1016/S0924-6509(08)70529-4.

[3] JOUNDAN I.A., NOUH S., and NAMIR A. Design of good linear codes for a decoder based on majority voting procedure. In: *2016 International Conference on Advanced Communication Systems and Information Security* (ACOSIS). 2016: 1-6. DOI: 10.1109/ACOSIS.2016.7843918.

[4] BASMASSI M.A., BENAMEUR L., and CHENTOUFI J.A. A novel greedy genetic algorithm to solve combinatorial optimization problem. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLIV-4/W3-2020,* 2020: 117-120. DOI:

10.5194/isprs-archives-XLIV-4-W3-2020-117-2020.

[5] MUDALIAR D.N., and MODI N.K. Applying m-Mutation Operator in Genetic Algorithm to Solve Permutation Problems. In: *2019 IEEE International Conference on System, Computation, Automation and Networking* (ICSCAN), Pondicherry, India, 2019: 19093805. DOI: 10.1109/ICSCAN.2019. 8878867.

[6] PHIWHORM K., and SAIKAEW K.R. A Hybrid Genetic Algorithm with Multi-Parent Crossover in Fuzzy Rule-Based. *International Journal of Machine Learning and Computing*, 2017, 7(5): 114-117. DOI: 10.18178/ijmlc.2017.7.5.631.

[7] HULIANYTSKYI L., and RIASNA I. Formalization and Classification of Combinatorial Optimization Problems. *Optimization Methods and Applications*, 2017: 239-250. ISBN: 978-3-319-68639-4. DOI: 10.1007/978-3-319- 68640-0_11.

[8] YAKOVLEV S., KARTASHOV O., and YAROVAYA O. On class of genetic algorithms in optimization problems on combinatorial configurations. In: *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies* (CSIT), 2018, 1: 374-377. DOI: 10.1109/STC-CSIT.2018.8526746.

[9] WAGDY A., HADI A., and KHATER A. Gaining-sharing knowledge-based algorithm for solving optimization problems: a novel nature-inspired algorithm. *International Journal of Machine Learning and Cybernetics*, 2020, 11. DOI: 10.1007/s13042-019-01053-x.

[10] AGRAWAL P., TALARI G., and WAGDY A. A novel binary gaining-sharing knowledge-based optimization algorithm for feature selection. *Neural Computing and Applications*, 2021, 33: 1-20. DOI: 10.1007/s00521-020-05375-8.

[11] AGRAWAL P., TALARI G., and WAGDY A. Chaotic gaining sharing knowledge-based optimization algorithm: an improved metaheuristic algorithm for feature selection. *Soft Computing*, 2021, 25(2). DOI: 10.1007/s00500-021-05874-3

[12] HASSANAT A., ALMOHAMMADI K., ALKAFAWEEN E., ABUNAWAS E., HAMMOURI A., and PRASATH V.B.S. Choosing Mutation and Crossover Ratios for Genetic Algorithms – A Review with a New Dynamic Approach. *Information*, 2019, 10(12): 390. https://doi.org/10.3390/info10120390

[13] KOOHESTANI B. A crossover operator for improving the efficiency of permutation-based genetic algorithms. *Expert Systems with Applications*, 2020, 151: 113381.

[14] LIU C., and KROLL A. Performance impact of mutation operators of a subpopulation-based genetic algorithm for multi-robot task allocation problems. *SpringerPlus*, 2016, 5: Article number 1361. DOI: 10.1186/s40064-016-3027-2.

[15] ASKALI M., AZOUAOUI A., NOUH S., and BELKASMI M. On the Computing of the Minimum Distance of Linear Block Codes by Heuristic Methods. *International Journal of Communication*, 2012, 5: 774-784.

[16] Simeon Ball. *A Course in Algebraic Error-Correcting Codes*. Textbook. Birkhäuser, 2020.

参考文:

[1] HAILY A. 和 HARZALLA D. 關於某些類系統碼的自同構群。載於：亞洲數學與計算機研究雜誌，2016 年，13(2)。

[2] 4 有限域。在：糾錯碼理論。F.J. MACWILLIAMS 和 N.J.A. SLOANE（編輯）。北荷蘭數學圖書館，1977，16 ： 93-124 。 https://doi.org/10.1016/S0924-6509(08)70529-4。

[3] JOUNDAN I.A.、NOUH S. 和 NAMIR A. 基於多數表決程序的解碼器的良好線性碼設計。在：2016 年高級通信系統和信息安全國際會議。 2016：1-6。 DOI：10.1109/ACOSIS.2016。 7843918。

[4] BASMASSI M.A.、BENAMEUR L. 和 CHENTOUFI J.A.一種求解組合優化問題的新型貪心遺傳算法。在：國際攝影測量、遙感和空間信息科學檔案 XLIV-4/W3-2020，2020：117-120。 DOI：10.5194/isprs-archives-XLIV-4-W3-2020-117-2020。

[5] MUDALIAR D.N. 和 MODI N.K.在遺傳算法中應用 m-突變算子解決置換問題。在：2019 年電氣和電子工程師學會系統、計算、自動化和網絡國際會議，印度本地治裡，2019：19093805。DOI：10.1109/ICSCAN.2019。8878867。

[6] PHIWHORM K. 和 SAIKAEW K.R.基於模糊規則的多親交叉混合遺傳算法。在：國際機器學習與計算雜誌，2017，7（5）：114-117。 DOI：10.18178/ijmlc.2017.7.5.631。

[7] HULIANYTSKYI L. 和 RIASNA I. 組合優化問題的形式化和分類。在：優化方法和應用，2017：239-250。國際標準書號：978-3-319-68639-4。 DOI：10.1007/978-3-319-68640-0_11。

[8] YAKOVLEV S.、KARTASHOV O. 和 YAROVAYA O. 關於組合配置優化問題中的遺傳算法類。在：2018 IEEE 第十三屆計算機科學與信息技術國際科學技術會議，2018，1：374-377。 DOI：10.1109/STC-CSIT.2018.8526746。

[9] WAGDY A.、HADI A. 和 KHATER A. 用於解決優化問題的基於知識共享的算法：一種新穎的自然啟發算法。國際機器學習與控制論雜誌，2020，11。DOI：10.1007/s13042-019-01053-x。

[10] AGRAWAL P.、TALARI G. 和 WAGDY A. 一種新穎的基於二元增益共享知識的特徵選擇優化算法。神經計算與應用，2021，33：1-20。 DOI：10.1007/s00521-020-05375-8。

[11] AGRAWAL P.、TALARI G. 和 WAGDY A. 混沌獲得共享基於知識的優化算法：一種改進的用於特徵選擇的元啟發式算法。軟計算, 2021, 25(2)。 DOI: 10.1007/s00500-021-05874-3

[12] HASSANAT A.、ALMOHAMMADI K.、ALKAFAWEEN E.、ABUNAWAS E.、HAMMOURI A. 和 PRASATH V.B.S.為遺傳算法選擇突變和交叉比率——一種新的動態方法的回顧。信息, 2019, 10(12): 390. https://doi.org/10.3390/info10120390

[13] KOOHESTANI B. 一種用於提高基於排列的遺傳算法效率的交叉算子。具有應用程序的專家系統，2020，151：113381。

[14] LIU C. 和 KROLL A. 變異算子對基於亞群的遺傳算法對多機器人任務分配問題的影響。施普林格加，2016年，5：文章編號 1361。DOI：10.1186/s40064-016-3027-2。

[15] ASKALI M.、AZOUAOUI A.、NOUH S. 和 BELKASMI M. 關於通過啟發式方法計算線性塊碼的最小距離。國際傳播雜誌，2012，5：774-784。

[16] 西蒙·鮑爾。代數糾錯碼課程。教科書。比爾克豪瑟，2020。