Fourteenth International Workshop on Algebraic and Combinatorial Coding Theory
September 7–13, 2014, Svetlogorsk (Kaliningrad region), Russia *pp. 7–13*

Incremental calculation of decoding failure probability for iterative decoding of Reed-Solomon product code

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

Abstract. Products of Reed-Solomon codes are important in applications because they offer a combination of large blocks, low decoding complexity, and good performance [3,4]. We give exact and approximate methods for calculation of an increment of failure probability for any iteration.

1 Product code construction and decoding algorithm

We consider an $(n \times m)$ product code over $GF(q)$ with the component RS codes $\mathbb{C}_C(n,k,d)$ for columns and $\mathbb{C}_R(m,k,d)$ for rows of a codeword matrix. Let $\mathbf{A}$ be a $k \times k$ message matrix then $\mathbf{C} = \mathbf{G}_C^T \mathbf{A} \mathbf{G}_R$ is a codeword of product code, where $\mathbf{G}_R$, $\mathbf{G}_C$ are generating matrixes of row and column component codes. The product code has the minimal distance d^2 and has applicable decoding algorithm [1] that corrects all configurations up to $\frac{d^2-1}{2}$ errors and much more.

Investigation of iterative decoding for product codes has a long history, nevertheless J. Justesen and T. Høholdt [2, 3] gave a new explanation of the iterative process. Their main idea is in estimation of unavoidable degradation of the error model during iterative process. We will follow this idea might be in more constructive way.

At first, suppose that the component code decoder works without decoding errors. So the decoding result can be *correct decoding* (with correction up to $t = \frac{d-1}{2}$ errors) or *decoding failure* (when there are more than t errors). It is not so extravagant supposition because probability of decoding error is always less $1/t!$ [2, 3].

Next, in a simplified version product code decoder works as follows. We use list of undecoded rows L_R and list of undecoded columns L_C. At the beginning, all rows and columns are undecoded with error density equal to channel error probability in average. During iteration, the column component decoder examines undecoded columns from the list L_C for correction $\leq t$ errors if size of the list L_R is more $(d-1)$, or corrects $\leq (d-1)$ erasures (and errors)

from the list of undecoded rows, and deletes all successfully decoded columns from the list L_C. The similar rules work for the row component decoder.

Remark 1. *After the first iteration error density in all the rows and columns remaining in the lists L_C and L_R is higher than the channel error probability and higher than* $\min\left\{\frac{t+1}{|L_C|}, \frac{t+1}{|L_R|}\right\}$.

Remark 2. *The product code decoding failure condition: size of the list L_C (or L_R) has not changed during iteration (except the first one). Additionally, detection of necessary correction of position out of the lists L_C and L_R by any component decoder means detection of incorrect decoding on some of earlier iteration.*

Remark 3. *De facto, component decoders work on shortened component codes length of which is equal to size of the lists.*

2 General scheme of probabilities calculation

The probability of correct decoding of the product code after I iterations we define as the following sum $\Pr_C(I) = \sum_{i=1}^{I} \Delta_i$, where Δ_i is an increment of the probability on i-th iteration. Decoding failure probability (under accepted suppositions) is just supplement of $\Pr_C(I)$ to one.

Let we start from column decoding and let ℓ_i be the number of undecoded "bad" blocks after i-th iteration. Initial values are $|L_C| = \ell_0 = m$, $|L_R| = \ell_{-1} = n$.

After the first iteration, we get ℓ_1 undecoded "bad" columns. If $\ell_1 < d$, than on the next step the row component decoder corrects as erasures all n rows. Other case the row decoder corrects all possible rows with $\leq t$ errors and defines the value ℓ_2 of undecoded "bad" rows.

After each iteration, we have to recalculate estimate of the error density ρ_i in the remaining "bad" blocks taking into account their real (shortened) length. Initial value ρ_1 is the channel error probability. After the first iteration, we define ρ_2 as ratio of average number of errors in "bad" columns to their height m. We will continue in this manner on the next iterations.

3 Exact formulas for calculation

Definition 1. *Error density on the i-th iteration is as follows:*

$$\rho_i = \frac{1}{\ell_{i-2}} \frac{\lambda^* (\ell_{i-2}, \rho_{i-1})}{\lambda (\ell_{i-2}, \rho_{i-1})}, \quad \lambda^* (\ell, \rho) = \sum_{j=t+1}^{\ell} j \left(\begin{array}{c} \ell \\ j \end{array} \right) \rho^j (1-\rho)^{\ell-j},$$

$$\lambda (\ell, \rho) = \sum_{j=t+1}^{\ell} \left(\begin{array}{c} \ell \\ j \end{array} \right) \rho^j (1-\rho)^{\ell-j}.$$

Definition 2. *Probability of correct decoding of a block is*

$$\gamma (\ell_{j-1}, \rho_j) = \sum_{v=0}^{t} \left(\begin{array}{c} \ell_{j-1} \\ v \end{array} \right) (\rho_j)^v (1-\rho_j)^{\ell-v}.$$

Definition 3. *Probability of ℓ_j undecoded blocks conditioned to the state (ℓ_{j-1}, ℓ_{j-2}) of two last iterations is*

$$\Pr (\ell_j | \ell_{j-1}, \ell_{j-2}) = \left(\begin{array}{c} \ell_{j-2} \\ \ell_j \end{array} \right) (\gamma (\ell_{j-1}, \rho_j))^{\ell_{j-2}-\ell_j} (1 - \gamma (\ell_{j-1}, \rho_j))^{\ell_j}.$$

Definition 4. *Increment of the probability of correct decoding of the product code on the i-th iteration is as follows:*

$$\Delta_i = \sum_{\ell_1=d+\delta(i,1)}^{n} \Pr (\ell_1 | m, n) \sum_{\ell_2=d+\delta(i,2)}^{m-1} \Pr (\ell_2 | \ell_1, m) \sum_{\ell_3=d+\delta(i,3)}^{\ell_1-1} \Pr (\ell_3 | \ell_2, \ell_1) \ldots \times$$

$$\times \sum_{\ell_{i-1}=d+\delta(i,i-1)}^{\ell_{i-3}-1} \Pr (\ell_{i-1} | \ell_{i-2}, \ell_{i-3}) \sum_{\ell_i=0}^{d-1} \Pr (\ell_i | \ell_{i-1}, \ell_{i-2}), \ \delta (i,j) = \left\lfloor \frac{i-j-1}{2} \right\rfloor.$$

The upper and lower limits for summation are defined in accordance with the rules of the product code decoding failure.

Complexity of calculation Δ_i is exponential with iteration number i (it is more or less evident).

Example of exact formulas for few first iterations see in Table 1. Examples of calculation of increments and probabilities see on Figure 1.

4 Approximate recursive scheme calculation

To decrease the complexity from exponential to a polynomial function we can use the following way. Let see on the last three terms in general expression for

increment Δ_i. They are as follows: $\sum_{\ell_{i-2}=d+\delta(i,i-2)}^{\ell_{i-4}-1} \Pr(\ell_{i-2}|\ell_{i-3},\ell_{i-4})$, $\sum_{\ell_{i-1}=d+\delta(i,i-1)}^{\ell_{i-3}-1} \Pr(\ell_{i-1}|\ell_{i-2},\ell_{i-3})$, $\sum_{\ell_i=0}^{d-1} \Pr(\ell_i|\ell_{i-1},\ell_{i-2})$. All the previous terms give a trace to calculate the probabilities $\Pr(\ell_{i-2}|\ell_{i-3},\ell_{i-4})$, $\Pr(\ell_{i-1}|\ell_{i-2},\ell_{i-3})$. These terms are related with a random shortened $(\ell_{i-1}\times\ell_{i-2})$ subcode of the given product code. So, we could find the probability $\Pr(\ell_{i-1},\ell_{i-2})$ and estimate the Δ_i in the following form for $i\geq 3$ (initial values Δ_1 and Δ_2 are calculated by exact formulas):

$$\tau(i)=\ell_{-(i \bmod 2)}-\lfloor i/2\rfloor,\ \ell_{-1}=n,\ \ell_0=m,$$
$$\Delta_i\approx\sum_{\ell_{i-2}=d}^{\tau(i-2)}\sum_{\ell_{i-1}=d}^{\tau(i-1)}\Pr{}^{\bullet}(\ell_{i-1},\ell_{i-2})\sum_{\ell_i=0}^{d-1}\Pr(\ell_i|\ell_{i-1},\ell_{i-2}),$$

$$\Pr(\ell_i|\ell_{i-1},\ell_{i-2})=\left(\begin{array}{c}\ell_{i-2}\\ \ell_i\end{array}\right)(\gamma(\ell_{i-1},\rho_i))^{\ell_{i-2}-\ell_i}(1-\gamma(\ell_{i-1},\rho_i))^{\ell_i},$$

$$\gamma(\ell_{i-1},\rho_i)=\sum_{v=0}^{t}\left(\begin{array}{c}\ell_{i-1}\\ v\end{array}\right)(\rho_i)^v(1-\rho_i)^{\ell_{i-1}-v},$$

$$\rho_i(\ell_{i-2},\widehat{\rho}_{i-1})=\frac{1}{\ell_{i-2}}\frac{\lambda^*(\ell_{i-2},\widehat{\rho}_{i-1})}{\lambda(\ell_{i-2},\widehat{\rho}_{i-1})}\Rightarrow\rho_i,$$

$$\lambda(\ell_{i-2},\widehat{\rho}_{i-1})=\sum_{j=t+1}^{\ell_{i-2}}\left(\begin{array}{c}\ell_{i-2}\\ j\end{array}\right)\widehat{\rho}_{i-1}^{\,j}(1-\widehat{\rho}_{i-1})^{\ell-j},$$

$$\lambda^*(\ell_{i-2},\widehat{\rho}_{i-1})=\sum_{j=t+1}^{\ell_{i-2}}j\left(\begin{array}{c}\ell_{i-2}\\ j\end{array}\right)\widehat{\rho}_{i-1}^{\,j}(1-\widehat{\rho}_{i-1})^{\ell-j}.$$

In order to prepare the next iteration we should calculate the following:

$$\Pr{}^{\bullet}(\ell_i,\ell_{i-1})=\sum_{\ell_{i-2}=d+1}^{\tau(i-2)}\Pr{}^{\bullet}(\ell_{i-1},\ell_{i-2})\Pr(\ell_i|\ell_{i-1},\ell_{i-2};d\leq\ell_i<\ell_{i-2}),$$
$$d\leq\ell_i\leq\tau(i);$$
$$\Pr{}^{\bullet}(\ell_{i-2})=\sum_{\ell_{i-1}=d+1}^{\tau(i-1)}\Pr{}^{\bullet}(\ell_{i-1},\ell_{i-2}),\ d\leq\ell_{i-2}\leq\tau(i-2),$$

$$\widehat{\rho}_i(\widehat{\rho}_{i-1})=\frac{\sum_{\ell_{i-2}=d+1}^{\tau(i-2)}\Pr{}^{\bullet}(\ell_{i-2})\lambda^*(\ell_{i-2},\widehat{\rho}_{i-1})}{\sum_{\ell_{i-2}=d+1}^{\tau(i-2)}\Pr{}^{\bullet}(\ell_{i-2})\,\ell_{i-2}\lambda(\ell_{i-2},\widehat{\rho}_{i-1})}\Rightarrow\widehat{\rho}_i.$$

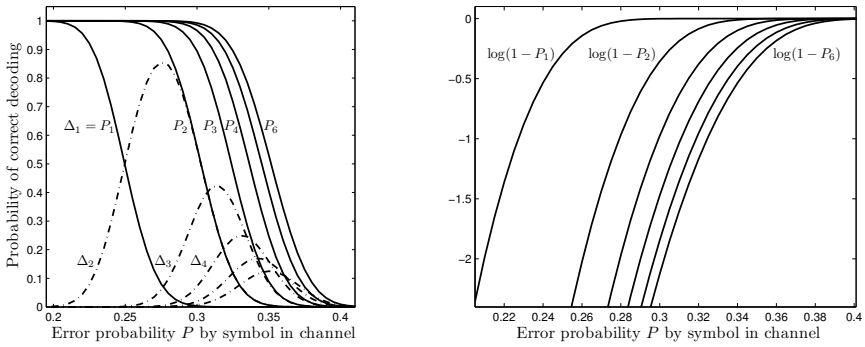

Figure 1: **Probability of correct decoding and logarithm of failure probability for** $[30, 16, 15]_{64} \times [30, 16, 15]_{64}$ **product code with component RS codes.** probabilities $P_1, P_2, \ldots, P_6$ of correct decoding (the solid curves); increments $\Delta_1, \Delta_2, \ldots, \Delta_6$ of probabilities P_i (the dashed-dotted curves); $1, 2, \ldots, 6$ are the numbers of iterations

In the last expression, we define average density of error in a "bad" block as ratio of average number of errors in that block to its average length.

Examples and comparison of exact and approximate calculations see on Figure 2.

Complexity of approximate procedure is linear on iteration number and approximately cubic on the product code size.

5 Conclusion

We have defined here exact and approximate procedures for calculation of probabilities of correct decoding or decoding failure for a product of Reed-Solomon codes under the strong condition that the probability of error of component decoder is negligible. The point of suggested method is definition of degradation of error model (error density ρ_i) during iterative decoding. As we can see from Figure 2 approximation error by iteration is small and accumulation of errors with the iteration number is visible which is usual effect for recurrent calculation.

We are sure that this way for probability calculation can be expanded on other product codes.

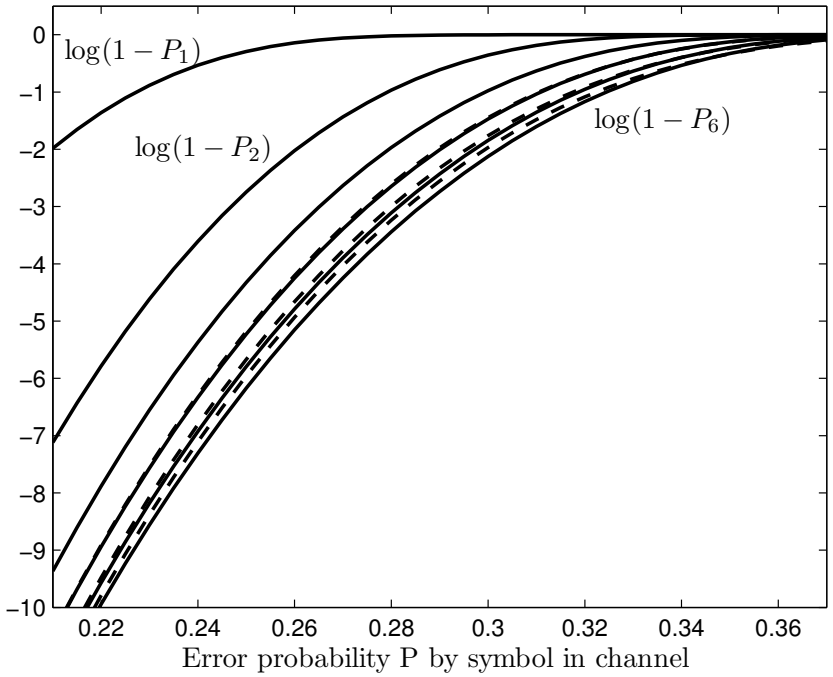

Figure 2: **Logarithm of failure probability for** $[30, 16, 15]_{64} \times [30, 16, 15]_{64}$ **product code with component RS codes.** exact calculations (solid curve) and approximation (dashed curve); $1 - P_i$ is failure probability after the i-th iteration

Table 1: Increments Δ_i of correct decoding probability

	ℓ_1	ℓ_2	ℓ_3	ℓ_4	ℓ_5
Δ_1	$\sum_{\ell_1=0}^{d-1} \Pr(\ell_1)$				
Δ_2	$\sum_{\ell_1=d}^{n} \Pr(\ell_1)$	$\sum_{\ell_2=0}^{d-1} \Pr(\ell_2\|\ell_1)$			
Δ_3	$\sum_{\ell_1=d}^{n} \Pr(\ell_1)$	$\sum_{\ell_2=d}^{m-1} \Pr(\ell_2\|\ell_1)$	$\sum_{\ell_3=0}^{d-1} \Pr(\ell_3\|\ell_2\ell_1)$		
Δ_4	$\sum_{\ell_1=d+1}^{n} \Pr(\ell_1)$	$\sum_{\ell_2=d}^{m-1} \Pr(\ell_2\|\ell_1)$	$\sum_{\ell_3=d}^{\ell_1-1} \Pr(\ell_3\|\ell_2\ell_1)$	$\sum_{\ell_4=0}^{d-1} \Pr(\ell_4\|\ell_3\ell_2)$	
Δ_5	$\sum_{\ell_1=d+1}^{n} \Pr(\ell_1)$	$\sum_{\ell_2=d+1}^{m-1} \Pr(\ell_2\|\ell_1)$	$\sum_{\ell_3=d}^{\ell_1-1} \Pr(\ell_3\|\ell_2\ell_1)$	$\sum_{\ell_4=d}^{\ell_2-1} \Pr(\ell_4\|\ell_3\ell_2)$	$\sum_{\ell_5=0}^{d-1} \Pr(\ell_5\|\ell_4\ell_3)$

References

[1] R. E. Blahut, *Theory and practice of error control codes*, Add-Wil. Publishing company reding, Massachusets, 1984.

[2] J. Justesen and T. Høholdt, Iterative Decoding of Product Codes, in *Proc. XII Int. Workshop on Algebraic and Combin. Coding Theory, ACCT2010, Novosibirsk, Russia, 2010,* 5–11.

[3] J. Justesen and T. Høholdt, Iterated Decoding of Product Codes and Graph Codes with RS Component Codes, ISIT 2010.

[4] J. Justesen, K. J. Larsen, and L. A. Pedersen, Error Correcting Coding for OTN, *IEEE Commun. Magaz.* September 2010