Проведено дослідження впливу теорії інформації на розвиток теорії завадостійкого кодування. Показані основні відмінності між ймовірнісним та детермінованим підходами при аналізі коректувальної здатності різних класів лінійних кодів.

Розроблені автоматні ієрархічні моделі для аналізу перестановочного декодування циклічних кодів і запропоновано генератор циклічних перестановок на основі двох автоматів Мура.

На основі автоматного представлення циклічних кодів проведено дослідження регулярних і нерегулярних станів лінійних послідовнісних схем (ЛПС). Показана можливість суттєвого спрощення декодування циклічних кодів на основі переведення нерегулярних синдромів ЛПС в регулярні за допомогою перестановок.

Розроблено формалізовані методи визначення коректувальної здатності циклічних кодів, що ітеративно декодуються (ІДЦК). Традиційний повний перебір всіх можливих варіантів порівняння кодових слів замінено направленим пошуком розв'язання поставленої задачі, що призводить до значної економії часу обчислень. Наведено алгоритм визначення коректувальної здатності ІДЦК відносно подвійних помилок.

Показано, що всі ітеративні коди підвищують свою коректувальну здатність зі збільшенням числа ітерацій і її можна задавати у відсотках для помилок різної кратності. Синдроми помилок розподіляються по окремим ітераціям, що дозволяє зменшити розрядність перевіряльного слова коду. В кінцевому результаті це призводить до збільшення швидкості ітеративних кодів в порівнянні з традиційними коректувальними кодами.

Проведено порівняльний аналіз ІДЦК і LDPC-кодів для визначення сфери їх оптимального застосування

Ключові слова: циклічні коди, низькогустинні коди, коректувальна здатність, ітеративне декодування, лінійна послідовнісна схема, перестановки

UDC 681.32
DOI: 10.15587/1729-4061.2019.157299

ON THE ERROR-CORRECTING CAPABILITIES OF ITERATIVE ERROR CORRECTION CODES

V. Semerenko
PhD, Associate Professor
Department of Computer Technique
Vinnytsia National
Technical University
Khmelnytske highway, 95,
Vinnytsia, Ukraine, 21021
E-mail: VPSemerenko@ukr.net

1. Introduction

The error correcting coding theory has passed a complex and controversial path in its development. The entire 70-year period has been the search for answers to the main questions: which code is the best and how to construct it?

These questions are interesting not only from the point of mathematics. Error correction codes have found wide application in various technical fields, such as satellite and mobile communications, data storage and archiving systems, etc. Even a small advance in the theory can yield a huge economic gain in practice.

Paper [1] justified the basic principles of construction of error correction codesand proposed the criterion of the best code. It is the maximum approximation to channel capacity for data transmission. But codes, which began to appear from the very beginning, were not bad from the practical side, but they did not meet the Shannon criteria. Several directions emerged in error correction coding. They develop separately and interact with each other rarely. Therefore, combined analysis and comparison of characteristics of various classes of codes is difficult. A comparative study of the error-correcting capabilities of error correction codes is particularly important from both points of view – theoretical and practical ones.

The main trend in development of modern communication systems is a constant increase in transmission rates and practical development of the terabyte range. As in previous years, the main reserve in improvement of the quality and rate of transmission is the use of error correction coding. Modern technologies require new ideas and new ways to convert data. Iterative error correction codes are very promising for solution of such problems.

2. Literature review and problem statement

The basis of the coding theory is the theory of information. We can define encoding (decoding) of information as its transformation, when an amount of information remains unchanged, but a qualitative nature of information carrier changes [2]. To perform any transformation of information, it is necessary to present it mathematically and to specify a method for its measurement.

In [3] it is proposed a combinatorial method for measurement of information based on a choice from a certain set of possibilities.

The term "bit" has been widely applied to denote a unit of information since 1948 [1]. According to Shannon, an amount of information is equal to removal of uncertainty (entropy) before an experiment and after an experiment. The method requires statistical characteristics of individual symbols and messages.

In [4] it is presented an algorithmic method for measurement of an amount of information: we can consider

$l(p)$ minimum length of p "program", which is necessary for conversion of x to y as are lative complexity of y object for a given x object.

Since the error correcting coding theory appeared as a response to needs of communication systems (primarily space communications), the coding theory began to develop based mainly on the probabilistic approach, i. e. Shannon approach to the information theory.

It was the optimal choice from the practical point of view, but many questions arose from the theoretical point of view. Some of them still have no resolution. Researchers noted disadvantages of the probabilistic approach in the 60s and 70s: "information theory should precede probability theory; it should not rely on it" [5]. Further research confirmed that "the amount of information is not necessarily connected with random events" [6]. Gaps in the theory manifested themselves in practice soon.

Founders of the error correcting coding theory considered a code, which corresponded to various theoretical limits and relationships between code parameters and its error-correcting capabilities to the maximum extent, as the "best code". However, codes known by that time had significantly worse characteristics than codes predicted by the theory. Therefore, the objective was to build codes, for which "theoretical interest depended on how realistic was creation of equipment for their practical use" [7].

Improvement of some characteristics of codes leads, as a rule, to deterioration of another ones [8]. We should give preference to the problem of increasing of ability of codes to detect and correct errors from the practical point of view.

We can limit ourselves to the class of linear codes among all error correction codes. And we should distinguish probabilistic codes (for example, *low-density parity-check* (LDPC)-codes) and deterministic codes (for example, cyclic codes). These codes are most common now in various technical areas.

The basic idea of encoding of linear codes is to add additional check binary digits to a source information word either in explicit or implicit form and obtaining of codeword Z. As a result, we get a "coding gain," that is, ability to detect and to correct errors in Z [9]. It is quite difficult to evaluate the "coding gain"analytically.

Usually, analysis of various codes occurs by comparison of curves (Fig. 1). This curves show the dependence of ρ_b average probability of an erroneous bit on E_b/N_o ratio for these codes (E_b is bit energy, N_0 is spectral density of noise power). All curves should be at the right of the vertical ordinate with a value of –1.6 dB, which indicates a Shannon limit. We can determine a distance from a code curve to Shannon limit using such a graph (we can call it BER – *bit error rate* briefly), and the closer it to Shannon limit is, the better is the code. One code will be better than another one in proportion to γ_e value.

There is an integer characteristic of the ability to detect and correct errors – the minimum code distance $d_{\min}$ in addition to the probabilistic estimate. By definition, $d_{\min}$ is equal to the smallest of all Hamming distances between different pairs of code words. $d_{\min}$ parameter $d_{\min}$ makes possible to determine amount of τ_d detected errors and τ_c corrected errors by a given code accurately:

$$d_{\min} \geq \tau_d + \tau_c + 1 = 2\tau_c + 1 = \tau_d + 1.$$

The basis of the universal method for the exact calculation of parameter $d_{\min}$ for an arbitrary block linear code is the analysis of weights spectrum of a code. We know weights distribution of the code, even in analytical form, for some subclasses of cyclic codes. However, there is no resolution of this problem for all codes, since it belongs to NP-hard problems [10].

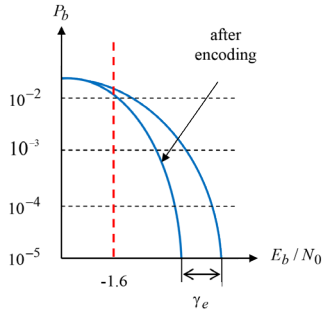

Fig. 1. Graph of the dependence of ρ_b on E_b/N_0 ratio

The complexity of calculation of $d_{\min}$ exact values contributed to emergence of various asymptotic bounds that establish the relationship between $d_{\min}$ and code redundancy. Hamming, Plotkin, and Elias bounds set the upper bound for $d_{\min}$, and Varshamov – Hilbert bound set the lower bound for $d_{\min}$ [7].

It is more acceptable to use BER curve for probabilistic codes, but the minimum code distance and the corresponding asymptotic bounds are more acceptable for deterministic codes.

And here we meet with serious problem – how to compare error-correcting capabilities of such different codes. We can try to accomplish this task using both criteria for all codes.

For example, we can show both LDPC codes and cyclic codes on a single BER curve together. However, this approach will be correct for random errors of arbitrary multiplicity only. Cyclic codes can also correct burst errors of various lengths. In [11] it is shown that the distance to the Shannon limit will be the same, that is, we will not receive a full error correction estimate by cyclic codes for all Fire codes.

Let us consider the opposite approach: estimation of the error-correcting capabilities of probabilistic codes using parameter $d_{\min}$. A huge amount of research on this issue appeared over the past decade. We can summarize the results obtained regarding LDPC codes and make the following conclusions:

1. We should still calculate minimum code distance $d_{\min}$ based on weight distribution [12].

2. Therefore, it is not surprising that the calculation of $d_{\min}$ for LDPC codes is an NP-hard problem [13].

3. We obtain values of parameter $d_{\min}$ for specific LDPC codes as a result of numerical experiments on a supercomputer [14]. We compensate the absence of mathematically sound methods for estimation of the error-correcting capabilities of codes by search algorithms that require huge computational resources.

4. The minimum distance of algebraic LDPC codes is usually higher than random LDPC codes [15]. The minimum distance of regular LDPC codes approaches the distance of the best random linear codes [16]. Therefore, the minimum distance of the classical (i. e. irregular) LDPC codes is low.

On the other hand, no one doubts the high error-correcting capabilities of LDPC codes, otherwise we would not apply these codes in practice.

We can resolve this paradox only by the recognition of the fact that the traditional $d_{\min}$ parameter is not an adequate estimate for error-correcting capabilities for probabilistic iterative codes. Accordingly, the analysis of the distance to the Shannon limit on BER curves would not be the best criterion for evaluation of similar properties of deterministic codes.

Consequently, there is currently no general methodology for mathematically justified comparison of error-correcting capabilities of different classes of codes.

3. The aim and objectives of the study

The aim of the study is to obtain results of a comparative analysis of error-correcting capabilities of linear error correction codes and to obtain deterministic and accurate estimates of error-correcting capabilities of iteratively decoded cyclic codes (IDCC) based on the mathematical apparatus of linear finite-state machines (LFSM).

It is necessary to solve the following tasks to achieve the objective:

– study of the ability to detect and correct errors with iterative error correction codes from the standpoint of the information theory and coding theory;

– mathematical substantiation of permutation decoding of cyclic codes;

– investigation of an influence of cyclic permutations on complexity of decoding and on error-correcting capabilities of cyclic codes;

– proposition of effective tools for estimation of error-correcting capabilities of IDCC for practical implementation.

4. Mathematical substantiation of permutation decoding of cyclic codes

We use the following cyclic permutations of positions of Z codeword [9, 11]:

$$i \to (i+\upsilon) \bmod n, \quad GF(2) \ \ \upsilon = 2,3,4,\ldots \tag{1}$$

We can consider formation of permutations as a result of operation of some permutation generator and represent its operation by Λ Moore finite-state automaton with S finite set of states, Y finite set of outputs, a transition function

$$S(t+1) = P \times S(t), \quad GF(n)$$

and output function

$$Y(t) = S(t), \quad GF(n),$$

where P is a permutation operator.

The operation of automaton Λ occurs in t discrete times. We call automaton Λ a high-level automaton.

The inputs of the automaton Λ receive only zero values, and the outputs of the automaton Λ are its state. The state in time t

$$S(t)=\{s(1), s(2), s(3),..., s(n)\}$$

is the positions of binary digits of the code word Z.

We can assume that calculations take place in Galois field $GF(n)$, because elements of S and Y sets are selected from integer alphabet $\{0, 1,\ldots , n-1\}$ for a cyclic (n, k)-code.

The state $S(1)$ coincides with the initial value of code word Z at the beginning of automaton Λ operation. The task of the generator based on automaton Λ is to calculate new positions of binary digits of code word Z in each time. It is equivalent to obtaining of the next state $S(t+1)$. We call the obtaining of the next state of automaton Λ iteration.

The position of first binary digit of state $S(t+1)$ is always the same, it is $s(1)=1$. We can calculate values of subsequent positions of binary digits according to (1). Since these calculations take place recursively, it is convenient to present them using another automaton Moore (we call it as low level automaton π) with transition and output functions:

$$s(i+1) = s(i) + \upsilon, \quad GF(n),$$

$$y(i) = s(i), \quad GF(n).$$

In this automaton, the value of position $s(i+1)$ depends both on the value of previous position $s(i)$ and on integer constant υ, which does not change during the entire session of automaton Λ operation.

It is convenient to investigate properties of sequences of positions $s(i)$ based on automaton π, that is, to investigate cycles of low-level permutations. Let us consider the transition graph of automaton π for $n=15$ and different values of constant υ.

If $i \to (i+1) \bmod n$, there is an initial cyclic sequence of positions $s(i)$, which coincides with the initial value of codeword Z. If $i \to (i+2) \bmod n$, we get a new cyclic sequence of positions $s(i)$ of $n=15$ length. If $i \to (i+3) \bmod n$, we get three separate cyclic sequences of positions $s(i)$, each of $n=5$ length.

The structure of low-level permutation cycles depends on the ratio between the parameters n and υ. We can note the main conformities:

– if there are no common multiple for parameters n and υ, then cycles of maximum length $L=n$ are formed;

– if there are a common multiple m for parameters for n and υ, then m cycles of length $L = n/m$ are formed.

As a result, a next state $S(t)$ of automaton Λ will appear based on one or several cycles of low-level permutations. Getting of w of different states $S(t)$ will mean the possibility of w iterations for permutation generators based on automaton Λ.

Table 1 shows the parameters of the permutation generator for $n=15$ and possible values of υ constant.

Computer modeling shows that this generator property is possible if $L=n$ only. A permutation of $i \to (i+2) \bmod n$ provides this in the most cases.

Sometimes there are pairs of positions called pendulum positions. They only replace each other or stay on one place permanently. There is no point to continue iterations specifically for pendulum positions after permutations of all other positions.

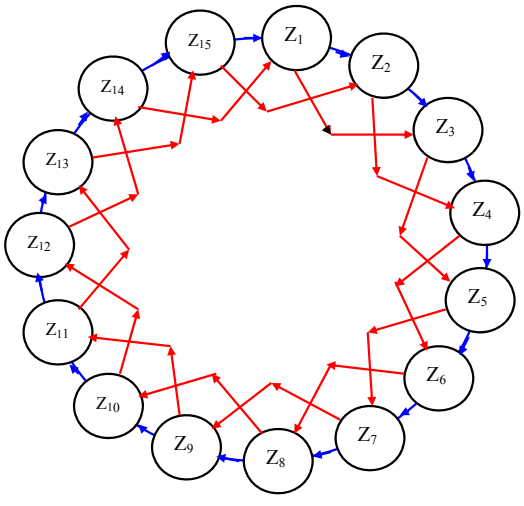

Fig. 2. Cyclic permutations of $i \to (i+\upsilon) \bmod 15$ for $\upsilon=1$ (blue) and $\upsilon=2$ (red)

We can call this effect, when it is not possible to correct some errors at decoding, the correction threshold. Such situation is typical for all iterative codes (it is known as "error floor" for LDPC codes and turbo codes [17]). We know the exact condition for appearance of a correction threshold in IDCC unlike in probabilistic codes.

$n/3+1$ and $2n/3+1$.

Pendulum positions exist only for (n, k)-codes with odd n length multiple of three at permutations with $\upsilon=2$ parameter. Their positions in a codeword are as follows:

$n/3+1$ and $2n/3+1$.

It is necessary to generate check sets additionally for pendulum positions.

Table 1

Possible parameters of the permutation generator

n	$\upsilon=2$	w	L
15	2,7,8,13	4	15
15	4,11,14	2	15
15	3,6,9,12	3	5
15	5,10	5	3

It is very simple to determine a number of iterations for a given cyclic code. If the previously considered automaton Λ passed to state $S(t)$ again from initial state $S(t)$ on $(w+1)$-th time of operation, therefore, such generator has w iterations of permutations. The generator with the highest number of iterations is the best, because it corrects more errors.

5. Influence of permutations on the complexity of decoding of traditional cyclic codes

Let us consider the influence of permutations on complexity of decoding of cyclic codes in more detail.

The task of decoding cyclic codes began to be considered as solved after the appearance in the 60s of various algebraic decoding methods, in particular, the Berlekamp-Massey method [18].

We should clarify that the problem is really solved, but only theoretically. Complexity of calculations is increasing rapidly with an increase in the number of errors and the length of a codeword. Therefore, various probabilistic methods for decoding of linear codes appeared [9]. However, complexity of calculations increased even more (mainly due to the use of non-integer arithmetic), and guarantees of obtaining of exact solutions disappeared at the same time.

Therefore, the problem of finding of exact solutions with acceptable calculational complexity remains the most important unsolved problem in the error correcting coding theory.

A promising direction in this regard is a use of automaton representation of cyclic codes. We use one more automaton model [19]. Its basis is a special type of linear finite automaton – linear finite-state machines (LFSM). A transition (state) function describes LFSM with r memory elements, l inputs and m outputs over $GF(2)$ Galois field

$$S(t+1)=A\times S(t)+B\times U(t), \quad GF(2) \tag{2}$$

and output function

$$Y(t)=C\times S(t)+D\times U(t), \quad GF(2),$$

where

$$A=\left|a_{ij}\right|_{r\times r},\ B=\left|b_{ij}\right|_{r\times l},\ C=\left|c_{ij}\right|_{m\times r} \text{ and } D=\left|d_{ij}\right|_{m\times l}$$

are the characteristic matrix of LFSM; $S(t)=\left|s_i\right|_r$ is a state word; $U(t)=\left|u_i\right|_l$ is the input word; $Y(t)=\left|y_i\right|_m$ is the output word.

The decoding process of cyclic (n, k)-codes based on automaton models consists of two stages:

– determination of presence or absence of an error;

– determination of error parameters, if any.

The first stage consists in calculation of state $S(n)$: LFSM will go into this state after supply to the input of n-digit codeword Z using the recursive formula following from (2):

$$S(j+1)=A\times S(j)+B\times z_j, \quad GF(2),\ z_j\in Z,\ j=1\div n.$$

The state $S(n)$ is usually called a syndrome. A zero value of this state indicates absence of errors in a transmitted codeword within detecting capabilities of the selected cyclic code. If there is an error of multiplicity τ in a code word, which we denote $Z_{err}^{(\tau)}$, we will obtain a non-zero error syndrome – $S_{err}^{(\tau)}()$.

The peculiarity of cyclic codes is that we get n shifts of j-th error pattern $E_j^{(\tau)}$ of multiplicity τ after n shifts of cyclic code word $Z_{err}^{(\tau)}$.

Each $E_j^{(\tau)}$ shift corresponds to j-th error syndrome $S_{err}^{(\tau)}()$. All n shifts of $E_j^{(\tau)}$ correspond to n syndromes of $S_{err}^{(\tau)}()$, which form zero cycles (ZC) for errors of multiplicity τ [20].

It is sufficient to select one syndrome only in each ZC to correct errors in accepted code word $Z_{err}^{(\tau)}$. We should give preference to the regular syndrome $S_{err}^{(\tau)}()$, which is an r-digit cyclic word containing τ units and $(r-\tau)$ zeros, and with a unit in the lower (left) position $(r=n-k)$ bit [21].

The regular syndrome (we call ZC, which contains it, regular also) corresponds to this configuration of erroneous binary digits in $Z_{err}^{(\tau)}$, when all of them fall into the r-position $X_{err}^{(\tau)}$ $(X_{err}^{(\tau)}\subset Z_{err}^{(\tau)})$ check window. It is sufficient to perform

the following operation to obtain positions of $X^{(\tau)}$ check window without errors

$$X^{(\tau)} = X^{(\tau)}_{err} + S^{\tau}_{err}(n), \quad GF(2).$$

Irregular $S^{(\tau)}_{err}()$ syndrome (we call ZC, which contains it, irregular also) corresponds to such a pattern of erroneous digits, when they are located along the entire length of code word $Z^{(\tau)}_{err}$ and, as a result of a shift, never enter r-position check window. It is necessary either to keep a cumbersome correspondence table between $S^{(\tau)}_{err}()$ and $Z^{(\tau)}_{err}$, for such errors or to perform complex algebraic transformations.

For a cyclic (n, k)-code in the presence of τ errors, we determine amount of regular ZC from formula:

$$N^{(\tau)}_{reg} = \binom{n-k-1}{\tau-1}, \tag{3}$$

and irregular ZC:

$$N^{(\tau)}_{not} = \left\lfloor \binom{n}{\tau}\frac{1}{n} \right\rfloor - \binom{n-k-1}{\tau-1}, \tag{4}$$

where $\binom{j}{i}$ is the number of combinations from j to i.

It follows from (3) and (4) that we obtain only regular error syndromes within the limits of the error-correcting capabilities of (n, k)-code, if the inequality

$$\binom{n-k-1}{\tau-1} \geq \left\lfloor \binom{n}{\tau}\frac{1}{n} \right\rfloor \text{ is correct for } \tau = 1...\tau_{\min}.$$

For example, (15,7)-BCH code, which corrects two errors, meets this requirement, it has all regular ZCs for τ=2 and, therefore, it is easy to decode it. But a quadratic-residue (17,9)-code with the same error-correcting capabilities ($d_{\min}$=5) has one irregular ZC for τ=2.

In general, a amount of irregular ZC increases and, accordingly, difficulty of correction of such errors increases, with an increase in the length of n code and τ number of corrected errors.

And here we can use permutation decoding. Transformation of irregular syndromes into regular ones occurs under the influence of cyclic permutations. A larger number of error patterns fall into the check window and get corrected due to the shift of positions of code word $Z^{(\tau)}_{err}$ at each iteration.

Fig. 3–5 shows curves of a gradual increase in regular ZCs with an increase in iterations for several codes.

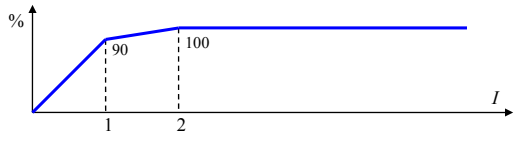

Fig. 3. Share (in %) of regular error syndromes by iterations for quadratic-residual (17,9)-code, error multiplicity τ=2

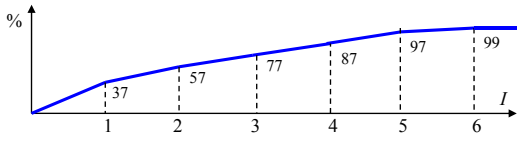

Fig. 4. Share (in %) of regular error syndromes by iterations for (63,51) BCH code, error multiplicity τ=2

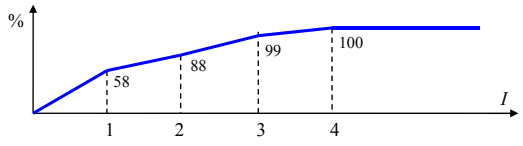

Fig. 5. Share (in %) of regular error syndromes by iterations for (23,12)-Golay code, error multiplicity τ=3

Example 1. In the known (23,12) Golay code with generator polynomial

$$g(x) = 1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11}$$

regular ZCs correspond to all single errors, 91 % of double errors and 58 % of triple errors. Let us analyze only the influence of permutations on complexity of error detection without considering the decoding procedure described in detail in [21]. Let us assume we obtained a code word with three errors (marked in red) via the communication channel:

$$Z^{(3)}_{err} = 1\,0\,0\,0\,0\,0\,0\,0\,\mathbf{1}\,0\,0\,0\,1\,1\,0\,0\,0\,\mathbf{0}\,1\,1\,\mathbf{1}\,1\,0. \tag{5}$$

The pattern of error positions in (5) does not make them fall into the 11-position check window of this code, which corresponds to the irregular error syndrome. After the first permutation of $i \to (i+2) \bmod 23$ form, we get the code word

$$Z^{(3)(1:2)}_{err} = 1\,0\,0\,0\,\mathbf{1}0\,1\,0\,0\,1\,\mathbf{1}0\,0\,0\,0\,0\,0\,0\,1\,0\,\mathbf{0}1\,1.$$

After the second $i \to (i+2) \bmod 23$ permutation, we get the code word

$$Z^{(3)(1:4)}_{err} = 1\,0\,1\,1\,0\,1\,0\,0\,0\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1. \tag{6}$$

The code word (6) corresponds already to the regular error syndrome, it falls into the 11-position check window (highlighted in color) and it is easy to correct it. In conclusion, it remains to make only two reverse permutations and to get the corrected Z codeword.

Thus, we do not need neither complex algebraic transformations nor storage of additional information for decoding of the Golay code.

IDCC phenomenon manifests itself in simplification of the decoding process: if there are iterations, amount of corrected errors (i. e., parameter $d_{\min}$) does not change, and calculational complexity decreases sharply. As a result, there is a significant reduction of decoding time for high multiplicity errors.

6. Determination of error-correcting capabilities of cyclic codes using permutations

It is not possible yet to solve the problem of determination of the error-correcting capabilities of error correction codes in a general form. We can only talk about individual classes of codes today. There is the resolution of the problem only at the level of various types of approximation of experimental data with analytical expressions for probabilistic codes, in particular for LDPC codes [14].

The basis of the task of determination of the error-correcting capabilities of codes is a check algorithm, which determines its belonging to NP-hard problem. It is necessary to simplify a search for code words (error syndromes) as

much as possible using various properties of specific classes of codes.

Cyclic codes are the most suitable for this objective.

Firstly, it is possible to replace n error patterns of n-digit code word with a single error pattern due to the cyclical property. In addition, there are τ equivalent error patterns for each error multiplicity τ. It is enough to use only one (basic) error pattern. As a result, it is possible to reduce an amount of data analyzed in $n\tau$ times.

Secondly, it is possible to reduce search and make it more formalized using cyclic permutations of positions of code words.

Let us consider the method of determination of the error-correcting capabilities of IDCC in more detail. The basis of the method is a generalized algorithm for accurate determination of a degree of correction of errors of various multiplicities. There are different options of the algorithm for each error multiplicity τ. They differ in complexity of calculations. An increase in error multiplicity τ leads to an increase in complexity and execution time of the algorithm. Moreover, it is possible to perform the indicated options of the algorithm for each error multiplicity in parallel.

Based on this method, it is possible not only to determine an exact number of corrected errors of various multiplicities, but also their dependence on code parameters.

7. Algorithm for determination of error-correcting capabilities of IDCC relatively to double errors

Let us assume we have a cyclic (n, k)-code with the code word

$$Z=z_1 z_2 \ldots z_{n-1} z_n, \quad (7)$$

1. To set the iteration number w=1. To generate the current code word Z_c, which coincides with the source word (7).

2. To form three check windows of n positions of code word Z_c:

– left check window X_{lt} (lower r positions of the current code word Z_c)

$$z_1, z_2, \ldots, z_{r-1}, z_r,$$

– right check window X_{rg} (higher $r-1$ positions of the current code word Z_c)

$$z_m, z_{m+1}, \ldots, z_n, (m=n-2r-2)$$

and main check window $X^{(w)}$ (all positions of the source word (7)).

3. To mark all positions of code word Z located in windows X_{lt} and X_{rg} during w iteration in $X^{(w)}$ main check window $X^{(w)}$.

If all positions in window $X^{(w)}$ are marked, then all digits of word Z are checked during w iterations, go to step 5.

4. To increase the iteration number $w=w+1$. To perform permutation (for example, of $i\rightarrow(i+2)\mathrm{mod}n$ form) of positions of code word Z_c and to form the next current code word Z_c of the permutated positions.

If the permutated code word Z_c coincides with the source word (7), then go to step 5, otherwise go to step 2.

5. The end.

The main idea of the algorithm is that positions of error patterns of multiplicity τ fall into left and right check windows during several iterations (can occur simultaneously). The presence of such patterns ensures verification of the corresponding errors of multiplicity τ.

This algorithm is convenient to perform in the table.

Example 2. Let us determine the degree of error-correcting capabilities with respect to double errors of a cyclic (31,26) Hamming code with a primitive generator polynomial $g(x)$ of degree 5 (Table 2). Such errors correspond to 30 double errors patterns

$z_1,z_2,$
$z_1,z_3,$
. . .
$z_1,z_{31}.$

After formation of three check windows, we can see that checking of positions z_1,z_2,z_3,z_4,z_5 and positions z_{28},z_{29},z_{30}, z_{31} (highlighted in red) occurs at the first iteration. New positions z_7 and z_9 fall into the left check window, and positions z_{24} and z_{26} fall into the right check window at the second iteration with the permutation of $i\rightarrow(i+2)$mod31 form. We notice these new positions in $X^{(w)}$ main check window.

New positions z_{13} and z_{17} fall into the left check window, and positions z_{16} and z_{20} fall into the right check window at the third iteration. There are new positions added in both extreme windows at the fourth iteration: z_{25} andz_8. At the fifth iteration, there are new positions in both extreme windows: z_{18} and z_{15}. Here permutations lead to the source codeword.

Table 2

Checking of codeword positions Z in iterations

Iterations	X_{lt} Left check window	$X^{(w)}$Main check window	X_{rg} Right check window
1	$z_1z_2z_3z_4z_5$	$z_1z_2z_3z_4z_5z_6z_7z_8z_9z_{10}z_{11}z_{12}z_{13}z_{14}z_{15}z_{16}$ $z_{17}z_{18}z_{19}z_{20}z_{21}z_{22}z_{23}z_{24}z_{25}z_{26}z_{27}z_{28}z_{29}z_{30}z_{31}$	$z_{28}z_{29}z_{30}z_{31}$
2	$z_1z_3z_5z_7z_9$	$z_1z_2z_3z_4z_5z_6z_7z_8z_9z_{10}z_{11}z_{12}z_{13}z_{14}z_{15}z_{16}$ $z_{17}z_{18}z_{19}z_{20}z_{21}z_{22}z_{23}z_{24}z_{25}z_{26}z_{27}z_{28}z_{29}z_{30}z_{31}$	$z_{24}z_{26}z_{28}z_{30}$
3	$z_1z_5z_9z_{13}z_{17}$	$z_1z_2z_3z_4z_5z_6z_7z_8z_9z_{10}z_{11}z_{12}z_{13}z_{14}z_{15}z_{16}$ $z_{17}z_{18}z_{19}z_{20}z_{21}z_{22}z_{23}z_{24}z_{25}z_{26}z_{27}z_{28}z_{29}z_{30}z_{31}$	$z_{16}z_{20}z_{24}z_{28}$
4	$z_1z_9z_{17}z_{25}z_2$	$z_1z_2z_3z_4z_5z_6z_7z_8z_9z_{10}z_{11}z_{12}z_{13}z_{14}z_{15}z_{16}$ $z_{17}z_{18}z_{19}z_{20}z_{21}z_{22}z_{23}z_{24}z_{25}z_{26}z_{27}z_{28}z_{29}z_{30}z_{31}$	$z_{31}z_8z_{16}z_{24}$
5	$z_1z_{17}z_2z_{18}z_3$	$z_1z_2z_3z_4z_5z_6z_7z_8z_9z_{10}z_{11}z_{12}z_{13}z_{14}z_{15}z_{16}$ $z_{17}z_{18}z_{19}z_{20}z_{21}z_{22}z_{23}z_{24}z_{25}z_{26}z_{27}z_{28}z_{29}z_{30}z_{31}$	$z_{30}z_{15}z_{31}z_{16}$

Thus, it is possible to identify and correct 21 double error patterns, which is 68 % of all possible double errors for a given code, after 5 iterations. Thus, we replace the traditional complete checking of all possible options of comparison of code words with the algorithm of directional search for a solution to the problem, which leads to significant savings in calculational time.

8. The relationship between error-correcting capabilities and parameters of IDCC

Iterative decoding methods of deterministic codes give possibility to approach the solution of the key problem of coding: finding the minimum code redundancy to provide the specified error-correcting capabilities of a code, from a new perspective.

We can find firstly the exact degree (expressed, for example, in %) of error correction of the given multiplicity τ for (*n, k*)-code. We can gradually increase a degree of correction of given errors gradually increasing (*n*–*k*) value, which is equivalent to increasing of the length of check window *X*.

For example, it is sufficient to increase the word length of a check word by one to correct all double errors from Example 2 completely. For this purpose, we can use $g(x)(x+1)$, generator polynomial, which is equivalent to using the (31,25) Abramson code.

The iteratively decoded cyclic (15,11)-code given in [11] gives possibility to correct 40 % of triple errors. It is necessary to increase the check word to digit 7 to increase error-correcting capabilities to 100 %.

If necessary, it is possible to reduce error-correcting capabilities of a code.

Of course, there remains the task of distinguishing between errors of different multiplicities (either using a slave cyclic code [11] or by the traditional way using additional information from demodulator [9]).

In traditional cyclic codes, we select number of digit of a check word Ψ (a length of check window X) based on the requirement that it can represent all error syndromes of a given multiplicity τ. During iterative decoding, error syndromes are distributed on separate iterations, which reduce the number of digit of a check word Ψ (check window X) and the possibility of placing error syndromes of higher multiplicity in it is provided [6]. A decrease length of word Ψ increases length k of informational word I, which leads to an increase in code rate.

In known codes (for example, BCH codes), for neighboring τ and τ+1 values, parameters *n*, *k* and *r* of a code change at large intervals. The indicated intervals are much smaller during iterative decoding. It is convenient for practical implementation.

9. The study of error-correcting capabilities of probabilistic codes

Let us consider LDPC codes in more detail. These codes are the most complete embodiment of the Shannon theorem on the transmission of information via a communication channel with interference in practice. If a code word length is infinitely long (which is equivalent to an infinitely long decoding time), it is possible to correct almost all errors. If the task was to correct all errors during one time, then we would need a decoder of infinite complexity. Creation of such an error correction code is not possible in practice.

LDPC codes partially bypassed of the problem of complexity thanks to the multi-time (iterative) decoding method. Decoding complexity decreases to acceptable values due to an increase in decoding times (iterations) [22].

It is possible to solve the problem of error correction capabilities in a similar way: amount of errors detected and corrected increases due to an increase in decoding iterations.

A characteristic feature of probabilistic iteratively decoded codes is that it is not possible to express a degree of their error-correcting capabilities by accurate analytical estimates.

We can assume that to probabilistic codes will correspond to probabilistic estimates for each error multiplicity most closely. Therefore, we can represent error-correcting capabilities of LDPC codes for errors of multiplicity τ_i and less as a sequence of non-integer numbers

$$p_1(), p_2(), \dots p_{i-1}(), p_i(),$$

where $p_i()$ is the probability of correction of errors of multiplicity τ_i,

$$p_1()=1, p_i() \le p_{i-1}().$$

Value $p_1()$ for a single error is always equal to 1, and the remaining values $p_i()$ will increase gradually at each iteration from 0 to 1, not reaching unity for large error multiplicities.

The similar situation with a change in the error-correcting capabilities occurs in case of iterative decoding of cyclic codes based on permutations of codeword positions [11]. We can also express the degree of error correction by a sequence of non-integer numbers (or as a percentage of the total number of errors) [23].

Factors of the error-correcting capabilities (changeability and non-integer representation) of iterative codes of various classes make incorrect to compare them with the error-correcting capabilities of traditional deterministic codes based on a single, constant and integer parameter d_{min}.

10. Discussion of criteria for error-correcting capabilities of codes

There are many unsolved problems in the modern theory of error correction coding. The most important of them is the problem of effective (not NP-hard) determination of the error-correcting capabilities of a code.

On the one hand, there is a wide variety of error-correcting codes. On the other hand, there are two main criteria for estimation of codes:

– by the distance to the Shannon limit on BER curves;

– based on the minimum code distance and asymptotic bounds.

The ancestor of the probabilistic branch of the information theory, C. Shannon, also considered codes only from the probabilistic side and, therefore, the first criterion until today is the main one for most of error correction codes.

However, for deterministic codes other criteria (code distance dmin and asymptotic bounds) are known [7]. These bounds give possibility to estimate properties of codes for detection of errors depending on the introduced redundancy. The proposed IDCCs are closest ones to the asymptotic Hamming bound. From this point of view they have a significant advantage relative to well-known codes, in particular, BCH codes [20]. In practice, the advantage manifests itself in an increase in code rates and transferring of more useful information per unit of time (Table 3).

We can note regarding parameter $d_{\min}$ that it gives a very approximate estimate of the error-correcting capabilities even for deterministic codes, since it is possible to correct errors beyond $d_{\min}$ for most codes [23]. We can obtain the most accurate estimates by direct determination of amount of corrected errors, which is relatively easy to do for IDCC codes.

LDPC codes have a deserved advantage among probabilistic codes due to the maximum proximity to the Shannon limits. Attempts to estimate them using code distance $d_{\min}$ are incorrect and only diminish their advantages.

Table 3

Parameters of iterative and traditional codes

Code	Code rate, k/n	Length of a check word,r	Number of corrected errors	Degree of error correction	Number of iterations
(15,11) Hamming	73 %	4	1	100 %	1
(15,7) BCH	47 %	8	2	100 %	1
(15,5) BCH	33 %	10	3	100 %	1
(31,16) BCH	52 %	15	3	100 %	1
(31,21) BCH	68%	10	2	100 %	1
(15,11) IDCC	73 %	4	2	87 %	3
(15,11) IDCC	73 %	4	3	40 %	4
(17,10) IDCC	59 %	7	3	100 %	4
(31,25) IDCC	81 %	6	2	100 %	5

If we use the corresponding criteria for each type of a code, then we will see interesting patterns between the considered probabilistic and deterministic iterative codes.

Firstly, all iterative codes increase their error-correcting capabilities gradually, with an increase in a series of iterations.

Secondly, we can set the error-correcting capabilities as a percentage for errors of different multiplicity for the indicated iterative codes.

Thirdly, iterative codes are high-rate codes.

We can choose the minimum possible length of a check word to ensure the specified error-correcting capabilities of a code in IDCC. This feature makes iterative codes different from other error correction codes, which form their error-correcting capabilities immediately in the form of an integer parameter $d_{\min}$.

Each type of iterative code has its own scope of application, where its advantages are the most useful. From this standpoint, LDPC codes are best for large code lengths, for which approximate estimates of their correcting capabilities are possible with a validation less than 100 %, and IDCC are best for small lengths of codes with accurate estimates of their error-correcting capabilities.

Thus, studies have showed the following:

– it is necessary to distinguish probabilistic and deterministic error correction codes;

– an iterative decoding approach is possible not only for probabilistic codes, but also for deterministic codes;

– there is much in common between iterative probabilistic and deterministic codes, however, the criteria for estimation of their error-correcting capabilities are different;

– the advantages of the proposed permutation decoding are reducing of complexity of calculations and complete formalization of actions in algorithms for determination of error-correcting capabilities of deterministic codes;

– further studies are required to simplify algorithms for determination of error-correcting capabilities of codes.

Of course, these arguments are preliminary, the development of the error correction codes theory continues.

11. Conclusions

1. We performed mathematical substantiation of permutation decoding of cyclic codes. We proposed a generator of cyclic permutations based on two Moore automata. Formation of one state of an automaton of the high level occurs with the help of an automaton of the low level. Cycles of permutations of codeword positions are formed with the help of states of the high-level automaton.

2. We carried out the study of regular and irregular states of an automaton based on the automaton representation of cyclic codes and gave quantitative estimates. We proved that irregular syndromes are transformed into regular ones under the influence of cyclic permutations. And complexity of decoding cyclic codes decreases with an increase in amount of regular syndromes.

3. We proposed methods for accurate determination of error-correcting capabilities of IDCC based on cyclic permutations. For these codes, as well as for other classes of codes, complexity of resolution of this problem will increase significantly with an increase in error rate, but it will be slower in $n\tau$ times due to a simpler code structure.

4. We substantiated the relationship between error-correcting capabilities and IDCC parameters. We showed a significant increase in IDCC rate in comparison with the known BCH codes (from 1.2 to 2.2 times for the examples given).

References

1. Shennon K. Raboty po teorii informacii i kibernetike. Moscow, 1963. 829 p.
2. Ursul A. D. Priroda informacii. Filosofskiy ocherk. Moscow: Politizdat, 1968. 288 p.
3. Hartley R. V. L. Transmission of Information // Bell System Technical Journal. 1928. Vol. 7, Issue 3. P. 535–563. doi: https://doi.org/10.1002/j.1538-7305.1928.tb01236.x
4. Kolmogorov A. N. Tri podhoda k opredeleniyu ponyatiya “kolichestvo informacii” // Problemy peredachi informacii. 1965. Vol. 1, Issue 1. P. 3–11.
5. Kolmogorov A. N. Teoriya informacii i teoriya algoritmov. Moscow: Nauka, 1987. 304 p.

6. Bulychev I. I., Soroka M. Yu. About the nature and the essense of information // Noosfernye issledovaniya. 2016. Issue 1-2 (13-14). P. 191–207.
7. Piterson U., Ueldon E. Kody, ispravlyayushchie oshibki. Moscow: Mir. 1976, 596 p.
8. Sklyar B. Cifrovaya svyaz'. Teoreticheskie osnovy i prakticheskoe primenenie. Moscow: Izd. dom «Vil'yams», 2004. 1104 p.
9. Klark Dzh. ml., Keyn Dzh. Kodirovanie s ispravleniem oshibok v sistemah cifrovoy svyazi. Moscow: Radio i svyaz', 1987. 392 p.
10. Dumer I., Micciancio D., Sudan M. Hardness of approximating the minimum distance of a linear code // IEEE Transactions on Information Theory. 2003. Vol. 49, Issue 1. P. 22–37. doi: https://doi.org/10.1109/tit.2002.806118
11. Semerenko V. Iterative harddecision decoding of combined cyclic codes // Eastern-European Journal of Enterprise Technologies. 2018. Vol. 1, Issue 9 (91). P. 61–72. doi: https://doi.org/10.15587/1729-4061.2018.123207
12. Garrammone G., Declercq D., Fossorier M. P. C. Weight Distributions of Non-Binary Multi-Edge Type LDPC Code Ensembles: Analysis and Efficient Evaluation // IEEE Transactions on Information Theory. 2017. Vol. 63, Issue 3. P. 1463–1475. doi: https://doi.org/10.1109/tit.2016.2647724
13. Computing the Minimum Distance of Nonbinary LDPC Codes / Liu L., Huang J., Zhou W., Zhou S. // IEEE Transactions on Communications. 2012. Vol. 60, Issue 7. P. 1753–1758. doi: https://doi.org/10.1109/tcomm.2012.050812.110073a
14. Uryvskiy L. A., Osipchuk S. A. Issledovanie svoystv pomekhoustoychivyh kodov klassa LDPC. Naukoemkie tekhnologii v infokommunikaciyah: obrabotka informacii, kiberbezopasnost', informacionnaya bor'ba: kolekt. monogr. / V. M. Bezruk, V. V. Barannik (Eds.). Kharkiv, 2017. P. 137–139.
15. Error-Correction Coding and Decoding. Bounds, Codes, Decoders, Analysis and Applications / Tomlinson M., Tjhai C. J., Ambroze M. A., Ahmed M., Jibril M. Springer, 2017. doi: https://doi.org/10.1007/978-3-319-51103-0
16. Distance Properties of Short LDPC Codes and Their Impact on the BP, ML and Near-ML Decoding Performance / Bocharova I. E., Kudryashov B. D., Skachek V., Yakimenka Y. // Lecture Notes in Computer Science. 2017. P. 48–61. doi: https://doi.org/10.1007/978-3-319-66278-7_5
17. Butler B. K., Siegel P. H. Error Floor Approximation for LDPC Codes in the AWGN Channel // IEEE Transactions on Information Theory. 2014. Vol. 60, Issue 12. P. 7416–7441. doi: https://doi.org/10.1109/tit.2014.2363832
18. Berlekemp E. Algebraicheskaya teoriya kodirovaniya. Moscow: Mir, 1971. 477 p.
19. Semerenko V. P. Burst-Error Correction for Cyclic Codes // IEEE EUROCON 2009. 2009. doi: https://doi.org/10.1109/eurcon.2009.5167864
20. Semerenko V. P. Parallel Decoding of Bose-Chaudhuri-Hocquenghem Codes // Engineering Simulation. 1998. Vol. 16, Issue 1. P. 87–100.
21. Semerenko V. P. Teoriya tsyklichnykh kodiv na osnovi avtomatnykh modelei: monohrafiya. Vinnytsia: VNTU, 2015. 444 p.
22. Gallager R. Kody s maloy plotnost'yu proverok na chetnost'. Moscow: Mir, 1966. 144 p.
23. Semerenko V. P. Estimation of the correcting capability of cyclic codes based on their automation models // Eastern-European Journal of Enterprise Technologies. 2015. Vol. 2, Issue 9 (74). P. 16–24. doi: https://doi.org/10.15587/1729-4061.2015.39947