A new idea for graph code decoding

V. B. Afanassiev, afanv@iitp.ru
S. A. Popov spopov@iitp.ru
Institute for Information Transmission Problems of Russian Academy of Sciences
(Kharkevich Institute), Bol'shoi Karetnyi per. 19, GSP-4, Moscow, 127994, Russia

Introduction

Let (n, k, d) be the parameters of a component code. Giving a bipartite graph code we will consider an equivalent representation of their codewords as matrix **M** with the following properties:

- $m \geq n$.
- Each row (column) of **M** contains *n* open cells and *n* closed cells;
- Total number of open cells is equal to the length $N = mn$ of a graph code.
- Open cells are numbered in any order from 1 to *N* and *i*-th open cell is filled by *i*-th symbol of a codeword.
- Symbols written in open cells of any row (column) have to belong to a codeword of a component code.
- A codeword in **M** consist of *2m* component *codewords* in *m* rows and *m* columns.
- Each code symbol belongs to two component codewords (row and column).

Reference decoding algorithm

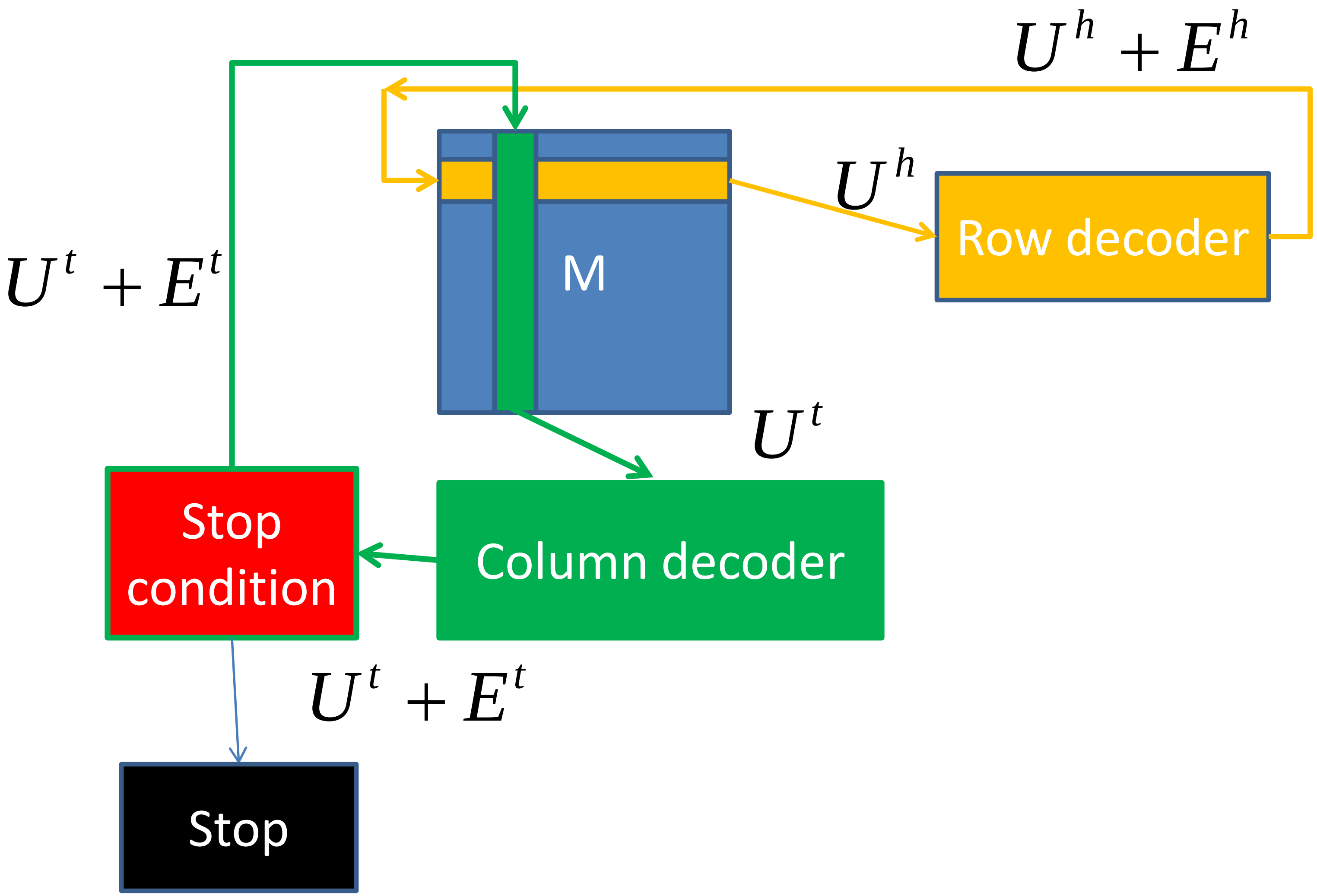
Giving a received word with errors and erasures:

1. *Decode all rows by decoding algorithm of a component code and apply the corrections to the received word.*
2. *Decode all columns by decoding algorithm of a component code and apply the corrections to the received word.*
3. *Check the stop condition and continue steps 1 – 3 if not.*

In [1, 3] was shown that the reference algorithm works well when component code corrects 5 or more errors and erasures.

For component Reed-Solomon code the conditional probability of decoding errors when more than T errors occur is approximately $1/T!$. This value is insignificant when the minimal distance of component code is >=10.

Reference decoding algorithm



The goal

- a new iterative decoding algorithm of a graph code when the component codes has a small distance (or low error-correcting capability) $T<=2$.

The main idea

M is filled by a received word with errors and erasures.

Decode all rows and columns of **M** by a component code decoder and save all corrections in array **E** .

Notations:

- $U^h = \left(U_1^h, U_2^h, \ldots, U_n^h\right), h = 1, 2, \ldots, 2m,$ is a row (col.) of **M**.
- $U^{j(h,i)} = \left(U_1^{j(h,i)}, U_2^{j(h,i)}, \ldots, U_n^{j(h,i)}\right),$ column (or row) of **M**

$$j(h,i) = 1, 2, \ldots, 2m, i = 1, 2, \ldots, n, \; j(h,i) \neq h,$$

is a component *block* that intersects U^h in *i*-th position.

- $U_i^h = U_{l(h,i)}^{j(h,i)} = u \in \{GF(q) \cup \varnothing\}$ is one of ***N*** received symbols.

The main idea

$$E^h = \left(e_1^h, e_2^h, \ldots, e_n^h\right),$$

$$\alpha_i^h = e_i^h \rightarrow u \leftarrow e_{l(h,i)}^{j(h,i)} = \beta_i^h$$

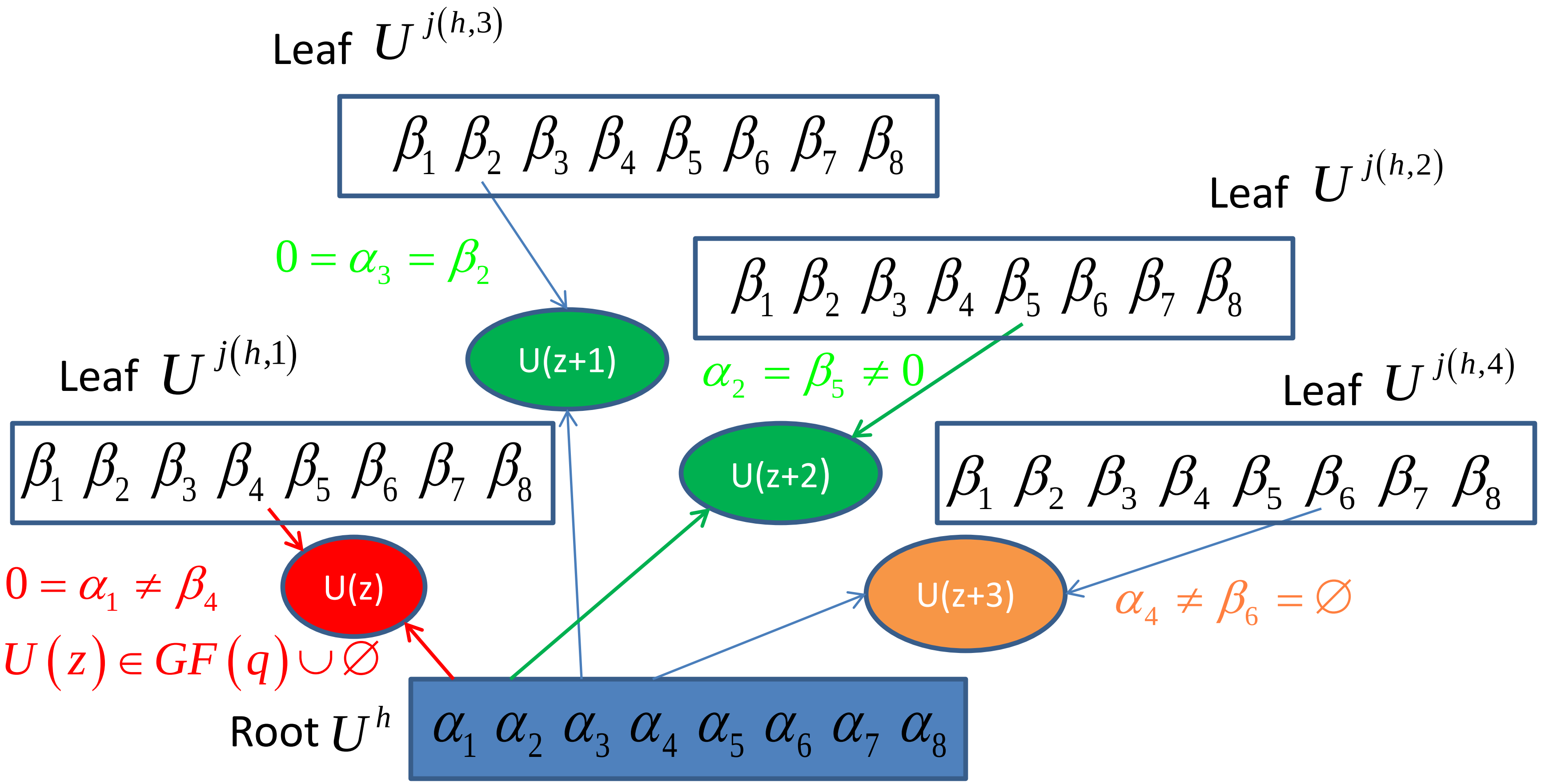
$\alpha = \beta$ - *coincidence* of corrections from both sides, or

$\alpha \neq \beta$ - *conflict* (or different corrections from both sides).

Facts:

- *if a conflict takes place then one of two decoding results of component blocks is wrong,*
- *the more is number of conflicts the less probable is correctness of the decoding result* E^i.

A part of code tree. $RS(8,6,3)$



The main idea (algorithm)

- Calculate for every root E^h the ratio

$$\Lambda^h = \frac{\Pr\left(G \mid E_{root}, E_{leaf}, U_{root}\right)}{\Pr\left(B \mid E_{root}, E_{leaf}, U_{root}\right)} = \frac{\Pr\left(E_{leaf} \mid G, E_{root}, U_{root}\right)}{\Pr\left(E_{leaf} \mid B, E_{root}, U_{root}\right)} \frac{\Pr\left(G \mid E_{root}, U_{root}\right)}{\Pr\left(B \mid E_{root}, U_{root}\right)}, \; h = 1, \ldots, 2m,$$

G - *correct decoding* of a root (was transmitted) and B, on the contrary, be a *wrong decoding* (was not transmitted).

- Find all the roots such that $\Lambda^h > \Lambda^{j(h,i)}$ over the root conflict positions $l(h,i), i = 1, \ldots, n, h = 1, \ldots, 2m$, and apply the corrections to the received (long) word in **M**.
- Do nothing with all other roots.

Proposition 1

$$\frac{\Pr\left(G \mid E_{root}, U_{root}\right)}{\Pr\left(B \mid E_{root}, U_{root}\right)} = \frac{\Pr\left(G \mid t_{root}\right)}{\Pr\left(B \mid t_{root}\right)}$$

Proposition 2

$$\frac{\Pr\left(E_{leaf} \mid G, E_{root}, U_{root}\right)}{\Pr\left(E_{leaf} \mid B, E_{root}, U_{root}\right)} = \frac{\Pr\left(\beta_{leaf} \mid G, \alpha_{root}, t_{leaf}, U_{root}^{\varnothing}\right)}{\Pr\left(\beta_{leaf} \mid B, \alpha_{root}, t_{leaf}, U_{root}^{\varnothing}\right)}$$

Proposition 3

$$\frac{\Pr\left(\beta_{leaf} \mid G, \alpha_{root}, t_{leaf}, U_{root}^{\varnothing}\right)}{\Pr\left(\beta_{leaf} \mid B, \alpha_{root}, t_{leaf}, U_{root}^{\varnothing}\right)} = \frac{\Pr\left(\left(\beta_1^h, \beta_2^h, \ldots, \beta_n^h\right) \mid G, \left(\alpha_1^h, \alpha_2^h, \ldots, \alpha_n^h\right), \left(t_1^h, t_2^h, \ldots, t_n^h\right), U_{root}^{\varnothing}\right)}{\Pr\left(\left(\beta_1^h, \beta_2^h, \ldots, \beta_n^h\right) \mid B, \left(\alpha_1^h, \alpha_2^h, \ldots, \alpha_n^h\right), \left(t_1^h, t_2^h, \ldots, t_n^h\right), U_{root}^{\varnothing}\right)} =$$

$$= \prod_{i=1}^{n} \frac{\Pr\left(\beta_i^h \mid G, \alpha_i^h, t_i^h, U_{root}^{\varnothing}\right)}{\Pr\left(\beta_i^h \mid B, \alpha_i^h, t_i^h, U_{root}^{\varnothing}\right)}$$

Proposition 4

$$\Lambda = \frac{\Pr\left(\beta \mid G, \alpha, t', U_{root}^{\varnothing}\right)}{\Pr\left(\beta \mid B, \alpha, t', U_{root}^{\varnothing}\right)} = \begin{cases} \dfrac{1}{\left[\Pr\left(G' \mid t'\right) / \Pr\left(B' \mid t'\right)\right] + 1}, \alpha \neq \beta, \\ \dfrac{\Pr\left(G' \mid t'\right) / \Pr\left(B' \mid t'\right) + \left(1 - t'/n\right)}{\left[\Pr\left(G' \mid t'\right) / \Pr\left(B' \mid t'\right) + 1\right]\left(1 - t'/n\right)}, \alpha = \beta = 0 \,\&\, u \neq \varnothing', \\ \dfrac{\Pr\left(G' \mid t'\right) / \Pr\left(B' \mid t'\right) + \left(t'/nq\right)}{\left[\Pr\left(G' \mid t'\right) / \Pr\left(B' \mid t'\right) + 1\right]\left(t'/nq\right)}, \alpha = \beta \neq 0 \,\&\, u \neq \varnothing', \\ \dfrac{\Pr\left(G' \mid t'\right) / \Pr\left(B' \mid t'\right) + \left(1/q\right)}{\left[\Pr\left(G' \mid t'\right) / \Pr\left(B' \mid t'\right) + 1\right]\left(1/q\right)}, \alpha = \beta \,\&\, u = \varnothing' \end{cases}$$

Proposition 5

$$\Lambda = \frac{\Pr\left(\beta \mid G, \alpha, \varnothing\right)}{\Pr\left(\beta \mid B, \alpha, \varnothing\right)} = \begin{cases} 1 - p^*, \alpha = \beta = 0, \\ p^*, \alpha \neq \beta = 0. \end{cases}$$

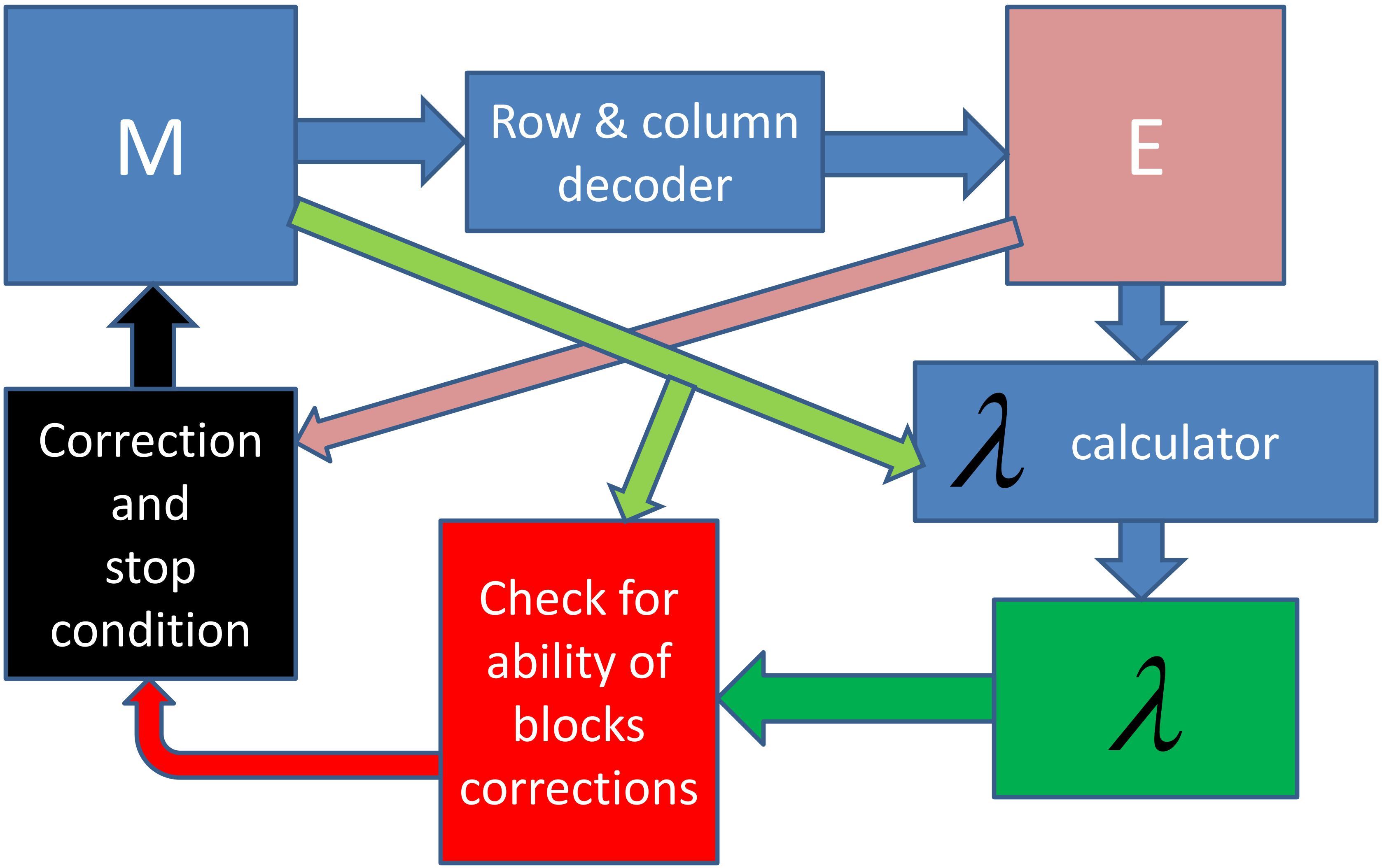
Definition

$$\lambda^h = \log \Lambda^h = \sum_{i=1}^{n} \lambda_i^h + \lambda_0^h, \lambda_i^h = \log \frac{\Pr\left(\beta_i^h \mid G, \alpha_i^h, t_i^h, U_{root}^{\varnothing}\right)}{\Pr\left(\beta_i^h \mid B, \alpha_i^h, t_i^h, U_{root}^{\varnothing}\right)}, \lambda_0^h = \log \frac{\Pr\left(G \mid t_{root}\right)}{\Pr\left(B \mid t_{root}\right)}.$$

Estimation

$$\lambda_i^h \approx \begin{cases} -\log\left[\Pr\left(G' \mid t_i^h\right) / \Pr\left(B' \mid t_i^h\right) + 1\right] \approx \delta\left(d - \gamma t_i^h\right) > 0, \quad \alpha \neq \beta \\ \log\left(1 - \dfrac{t_i^h / n}{\Pr\left(G' \mid t_i^h\right) / \Pr\left(B' \mid t_i^h\right) + 1}\right) - \log\left(1 - t_i^h / n\right) \approx c\dfrac{t_i^h}{n}, \quad \alpha = \beta = 0 \,\&\, u \neq \varnothing', c < 1, \\ \log\left(\dfrac{nq}{t_i^h}\right) \approx 2\log q - \log t_i^h \approx 2\log q - \left(t_i^h - 1\right)/2, \quad \alpha = \beta \neq 0 \,\&\, u \neq \varnothing', \\ \log q + \log\left(1 - \dfrac{1 - 1/q}{\Pr\left(G' \mid t_i^h\right) / \Pr\left(B' \mid t_i^h\right) + 1}\right) \approx \log q - \varepsilon, 0 \leq \varepsilon < 1, \quad \alpha = \beta \,\&\, u = \varnothing', \\ \log\left(1 - p^*\right), \quad t_i^h = \varnothing \,\&\, \alpha = \beta = 0 \\ \log p^*, \quad t_i^h = \varnothing \,\&\, \alpha \neq \beta = 0 \end{cases}$$

New Algorithm



Simulation results

- Code parameters: size of matrix **M** – 256x256,
- Component code: Generalized Reed-Solomon code over $GF(16)$, n=16, k=14 or 12, code distance – 3 or 5.
- General graph code: total length N=4096, total dimension K=3072 or 2048, code distance (estimate) – 28 or 461.
- Simulation results for conditional error probability per a code symbol are given in the table below:

  δ=1.5, γ=.5 for RS d=3 and

  δ=2.0, γ=.5 for RS d=5.

S Eras.	T error	New decoder	Refer. decoder	S Eras.	T error	New decoder	Refer. decoder	S Eras.	T error	New decoder	Refer. decoder
0	250	1.e-2	3.e-2	40	130	1.e-6	4.e-4	200	90	1.2e-5	1.2e-3
0	200	8.3e-4	7.4e-3	40	100	1.1e-7	5.3e-5	200	60	2.1e-6	1.3e-4
0	190	3.5e-4	4.8e-3	50	100	3.5e-7	6.7e-5	200	40	8.3e-7	2.5e-5
0	170	5.3e-5	1.7e-3	60	145	2.7e-5	1.4e-3	200	20	7.8e-8	2.3e-6
0	150	5.2e-6	5.6e-4	80	190	3.9e-3	1.5e-2	200	15	0	1.e-6
0	140	1.2e-6	3.e-4	80	170	2.e-4	7.3e-3	300	100	1.4e-3	1.4e-2
0	130	4.2e-7	1.5e-4	80	150	9.9e-5	2.8e-3	300	20	1.8e-7	2.7e-5
0	120	8.5e-8	8.2e-5	80	135	1.5e-5	1.2e-3	400	70	1.5e-3	1.6e-2
0	110	4.9e-8	4.2e-5	80	110	1.7e-6	2.6e-4	400	10	2.9e-8	4.6e-5
10	200	1.1e-3	8.5e-3	80	80	1.9e-7	3.4e-5	500	40	1.5e-3	1.4e-2
10	160	2.6e-5	1.2e-3	100	170	1.6e-3	9.9e-3	500	20	6.1e-6	2.2e-3
10	130	4.1e-7	1.9e-4	100	140	7.4e-5	2.5e-3	500	10	5.3e-7	3.7e-4
10	100	4.6e-8	2.7e-5	100	100	1.4e-6	2.1e-4	600	20	2.3e-3	1.2e-2
20	190	6.2e-4	6.5e-3	120	140	1.7e-4	3.7e-3	600	5	1.5e-6	7.2e-4
20	140	2.6e-6	4.5e-4	120	60	4.2e-7	1.8e-5	620	2	8.1e-7	2.4e-4
20	100	2.7e-8	3.4e-5	160	100	8.e-6	8.7e-4	750	0	0	0

GRS code over *GF*(16), *n*=16, *k*=12, code distance – 5.

Graph code: total length *N*=4096, total dimension *K*=2048, code distance (estimate) – 461.

S Eras.	T error	New decoder	Refer. decoder	S Eras.	T error	New decoder	Refer. decoder	S Eras.	T error	New decoder	Refer. decoder
0	720	8.3e-5	1.e-1	200	550	2.e-6	1.4e-2	650	350	1.7e-3	1.2e-1
0	700	1.e-6	7.6e-2	200	500	0	5.6e-5	650	300	6.e-6	4.6e-2
0	600	0	2.7e-5	250	450	0	2.3e-6	650	250	0	3.2e-4
0	580	0	0	250	440	0	0	650	220	0	1.8e-6
20	680	0	5.7e-2	400	460	1.5e-5	7.4e-2	650	200	0	0
40	670	1.5e-6	5.6e-2	400	400	0	1.2e-3	800	220	0	3.9e-2
70	650	0	4.7e-2	400	370	0	1.3e-5	800	100	0	0
70	600	0	1e-3	450	410	0	3.7e-2	1400	30	3.7e-3	7.e-2
150	600	0	4.1e-2	550	400	1.e-3	1.1e-1	1400	10	0	6.3e-4
150	520	0	7.1e-6	550	370	0	7.4e-2	1600	10	1.3e-1	1.4e-1
160	550	0	1.3e-3	550	350	0	3.4e-2	1600	0	0	0
160	500	0	0	550	300	0	1.7e-4	1650	0	2.5e-4	2.7e-4
				550	270	0	1.6e-6	1700	0	9.8e-2	9.8e-2

REFERENCES

[1] T. Hoeholdt, J. Justesen, “Iterated Decoding of Product Codes and Graph Codes

with RS Component Codes”, proceedings ISIT 2007.

[2] T. Hoeholdt, J. Justesen, "Graph codes with Reed-Solomon component codes," proceedings ISIT 2006, pp. 2022-26, Seattle, Washington, July, 2006.

[3] M. Schwartz, P.H. Siegel, A. Vardy, "On the asymptotic performance of iterative decoders for product codes," proceedings ISIT 2005, pp. 1758-62, Adelaide, Australia, September 2005.