Insertion of erasures as a method of q-ry LDPC codes decoding

Alexey Frolov, Victor Zyablov {alexey.frolov, zyablov}@iitp.ru
IITP RAS

Abstract. An iterative decoding algorithm based on adding and correcting of erasures is suggested. Correcting capabilities of the suggested algorithm are researched. The dependency of realized correcting capabilities of the algorithm on initial number of erasures is introduced. The comparison of error-correcting capabilities of our algorithm and a majority algorithm is presented.

1 Introduction

A method for constructing of long codes from short constituent codes, based on bipartite graphs, was introduced by Tanner in [1]. In his method, one of the two sets of nodes in a bipartite graph is associated with code symbols, while the other set is associated with constituent block codes whose length is equal to the node degree. These two sets of nodes are hereafter referred to as variable nodes and constraint nodes, respectively. An example of Tanner graph is shown in Fig. 1.

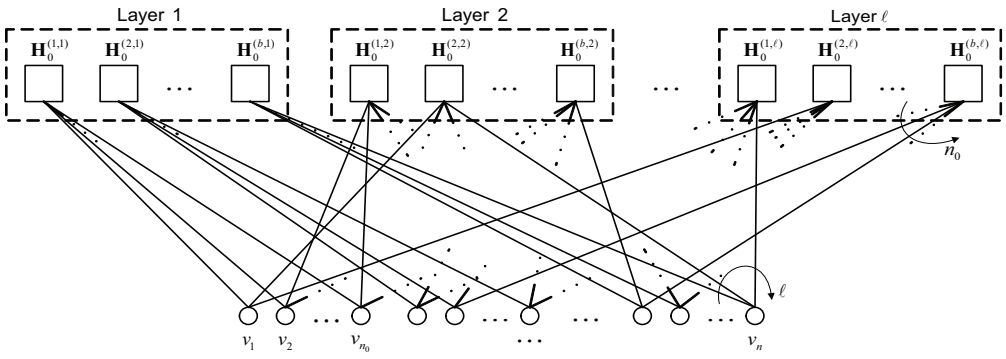

Figure 1: Tanner graph

Tanner's general code construction unifies many known code families that can be obtained by choosing different underlying bipartite graphs and associating different constituent codes with their constraint nodes.

The single parity check codes associated with the constraint nodes in the Tanner graph of an LDPC code can be replaced with other constituent block

codes (e.g., BCH codes, or Reed-Solomon codes [2]), which yields alternative constructions of LDPC codes, often referred to as generalized LDPC codes. The parity-check matrix of such an LDPC code is obtained by replacing every 1 in the graph's adjacency matrix with a column of the constituent code's parity-check matrix, and every 0 with the all-zero column.

2 Code structure

We will use a single parity-check code over $GF(q)$ as a constituent code. A parity-check matrix of the code consists of n_0 non-zero elements of $GF(q)$. The code has a minimum distance $d = 2$ and is able to correct at most 1 erasure.

Let $\mathbf{H}_b$ denote a block-diagonal matrix with b constituent parity-check matrices $\mathbf{H}_0$ on the main diagonal, that is,

$$\mathbf{H}_b = \begin{pmatrix} \mathbf{H}_0 & 0 & \cdots & 0 \\ 0 & \mathbf{H}_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{H}_0 \end{pmatrix}, \tag{1}$$

where b is very large. The matrix $\mathbf{H}_b$ is of size $b \times bn_0$. Let $\pi(\mathbf{H}_b)$ denote a random column permutation of $\mathbf{H}_b$. Then the matrix

$$\mathbf{H} = \begin{pmatrix} \pi_1(\mathbf{H}_b) \\ \pi_2(\mathbf{H}_b) \\ \vdots \\ \pi_\ell(\mathbf{H}_b) \end{pmatrix}, \tag{2}$$

constructed using $\ell \geqslant 2$ such permutations as layers, is a sparse $\ell b \times bn_0$ parity-check matrix of a q-ry LDPC code.

3 Decoding algorithm

The main difference of suggested algorithm from existing decoding algorithms for LDPC codes is in replacing of error-suspicious symbols with erasures. In each iteration error-suspicious symbols are replaced with erasures and then only the erasure correcting is performed within the iteration. The erasures which were added and were not corrected after the iteration was finished are removed. These two operations are repeated until the syndrome is changing.

Let us introduce the formal description of an algorithm:

1. **Syndrome calculation** The syndrome of the received vector is calculated. It consists of syndromes of constituent-codes. If a constituent code

contains erased symbols then its syndrome is not calculated and considered to be erased.

2. **Error-suspicious symbols erasing** For each non-erased symbol the syndromes of ℓ constituent codes containing the symbol are considered. If the constituent code syndrome is neither null nor erased we can calculate the decision. By a *decision* we mean a value that should be added to the observed symbol to zero the constituent code syndrome. Null and erased decisions correspond to constituent codes with null and erased syndromes accordingly. The subset of maximal cardinality (a) containing equal neither zero nor erased decisions is chosen. Let c denote the number of zero decisions and let e denote the number of erased decisions. If $a > c + e$ than the symbol is replaced with erasure. The syndromes of constituent codes containing the symbol are erased. The position of the symbol is placed to the list of added erasures.

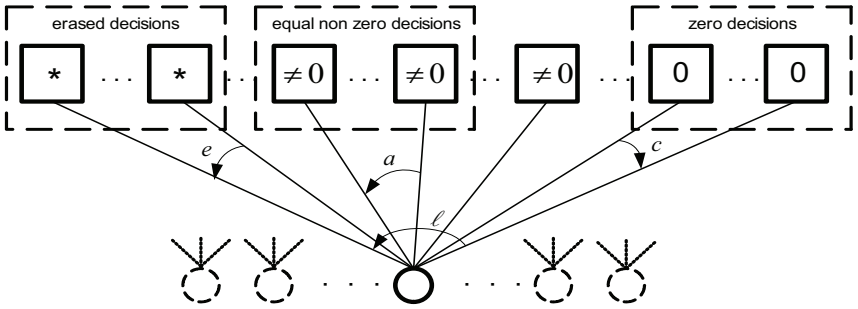

Figure 2: Error-suspicious symbols erasing

3. **Correcting of erasures** For each symbol from the list of erased symbols the subset of constituent codes containing the symbol is considered. Only the codes containing one erasure belong to the subset. For each code from this subset we can correct the erasure and form a list of possible symbol values. Then the most often value is found and the erased symbol is replaced with this value.

4. **Stop criterion** Added erasures are removed. The syndromes before and after iteration are compared. Return to Step 2 in case of not equal syndromes. In case of equal syndromes the syndrome weight is calculated. If the weight is equal to zero then the decoded vector is returned. Return denial of decoding if the weight is not equal to zero.

4 Numerical results

We used a code with such parameters while modeling process: $q = 16$; $n = 2048$; $R = \frac{1}{2}$; $n_0 = 16$; $\ell = 8$. Modeling process starts from 300 errors. This value decreases by 5 errors after 10 denials are got. The result of a modeling process is a dependency of denial probability on number of errors (number of erasures is fixed). Over than 10^6 tests are carried out for each dependency.

First of all we want to get the dependency of correcting capabilities of the suggested algorithm on the initial number of erasures. A family of dependencies is calculated with different initial number of erasures (0, 5, 10, 30, 50, 70, 90). The family is shown in Fig. 3.

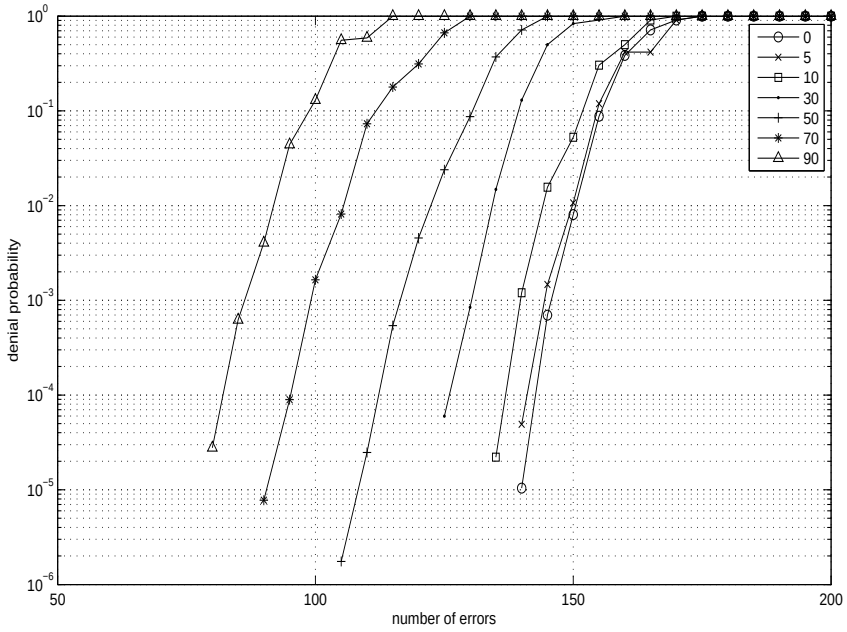

Figure 3: Family of dependencies

Let τ denote the number of erasures, e^* denote the number of errors in case of the denial probability is less than 10^{-4} (the greatest number of errors which meets the condition is chosen). We will use a value $d^* = 2e^* + \tau + 1$ to describe the realized correcting capabilities of the suggested algorithm. The dependency of realized correcting capabilities of the suggested algorithm on initial number of erasures is introduced in Table 1.

Table 1: The dependency of realized correcting capabilities on initial number of erasures

τ	0	5	10	30	50	70	90
e^*	142	140	136	126	110	94	81
d^*	285	286	283	283	271	259	253

As we see the realized correcting capabilities decrease while an initial number of erasures increases.

Now we would like to compare error-correcting capabilities of our algorithm and known decoding algorithms. So there are no erasures in the received vector and only error-correction is performed. The comparison of error-correcting capabilities of our algorithm and a majority algorithm (the description is given in [3]) is shown in Fig. 4.

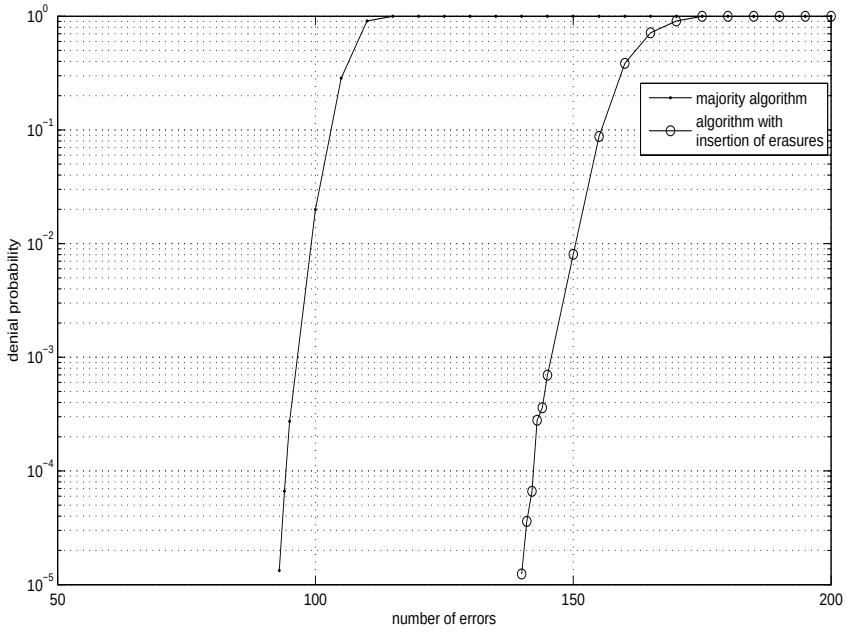

Figure 4: The comparison with a majority algorithm

As we see the results for our algorithm are better than results for majority algorithm.

5 Conclusion

An iterative decoding algorithm based on adding and correcting of erasures is suggested. The algorithm is capable to correct both errors and erasures. The dependency of realized correcting capabilities of the algorithm on initial number of erasures is introduced. Error-correcting capabilities of the algorithm are better than error-correcting capabilities of a majority algorithm.

References

[1] M. Tanner, "A recursive approach to low complexity codes", *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[2] N. Miladinovic, M. Fossorier, "Generalized LDPC Codes with Reed-Solomon and BCH Codes as Component Codes for Binary Channels", *Proc. IEEE Global Conf. on Communication (GLOBECOM)*, St. Louis, USA, November. 2005.

[3] V. Zyablov, A. Frolov, "The comparison of error-correcting capabilities of LDPC codes with constituent codes of different redundancy", *Information technologies and systems (ITaS'09): conference proceeding collect*, Moscow, IITP RAS, pp. 160 - 163, 2009 (in Russian).