ON IMPROVING THE DECODING OF RANDOM CODES BASED ON QUASIGROUPS

Aleksandra Popovska-Mitrovikj  
Smile Markovski  
Verica Bakeva

UKIM, Faculty of Computer Science and Engineering  
{aleksandra.popovska.mitrovikj, smile.markovski, verica.bakeva}@finki.ukim.mk

ABSTRACT

In this paper we examine the influence of the length of the messages on the performances of error-correcting codes based on quasigroup transformations proposed elsewhere. We made simulations for binary symmetric channel, for different lengths of messages with the same rate and find the packet-error probability (PER) and the bit-error probability (BER). Also we give some ideas for speed-up the decoding process and improving the performances of random codes based on quasigroups.

Key words: error-correcting code, random code, packet-error probability, bit-error probability, quasigroup, quasigroup transformation.

I. INTRODUCTION

Here we consider the codes defined in [3] and our task is to investigate the influence of the length of the messages to the code performances with the same rate. First we give some definitions and explanation of the coding and decoding algorithm of these codes.

A. Quasigroups and Quasigroups string transformation

A quasigroup \((Q, *)\) is a groupoid, i.e., a set \(Q\) with a binary operation \(\ast : Q^2 \rightarrow Q\), such that for all \(u, v \in Q\), there exist unique \(x, y \in Q\), satisfying the equalities \(u \ast x = v\) and \(x \ast y = v\). In the sequel we assume that the set \(Q\) is a finite set. The cardinality of this set, \(|Q|\), is called an order of the quasigroup. The main body of the multiplicity table of a quasigroup is a Latin square over the set \(Q\). Given a quasigroup \((Q, \ast)\) a new operation \(\backslash\), called a parastrophe, can be derived from the operation \(\ast\) as follows:

\[ x \ast y = z \Leftrightarrow y = x \backslash z. \]

Then the algebra \((Q, \ast, \backslash)\) satisfies the identities:

\[ x \backslash (x \ast y) = y \quad \text{and} \quad x \ast (x \backslash y) = y, \]

and \((Q, \backslash)\) is also a quasigroup.

Quasigroup string transformations are defined on a finite set \(Q\) (i.e., an alphabet \(Q\)) endowed with a quasigroup operation \(\ast\), and they are mappings from \(Q^+\) to \(Q^+\), where \(Q^+\) is the set of all nonempty words on \(Q\). Note that \(Q^+ = Q \cup Q^2 \cup Q^3 \ldots\). Here, we use two types of quasigroup transformations as explained below. Let \(l \in Q\) be a fixed element, called a leader. For every \(a_i, b_i \in Q\), \(e\)– and \(d\)–transformations are defined as follows:

\[ e_i(a_1a_2 \ldots a_n) = b_1b_2 \ldots b_n \Leftrightarrow b_{i+1} = b_i \ast a_{i+1}, \]

\[ d_i(a_1a_2 \ldots a_n) = b_1b_2 \ldots b_n \Leftrightarrow b_{i+1} = a_i \ast a_{i+1}, \]

for each \(i = 0, 1, \ldots, n - 1\), where \(b_0 = a_0 = l\). By using the identities \(x \backslash (x \ast y) = y\) and \(x \ast (x \backslash y) = y\), we have that \(d_i(e_i(a_1a_2 \ldots a_n)) = a_1a_2 \ldots a_n\) and \(e_i(d_i(a_1a_2 \ldots a_n)) = a_1a_2 \ldots a_n\). This means that \(e_i\) and \(d_i\) are permutations on \(Q^n\), mutually inverse. In the code design compositions of \(e_i\) and \(d_i\) are used.

Theorem 1: [4] Consider an arbitrary string \(a = a_1a_2 \ldots a_n\) where \(a_i \in Q\), and let \(\beta\) be obtained after \(k\) applications of an \(e\)–transformation on \(a\). If \(n\) is an enough large integer then, for each \(1 \leq t \leq k\), the distribution of substrings of \(\beta\) of length \(t\) is uniform. (We note that for \(t > k\) the distribution of substrings of \(\beta\) of length \(t\) is not uniform.)

Code design uses the alphabet \(Q = \{0, 1, \ldots, 9, a, b, c, d, e, f\}\) of nibbles and a quasigroup operation \(\ast\) on \(Q\), together with its parastrophe \(\backslash\).

B. Description of coding

Let \(M = m_1m_2 \ldots m_n\) be a block of \(N_{\text{block}}\) bits, where \(m_i\) is a nibble (4-bit letter); hence, \(N_{\text{block}} = 4r\). We first add redundancy as zero bits and produce block \(L = L^{(1)}L^{(2)} \ldots L^{(s)} = L_1L_2 \ldots L_m\) of \(N\) bits, where \(L^{(i)}\) are 4-nibble words, \(L_i\) are nibbles, so \(m = 4s\), \(N = 16s\). After erasing the redundant zeros from each \(L^{(i)}\), the message \(L\) will produce the original message \(M\). On this way we obtain an \((N_{\text{block}}, N)\) code with rate \(R = N_{\text{block}}/N\). The codeword is produced from \(L\) after applying the encryption algorithm given in Figure 1. For that aim, previously, a key \(k = k_1k_2 \ldots k_t\) of length \(n\) nibbles should be chosen. The obtained codeword of \(M\) is \(C = C_1C_2 \ldots C_m\), where \(C_i\) are nibbles.

C. Description of decoding

After transmission through a noise channel (for our experiments we use binary symmetric channel), the codeword \(C\) will be received as message \(D = D^{(1)}D^{(2)} \ldots D^{(s)} = D_1D_2 \ldots D_m\), where \(D^{(i)}\) are blocks of 4 nibbles and \(D_i\) are nibbles. The decoding process consists of four steps: (i) procedure for generating the sets with predefined Hamming distance, (ii) inverse coding algorithm, (iii) procedure for generating decoding candidate sets and (iv) decoding rule.

The probability that \(\leq t\) bits in \(D^{(i)}\) are not correctly transmitted is \(P(p; t) = \sum_{k=0}^{t} \binom{16}{k} p^k(1-p)^{16-k}\), where \(p\) is probability of bit-error in a binary symmetric channel. Let \(B_{\text{max}}\) be an integer such that \(1-P(p; B_{\text{max}}) \leq \eta b\) and \(H_i = \{a \in Q^4, H(D^{(i)}, a) \leq B_{\text{max}}\}, \) for \(i = 1, 2, \ldots, s\), where \(H(D^{(i)}, a)\) is the Hamming distance between \(D^{(i)}\)
The message M is longer, then for obtaining the same rate R, the number of blocks in the redundant message must be larger and codeword is longer than for shorter messages. So, for longer codeword we have larger packet-error probability. We check this result experimentally.

In [5], we gave experimental results, for different patterns of redundancy, for code (72,288) with rate R=1/4 for binary symmetric channel. For comparison we have made experiments with same quasigroup, key and rate for code (144,576) with twice longer messages. We compare the best results for two codes. For the code (72,288) the best results are obtained for the pattern: 11011001000000011010001000000110110010000000 110010010000000000000, and for (144,576) for the pattern: 11011001000000011010001000000110110010000000 1100100100000000000000011011001000000000110010001000 11000010110100000000011011001000000000110010001000 100000110101000000000110110010000000000000000000.

In the experiments we use key $k = 0123456789$ and the quasigroup given in [5]. The obtained results for PER and BER for different values of bit-error probability $p$ of binary symmetric channel and $B_{\text{max}} = 4$ are presented in Table 1 and Table 2.

Table 1: Experimental results for packet-error probability

<table>
<thead>
<tr>
<th>p</th>
<th>$RCBQ(72,288)$</th>
<th>$RCBQ(144,576)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.03</td>
<td>0.00171</td>
<td>0.00343</td>
</tr>
<tr>
<td>0.04</td>
<td>0.00594</td>
<td>0.01200</td>
</tr>
<tr>
<td>0.05</td>
<td>0.01594</td>
<td>0.03200</td>
</tr>
<tr>
<td>0.06</td>
<td>0.03594</td>
<td>0.06800</td>
</tr>
<tr>
<td>0.07</td>
<td>0.06656</td>
<td>0.13200</td>
</tr>
<tr>
<td>0.08</td>
<td>0.11313</td>
<td>0.22114</td>
</tr>
<tr>
<td>0.09</td>
<td>0.18875</td>
<td>0.32571</td>
</tr>
</tbody>
</table>

Table 2: Experimental results for bit-error probability

<table>
<thead>
<tr>
<th>p</th>
<th>$RCBQ(72,288)$</th>
<th>$RCBQ(144,576)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.03</td>
<td>0.00089</td>
<td>0.00228</td>
</tr>
<tr>
<td>0.04</td>
<td>0.00343</td>
<td>0.00750</td>
</tr>
<tr>
<td>0.05</td>
<td>0.00928</td>
<td>0.01746</td>
</tr>
<tr>
<td>0.06</td>
<td>0.02239</td>
<td>0.04121</td>
</tr>
<tr>
<td>0.07</td>
<td>0.04065</td>
<td>0.08373</td>
</tr>
<tr>
<td>0.08</td>
<td>0.06485</td>
<td>0.14621</td>
</tr>
<tr>
<td>0.09</td>
<td>0.11357</td>
<td>0.21843</td>
</tr>
</tbody>
</table>

From the results in Table 1 and Table 2 we can conclude that the packet-error probability and bit-error probability for the longer codewords are approximately twice larger than for the shorter codewords. In Table 3 the

II. THE INFLUENCE OF THE LENGTH OF THE MESSAGES TO THE CODE PERFORMANCES

In the known error-correcting codes, the probabilities of bit-error and packet error decrease when the length of the message increase. We will show that it is not case for random codes based on quasigroups. From the formula for theoretical packet-error probability given in Theorem 2, it is clear that for larger number of blocks in the codeword the packet-error probability is larger. This follow from the fact that in this case we have more iteration in the process of decoding and in each of them the correct block should be in the appropriate (corresponding) set $S$. When
A. The decoding speed

As it is almost always the case, the speed of the decoding process is one of the biggest problem for these random codes based on quasigroups as well. Depending on the patterns, the decoding process is slow in some iterations since the number of elements in the sets \( S \) is very large. In fact, if we distribute the redundant zeros more uniformly, the sets \( S \) are not going to be very large, but then many more-candidate-errors will appear. So, we need to put more redundant zeros at the end of the patterns. In fact, finding good equilibrium for placing the redundant zeros is an open problem and by experiments several enough satisfactory patterns are discovered.

In order to improve the decoding speed, we have experimented with another idea for decoding. Namely, instead of using a \((72,288)\) code, we have used together two \((72,144)\) codes, that decode a same message of length 72 bits. Then, for the obtained sets \( S^{(1)} \) and \( S^{(2)} \) we have made its intersection \( S = S^{(1)} \cap S^{(2)} \) in each iteration step. Note that the code rate in this case is again 1/4, since we produce two codewords of length 144 bits.

We have analyzed the number of elements in \( S \), when we have two processes of decoding for different parameters. First we apply the coding algorithm on the same redundant message twice with different key or quasigroups and we obtain twice longer codeword (obtained as concatenation of two codewords). The codeword is transmitted through a binary symmetric channel. The outgoing message is divided in two messages with equal lengths and we decode them parallel with the corresponding parameters. After each iteration we take an intersection of the two sets \( S^{(1)} \) and \( S^{(2)} \) obtained in both decoding processes. In the next iteration the both processes use the obtained intersection \( S \). The preliminary experiments show that this modification significantly reduces the number of elements in the sets \( S \). The problem here is that for obtaining code with rate 1/4 we need pattern for rate 1/2, but it is hard to make good pattern for this rate. In the initial experiments, with this modified algorithm, we obtained similar values for PER and BER as for \((72,288)\) code. What is very promising for this modified algorithm is that we have big improvement in the speed of the decoding process, which is in average more than four times faster.

B. Reducing the null-errors

In the case when null-errors appear, i.e., when the set \( S \) is empty, we are going to resolve this problem by backtracking. Namely, if in the \( i \)-th iteration we obtain \( S = \emptyset \), it means that in some previous step \( j < i \) we have lost the right word that had to be processed. This happened because \( B_{max} \) is too small. So, we have to go back to the \((i-1)\)-th or \((i-2)\)-th step and there to work with increased value of \( B_{max} \), i.e., we take \( B_{max} + 1 \) or \( B_{max} + 2 \). The experiments of this kind are under preparation and we expect improving of the PER and BER.
Anyhow, we expect the decoding speed to be increased, but not significantly.

C. Reducing the more-candidate-errors

Another idea that we consider for elimination the unsuccessful decoding with more-candidate-error is the following one. From the experiments we can see that when the decoding process ends with more elements in the last set $S$, almost always in this set is the correct message. So, in these cases we can randomly select a message from the set $S$ in the last iteration and it can be taken as the decoded message. If the selected message is the correct then the bit-error is 0, so the BER will also be reduced. In the experiments we have made with this modification we got that in around half of the cases, when the last set $S$ contained more than one element, the correct message is selected.

References


