Parallel Encoding Algorithm for LDPC Codes Based on Block-Diagonalization

Takayuki Nozaki

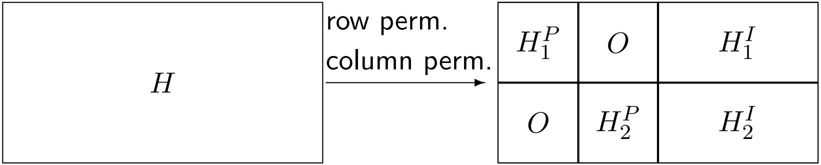
Yamaguchi University
This study is supported by a research granted from The Murata Science Foundation

ISIT2015
18th June 2015

Abstract

Purpose of Research: Reduce the encoding time for LDPC codes

Main Idea: Parallel encoding based on block-diagonalization of H_P



$$\begin{pmatrix} H_1^P & O & H_1^I \\ O & H_2^P & H_2^I \end{pmatrix} (\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{m}) = \boldsymbol{0} \quad \Rightarrow \quad \begin{cases} H_1^P \boldsymbol{p}_1 = -H_1^I \boldsymbol{m} \\ H_2^P \boldsymbol{p}_2 = -H_2^I \boldsymbol{m} \end{cases}$$

Contributions of Research

- Propose an efficient *parallel* encoding algorithm for LDPC codes
- Evaluate the number of operations of the proposed algorithm
  - The number of operations of each processor is almost equal
  - The encoding time becomes $1/K$ compared with conventional one

Background

Low-Density Parity-Check (LDPC) code

Linear code defined by a sparse parity check matrix $H \in \mathbb{F}^{M \times N}$

	Complexity	Parallelization
Encoder	$O(N + \delta^2)$	?
Decoder	$O(N)$	Possible

$(\delta = O(N),\ \delta \ll N)$

Researches of Encoding Algorithm

	Complexity	Parallelization
Generator matrix	$O(N^2)$	Possible
[Richardson 2001], [Kaji 2006]	$O(N + \delta^2)$	?
This study	$O(N + \delta^2)$	Possible

General Framework of Encoding

Encoding algorithm is divided into *precoding step* and *encoding step*

Precoding Step

- Input: Parity check matrix H
- Output: Systematic form $(H_P \mid H_I)$

$PHQ = (H_P \mid H_I)$ P, Q are permutation matrices

$$(H_P \mid H_I) \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{m} \end{pmatrix} = \boldsymbol{0} \quad \Rightarrow \quad H_P \boldsymbol{p} = -H_I \boldsymbol{m}$$

Encoding Step

- Input: Message $\boldsymbol{m}$
- Output: Parity part $\boldsymbol{p}$

Solve linear equation $H_P \boldsymbol{p} = -H_I \boldsymbol{m}$

Encoding algorithm is regarded as an algorithm finding P, Q such that $H_P \boldsymbol{p} = -H_I \boldsymbol{m}$ is efficiently solved

Outline of Proposed Algorithm

Precoding Step of Proposed Algorithm

1. (To realize parallel algorithm,) Transform H into singly bordered block-diagonal (SBBD) matrix [Aykanat 2004]

$$H \Rightarrow H_2^{\mathrm{SBBD}} = \begin{pmatrix} A_1 & O & B_1 \\ O & A_2 & B_2 \end{pmatrix}$$

2. (To efficiently solve the linear equations,) Rearrange row and column of A_i by conventional algorithm (e.g. Approximate triangularization [Richardson 2001])

Outline of the remaining slides

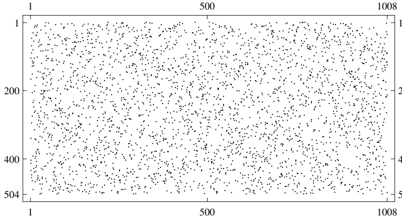
1. Approximate triangularization [Richardson 2001]
2. Singly bordered block-diagonalization [Aykanat 2004]
3. Propose parallel encoding algorithm
    1. Precoding step
    2. Encoding step
    3. Number of operation

[Richardson 2001] (1: Precoding Step)

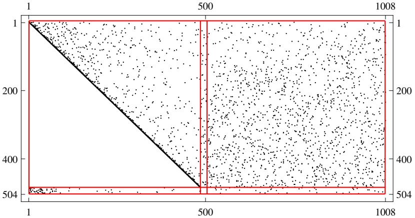
[Richardson 2001]

Transform H_P into *approximate triangular matrix* (ATM)

Complexity $O(N + \delta^2)$



$\Rightarrow$



$$H \Rightarrow H^{\text{ATM}} = \begin{pmatrix} T & C & H_u^I \\ D & E & H_l^I \end{pmatrix}$$

$$\begin{pmatrix} T & C & H_u^I \\ D & E & H_l^I \end{pmatrix} (\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{m})^T = \boldsymbol{0}^T$$

[Richardson 2001] (2: Encoding Step)

$$\begin{pmatrix} T & C & H_u^I \\ D & E & H_l^I \end{pmatrix} (\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{m})^T = \mathbf{0}^T$$

1. ${\boldsymbol{p}_2}^T = \phi^{-1}(DT^{-1}H_u^I - H_l^I)\boldsymbol{m}^T$
2. $T{\boldsymbol{p}_1}^T = -C{\boldsymbol{p}_2}^T - H_u^I \boldsymbol{m}^T$

where $\phi := E - DT^{-1}C \in \mathbb{F}^{\delta \times \delta}$

Number of Operations

- Number of multiplication (For non-binary case)

$$\mu = \mathrm{wt}(H_l^I) + \mathrm{wt}(H_u^I) + \mathrm{wt}(C) + \mathrm{wt}(D) + 2\mathrm{wt}(T) + \mathrm{wt}(\phi^{-1})$$

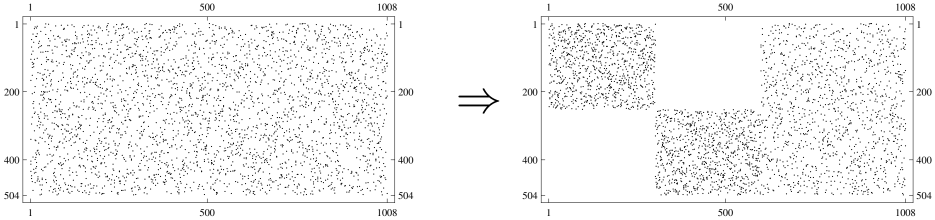
- Number of addition

$$\alpha = \mathcal{S}(H_l^I) + \mathcal{S}(H_u^I) + \mathcal{S}(C) + \mathcal{S}(D) + 2\mathcal{S}(T) + \mathcal{S}(\phi^{-1}) + M$$

where $\mathcal{S}(A) := \mathrm{wt}(A) - (\#\text{ of non-zero rows})$

Block-diagonalization (1: Singly bordered BD)

Singly bordered block-diagonalization (SBBD) [Aykanat 2004]



$$H \Rightarrow H_2^{\mathrm{SBBD}} = \begin{pmatrix} A_1 & O & B_1 \\ O & A_2 & B_2 \end{pmatrix}$$

(To simplify the notation, we assume the number of diagonal blocks $K = 2$ in this talk)

To transform H into SBBD matrix, hypergraph partition is used

1. Show a hypergraph representation of H
2. Solve the hypergraph partition problem

SBBD for H (1: Hypergraph representation of a matrix)

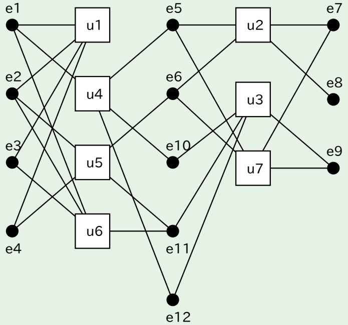
Row-net model [Çatalyürek 1999]

i-th column $\Rightarrow$ i-th net (or hyperedge) e_i

j-th row $\Rightarrow$ j-th vertex u_j

Example

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$



SBDD for H (2: Hypergraph partition)

K-way hypergraph partition $\Pi = \{\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_k\}$

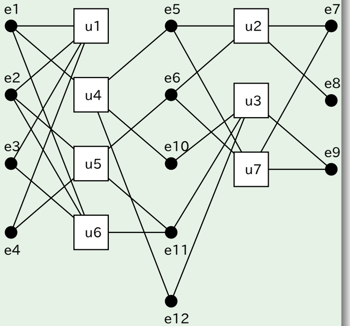
(1) $\emptyset \neq \mathcal{U}_i \subseteq \mathcal{U}$, (2) $\mathcal{U}_i \cap \mathcal{U}_j = \emptyset$ (for $i \neq j$), (3) $\bigcup_{i=1}^{K} \mathcal{U}_i = \mathcal{U}$

Cuts is the nets which connecting to more than one parts
Cutset $\mathcal{X}(\Pi)$ is the set of cuts for the partition Π

Example

$$K = 2$$
$$\mathcal{U}_1 = \{u_1, u_4, u_5, u_6\}$$
$$\mathcal{U}_2 = \{u_2, u_3, u_7\}$$

$$\mathcal{X}(\Pi) = \{e_5, e_6, e_{10}, e_{11}, e_{12}\}$$



SBBD for H (3: Hypergraph partition problem and Transformation of matrix)

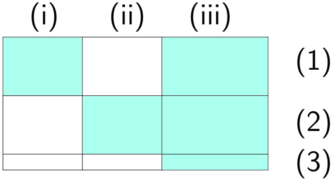
K-way hypergraph partition problem

minimize : $|\mathcal{X}(\Pi)|$
s.t. Balance condition $\max_i |\mathcal{U}_i| < |\mathcal{U}|(1+\epsilon)/K$

There exist some heuristic algorithms (e.g. PaToH [Çatalyürek])

Block-diagonalization of H ($K=2$)

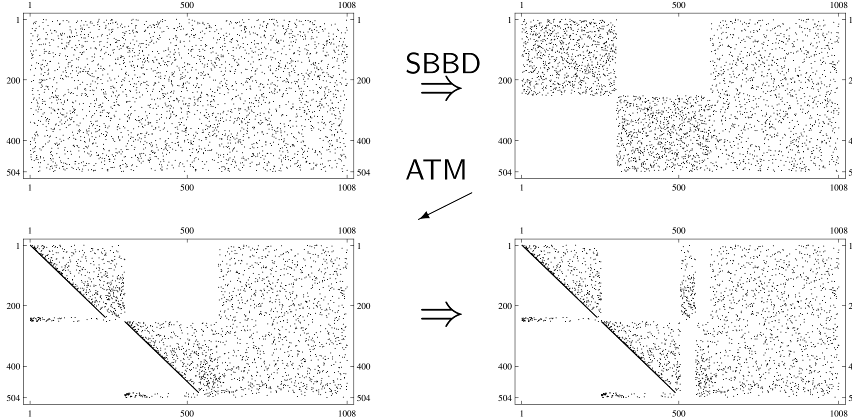
$H =$



(1) $\mathcal{U}_1$
(2) $\mathcal{U}_2$
(3) Vertexes only connected to cutset $\mathcal{X}(\Pi)$

(i) Nets only connecting to $\mathcal{U}_1$
(ii) Nets only connecting to $\mathcal{U}_2$
(iii) Cutset $\mathcal{X}(\Pi)$

Proposed Algorithm (1: Precoding step)



$$\begin{pmatrix} T_1 & C_1 & O & O & H^I_{1,u} \\ D_1 & E_1 & O & O & H^I_{1,l} \\ O & O & T_2 & C_2 & H^I_{2,u} \\ O & O & D_2 & E_2 & H^I_{2,l} \end{pmatrix} (\boldsymbol{p}_{1,1}, \boldsymbol{p}_{1,2}, \boldsymbol{p}_{2,2}, \boldsymbol{p}_{2,2}, \boldsymbol{m})^T = \mathbf{0}^T$$

Proposed Algorithm (2: Encoding Step)

$$\begin{pmatrix} T_1 & C_1 & O & O & H^I_{1,u} \\ D_1 & E_1 & O & O & H^I_{1,l} \\ O & O & T_2 & C_2 & H^I_{2,u} \\ O & O & D_2 & E_2 & H^I_{2,l} \end{pmatrix} (\boldsymbol{p}_{1,1}, \boldsymbol{p}_{1,2}, \boldsymbol{p}_{2,1}, \boldsymbol{p}_{2,2}, \boldsymbol{m})^T = \mathbf{0}^T$$

$\Rightarrow$

$$\begin{pmatrix} T_1 & C_1 & H^I_{1,u} \\ D_1 & E_1 & H^I_{1,l} \end{pmatrix} (\boldsymbol{p}_{1,1}, \boldsymbol{p}_{1,2}, \boldsymbol{m})^T = \mathbf{0}^T$$

$$\begin{pmatrix} T_2 & C_2 & H^I_{2,u} \\ D_2 & E_2 & H^I_{2,l} \end{pmatrix} (\boldsymbol{p}_{2,1}, \boldsymbol{p}_{2,2}, \boldsymbol{m})^T = \mathbf{0}^T$$

Parity parts $\boldsymbol{p}_1, \boldsymbol{p}_2$ are parallelly solved in dual processor system

Numerical Example (Computational Complexity)

Name [MacKay]	RU Algorithm		Proposed Algorithm			
	μ/α	(δ)	μ_1/α_1	(δ_1)	μ_2/α_2	(δ_2)
PEGReg504x1008	4599/3542	(21)	2369/1802	(23)	2384/1819	(23)
PEGReg252x504	2283/1739	(16)	1125/839	(13)	1124/843	(13)
PEGirReg504x1008	5068/4058	(1)	2587/2079	(2)	2599/2093	(1)
PEGirReg252x504	2560/2054	(1)	1284/1028	(2)	1289/1034	(1)
32000.2240.3.105	102659/98159	(7)	50366/48136	(10)	52180/49871	(10)
16383.2130.3.103	55472/51164	(16)	27453/25298	(15)	28055/25848	(19)
4095.737.3.101	14418/12915	(10)	7192/6428	(9)	7174/6412	(9)
10000.10000.3.631	144200/123378	(337)	87501/76950	(302)	96271/85278	(325)
8000.4000.3.483	45006/36709	(141)	23913/19638	(117)	25204/20868	(127)
4000.2000.3.243	20240/16075	(74)	10417/8279	(64)	10475/8298	(61)
504.504.3.504	4572/3517	(19)	2262/1721	(14)	2273/1729	(16)

μ_i : The number of multiplication of i-th processor

α_i : The number of addition of i-th processor

- The number of operations of each processor is almost equal
- The encoding time becomes $1/K$ compared with conventional one

Conclusion

- We have proposed a *parallel* encoding algorithm for LDPC codes
  - Main Idea : Block-diagonalization of parity part H_P
- We have evaluated the number of operations of the proposed algorithm
  - The number of operations of each processor is almost equal
  - The encoding time becomes $1/K$ compared with conventional one