UNIVERSIDAD NEBRIJA

Verification of the effectiveness of DS-LDPC codes for memory applications

M. Pilar Vélez

22 de junio de 2016

Encuentro de Algebra Computacional y Aplicaciones - EACA 2016
Logroño, 22-24 de junio de 2016

DS-LDPC Codes

Difference Set Low Density Parity Check code C of length N are

- Binary linear cyclic codes; i.e. k-dimensional vector subspaces of $\mathbb{F}_2^N$ over $\mathbb{F}_2$.
- (N, K) block codes that acts on a block of K bits of input data to produce N bits of output data; the rest $N - K$ bits are called parity check bits.
- Based in the construction of a perfect difference set.
- High-error correction capability codes.

Defining a DS-LDPC code

Take $q = 2^s$, $N = q(q+1)+1$

Perfect difference set: $P = \{l_0, \ldots, l_q\} \subset \mathbb{Z}^+ \cup \{0\}$ with $0 = l_0 < \cdots < l_q \leq N-1$ and the differences $l_j - l_k$ modulo N, $j \neq k$, are all different.

Sindrome polynomial:

Take $\theta(X) = 1 + X^{l_1} + X^{l_2} + \cdots + X^{l_q}$, the syndrome polynomial of the code is the polynomial of degree K

$$h(X) = \gcd(\theta(X), X^N - 1)$$

The code is generated by the polynomial of degree $N - K$:

$$g(X) = \frac{X^N - 1}{h(X)}$$

Error-correcting

Check-polynomials for P:

$$w_0 = X^{N-1} + X^{N-1-l_1} + X^{N-1-l_2} + \cdots + X^{N-1-l_q}$$

For $k = 1, \ldots, q$ take $(X^{l_k} w_0)$, that is, add $l_k (\text{mod } N)$ to the exponents of w_0:

$$\begin{aligned} w_k \quad &= X^{l_k-1} + X^{l_k-l_1-1} + \cdots + X^{l_k-l_{k-1}-1} + \\ &+ X^{N-1} + X^{N-1-(l_{k+1}-l_k)} + \cdots + X^{N-1-(l_q-l_k)} \end{aligned}$$

As P is a perfect difference set, two polynomials w_i and w_j have not common terms except X^{N-1}, then $\{w_0(X), w_1(X), \ldots, w_q(X)\}$ are $q+1$ **polynomials orthogonal on** X^{N-1}.

This implies that $\{w_0(X), w_1(X), \ldots, w_q(X)\}$ is a parity check set able to correct up to $q/2$ errors and then the code have minimun distance at least $d = q+2$ [Massey,1963].

Detecting errors: Check-sums

Codeword $x \in C$

$$e = y - x = e_0 e_1 \cdots e_{N-1}$$ Error vector

Received word y

$$\#\{\text{ errors in } y\} = \#\{1'\text{s in } e\}$$

Check-sums: $w_0(e), \ldots, w_q(e)$,

$$w_k(e) = \sum_{j \in \text{ exponent in } w_k} e_j \mod 2, \; k \in \{0, 1, \ldots, q\}$$

This process is called *evaluation* of e in w_k.

Detecting errors

Shift the error vector e in order to have all cycles:

$$\begin{aligned} \text{Cycle } 0: &\quad \mathcal{E}_0 = e_0 e_1 \cdots e_{N-1} \\ \text{Cycle } 1: &\quad \mathcal{E}_1 = e_{N-1} e_0 \cdots e_{N-2} \\ &\vdots \qquad\qquad \vdots \\ \text{Cycle } N-1: &\quad \mathcal{E}_{N-1} = e_1 e_2 \cdots e_0 \end{aligned}$$

If $t \leq d - 1 = q + 1$, we can detect at most t errors as follows:

$$y \in C \Longleftrightarrow w_k(\mathcal{E}_j) = 0 \ (\text{mod } \ 2), \ \forall k \in \{0, \ldots, q\}, \ \forall j \in \{0, \ldots, N-1\}$$

Example

$P = \{0, 2, 3\}$ is a perfect difference set with $q = 2$ and $N = 2(2+1)+1 = 7$.

Check-sums:

$$\begin{array}{lcl} w_0 = X^6 + X^4 + X^3 & & w_0(e) = e_6 + e_4 + e_3 \\ w_1 = X^1 + X^6 + X^5 & \rightarrow & w_1(e) = e_6 + e_5 + e_1 \\ w_2 = X^2 + X^0 + X^6 & & w_2(e) = e_6 + e_2 + e_0 \end{array}$$

Consider the error vector $e = 0010111$

$w_0(\mathcal{E}_0) = 1+1+0 = 0,\ w_1(\mathcal{E}_0) = 1+1+0 = 0$ and $w_2(\mathcal{E}_0) = 1+1+0 = 0$

$w_0(\mathcal{E}_1) = 1+0+1 = 0,\ w_1(\mathcal{E}_1) = 1+1+0 = 0$ and $w_2(\mathcal{E}_1) = 1+0+1 = 0$

$\ldots$

$w_0(\mathcal{E}_6) = 0+1+1 = 0,\ w_1(\mathcal{E}_6) = 0+1+1 = 0$ and $w_2(\mathcal{E}_6) = 0+0+0 = 0$

Remark that $q = 2$ then check sums detect up to 3 errors.

Usual lengths of DS-LDPC codes applied to memories are $N = 73, 273$ and 1057 (i.e. $q = 8, 16, 32$)

Conjecture

Improving performance of Majority logic decoder [Liu, Reviriego & Maestro, 2010] conjectured:

Conjecture : *Given a word read from a memory protected with DS-LDPC codes (code length $N = 73, 273, 1057$) and affected by up to five bit-flips, all errors can be detected in only three decoding cycles.*

To prove their conjecture, they try to check each possible combination of error vectors with up to 5 errors and they succeded partially. For four errors and $N = 1057$ they get checked the 2% of the cases; that is, one billion cases of possible $\binom{1057}{4} = 51.911.764.520$.

Theorem

Theorem: Let C be a DS-LDPC code with $q = 4, 8, 16, 32$. Let $x \in C$ and y the received vector with up to five errors. Then, the evaluation of the check-sums on three consecutive cycles of y is enough to detect the existence of errors.

Lemma: Under the same conditions of the theorem, if an error occurs in the $N - 1$ position, then at least one check-sum is non-zero.

Proof of the Lemma: As $q \geq 4$, the number of check-sums is bigger or equal than 5. Moreover, we have a factor 1 in each check-sum corresponding to exponent $N - 1$ and the other ones are at most 4. Then, some check-sum is non-zero.

Notation and Remark

Denote m_{ij} the position of the ith error of y in Cycle j, $j \in \{0, 1, \ldots, N-1\}$.

- If $m_{i0} = \alpha$ then $m_{ij} = \alpha + j \mod N$
- m_{ij} will be determined by one term of one of the polynomials, say $w_{f_{ij}}$.

$$m_{ij} = l_{f_{ij}} - l_{c_{ij}} - 1 \mod N$$

That means that the position of the error is the exponent of the term placed in the position c_{ij} of the polynomial $w_{f_{ij}}$.

Proof of the theorem (odd number of errors)

Suppose the received vector y has up to five errors, then the error vector e has at most 5 ones.

Odd number of errors:
If e has an odd number of ones, errors are detected in the first cycle. As check-sums does not share positions except $N-1$, some check-sum must be odd.

In other case the number of errors must be 2 or 4 and we suppose that the error is not detected in three consecutive cycles

Proof of the theorem (two errors)

Two errors:
If e has two ones, then the position of ones occurs in the same polynomial in each cycle, denote they by $w_{f_{10}}$, $w_{f_{11}}$ and $w_{f_{12}}$:

$$w_{f_{10}}: \begin{array}{l} m_{10} = l_{f_{10}} - l_{c_{10}} - 1 \\ m_{20} = l_{f_{10}} - l_{c_{20}} - 1 \end{array} \Rightarrow m_{10} - m_{20} = l_{c_{10}} - l_{c_{20}}$$

$$w_{f_{11}}: \begin{array}{l} m_{11} = l_{f_{11}} - l_{c_{11}} - 1 \\ m_{21} = l_{f_{11}} - l_{c_{21}} - 1 \end{array} \Rightarrow m_{11} - m_{21} = (m_{10}+1) - (m_{20}+1) = m_{10} - m_{20} = l_{c_{11}} - l_{c_{21}}$$

$$w_{f_{12}}: \begin{array}{l} m_{12} = l_{f_{12}} - l_{c_{12}} - 1 \\ m_{22} = l_{f_{12}} - l_{c_{22}} - 1 \end{array} \Rightarrow m_{12} - m_{22} = (m_{10}+2) - (m_{20}+2) = m_{10} - m_{20} = l_{c_{12}} - l_{c_{22}}$$

$$m_{10} - m_{20} \neq 0 \text{ and } l_{c_{ij}} \in P \quad \Rightarrow \quad l_{c_{10}} = l_{c_{11}} = l_{c_{12}} \text{ and } l_{c_{20}} = l_{c_{21}} = l_{c_{22}}$$

Proof of the theorem (two errors)

Looking to the positions of the first error we have:

$$\begin{aligned} m_{10} &= l_{f_{10}} - l_{c_{10}} - 1 \\ m_{11} &= l_{f_{11}} - l_{c_{10}} - 1 \end{aligned} \Longrightarrow m_{11} - m_{10} = (m_{10} + 1) - m_{10} = 1 = l_{f_{11}} - l_{f_{10}}$$

and

$$\begin{aligned} m_{11} &= l_{f_{11}} - l_{c_{10}} - 1 \\ m_{12} &= l_{f_{12}} - l_{c_{10}} - 1 \end{aligned} \Longrightarrow m_{12} - m_{11} = (m_{11} + 1) - m_{11} = 1 = l_{f_{12}} - l_{f_{11}}$$

$$1 = l_{f_{11}} - l_{f_{10}} = l_{f_{12}} - l_{f_{11}} \quad \Rightarrow \quad l_{f_{10}} = l_{f_{11}} = l_{f_{12}} \text{ and } 1 = l_{f_{12}} - l_{f_{11}}$$

Contradiction

Proof of the theorem (four errors)

Four errors:

If e has four ones, let m_{10}, m_{20}, m_{30} and m_{40} the initial position of these ones and suppose that $m_{ij} \neq N-1$ for any $i \in \{1,2,3,4\}$ and for any $j \in \{0,1,2\}$.

Case 1: Errors team up in pairs and each pair shares polynomial along the three cycles.

Case 2: Errors team up in pairs and each pair shares polynomial in two of the three cycles. In the other cycle they mix.

Case 3: Each pair of errors shares the same polynomial only once in the three cycles.

Relating positions of ones in check-sums analogously to the case of two erros we get that cases 1 and 2 are imposible.

Proof of the theorem (four errors)

In case 3 we obtain the following system of 8 perfect difference linear equations with 12 unknowns, $l_{f_{ij}}$ and $l_{c_{ij}}$ lying in P:

$$\begin{aligned}
l_{f_{10}} - l_{c_{10}} &= l_{f_{11}} - l_{c_{11}} - 1 & \qquad l_{f_{30}} - l_{c_{30}} &= l_{f_{11}} - l_{c_{31}} - 1 \\
l_{f_{11}} - l_{c_{11}} &= l_{f_{12}} - l_{c_{12}} - 1 & \qquad l_{f_{11}} - l_{c_{31}} &= l_{f_{22}} - l_{c_{32}} - 1 \\
l_{f_{10}} - l_{c_{20}} &= l_{f_{21}} - l_{c_{21}} - 1 & \qquad l_{f_{30}} - l_{c_{40}} &= l_{f_{21}} - l_{c_{41}} - 1 \\
l_{f_{21}} - l_{c_{21}} &= l_{f_{22}} - l_{c_{22}} - 1 & \qquad l_{f_{21}} - l_{c_{41}} &= l_{f_{12}} - l_{c_{42}} - 1
\end{aligned}$$

Additionaly, we have the following restrictions:

- $m_{ij} \neq N - 1$.
- In each cycle $\mathcal{C}_j$, $m_{ij} \neq m_{i'j}$ for $i \neq i'$. In particular, $l_{c_{10}} \neq l_{c_{20}}$, $l_{c_{30}} \neq l_{c_{40}}$, $l_{c_{11}} \neq l_{c_{31}}, \ldots$
- In each cycle, the pairs of errors lie in different polynomials:

  $l_{f_{10}} \neq l_{f_{30}}$, $l_{f_{11}} \neq l_{f_{21}}$ and $l_{f_{12}} \neq l_{f_{22}}$.

Proof of the theorem (four errors)

From the computational point of view to resolve this system of equations is expensive. We tried to solve it using Maple, Cocoa and Matlab whitout succes.

Ad hoc algorithm in Maple:

Step 1: Choose a $l_{c_{ij}}$ variable appearing just in one equation (E_1).

Step 2: Solve (E_1) using the "auxiliary algorithm" and check restrictions.

Step 3: Solve the other equation in the same block as (E_1) and check restrictions.

Step 4: Return to steps $1, 2$ and 3 with the rest of the equations (two times more).

Step 5: The remaining block consists in two equations of length 1 that can be solved straightforward. Check restrictions.

The " auxiliary algorithm" is also implemented in Maple.

Proof of the theorem (four errors)

$$\begin{cases} l_{f_{10}} - l_{c_{10}} = l_{f_{11}} - l_{c_{11}} - 1 \\ l_{f_{11}} - l_{c_{11}} = l_{f_{12}} - l_{c_{12}} - 1 \end{cases}$$
$$\begin{cases} l_{f_{10}} - l_{c_{20}} = l_{f_{21}} - l_{c_{21}} - 1 \\ l_{f_{21}} - l_{c_{21}} = l_{f_{22}} - l_{c_{22}} - 1 \end{cases}$$
$$\begin{cases} l_{f_{30}} - l_{c_{30}} = l_{f_{11}} - l_{c_{31}} - 1 \\ l_{f_{11}} - l_{c_{31}} = l_{f_{22}} - l_{c_{32}} - 1 \end{cases}$$
$$\begin{cases} l_{f_{30}} - l_{c_{40}} = l_{f_{21}} - l_{c_{41}} - 1 \\ l_{f_{21}} - l_{c_{41}} = l_{f_{12}} - l_{c_{42}} - 1 \end{cases}$$

We ran out the algorithm for $N = 21, 73, 273$ and 1057 and we found that the system has no solution.
For the next code length $N = (2^6)(2^6 + 1) + 1 = 4161$, the computer crashed.

Appendix: Perfect difference linear equations

$$a_1 (l_{i_1} - l_{j_1}) + a_2 (l_{i_2} - l_{j_2}) + \cdots + a_r (l_{i_r} - l_{j_r}) = b$$

$$a_\alpha \in \mathbb{Z}_N, \ \alpha = 1, \ldots, r \text{ and } i_1, \ldots, i_r, j_1, \ldots, j_r \in \{0, 1, \ldots q\}.$$

- the unknowns, l_j, belong to the set P.
- any equation of the type $l_{i_\alpha} - l_{j_\alpha} = X_\alpha$, $X_\alpha \in \mathbb{Z}_N$, has one and only one solution in P unless $X_\alpha = 0$, in which case has $q + 1$ solutions.
- the equation $a_1 X_1 + \cdots + a_r X_r = b$, $a_1, \ldots, a_r, b \in \mathbb{Z}_N$, has solution if and only if $\gcd(a_1, \ldots, a_r) \mid b$ and by using the Extended Euclidean Algorithm we can compute the solution set.

Conclusions

1. Our approach guarantees the effectiveness of the error decoding algorithm proposed by [Liu, Reviriego & Maestro, 2010] for $N = 21, 73, 273, 1057$ and codewords affected for up to 5 errors.

2. For $N = 273$, [Liu, Reviriego & Maestro, 2010] found codewords afected six and by eight bit-flips that were not detected in the first three cycles. Nevertheless, for $N = 1057$ after simulate 5 billion cases of the possible $\binom{1057}{6}$, all of them were detected.

UNIVERSIDAD
NEBRIJA