A Note on a Decoding Algorithm of Codes on Graphs with Small Loops

Naoto KOBAYASHI
School of Sci. & Eng.
Waseda University
Tokyo, Japan
Email: naoto@ruri.waseda.jp

Toshiyasu MATSUSHIMA
School of Sci. & Eng.
Waseda University
Tokyo, Japan
Email: toshi@matsu.mgmt.waseda.ac.jp

Shigeichi HIRASAWA
School of Sci. & Eng.
Waseda University
Tokyo, Japan
Email: hirasawa@hirasa.mgmt.waseda.ac.jp

Abstract—The best-known algorithm for the decoding of low-density parity-check (LDPC) codes is the sum-product algorithm (SPA). The SPA is a message-passing algorithm on a graphical model called a factor graph (FG). The performance of the SPA depends on a structure of loops in a FG. Pearl showed that any loops in a graphical model could be erased by the clustering method. This method clusters plural nodes into a single node. In this paper, we show several examples about a decoding on a FG to which the clustering method is applied. And we propose an efficient decoding algorithm for it.

I. INTRODUCTION

In recent years, low density parity check (LDPC) codes [1] have been widely studied. LDPC codes are decoded by the sum-product algorithm (SPA) [2]. The SPA is a message-passing algorithm on a graphical model called a factor graph (FG) [4], and it computes the exact posterior probability if a FG has no loop. The FG that represents any LDPC codes generally has loops. Although the SPA doesn’t always compute the exact posterior probability on a FG with loops, it computes a good approximation of a posteriori probability for the decoding of LDPC codes.

The performance of the SPA depends on a structure of loops in a FG. For example, if a FG includes length-4 loops, errors often occur at the bits corresponding to the loops [2]. To avoid this problem, LDPC codes without small loops are used. Pearl showed that any loops in a graphical model could be erased by the clustering method [5][6]. This method collapses plural nodes into a single node. It can be also applied to a FG [4]. For example, clustering two nodes that correspond to a length-4 loop can erase the loop [7]. We call the FG to which the clustering method is applied a “Cluster Factor Graph (CFG).”

Since the structure of the FG that represents any LDPC codes is very complex, we cannot erase all loops in it by the clustering method. Thus, we consider erasing only some small loops in the FG. We can, however, expect to improve the performance of the SPA on such the CFG, because in the clustering node that contains collapsed plural random variables, the local joint probability of them is calculated. In particular, for a binary erasure channel (BEC), we can prevent the decoding error occurred by stopping sets [12] with the clustering methods. We show the performance of a decoding with the clustering method in this paper.

The computational complexity of the SPA on a CFG is larger than those of the SPA on the original FG, because the number of computation messages is generally increased. For binary party check codes, the computational complexity of the SPA on a CFG can be decreased. In this paper, we explain logic of it and propose an efficient decoding algorithm on a CFG.

This paper is organized as follows: In Section II, we shortly explain a FG and the SPA. In Section III, we explain the clustering method. In Section IV, we interpret the applications of this method for a decoding algorithm and propose an efficient decoding algorithm on a CFG. In Section V, we show several examples for a decoding of LDPC codes by simulations. In Section VI, we discuss the results of the simulations. In Section VII, we summarize the paper.

II. PRELIMINARIES

A. Notation

We assume a codeword $X := \{X_1, X_2, ..., X_N\} \in \{0, 1\}^N$ is transmitted over a noisy memory-less channel with transition probabilities $f_n(= p(Y_n|X_n = a))$, and let $Y := \{Y_1, Y_2, ..., Y_N\}$ be a received sequence. On receiving $Y$, we estimate a codeword $\hat{X} := \{\hat{X}_1, \hat{X}_2, ..., \hat{X}_N\} \in \{0, 1\}^N$. Let $n$ be an index of $n$th codeword symbol ($1 \leq n \leq N$). Let $H$ be a parity-check matrix whose row and column length are $M$ and $N$, and let $H_{mn}$ be the value of $m$th rows and $n$th columns of $H$ ($1 \leq m \leq M$). An index of each row of the parity-check matrix is denoted by $m$. We define $N(m) := \{n|H_{mn} = 1\}$ and $M(n) := \{m|H_{mn} = 1\}$ as sets of index to indicate the position of symbol “1” for each rows and columns.

If $S$ is any sets, we define $|S|$ as the number of element of $S$ and define $S_l(1 \leq l \leq |S|)$ as the $l$th element of $S$.

B. Factor Graph

A Factor Graph (FG) is a graphical model which has two types of node, a variable node and a check node. The details of a FG are described in [4]. Any liner codes can be represented by a FG, assuming that each symbol and received symbol represent as random variables, and a parity check matrix and a channel assumption represent as constrained conditions. We henceforth describe a variable node corresponding to a codeword symbol $n$ as a “variable node $x_n$,” and a check node
corresponding to a \( m \)th row of a parity-check as a “check node \( h_m \).” For each \( x_n \), a variable node \( x_n \) takes on elements in alphabet \( A_{x_n} \). On binary codes, \( A_{x_n} = \{0, 1\} \) for all \( x_n \).

Fig. 1 shows an example of the FG for given the parity check matrix for the (7, 4) Humming code as Eq. (1). We omit variable nodes corresponding to received words and check nodes corresponding to a channel assumption.

\[
H = \begin{pmatrix}
1 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1
\end{pmatrix}
\] (1)

C. Sum-Product Algorithm

The Sum-Product Algorithm (SPA) is a decoding algorithm for LDPC codes. It is an iterative message-passing algorithm on a graph. The details of the algorithm are described in [1][2]. Although there are several types of descriptions about the SPA, in this paper, we deal with the SPA on a probabilistic domain in [3].

III. Clustering Method

A. Clustering Method

The clustering method [5][6] is to collapse plural nodes into a single node on graphical models. This method can be also applied to a FG [4]. We consider that variable nodes are clustered into a single node\(^1\). We call a FG to which the clustering method applies a Cluster Factor Graph (CFG), and a new node prepared by clustering is called a “cluster node”\(^2\). On the other hand, variable nodes which are clustered are called “a variable node is contained in a cluster node.” Fig. 2 shows an example of the clustering method. This shows that the variable nodes \( x_4, x_6 \) and \( x_7 \) are clustered on the FG in Fig. 1.

B. Notation for the Clustering Method

We define new sets for the clustering method. Let \( k(1 \leq k \leq N_c) \) be an index of each cluster node, where \( N_c \) is the number of cluster nodes in a CFG. We describe a cluster node with an index \( k \) as a “cluster node \( c_k \).” The variable nodes contained in cluster node \( c_k \) are denoted by

\[
C_k := \{n|x_n \text{ is contained in } c_k(1 \leq n \leq N)\}. \quad (2)
\]

By procedures for the clustering method, we have \( C_i \cap C_j = \phi \), where \( i \neq j \). A cluster node \( c_k \) takes on elements of \( A_{c_k} \), where

\[
A_{c_k} := \bigotimes_{n \in C_k} A_{x_n}, \quad (3)
\]

(\( \bigotimes \) implies direct product; it is ordered by \( C_k \)). For simply descriptions, we equate a variable node which is not clustered with a cluster node which contains one variable node, that is \(|C_k| = 1\). For example in Fig. 2, we define cluster nodes as follows,

\[
C_1 = \{4, 6, 7\}, \quad (4)
\]

\[
C_2 = \{1\}, C_3 = \{2\}, C_4 = \{5\}, C_5 = \{3\}. \quad (5)
\]

Let \( M'(k) \) be the set of all check nodes which are connected to the cluster node \( c_k \). We have

\[
M'(k) = \bigcup_{l=1}^{\left|C_k\right|} M(C_{kl}). \quad (6)
\]

Let \( N'(m) \) be a modified set of \( N(m) \) by the clustering method. \( N'(m) \) has indexes of cluster nodes. For example in Fig. 2, we have \( N'(1) = \{1, 2, 4\}, N'(2) = \{1, 3\}, N'(3) = \{1, 4, 5\} \).

The check node \( h_1 \) is connected to the variable node \( x_4 \) and \( x_7 \) in Fig. 1; however, it is connected to the cluster node \( \{x_4, x_6, x_7\} \) in Fig. 2. With the notice of only the CFG in Fig.2, we confuse “the check node \( h_1 \) is connected to only the variable node \( x_4 \) and \( x_7 \)” with “the check node \( h_1 \) is connected to the variable node \( x_4, x_6 \) and \( x_7 \).” To solve this problem, we define the set \( D_{km} \) that is defined between a cluster node \( c_k \) and each check node which is connected to \( c_k \), and it is given by

\[
D_{km} := \{n|n \in C_k \cap N(m)\}. \quad (7)
\]

For example in Fig. 2, we have \( D_{11} = \{4, 7\} \) from \( N(1) = \{1, 4, 7\} \) and Eq.(4).

---

\(^1\)Check nodes can be clustered as well. However, we don’t deal with them in this paper.

\(^2\)“Cluster Graph [9]” and “Extended Junction Graph [10]” are graphical models that can deal with plural variable nodes as cluster nodes. A CFG is a subclass of them because each variable node is contained in only one cluster node in the model.
C. Characteristics of a CFG and the Clustering Method

Pearl showed that the clustering method could erase independent loops in a graphical model. First, we attempt to erase all loops in the FG that represent any LDPC codes. It is meaningless because such the CFG generally has only one cluster node which contains all variable nodes in the original FG. Thus, we henceforth consider to erase only some small loops in the FG. In addition, the length of a loop is decreased if the collapsed nodes are in the loop. We need to investigate which loops should be clustered.

IV. A Decoding Algorithm on a CFG

A. The Sum-Product Algorithm on a CFG

The SPA calculates local joint probabilities of each elements of $A_{ck}$ of the cluster node $c_k$. Thus, we expect to improve the performance of the SPA by the clustering method. If $|D_{km}| \neq |C_k|$, a message from a cluster node $c_k$ to a check node $h_m$ has elements which each alphabet is $\otimes_{a \in D_{km}} A_{xm}$. For calculate this message with an alphabet $a$, we need to sum the values of the local joint probabilities over the all elements that bits of the alphabet corresponding to each index in $D_{km}$ is $a$. This operation is called a marginalization.

For the SPA, the number of elements of messages between a cluster node and the neighbor check nodes generally equals the number of elements on which the cluster node takes. Thus, the computational complexity of the SPA on a CFG is larger than those of the SPA on the original FG. However, assuming that using any binary parity check codes, we prepare only two elements of messages “the even-message” and “the odd-message”. For the CFG that represent any binary parity check codes, all check nodes in the CFG are constraints of parity check. This constraint implies that the sum of the values of the neighbor variable nodes is zero modulo 2. If some of variable nodes in them are collapsed, we require only the following information to verify the constraint. That is, “the sum of them is zero modulo 2” or “the sum of them is one modulo 2”.

For example in Fig. 1 and Fig. 2, the check node $h_1$ implies

$$X_1 + X_4 + X_5 + X_7 = 0 \mod 2.$$  

According to the above discussions, the minimum requisite information of messages between $h_1$ and $c_1$ are not $(X_4 = 0, X_7 = 0), (X_4 = 0, X_7 = 1), (X_4 = 1, X_7 = 0)$ and $(X_4 = 1, X_7 = 1)$ but $(X_4 + X_7 = 0 \mod 2)$ and $(X_4 + X_7 = 1 \mod 2)$. For these reasons, the Horizontal step of the SPA on a CFG are same computations of the SPA on a FG. On the other hand, we entail the process of calculation each element of cluster nodes and the process of a marginalization on the Vertical step. We show these logics as the proposed algorithm to be hereinafter described.

B. For the Binary Erasure Channel

In this subsection, we consider the communication over the binary erasure channel (BEC). If erasures occur on all bits which are contained in any stopping sets [12], the SPA cannot decode by the received sequence. By the clustering method, we can prevent such a decoding error that is occurred by stopping sets. This is due to the equivalent of “a calculation of local joint probabilities” and “solving of a simultaneous equation”.

For example, we deal with the $(7, 4)$ Humming code which parity check matrix is Eq. (1). Assuming that a received sequence is $\{Y_1 = 0, Y_2 = 0, Y_3 = 0, Y_4 = e, Y_5 = 0, Y_6 = e, Y_7 = e\}$, where $e$ implies an erasure. Since $\{x_4, x_6, x_7\}$ are a stopping set, the SPA on the FG cannot decode it. That is, all messages from each check node to the variable nodes $x_4, x_6$ and $x_7$ are always an erasure. On the SPA on the CFG, the messages from each check node to the cluster node $c_1$ are

$$X_4 + X_7 = 0 \mod 2, \quad (9)$$
$$X_4 + X_6 + X_7 = 0 \mod 2, \quad (10)$$
$$X_6 + X_7 = 0 \mod 2. \quad (11)$$

These are a simultaneous equation, and we can solve these and obtain $X_4 = X_6 = X_7 = 0$. If certain conditions are satisfied, we improve the performance of the SPA for the BEC to collapse some of variable nodes in a stopping set by the clustering method. We describe the conditions in appendix.

C. The Proposed Decoding Algorithm

In this subsection, we propose an efficient decoding algorithm on a CFG based on the above descriptions. We descriot the algorithm as like the SPA in [3]. For the BEC, we can define the similar efficient algorithm to refer [12]. In this case, we can replace the routines in the Vertical step by solution algorithms of a simultaneous equation.

Let $q^a_{km}$ and $r^a_{mk}$ be messages between the check node $h_m$ and the cluster node $c_k$, where $a$ is a type of messages, “even” or “odd”. Let $\lambda(A_{ck})$ be the function that returns all alphabets in $A_{ck}$. And let $\lambda_e(A_{ck}, \{x_i\})$ be the function that returns all alphabets in $A_{ck}$, where the bits corresponding to $\{x_i\}$ have even weight ($\{x_i\}$ implies any sets of an index of variable nodes). Let $\lambda_o(A_{ck}, \{x_i\})$ be the function that returns all elements in $A_{ck}$, where the bits corresponding to $\{x_i\}$ have odd weight. Consequently, we have

$$\lambda(A_{ck}) = \lambda_e(A_{ck}, \{x_i\}) \cap \lambda_o(A_{ck}, \{x_i\}). \quad (12)$$

For example in Fig. 2, we have

$$\lambda_e(A_{c_1}, D_{11}) = \{000, 010, 101, 111\}, \quad (13)$$
$$\lambda_o(A_{c_1}, D_{11}) = \{001, 100, 011, 110\}. \quad (14)$$

(Notice that $x_1$ corresponds to the first bit of $A_{c_1}$; $x_7$ corresponds to the third bit of $A_{c_1}$). We define $t_j$ as the $j$th bit of a bit sequence $t$. $\alpha$ is a normalization constant.

On the Initialization and the Horizontal step, compute through each pair of $(m,k)$, where $k \in N^2(m)$ for all $m(1 \leq m \leq M)$:

3We can, obviously, compute this by a probabilistic calculation.
Initialization.

\[
\begin{align*}
q_{km}^{\text{even}} &= \sum_{t \in \lambda_e(A_{ck}, C_k)} \prod_{l=1}^{C_k} f_{C_k}^t, \\
q_{km}^{\text{odd}} &= \sum_{t \in \lambda_o(A_{ck}, C_k)} \prod_{l=1}^{C_k} f_{C_k}^t
\end{align*}
\]

Horizontal step.

\[
\begin{align*}
r_{mk}^{\text{even}} &= ((1 + \delta r_{mk})/2), \\
r_{mk}^{\text{odd}} &= ((1 - \delta r_{mk})/2),
\end{align*}
\]

where

\[
\delta r_{mk} = \prod_{k' \in N'(m) \setminus k} (q_{k'm}^{\text{even}} - q_{k'm}^{\text{odd}}).
\]

(*) If \(|N'(m)| = 1\), we define \(r_{mk}^{\text{even}} = 1\) and \(r_{mk}^{\text{odd}} = 0\).

Vertical step.

Compute for all \(t \in \lambda(A_{ck})\),

\[
Q_k = \prod_{t=1}^{C_k} f_{C_k}^t \prod_{m \in M'(n)} r_{mk}^{a_{km}(t)},
\]

where

\[
a_{km}(t) = \begin{cases} 
\text{"even"} & t \in \lambda_e(A_{ck}, D_{km}), \\
\text{"odd"} & \text{other}.
\end{cases}
\]

And compute for each \(m \in M'(k)\),

\[
\begin{align*}
q_{km}^{\text{even}} &= \alpha (\sum_{t \in \lambda_e(A_{ck}, D_{km})} Q_k^t) / r_{mk} \\
q_{km}^{\text{odd}} &= \alpha (\sum_{t \in \lambda_o(A_{ck}, D_{km})} Q_k^t) / r_{mk}
\end{align*}
\]

Pseudo-posterior probabilities.

Compute for all \(n\), where \(n \in C_k\),

\[
\begin{align*}
\phi_n^{0} &= \alpha \sum_{t \in \lambda_e(A_{ck}, \{n\})} Q_k^t \\
\phi_n^{1} &= \alpha \sum_{t \in \lambda_o(A_{ck}, \{n\})} Q_k^t
\end{align*}
\]

V. NUMERICAL EXPERIMENTS

In this section, we examine the performance of a decoding on the CFG by simulations. We examine MacKay’s regular LDPC code\(^4\) with a code length of 1008, column weight of 3, and row weight of 6. The rate of the code is 0.5, and it has no length-4 loops. We consider collapsing variable nodes that is contained in length-6 loops in the FG that represent this code. The number of length-6 loops of this FG is 165. We decide loops that are clustered to find the length-6 loops that satisfy the conditions in appendix. The CFG that is constructed by this algorithm has no length-4 loops. By this clustering algorithm, we cluster 98 length-6 loops in this FG; however the number of length-6 loops in the made-up CFG is 191. This is caused that length-8 loops which have collapsed nodes transform into length-6 loops. Using these graphs, we compare the SPA with the proposed decoding algorithm (denoted by “CSP”). We evaluate them by a word error (erasure) rate, a bit error (erasure) rate and a computational complexity. For each algorithm, \(2 \times 10^4\) symbols are transmitted over them. The maximum number of iterations is 100 for each algorithm.

In Fig. 3, we show the result of the decoding of both algorithms over the AWGN and the result of them over the BEC in Fig. 4. The computational complexity of both algorithms are \(O(n)\), if \(|N(m)|\), \(|M(n)|\) and \(|C_k|\) are constant for all \(m\), \(n\) and \(k\). Hence, we count the number of the arithmetic operation (addition and multiplication) in one iteration of each algorithm. We show them in Table I. In Table II, we show them in specific figures for the decoding of the above code.

\(^4\)http://www.inference.phy.cam.ac.uk/mackay/codes/504.504.3.504

![Fig. 3. Simulation result over the AWGN.](image1)

![Fig. 4. Simulation result over the BEC.](image2)

<table>
<thead>
<tr>
<th>V. N. NUMERICAL EXPERIMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE II</td>
</tr>
<tr>
<td>-------------------------------</td>
</tr>
<tr>
<td>THE NUMBER OF ARITHMETIC OPERATION (SPECIFIC FIGURES)</td>
</tr>
<tr>
<td>Horizontal Step.</td>
</tr>
<tr>
<td>Vertical Step.</td>
</tr>
<tr>
<td>PP.</td>
</tr>
<tr>
<td>total</td>
</tr>
</tbody>
</table>

\(^4\)http://www.inference.phy.cam.ac.uk/mackay/codes/504.504.3.504
VI. DISCUSSION

Fig. 4 illustrates that the decoding performance of the proposed algorithm over the BEC is better than that of the SPA. We think that this is attributed to the decrement of the decoding error that is occurred by stopping sets. On the other hand, the decoding performance of the proposed algorithm over the AWGN is a little better than that of the SPA, as shown in Fig. 3. Notice that the decoding performance with the clustering method is (a little) better than that without the clustering method, although the number of length-6 loops is increased by the clustering method. We may consider that the number of small loops in a FG and the performance of the SPA are not simple proportionality relations.

Table I and Table II illustrate that the number of arithmetic operations of the proposed algorithm is not so enormous as compared with the SPA if $|C_k|$ are small constant numbers for all $k$. For an implementation, it is more complicate because of a handling of $\lambda_e, \lambda_o$ and $a_k$. Furthermore, the required memory size of the proposed algorithm is more than that of the SPA for the memory of $C_k$ and $D_{km}$.

VII. CONCLUDING REMARKS

In this paper, we show several examples about a decoding on a FG to which the clustering method is applied. We also propose an efficient decoding algorithm for it. In this paper, we use the existing LDPC code, and the clustering algorithm is not so reflected. For the future, we should investigate a good class of codes that is suited to the clustering method. Furthermore, a decoding failure of LDPC codes over the BEC is evaluated exactly for the SPA [12]. We should evaluate it for the proposed algorithm.

ACKNOWLEDGMENT

The authors would like to acknowledge entire member of Hirasawa Lab. and Matsushima Lab. for their helpful suggestions to this work. This research is partially supported by Category No.15560338 of Grant-in-Aid for Scientific Research, Japan Society for the Promotion of Science.

REFERENCES


APPENDIX

If a loop in a FG satisfied following conditions then the SPA can determine the variable node $x^*$ that is contained in a stopping set by clustering the loop. Let $L$ be variable nodes in a loop. Let $S$ be variable nodes in a stopping set.

Conditions.
- $L$ is a subset of $S$.
- All check nodes that are connected to $L$ are only connected to following variable nodes: 1) Variable nodes in $L$, 2) Only one variable node $x^*$ in $S$ and not in $L$, 3) Variable nodes not in $S$ and not in $L$.

TABLE I

<table>
<thead>
<tr>
<th>SPA</th>
<th>CSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal Step.</td>
<td>$\sum_{m=1}^{M}</td>
</tr>
<tr>
<td>Vertical Step.</td>
<td>$\sum_{m=1}^{M} (\sum_{n \in N(m)} (2</td>
</tr>
<tr>
<td>P.P.</td>
<td>$(5N) (*)$</td>
</tr>
</tbody>
</table>

(*) This formula is derived if the SPA uses the pre-normalization values on the Vertical Step.