UDC 519.72
doi:10.31799/1684-8853-2022-6-41-52
EDN: UWXZHN

Articles

Decoding of linear codes for single error bursts correction based on the determination of certain events

A. A. Ovchinnikov[a], *PhD, Tech., Associate Professor, orcid.org/0000-0002-8523-9429, mldoc@guap.ru*
A. M. Veresova[a], *Post-Graduate Student, orcid.org/0000-0002-3792-9249*
A. A. Fominykh[a], *Master, orcid.org/0000-0002-1412-5766*
[a]*Saint-Petersburg State University of Aerospace Instrumentation, 67, B. Morskaia St., 190000, Saint-Petersburg, Russian Federation*

Introduction: *In modern systems for communication, data storage and processing the error-correction capability of codes are estimated for memoryless channels. In real channels the noise is correlated, which leads to grouping error in bursts. A traditional method to fight this phenomenon is channel decorrelation, which does not allow developing of coding schemes, mostly utilizing the channel capacity. Thus the development of bursts decoding algorithms for arbitrary linear codes is the actual task.* ***Purpose:*** *To develop a single error burst decoding algorithm for linear codes, to estimate the decoding error probability and computational complexity.* ***Results:*** *Two approaches are proposed to burst error correction. The first one is based on combining the window sliding modification of well-known bit-flipping algorithm with preliminary analysis of the structure of parity check matrix. The second one is based on the recursive procedure of constructing the sequence of certain events which, in the worst case, performs the exhaustive search of error bursts, but in many cases the search may be significantly decreased by using the proposed heuristics. The proposed recursive decoding algorithm allows a guaranteed correction of any single error bursts within burst-correction capability of the code, and in many cases beyond the burst-correction capability. The complexity of this algorithm is significantly lower than that of a bit flipping algorithm if the parity-check matrix of the code is sparse enough. An alternative hybrid decoding algorithm is proposed utilizing the bit-flipping approach and showing the error probability and completion time comparable to the recursive algorithm, however, in this case the possibility of a guaranteed burst correction hardly can be proved.* ***Practical relevance:*** *The proposed decoding methods may be used in modern and perspective communication systems, allowing energy saving and increasing reliability of data transmission by better error performance and computational complexity.*

Keywords *— channels with memory, low-density parity-check codes, burst error correction.*

For citation: Ovchinnikov A. A., Veresova A. M., Fominykh A. A. Decoding of linear codes for single error bursts correction based on the determination of certain events. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2022, no. 6, pp. 41–52. doi:10.31799/1684-8853-2022-6-41-52, EDN: UWXZHN

Introduction

Error-correcting coding is very important mean to provide the reliable data communication [1–3]. For the last 20 years, significant attention was paid to investigating of low-density parity-check (LDPC) codes. Despite of moving during the last years the research focus to the area of polar codes [4–6], LDPC codes remain part of many contemporary standards, including digital video broadcasting (DVB), local wireless networks (802.11 WiFi), optical fiber communication (G.975.1), mobile networks of fifth generation (5G), etc.

For the last decades, traditionally the analysis of error-correcting performance of different codes in communication systems is hold by intensive modeling in the channel with additive white Gaussian noise [7]. This is mainly connected with the fact that modern code constructions such as turbo codes or LDPC codes are pseudo-random, and while providing low error probabilities, which is agreed with Shannon's random coding theorems, they are hardly suited for theoretical analysis.

Along with it, assumption of Gaussian nature of noise is justificated, from the one hand, by simplicity of this model, and from the other is provided by receiving decorrelation procedures, while in fact in many channels the noise process is not independent. To describe the typical errors ion such channels, the notion of single error burst of length b in error vector of length n is used — this is any binary vector consisting of n elements, containing all its non-zero elements within b subsequent positions. Since the number of individual errors in such burst may exceed the error-correction capability of the code (for independent errors), they cannot be managed by classical algebraic decoders of cyclic codes. Due to pseudo-random nature, absence of strict structure and per-symbol decoding procedures of LDPC codes they are less sensitive to errors grouping, however, when using LDPC codes, artificial interleaving procedures are used to decorrelate the channel [8].

At the same time, it is known from information theory, that if an interleaving is applied to channel, and the property of noise correlation is not taken into account during decoding, we get an equivalent (in terms of average number of erroneous bits in channel) memoryless channel with less capacity [9]. Burst correction has low attention in classical coding theory, and is considered mainly for cyclic codes

[10–12]. In the recent publication [13] the burst erasure correction is considered, but only for the family of block-permutation LDPC codes and under consumption of fixed burst positions (phased burst). Thus the actual task is development of coding and decoding procedures for channels with memory, in particular, for single burst correction.

In [14] it was shown that the maximal burst length that can be corrected by LDPC code may be calculated using the polynomial-time procedure, taking the parity-check matrix as input. In fact the sparseness property of parity-check matrix does not taken into account in this procedure, so it may be applied to any linear code. However, the development of correspondent decoding algorithm realizing burst-correction capability of the code remain an open problem.

In this paper an approaches using sliding windows to single bursts correction is considered. These approaches are based on analysis of LDPC code parity-check matrix, however, obtained algorithms may be used for any linear codes, though with increasing the density of parity-check matrices the decoding complexity is also increasing.

Low-density parity-check codes

Low-density parity-check codes were proposed by R. Gallager [15]. These codes usually have low minimal distance, however, good weight spectral properties and iterative decoding algorithms allow providing very low decoding error probabilities [16, 17]. LDPC codes are usually defined as elements of probabilistic ensembles, characterized by weight distributions of columns of parity-check matrix. From the analysis of the properties of LDPC ensembles it is known that irregular constructions (those with variable column weights) with well optimized weight distributions provide fast decreasing of error probability in the area of low signal-to-noise ratio [18]. However, these constructions are usually suffer from the effect of error-floor (low decreasing of error probability with signal-to-noise ratio increasing), and besides, encoding and decoding procedures for them may be not so effectively implemented as for regular constructions. That is why today in practice the block-permutation construction of LDPC codes is most widely used (including most communication standards), when the parity-check matrix has the form [19, 20]

$$\mathbf{H} = \begin{bmatrix} \mathbf{C}^{i_{11}} & \mathbf{C}^{i_{12}} & \dots & \mathbf{C}^{i_{1\rho}} \\ \mathbf{C}^{i_{21}} & \mathbf{C}^{i_{22}} & \dots & \mathbf{C}^{i_{2\rho}} \\ \dots & \dots & \dots & \dots \\ \mathbf{C}^{i_{\gamma 1}} & \mathbf{C}^{i_{\gamma 2}} & \dots & \mathbf{C}^{i_{\gamma\rho}} \end{bmatrix}, \qquad (1)$$

where $\mathbf{C}$ — $(m \times m)$-matrix of cyclic permutation:

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix}.$$

Construction given in (1) defines regular quasi-cyclic code, for which especially computationally effective encoding and decoding algorithms may be implemented, and to obtain the advantages of optimized irregular distributions the masking may be used by replacing some cyclic permutation blocks in (1) by zero blocks.

It is known that the performance of LDPC decoders may be negatively affected by "bad" placements of non-zero elements in the parity-check matrix. As the simplest requirement to avoid this we will assume that there are no two rows or columns having more than one non-zero element in common positions (in this case one say that the matrix contains no cycles of length 4).

Calculation of the minimal distance of linear code, which defines the maximal number of correctable independent errors is NP-hard problem, at the same time, calculation of the maximal length of correctable single burst may be performed in polynomial time using the procedure described in [14]. This procedure is based on the fact that if the code can correct all single error bursts of length $b_{\max}$ and less, then all error vectors, forming such bursts, should be in different cosets, or equivalently, have different syndrome. Thus, for any bursts $\mathbf{e}_1$ and $\mathbf{e}_2$ of length $b_{\max}$ (or less) should holds $\mathbf{e}_1\mathbf{H}^{\mathrm{T}} \neq \mathbf{e}_2\mathbf{H}^{\mathrm{T}}$, besides, multiplication of error vector forming the burst of length b by the parity-check matrix is equivalent to linear combination of columns of $\mathbf{H}$ in sub-matrix limited by b columns. That is, in matrix $\mathbf{H}$ there should be no two submatrices of $b_{\max}$ subsequent columns, whose linear combination has the same result. This is guaranteed in case when the rank of concatenation of these two submatrices by columns is $2b_{\max}$.

From the same considerations in the block-permutation matrix (1) the maximal length of correctable error burst is always less than the block size, $b_{\max} < m$, since sum of all columns within the block is all-ones column. However, in case when the parity-check matrix $\mathbf{H}$ is masked by all-zero blocks, this may increase the length of correctable error burst. Experiments show that without masking, with high probability, especially when m is prime and $\mathbf{H}$ has no cycles of length 4, the equation $b_{\max} = m - 1$ holds.

The classical decoding method of LDPC codes in the channel with soft (continuous) output (e.g. additive white Gaussian noise channel) is belief propagation algorithm. However, in case of the channel

with binary output the simpler bit flipping algorithm may be used. In this paper, we consider only binary errors and we will consider only decoders with hard input and hard output.

The bit-flipping algorithm (Fig. 1) may be easily adapted to correction of single error burst by organizing the sliding window of size b_w, with the beginning running through possible starting positions of the burst from 0 to $n - b_w$ (the decoding procedure may be significantly speeded up, if the estimations of supposed burst beginning are obtained, however, in the paper we are not considering such improvements). Clearly, one should select $b_w \geq b_{\max}$. In that algorithm the subvector of $\mathbf{y}$ with elements on positions $i, \ldots, i + b_w - 1$ is denoted as $\mathbf{y}(i, \ldots, i + b_w - 1)$, and for the matrix $\mathbf{H}$ the submatrix formed by columns with numbers $i, \ldots, i + b_w - 1$, is denoted as $\mathbf{H}(i, \ldots, i + b_w - 1)$. By default, matrix multiplication is performed modulo 2, except when the operator "$\otimes$" is used denoting operations in decimal arithmetic.

In case when the all-zero syndrome is obtained, the algorithm returns the decoded vector which is a codeword, and informs about successful decoding. If for all possible positions of the sliding window after all decoding iterations the all-zero syndrome is not obtained, decoder signaling about failure and returns the word received from the channel itself as the decoded word. It should be noted that of the burst length does not exceeds the maximal correctable length $b_{\max}$, then the state "success" always means correct decoding, otherwise "success" may be caused either by correct decoding or decoding error, that is, finding the wrong (false) codeword.

This algorithm may be modified for the "failure" state case by determining the decoding result according to some criterion, which would allow decreasing the number of bit errors. However, in this paper we will consider the error probability per transmitted word as the target decoding characteristics, so we will not consider such modifications.

For the channels with memory in [21] the modification of belief propagation algorithm is proposed, based on preliminary channel state estimation and subsequent decoding. The advantage of this algorithm is the possibility of correcting multiple bursts and individual errors, disadvantage is growing complexity comparing to binary and integer operations, and necessity to provide the channel transition probabilities at the decoder's input.

Algorithms for single error burst decoding

We will develop the algorithm for error burst correction, starting with the structure of $(r \times n)$ block-permutation matrix (1) of LDPC code (here r denotes the number of rows in $\mathbf{H}$ rather than actual number of redundant bits). As it was mentioned, for this construction (without masking by all-zero blocks) $b_{\max} < m$, where m is block size. Let the error burst of length $b \leq b_{\max}$ occurred in the channel, for simplicity we assume $b = b_{\max}$, and for the received word the syndrome $\mathbf{S}$ is calculated. The subvector of error vector $\mathbf{e}$ containing the non-zero elements (i.e. the subvector on burst positions) denote as $\mathbf{e}_b$. Assume that we have the exact knowledge about burst position and we may select the $(r \times b)$-submatrix $\mathbf{H}_b$ containing columns of the parity-check matrix $\mathbf{H}$ corresponding to burst positions. Compute the vector $\mathbf{W} = \mathbf{1}_b \mathbf{H}_b^{\mathrm{T}}$, where $\mathbf{1}_b$ is all-one vector of length b, i.e. $\mathbf{W}$ is the sum of all columns of $\mathbf{H}_b$. Consider two cases: the burst is either completely within some block of $\mathbf{H}$, or, taking into account $b < m$, disturbing no more than two adjacent blocks of $\mathbf{H}$.

The first case is shown in Fig. 2, *a*, for the parity-check matrix with block size $m = 7$ and maximal correctable burst length $b_{\max} = 6$. Since every block of $\mathbf{H}$ is cyclic permutation matrix and contains exactly one non-zero element in the row, the vector $\mathbf{W}$ may contain only zeros and ones. If some i-th syndrome's component is not zero, the correspondent i-th component of $\mathbf{W}$ should also be non-zero (otherwise successful decoding for current submatrix $\mathbf{H}_b$ is not possible, since it contains no columns which would give non-zero syndrome element in i-th row). Hence, there are the only column in $\mathbf{H}_b$ which con-

INPUT: $(r \times n)$-matrix $\mathbf{H}$, received word $\mathbf{y}$, window size b_w, maximal number of iterations N.
OUTPUT: decoded word $\hat{\mathbf{a}}$ and status "success" or "failure" of the decoder.
1. Calculate $\mathbf{S} = \mathbf{y}\mathbf{H}^{\mathrm{T}}$; if $\mathbf{S} = \mathbf{0}$, return $\hat{\mathbf{a}} = \mathbf{y}$ and "success".
2. For $i = 0$ to $n - b_w$ // cycle for sliding window positions:
2.1. For current window position set $\mathbf{S}_b = \mathbf{S}$, $\mathbf{y}_b = \mathbf{y}(i, \ldots, i + b_w - 1)$, $\mathbf{H}_b = \mathbf{H}(i, \ldots, i + b_w - 1)$.
2.2. For $j = 1$ to N // cycle for decoding iterations:
2.2.1. Calculate $\mathbf{f}_b = \mathbf{S}_b \otimes \mathbf{H}_b$ (in decimal arithmetics).
2.2.2. Determine the set $pos = \arg\max \{\mathbf{f}_b\}$ of positions numbers of $\mathbf{f}$ with maximal values.
2.2.3. Invert elements of $\mathbf{y}_b$ in pos positions.
2.2.4. Renew $\mathbf{S}_b = \mathbf{y}_b \mathbf{H}_b^{\mathrm{T}}$, if $\mathbf{S}_b = \mathbf{0}$, add (modulo 2) to subvector $\mathbf{y}(i, \ldots, i + b_w - 1)$ the vector $\mathbf{y}_b$, return $\hat{\mathbf{a}} = \mathbf{y}$ and "success".
3. Return $\hat{\mathbf{a}} = \mathbf{y}$ and "failure".

■ ***Fig. 1.*** Windowed bit-flipping algorithm

tributes to non-zero syndrome element. The number of this column corresponds to erroneous position within burst, we may subtract this column from the syndrome and continue decoding. As can be seen from the Fig. 2, *a*, for given example consideration of first three non-zero elements of the syndrome leads to all-zero syndrome and reconstructs the error burst $\mathbf{e}_b$. Consider also the vector $\mathbf{f}_b = \mathbf{S}_b \otimes \mathbf{H}_b$, which we use in windowed bit-flipping algorithm (see Fig. 1) — for considered example all its maximal components correspond to erroneous burst positions, and the others are zero (windowed bit-flipping algorithm would correct such a burst by one iteration).

Now we consider more complicated case when the burst affected two adjacent blocks of $\mathbf{H}$ (Fig. 2, *b*). In this case the rows of $\mathbf{H}_b$ may be all-zeros, contain one non-zero element, or two elements from different blocks (in this example the border between blocks are between second and third columns of $\mathbf{H}_b$). In figure the burst $\mathbf{e}_b$ and vector $\mathbf{f}_b$ are also shown. In this case the comparison by their elements does not allow, as in Fig. 2, *a* to determine error bits in $\mathbf{e}_b$ by the values of $\mathbf{f}_b$ with such confidence — for example, the value 2 which is close to maximal may correspond both to first position without error and to last with error.

Basing on considered examples we may determine the following conditions by considering elements $\mathbf{W}$ and $\mathbf{S}$, from which certain decoding states or events is followed, and formulate the corresponding decoding steps:

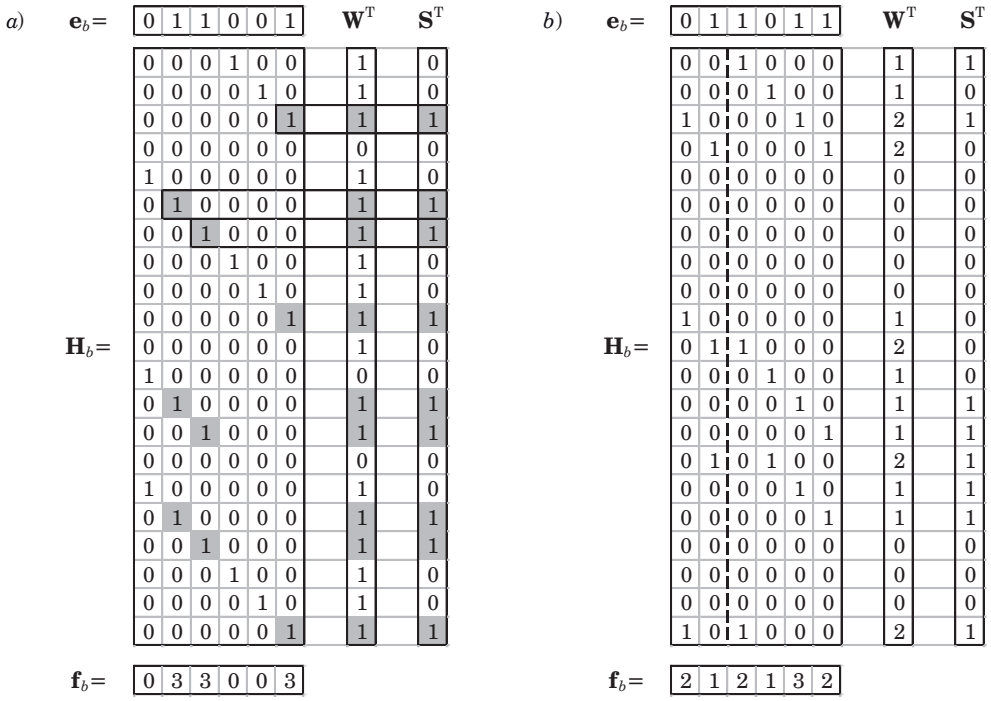
— ($\mathbf{W}_i = 0$, $\mathbf{S}_i = 1$): in this case the syndrome $\mathbf{S}$ cannot be obtained by any combination of bit errors within current burst position (i.e. current position of sliding window), decoding should be stopped and new window position should be defined (event “impossible”);

— ($\mathbf{W}_i = 1$, $\mathbf{S}_i = 1$): in this case the column correspondent to erroneous position within burst may be uniquely determined, correspondent bit of $\mathbf{e}_b$ on this position is set to 1, and the column is marked as considered (for given window position this bit within the burst would never change its value), then $\mathbf{W}$ and $\mathbf{S}$ may be renewed and decoding continues within current window (event “certain 1”);

— ($\mathbf{W}_i = 1$, $\mathbf{S}_i = 0$): in this case the column with one non-zero element in i-th row is certainly did not take part in computation of $\mathbf{S}$, the correspondent bit of $\mathbf{e}_b$ on position correspondent to this column may be set to 0, the column is marked as considered, which leads to renewal of $\mathbf{W}$, and decoding continues within current window (event “certain 0”);

— ($\mathbf{W}_i > 1$, $\mathbf{S}_i = 1$): for the considered block-permutation constructions and $b < m$ this event is possible only for $\mathbf{W}_i = 2$, but in general case one may assume larger number of ones in i-th row of $\mathbf{H}_b$, in this case we cannot certainly determine the values of any positions of the burst from i-th components of $\mathbf{W}$ and $\mathbf{S}$ (event “ambiguity”).

We may note that all listed situations are based on considering the non-zero elements of syndrome, except special case “certain 0”. First three cases determine certain events (with conditional probability



■ ***Fig. 2.*** Decoding by certain events: *a* — burst within the block; *b* — burst in two blocks

equal to 1, for the given window position): impossibility of decoding, certain zero and one values within the burst, conditioning that the burst is within the positions of current window. Basing on these events, the procedure may be determined looking through elements of **S** and **W** and performing described decoding steps, while the event "impossible" would be determined or the all-zero syndrome obtained. However, during the decoding the situation is possible, when for all non-zero syndrome elements the event "ambiguity" is determined. An example of such situation is given in Fig. 3, *a* for the case when two erroneous positions within the burst is found, and correspondent columns are excluded from consideration (in figure they are shadowed). Using the rule for the event "certain 0" in Fig. 3, *a* will lead to very fast successful decoding, but this event may be absent (Fig. 3, *b*). Note that the parity-check matrix in Fig. 3, *b* is not block-permutation, and the burst of length 9 is considered, while the code from this example may correct all bursts of length no more than 7.

In the case described in Fig. 3, *b*, the resolving of ambiguity is possible by using some heuristics [22], for example. As one variant we consider the hybrid decoder, when in case of ambiguity the decoding is switched to conventional bit-flipping within current window, until successful decoding or reaching the maximal number of iterations. Such approach defines in fact two-stage decoding which may be considered as windowed bit-flipping decoding with preliminary processing based on certain events.

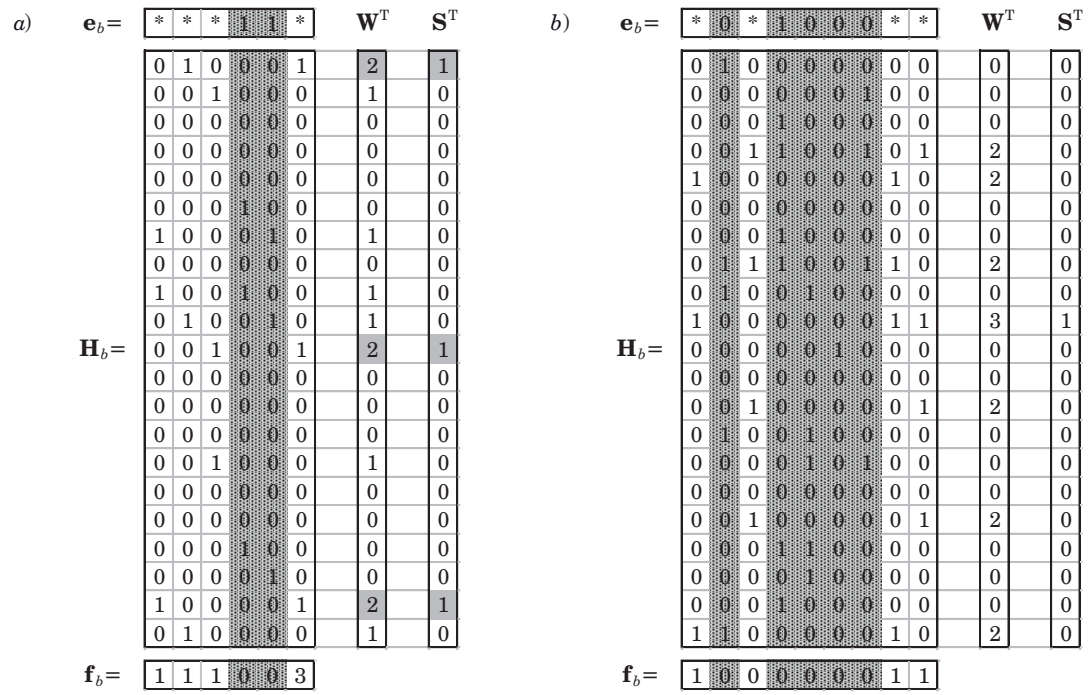
However, to ensure guaranteed decoding of burst (at least within the burst-correction capability of the code) another approach may be considered. In case of uncertainty we make trial selection of erroneous position and return to certain events consideration basing on new **W** and **S**, until success or impossibility of the decoding will be determined, in the latter case the algorithm should trace back and select another trial erroneous bits. If in the process of one trial we again meet the ambiguity, the next trial is made and so on, thus leading to tree-like exhaustive search. Surely, such search would lead to significant grow of complexity. To reduce the search, we will select as trial erroneous positions those with maximal value in vector $\mathbf{f}_b = \mathbf{S}_b \otimes \mathbf{H}_b$.

Basing on these considerations, the following recursive algorithm may be proposed as general method of single burst decoding:

1) for the current window position determine certain values of error burst positions basing on conditions on **S** and **W**, until decoding ends or event "ambiguity" is determined (for all non-zero elements of **S**);

2) for the "ambiguity" decoding state select the position with maximal value of current vector $\mathbf{f}_b$ as the next erroneous position and return to step 1, in case of failure select the next maximal value of $\mathbf{f}_b$ and so on; if trial succeeded or all (non-zero) elements of $\mathbf{f}_b$ are considered, go to step 3;

3) if the burst $\mathbf{e}_b$ giving the all-zero syndrome is obtained during preceding steps, return codeword $\hat{\mathbf{a}}$, which is vector **y** added (modulo 2) by burst $\mathbf{e}_b$ om



■ ***Fig. 3.*** Uncertainty during decoding: *a* — resolvable by event "certain 0"; *b* — without resolving by certain events

window positions, otherwise decoding in current window is failed.

Described algorithm should be performed for all possible positions of sliding window, its pseudocode is given in Figures 4 and 5.

In fact, in the worst case for the given window position the algorithm performs exhaustive search by vectors $\mathbf{e}_b$, which guarantees finding the error burst giving the all-zero syndrome, however, extremely time-consuming. However, if described algorithm is applied to LDPC codes, especially for block-permutation constructions, and if the burst length does not exceeds burst correction capability of the code, in many cases decoding is ended by the first step (processing of certain events), either successfully or not, then all the positions of error burst may be determined by no more than b steps.

To fasten the decoding speed (by cost of failure probability increasing, naturally), the recursion depth or the number of considered elements in $\mathbf{f}_b$ may be limited.

Analysis of decoding algorithms and simulation results

The algorithms described in preceding sections were implemented in MatLab environment under MS Windows 10, the simulations where made using CPU Intel Core i7-9700@3,00 GHz, RAM 32 Gb. The following algorithms were considered:

— windowed bit-flipping algorithm (WBF);

— hybrid decoding with processing of certain events basin on $\mathbf{W}$ and $\mathbf{S}$ with switching to windowed bit-flipping algorithm in case of ambiguity (WS + WBF);

— recursive algorithm.

The following code constructions were selected for analysis:

— code 1: random (66.35) block-permutation LDPC code (1) with parameters $\gamma = 3, \rho = 6, m = 11$ (density of the parity-check matrix is $\chi \approx 0.09$), for this code $b_{\max} = 10$;

INPUT: $(r \times n)$-matrix $\mathbf{H}$, received word $\mathbf{y}$, window size b_w.
OUTPUT: decoded word $\hat{\mathbf{a}}$, status “success” or “failure” of the decoding.
1. Calculate $\mathbf{S} = \mathbf{y}\mathbf{H}^{\mathrm{T}}$; if $\mathbf{S} = \mathbf{0}$, return $\hat{\mathbf{a}} = \mathbf{y}$ and “success”.
2. For $i = 0$ до $n - b_w$ // cycle for sliding window positions:
2.1. Set $\mathbf{H}_b = \mathbf{H}(i, \ldots, i + b_w - 1)$, $\mathbf{e}_b = (0, \ldots, 0)$.
2.2. Calculate $\mathbf{W} = \mathbf{1}_b\mathbf{H}_b^{\mathrm{T}}$.
2.3. Call procedure [$\mathbf{e}_b$, status] = try_decode($\mathbf{H}_b$, $\mathbf{S}$, $\mathbf{W}$, $\mathbf{e}_b$), given in Fig. 5.
2.4. If status = “success”, add (modulo 2) to subvector $\mathbf{y}(i, \ldots, i + b_w - 1)$ the vector $\mathbf{e}_b$, return $\hat{\mathbf{a}} = \mathbf{y}$ and “success”.
3. Return $\hat{\mathbf{a}} = \mathbf{y}$ and “failure”.

■ ***Fig. 4.*** General scheme of single error burst decoding algorithm

PROCEDURE: [$\mathbf{e}_b$, status] = try_decode($\mathbf{H}_b$, $\mathbf{S}$, $\mathbf{W}$, $\mathbf{e}_b$).
INPUT: $(r \times b_w)$-matrix $\mathbf{H}_b$, syndrome $\mathbf{S}$, vector $\mathbf{W}$, current burst $\mathbf{e}_b$.
OUPUT: renewed $\mathbf{e}_b$, status “success” or “failure”.
1. If $\mathbf{S} = \mathbf{0}$, return [$\mathbf{e}_b$, “success”].
2. If $\exists\, i$: $\mathbf{W}_i = 0$ and $\mathbf{S}_i = 1$ (“impossible” event is determined), return [$\mathbf{e}_b$, “failure”].
3. For the first $i = 0, \ldots, r$, for which $\mathbf{W}_i = 1$ and $\mathbf{S}_i = 0$ (“certain 0” event is determined) or $\mathbf{S}_i = 1$ (“certain 1” event is determined):
3.1. Find j: the number of (the only) nonzero element in i-th row of $\mathbf{H}_b$.
3.2. Set $\mathbf{e}_b(j) = \mathbf{S}_i$.
3.3. If $\mathbf{S}_i = 1$, then renew the syndrome: sum (modulo 2) $\mathbf{S}$ and the column j of matrix $\mathbf{H}_b$.
3.4. Mark the column j of $\mathbf{H}_b$ as considered (technically it may be zeroed).
3.5. Renew the vector $\mathbf{W}$.
3.6. Call [$\mathbf{e}_b$, status] = try_decode($\mathbf{H}_b$, $\mathbf{S}$, $\mathbf{W}$, $\mathbf{e}_b$).
3.7. If status = “success”, return [$\mathbf{e}_b$, status].
3.8. If status = “failure”, set $\mathbf{e}_b(j) = 0$ and return [$\mathbf{e}_b$, status].
4. Calculate $\mathbf{f}_b = \mathbf{S} \otimes \mathbf{H}_b$ (in decimal numbers). Sort $\mathbf{f}_b$ together with the number of its positions by decrease, obtaining the vector $\mathbf{f}_{b,\mathrm{sort}}$ and corresponding array of indexes ind. Determine $f_{>0}$ — the number of non-zero elements in $\mathbf{f}_b$.
5. For $i = 1, \ldots, f_{>0}$:
5.1. Set $j = ind(i)$ (j — position of the next element in $\mathbf{f}_b$ by the value).
5.2. Set $\mathbf{e}_b(j) = 1$.
5.3. Calculate trial renewal of syndrome $\mathbf{S}'$ by adding the j-th column of $\mathbf{H}_b$ to $\mathbf{S}$.
5.4. Set the trial renewal of $\mathbf{H}'_b$, marking j-th column as considered.
5.5. Calculate trial renewal of $\mathbf{W}'$.
5.6. Call [$\mathbf{e}_b$, status] = try_decode($\mathbf{H}'_b$, $\mathbf{S}'$, $\mathbf{W}'$, $\mathbf{e}_b$).
5.7. If status = “success”, return [$\mathbf{e}_b$, “success”].
5.8. If status = “failure”, restore the value $\mathbf{e}_b(j) = 0$ (if trial versions $\mathbf{H}'_b$, $\mathbf{S}'$, $\mathbf{W}'$ where not in separate memory, their values also should be restored by the moment of entrance in the cycle).
6. If this step is reached, then the search through elements of $\mathbf{f}_b$ is completed without success, return [$\mathbf{e}_b$, “failure”].

■ ***Fig. 5.*** Recursive procedure of decoding trial

– code 2: random (66.33) code with density $\chi \approx 0.1$ and $b_{\max} = 12$;

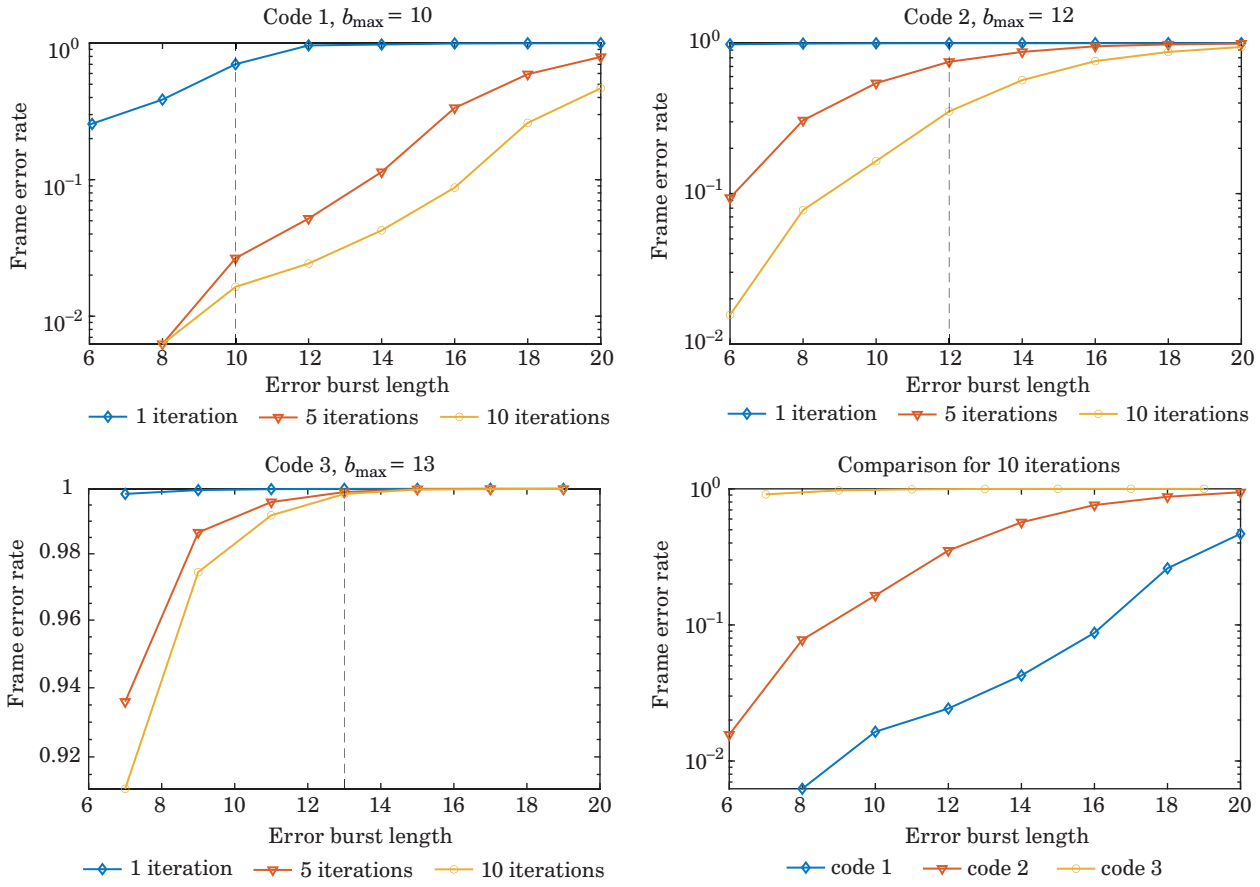
– code 3: random (66.33) code with density $\chi \approx 0.5$ and $b_{\max} = 13$.

Relatively small lengths of the codes were selected to fasten simulations in MatLab, since for the code 3 (which is defined by the dense parity-check matrix) simulations take significant time. At the same time, some experiments with longer codes gave qualitatively the same results, so in what follows we consider the codes with listed parameters.

For error vector generation the burst beginning was selected randomly and uniformly, the first and last positions of the burst are set to ones, erroneous bits within the burst were generated with probability 0.5, decoding fastening by means of estimation the burst position was not applied. Note that the simulation was also made for burst lengths exceeding the correction capabilities of the codes, so the decoding error event consists both of decoding failure and decoding error, that is finding false codeword.

In Fig. 6 the results of evaluating the codeword (frame) error rate is given for windowed bit-flipping algorithm with 1, 5, 10 iterations, for the single burst lengths from 6 to 20. As can be seen from the curves, windowed bit-flipping algorithm did not realize the burst-correction capability in any case, and since the algorithm is based on the sparsity of the parity-check matrix, for dense matrix (code 3) the error probability rapidly tends to one. For the matrix (code 2) with density close to those of block-permutation construction (code 1) the error probability is remarkably higher comparing to code 1. This may be connected to the fact that random matrix of code 2 was not optimized in its structure for using traditional iterative decoders for LDPC codes. During decoding, all error events were caused by decoding failures, no any false codewords were found.

In Fig. 7 the results for recursive decoding is given. The average time of one codeword decoding (in seconds) were also estimated. Since no special optimizations of decoders implementations were not performed, the goal of time estimation is to approximately evaluate the comparative complexity in different cases. For code 1 and code 2 the decoding algorithm was fully completed. During simulations of code 1 and code 2 there were no any cases of ambiguity shown in Fig. 3. Besides, decoding did not lead to any failure events, so all decoding errors were caused by false codewords, and in many cases the decoding for burst lengths exceeding the correction capability was successful. As can be seen from

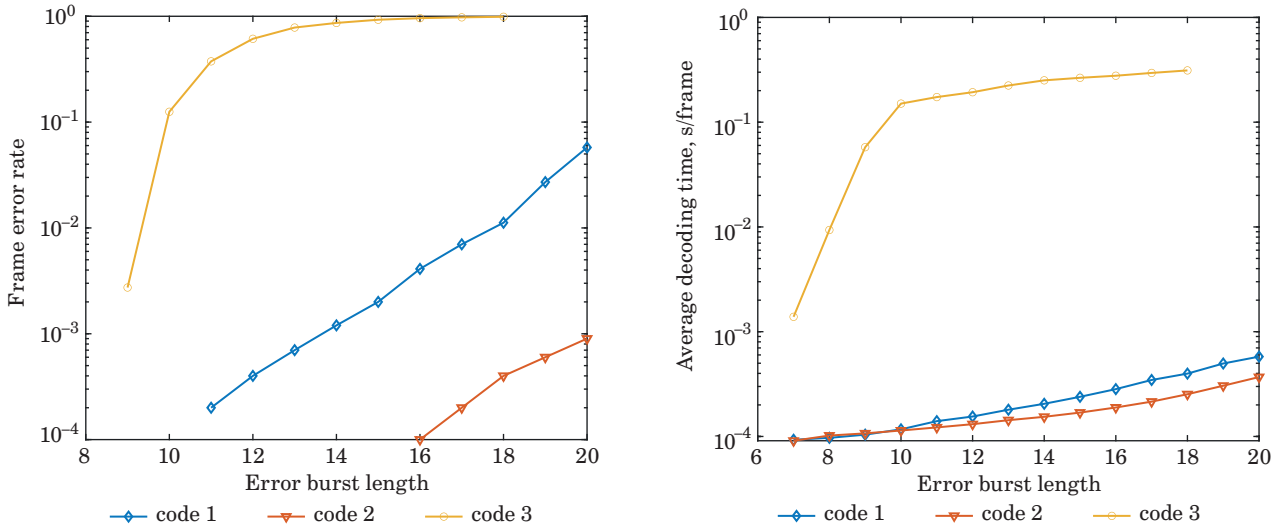


■ ***Fig. 6.*** Simulation results for windowed bit-flipping algorithm

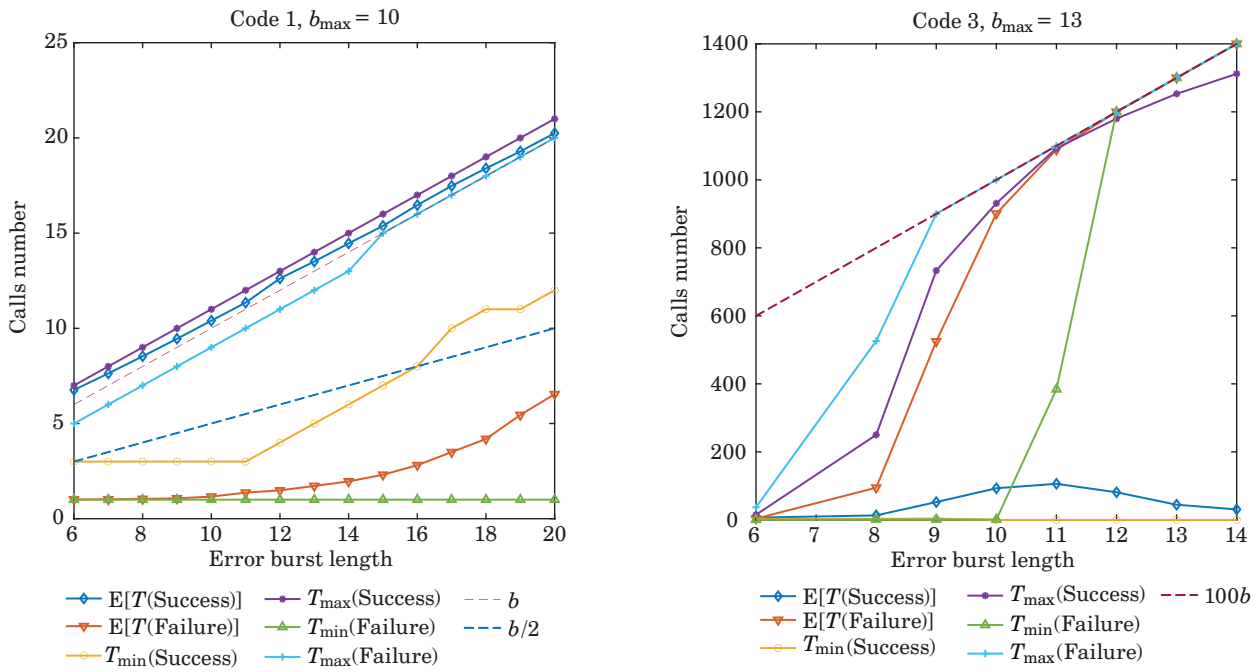
Fig. 7, code 2 provides better error probability comparing to code 1, with comparable but slightly less decoding complexity.

As for code 3, which is defined by dense matrix, the simulation shows fundamentally different results. The decoding of one codeword might be up to 1000 times slower than for code 1 and code 2, in fact in this case the decoder with high probability tends to brute force of error bursts. Thus for the code 3 in Fig. 7 the artificial limitation of the number of calls of `try _ decode` procedure was set, equal to $100b_w$. In this case decoder stops to find the false codewords and started to fail. From the window size of about 10, where the time curve of code 3 decoding changes its fast growth by gentle increasing (see Fig. 7), the decoding failure in almost all cases is connected to reaching the maximal recursion depth for all window positions, and not with determining "impossible" event. Even in case of successful decoding, the recursion depth was close to maximal.

In Fig. 8 for the code 1 and code 3 are given: $T_{\min}$(Success), $T_{\max}$(Success), E(T(Success)) — minimal, maximal and average number of calls of `try _ decode` procedure (see Fig. 5) for successful

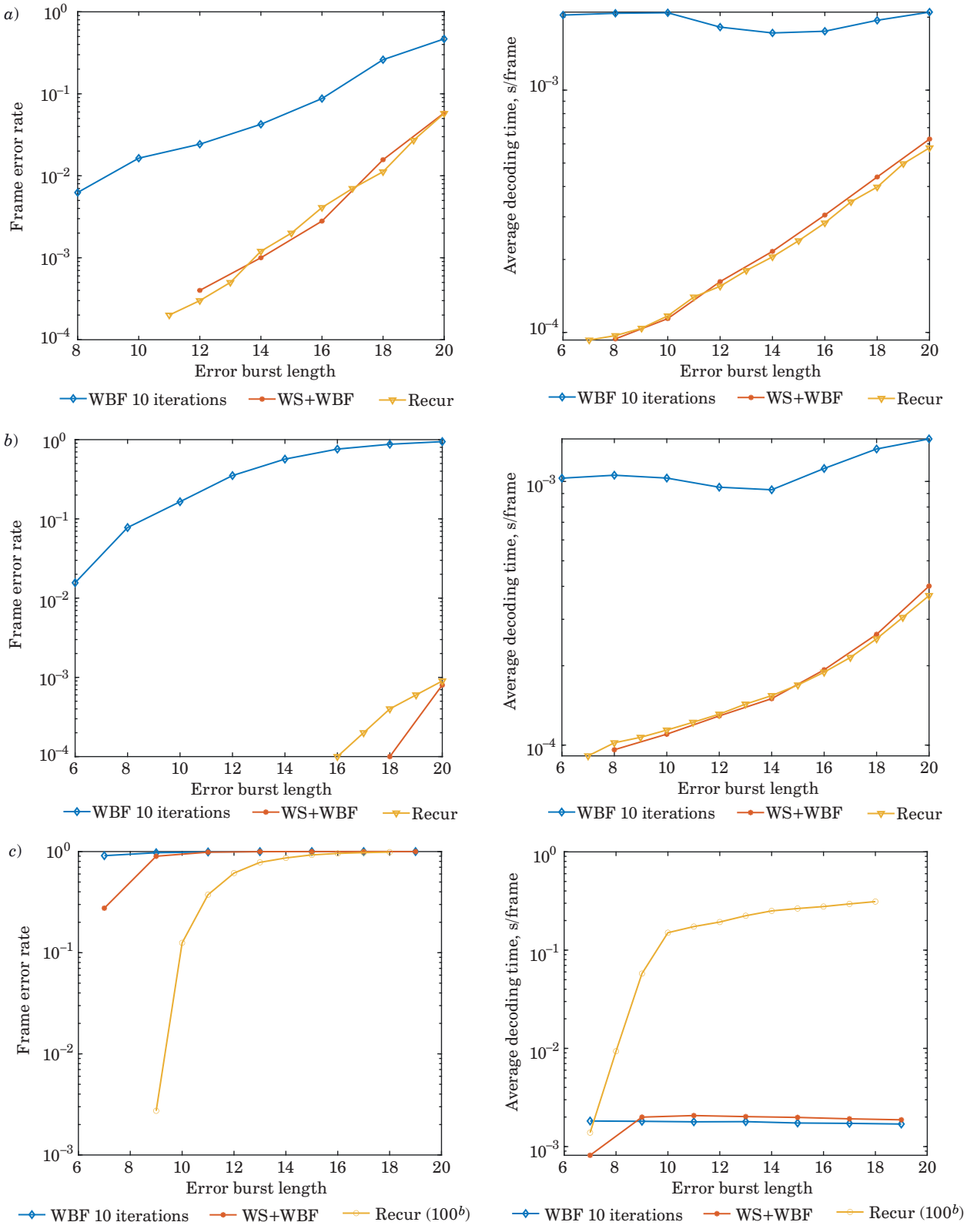


■ ***Fig. 7.*** Simulation results for recursive algorithm of single burst correction



■ ***Fig. 8.*** The number of calls of try_decode procedure

decoding in the window, $T_{\min}$(Failure), $T_{\max}$(Failure), E(T(Failure)) — similar values for failed decoding in the window. As one can see, for the code 1 the maximal number of calls of `try _ decode` for given window position is approximately equal to window size b_w both for correct and not correct window position.



■ ***Fig. 9.*** Comparison of different decoders for code 1 (*a*), code 2 (*b*) and code 3 (*c*)

Minimal number of calls may be approximated as $b_w/2$ for correct window position and 1 otherwise (in this case the event "impossible" is determined by the first call). For the code 3 the maximal recursion depth (and average for failed decoding) rapidly increasing and approaches upper limit of 100 window sizes even for burst lengths below correction capability.

In Fig. 9, *a* the evaluation of error probability and decoding time for windowed bit-flipping, hybrid and recursive algorithms for code 1 are presented. As can be seen, algorithms using analysis of certain events show approximately the same error probability (an order less than for bit-flipping algorithm) and comparable decoding time, the error events (caused only by false codewords) appears only for burst lengths exceeding the capability of the code. Recall that for these algorithms no uncertainty during decoding were reached, so the decoding completed only by determination of certain events.

In Fig. 9, *b*, *c* the similar results for code 2 and code 3 are given. For code 2 the gain obtained by decoders analyzing the certain events comparing to bit-flipping algorithm is even larger than for code 1, this may be reasoned both by worse performance of bit-flipping algorithm for code 2, and by the fact that code 2 has slightly larger burst-correction capability than code 1, and perhaps — better weight spectrum for correcting single bursts, so the decoding far beyond correction capability for this code is possible. For the code 3 given by dense matrix, the recursive algorithm (with recursion depth limited by 100 window sizes) shows significant gain for bursts length up to 10 (recall that code 3 has $b_{\max} = 13$), and two other decoders tends to error probability equal to 1. This gain is achieved by cost of significant decoding time, in all cases, when error probability close to 1, decoding time stops to change with burst length increasing, since in practically all cases decoding for all window positions is failed.

Conclusion

In this paper the decoding of single error bursts with linear codes is considered. The analysis of windowed decoding is hold, basing on determination of certain events using structure of the parity-check matrix. The two-stage hybrid decoder is considered, combining the consideration of certain events with bit-flipping algorithm. The recursive decoding algorithm is proposed, which guarantees correction of single error bursts with correction capability, the search in this decoder is optimized by selecting the least reliable position, determined by the column with most coincides of non-zero elements with syndrome. The evaluation of error probability and average decoding time per transmitted codeword is performed by computer simulation. The simulation results show that proposed algorithms allow to correct large number of single error bursts with lengths exceeding the burst-correction capability. Proposed recursive algorithm may be applied to any linear code, however taking into account, that the decoding complexity is reasonable (and significantly less than for bit-flipping algorithm) only for codes which parity-check matrix is sparse enough.

Financial support

The paper was prepared with the financial support of the Russian Science Foundation, project No. 22-19-00305 "Spatial-temporal stochastic models of wireless networks with a large number of users".

References

1. Moon T. K. *Error correction coding: Mathematical methods and algorithms*. Wiley, 2020. 992 p.
2. Lin Z. *Design of network coding schemes in wireless networks*. Boca Raton, FL, CRC Press, 2022. 166 p.
3. Gazi O. *Forward error correction via channel coding*. Cham, Springer, 2020. 319 p.
4. Ghaddar N., Kim Y.-H., Milstein L. B., Ma L., and Yi B. K. Joint channel estimation and coding over channels with memory using polar codes. *IEEE Transactions on Communications*, Oct. 2021, vol. 69, no. 10, pp. 6575–6589. doi:10.1109/TCOMM.2021.3098822
5. Sasoglu E., Tal I. Polar coding for processes with memory. *IEEE Transactions on Information Theory*, 2019, vol. 65, no. 4, pp. 1994–2003. doi:10.1109/TIT.2018.2885797
6. Shao S., Hailes P., Wang T.-Y., Wu J.-Y., Maunder R. G., Al-Hashimi B. M., Hanzo L. Survey of Turbo, LDPC, and polar decoder ASIC implementations. *IEEE Communications Surveys & Tutorials*, 2019, vol. 21, no. 3, pp. 2309–2333. doi:10.1109/COMST.2019.2893851
7. Stevens A. *Monte-Carlo simulation: an introduction for engineers and scientists*. First ed. Boca Raton, FL, CRC Press, 2022. 112 p.
8. Jihwan S., Lee H.-K. Burst error correction for convolutional code concatenated with Hamming code with a block interleaver. *2020 Intern. Conf. on Artificial Intelligence in Information and Communication (ICAIIC)*, Fukuoka, Japan, 2020, pp. 531–533. doi:10.1109/ICAIIC48513.2020.9065198
9. Loyka S., Charalambous C. D. On the capacity of Gaussian MIMO channels with memory. *IEEE Communications Letters*, 2022, vol. 26, no. 8, pp. 1760–1763. doi:10.1109/LCOMM.2022.3174774
10. Veresova A. M., Ovchinnikov A. A. Comparison of the probability of Reed — Solomon and LDPC codes decoding error in the Gilbert — Elliott channel. *2022*

Wave Electronics and its Application in Information and Telecommunication Systems (WECONF), Saint-Petersburg, Russia, 2022, pp. 1–4. doi:10.1109/WECONF55058.2022.9803501

11. Kulhandjian M., Kulhandjian H., D'Amours C. Improved soft decoding of Reed — Solomon codes on Gilbert — Elliott channels. *2019 IEEE Intern. Symp. on Information Theory (ISIT)*, Paris, France, 2019, pp. 1072–1076. doi:10.1109/ISIT.2019.8849456

12. Song L., Huang Q., Wang Z. Construction of multiple-burst-correction codes in transform domain and its relation to LDPC codes. *IEEE Transactions on Communications*, 2020, vol. 68, no. 1, pp. 40–54. doi:10.1109/TCOMM.2019.2948341

13. Xiao X., Vasić B., Lin S., Li J., and Abdel-Ghaffar K. Quasi-cyclic LDPC codes with parity-check matrices of column weight two or more for correcting phased bursts of erasures. *IEEE Transactions on Communications*, May 2021, vol. 69, no. 5, pp. 2812–2823. doi:10.1109/TCOMM.2021.3059001

14. Veresova A. M., Ovchinnikov A. A. About one algorithm for correcting bursts using block-permutation LDPC-codes. *2019 Wave Electronics and its Application in Information and Telecommunication Systems (WECONF)*, Saint-Petersburg, Russia, 2019, pp. 1–4. doi:10.1109/WECONF.2019.8840580

15. Gallager R. G. *Low density parity check codes*. Cambridge, MA, MIT Press, 1963. 90 p.

16. Zhu K., Wu Z., Comprehensive study on CC-LDPC, BC-LDPC and Polar code. *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2020, pp. 1–6. doi:10.1109/WCNCW48565.2020.9124897

17. Jeong S., Ha J. MET-LDPC code ensembles of low code rates with exponentially few small weight codewords. *IEEE Transactions on Communications*, 2021, vol. 69, no. 6, pp. 3517–3527. doi:10.1109/TCOMM.2021.3063348

18. Ovchinnikov A. A., Fominykh A. A. About some irregular degree distributions of LDPC codes in two-state channels. *2021 Wave Electronics and its Application in Information and Telecommunication Systems (WECONF)*, Saint-Petersburg, Russia, 2021, pp. 1–4. doi:10.1109/WECONF51603.2021.9470627

19. Xiao X., Vasic B., Lin S., Abdel-Ghaffar K., Ryan W. E. Reed — Solomon based quasi-cyclic LDPC codes: designs, girth, cycle structure, and reduction of short cycles. *IEEE Transactions on Communications*, 2019, vol. 67, no. 8, pp. 5275–5286. doi:10.1109/TCOMM.2019.2916605

20. Li J., Gong Y., Lin S., Abdel-Ghaffar K. Balanced incomplete block designs, partial geometries, and their associated QC-LDPC codes. *2021 11th Intern. Symp. on Topics in Coding (ISTC)*, Montreal, QC, Canada, 2021, pp. 1–5. doi:10.1109/ISTC49272.2021.9594122

21. Eckford A. W., Kschischang F. R., Pasupathy S. Analysis of low-density parity-check codes for the Gilbert — Elliott channel. *IEEE Transactions on Information Theory*, 2005, vol. 51, no. 11, pp. 3872–3889. doi:10.1109/TIT.2005.856934

22. Ovchinnikov A., Fominykh A. About burst decoding for block-permutation LDPC codes. *Internet of Things, Smart Spaces, and Next Generation Networks and Systems: 20-th Intern. Conf., NEW2AN 2020, and 13-th Conf., ruSMART 2020*, Saint-Petersburg, Russia, 2020, pp. 393–401. doi:10.1007/978-3-030-65726-0_35

УДК 519.72
doi:10.31799/1684-8853-2022-6-41-52
EDN: UWXZHN

Декодирование линейных кодов при исправлении одиночных пакетов ошибок на основе определения достоверных событий

А. А. Овчинников[a], канд. техн. наук, доцент, orcid.org/0000-0002-8523-9429, mldoc@guap.ru
А. М. Вересова[a], аспирант, orcid.org/0000-0002-3792-9249
А. А. Фоминых[a], магистр, orcid.org/0000-0002-1412-5766
[a]Санкт-Петербургский государственный университет аэрокосмического приборостроения, Б. Морская ул., 67, Санкт-Петербург, 190000, РФ

Введение: в современных системах связи, хранения и обработки данных помехоустойчивость различных кодов, исправляющих ошибки, оценивается для каналов без памяти. В реальных каналах связи шум представляет собой коррелированный случайный процесс, что приводит к группированию ошибочных бит в пакеты. Классический подход для борьбы с пакетированием ошибок состоит в применении процедуры декорреляции канала, что не позволяет строить кодовые схемы, наиболее полно реализующие пропускную способность канала. Таким образом, актуальной является задача построения алгоритмов декодирования для исправления пакетов ошибок для произвольных линейных кодов. **Цель:** разработать алгоритм декодирования одиночных пакетов ошибок для линейных кодов, оценить вероятность ошибки и вычислительную сложность разработанного алгоритма. **Результаты:** предложены два подхода к исправлению пакетов ошибок. Первый основан на комбинировании оконной модификации известного алгоритма инвертирования бит с предварительным анализом структуры проверочной матрицы. Второй основан на рекурсивной процедуре построения последовательности достоверных событий, в наихудшем случае осуществляющей полный перебор пакетов ошибок, который во многих случаях может быть значительно сокращен с помощью предложенной эвристики. Предложенный рекурсивный алгоритм декодирования позволяет гарантированно исправлять любые одиночные пакеты ошибок в пределах корректирующей способности кода, а с высокой вероятностью и сверх корректирующей способности. Сложность этого алгоритма значительно ниже сложности алгоритма

инвертирования бит, если проверочная матрица линейного кода является достаточно разреженной. Альтернативный гибридный алгоритм декодирования с использованием инвертирования бит для низкоплотностных кодов показывает вероятность ошибки и время выполнения, сравнимые с рекурсивным алгоритмом, однако возможность гарантированного исправления пакетов ошибок с его помощью вряд ли может быть доказана. **Практическая значимость:** предложенные методы декодирования могут быть использованы в современных и перспективных системах связи, позволяя экономить энергию и повышать надежность передачи данных за счет лучшей эффективности исправления ошибок и меньшей вычислительной сложности.

Ключевые слова — каналы с памятью, коды с малой плотностью проверок на четность, исправление пакетов ошибок.

Для цитирования: Ovchinnikov A. A., Veresova A. M., Fominykh A. A. Decoding of linear codes for single error bursts correction based on the determination of certain events. *Информационно-управляющие системы*, 2022, № 6, с. 41–52. doi:10.31799/1684-8853-2022-6-41-52, EDN: UWXZHN

For citation: Ovchinnikov A. A., Veresova A. M., Fominykh A. A. Decoding of linear codes for single error bursts correction based on the determination of certain events. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2022, no. 6, pp. 41–52. doi:10.31799/1684-8853-2022-6-41-52, EDN: UWXZHN

УДК 519.21: 537.86
ББК 22.17

Хименко В. И.
Выбросы случайных процессов и проблема пересечений уровней
Москва: ТЕХНОСФЕРА, 2022. – 582 с. ISBN 978-5-94836-658-6

Характеристики выбросов, пересечения заданных уровней, экстремальные значения случайных процессов – это класс характеристик, позволяющих описывать структуру и вероятностное поведение случайных функций. По своему содержанию такие характеристики относятся к направлению междисциплинарных исследований. Необходимость их изучения связана с решением многочисленных задач из различных областей физики, техники и естествознания.

Содержание данной работы отражает современное состояние исследований в области прикладной теории выбросов и общей проблемы «пересечений уровней». Здесь делается попытка систематизации, обобщения и развития основных результатов, попытка рассмотрения проблематики превышений заданных уровней «в целом» для наиболее распространенных классов случайных функций. Представлено большое количество новых результатов. Это относится к анализу вероятностной структуры временных рядов, непрерывных случайных процессов, случайных потоков событий и случайных пространственно-временных полей. Показаны возможности общей классификации прикладных задач и особенности их решения на основе использования характеристик пересечений уровней.

Для широкого круга специалистов, аспирантов и студентов, для тех, кто изучает, исследует и применяет на практике модели и методы анализа различных по своей физической природе случайных данных.