Variable-to-Check Residual Belief Propagation for LDPC codes

Jung-Hyun Kim

The Graduate School

Yonsei University

School of Electrical and Electronic Engineering

Variable-to-Check Residual Belief Propagation for LDPC codes

Jung-Hyun Kim

A Dissertation Submitted to the

Graduate School of Yonsei University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Supervised by

Professor Hong-Yeop Song, Ph.D.

School of Electrical and Electronic Engineering
The Graduate School

YONSEI University

June 2008

This certifies that the thesis of
Jung-Hyun Kim is approved.

Thesis Supervisor: Hong-Yeop Song

Kwang Soon Kim

Sooyong Choi

The Graduate School
Yonsei University
June 2008

감사의 글

설렘과 즐거움으로 가득했던 지난 2년간의 시간들을 되돌아보며 부족한 저를 이끌어주신 많은 분들께 감사의 말씀을 전합니다.

무엇보다 지금의 저를 있게 해주신 주님께 감사 드립니다. 항상 저의 든든한 후원자가 되어주시는 아버지, 언제나 사랑과 믿음으로 보살펴주시는 어머니, 보다 높은 곳을 향해 나아갈 수 있도록 이끌어주는 누님, 제게 커다란 힘이 되어주시는 외할머니, 항상 응원해주시는 큰이모, 작은이모, 그리고 보이지 않는 곳에서 도움을 주신 친척분들께 감사 드립니다.

학문에 임하는 자세와 열정을 가르쳐주신 존경하는 송홍엽 교수님, 아낌없는 조언과 격려로 보다 나은 논문이 될 수 있도록 지도해주신 김광순 교수님과 최수용 교수님께도 깊은 감사를 드립니다.

과분한 관심과 기도로 제게 힘을 주시는 한빛교회 가족들과 참빛교회 가족들, 고민이 있을 때마다 의지가 되어준 소수정예 친구들, 그리고 함께 동고동락한 연구실 선후배님들께 감사 드립니다.

마지막으로 제가 사랑하고 존경하는 모든 분들과 저를 아껴주고 믿어주는 모든 분들께 감사의 마음과 함께 그 동안의 결실이 담긴 이 논문을 바칩니다.

2008년 6월

김 정 현 드림

Contents

List of Figures

List of Tables

ABSTRACT

Variable-to-Check Residual Belief Propagation for LDPC Codes

Jung-Hyun Kim
School of Electrical
and Electronic Eng.
The Graduate School
Yonsei University

We propose Variable-to-Check Residual Belief Propagation (VCRBP) decoding for low-density parity-check (LDPC) codes. The proposed method is a dynamic scheduling belief propagation algorithm that updates first the variable-to-check message which has the largest residual. The residual value is calculated from the difference of message before and after update. VCRBP guarantees better error correcting performance by using more reliable residual compared to residual belief propagation (RBP) recently applied to LDPC decoding by Vila Casado *et al.* Moreover, VCRBP reduces decoding complexity by removing unnecessary ordering and additional step of RBP. Also, we propose node-wise VCRBP (N-VCRBP) decoding that updates simultaneously all outgoing messages from variable node corresponding to the largest residual. By doing so, N-VCRBP decoding can obtain very close error correcting performance with lower decoding complexity compared to VCRBP decoding. For reducing the complexity, we present two kinds of methods for both RBP and VCRBP decoding, named forced convergence-based RBP/VCRBP (FC-RBP/VCRBP) decoding and sign-based RBP/VCRBP (S-RBP/VCRBP) decoding. Both of them significantly reduce decoding complexity with only negligible deterioration in error correcting performance. FC-RBP/VCRBP decoding reduces the complexity by skipping update of convergent messages. A message converge or not can be determined by the magnitude of its residual. S-RBP/VCRBP decoding reduces the complexity by using a new ordering measure based on the change of sign. This method is a very simple version of RBP/VCRBP decoding because it is likely that the message which has the largest residual change its sign after update. Simulation results show clear advantages of proposed methods. Specially, VCRBP decoding out-performs with only maximum 8 iterations by about 0.3 dB compared to RBP decoding at an FER of 10^{-4}. All of the proposed methods have very fast decoding convergence speed in terms of the number of iterations and they are more effective in the codes with short block length and containing small stopping sets.

A second major theme of this thesis is the design of joint low-density parity-check (LDPC) codes for multi-user relay channel. The joint LDPC code is a channel-network coding method which guarantee high level of performance and achieved service quality. In this thesis, we describe how to apply proposed model to relay channel and how to optimize. Additionally, we show how efficient joint LDPC code method using intelligent relay is in wireless ad-hoc networks.

Key words : Low-Density Parity-Check (LDPC) codes, residual belief propagation (RBP) algorithm, fast convergence decoding, informed dynamic scheduling (IDS) decoder, channel-network coding, cooperative relay

Chapter 1

Introduction

1.1 Motivation

1.1.1 VCRBP Decoding for LDPC Codes

Next generation communication systems require faster and more accurate channel decoders due to very high data transmission rates, e.g. 4G wireless communications. By that reason, a number of decoding algorithms have been developed to cope with the requirements. Particularly in low-density parity-check (LDPC) codes [1], belief propagation (BP) algorithm have been extensively studied for faster decoding convergence of LDPC codes. The most typical method is the serial scheduled decoding. The serial scheduled decoding updates toward variable nodes or check nodes in serial manner. Specially, shuffled belief propagation (SBP) decoding [2] updates toward variable node and layered belief propagation (LBP) decoding [3] updates toward check node. Theoretical analysis and Monte-Carlo simulation results show that serial scheduled decoding converges twice faster than, so called, flooding, which updates all the nodes simultaneously. Since SBP decoding and LBP decoding were introduced, various improvement on them have been tried, some of which are replica SBP decoding [4], row-column message passing scheduling decoding [6], joint row-column decoding [7] and edge-based scheduled BP decoding [8].

Recently, more effective serial scheduling method was introduced using dynamic scheduling method. This dynamic scheduling method, called residual belief propagation (RBP) [9], accelerates the decoding convergence faster than non-dynamic serial scheduling in terms of the number of iterations [10], [11]. However, RBP decoding has some shortcomings both in the error correcting performance (because of its greediness [10], [11]) and in decoding complexity yet.

In this thesis, we propose an advanced algorithm overcoming the inadequacy of RBP decoding, named variable-to-check residual belief propagation (VCRBP) decoding. VCRBP decoding calculates the residual from the difference of variable-to-check message before and after update, while RBP decoding in [10] and [11] calculates the residual from the difference of check-to-variable message before and after update. It is interesting to note that only the above difference in the original RBP and the proposed one here results in some non-trivial improvement on the error correcting performance with only a small number of iterations. Reducing the complexity follows easily by reducing the number of ordering processes of the residuals and by avoiding unnecessary computations.

Also, we propose node-wise VCRBP (N-VCRBP) decoding that updates all outgoing messages from variable node corresponding to the largest residual. By doing so, N-VCRBP decoding can obtain very close error correcting performance with lower decoding complexity compared to VCRBP decoding. In addition, we can reduce the complexity more by removing edges from candidate set to avoid useless updates.

For reducing the complexity, we present two kinds of methods for both RBP and VCRBP decoding, named forced convergence-based RBP/VCRBP (FC-RBP/VCRBP) decoding and sign-based RBP/VCRBP (S-RBP/VCRBP) decoding. FC-RBP/VCRBP decoding reduces decoding complexity by exploiting the fact that a large number of variable nodes converge to a strong belief after very few iterations, i.e., these bits have already been reliably decoded and we can skip updating their belief in subsequent iterations. S-RBP/VCRBP decoding selectively chooses an message based on change of its sign instead of magnitude of its residual. We provide experimental evidences that, with very close error correcting performance, the S-RBP/VCRBP decoding can improve decoding complexity that is significantly better than that of both RBP and VCRBP decoding.

1.1.2 Joint LDPC Codes for Multi-User Relay Channel

Recently, how to achieve the high data transmission rate of next generation wireless communication systems in cellular architecture is also one of the main research topics. With the existing cellular architecture, the goal cannot be realized over a wide coverage area. Therefore, various novel cellular network architectures have been studied. The multi-hop relaying concept would be considered as one of the solutions. Adding relays in the cell reduces the signal transmission distances, resulting in lower propagation loss and higher average signal-to-noise ratio (SNR) to the mobile user. Cellular relay network [24] is generalized form of the relaying concept. The basic idea of cellular relay network is that the large number of relays cover the whole cell area and each relay serves a small area with a small amount of power. Because each relay needs low power, channel could be reused when its interference is not severe. Therefore, the system performance can be improved by both reduced propagation loss and reuse of channels.

For fundamental understanding of role of relays in networks, a basic structure with one relay has been studied in various points of view such as space time code (STC) [25], multi-input multi-output (MIMO) [26], and error correcting code (ECC). Among these, we are specially interested in applying ECC to a relay network. Turbo codes [27] and LDPC codes [28] provide powerful performance improvement of potential error correcting ability. LDPC codes are considered to be more efficient than Turbo codes for future wireless systems with high speed communication due to lower decoding complexity. The most typical method of LDPC codes for the relay channel, bilayer LDPC codes [29], [30] have simple structure as well as good performance.

For the next generation communication systems, cooperation is another hot issue. User cooperative model is one of the examples of network coding. And it is possible to use a modulo 2 addition of two user data [31], [32] or multi-user data [33], [34], [35], [36].

In this thesis, we propose joint LDPC codes which combine channel coding for relay channel and user cooperation. Furthermore, we apply joint LDPC codes into ad-hoc network model. This application not only guarantees achieved service quality but also has good performance.

1.2 Overview

The remainder of this thesis is organized as follows. In Chapter 2, we propose VCRBP decoding for LDPC codes and its modified methods, node-wise VCRBP, FC-RBP/VCRBP and S-RBP/VCRBP. In Chapter 3, we propose joint LDPC codes for multi-user relay channel and introduce its application models. Finally, we summarize this thesis in Chapter 4.

Chapter 2

Variable-to-Check RBP (VCRBP) Decoding for LDPC Codes

2.1 Residual Belief Propagation (RBP) Decoding for LDPC Codes

2.1.1 Belief Propagation Decoding for LDPC Codes

LDPC codes decoding can be effectively performed using the BP algorithm on a code graph. The LDPC code graph is a bipartite graph composed of *N* variable nodes v_j for $j \in 1, ..., N$ that represent the codeword bits and M check nodes c_i for $i \in 1, ..., M$ that represent the parity check equations. Each node generates and propagates messages to its neighbors based on its current incoming messages. Here, the ”neighbor” is any node on the other side of the bipartite graph connected by an edge. By this exchange of messages, BP decoding is achieved. The exchanged messages correspond to the log-likelihood ratio (LLR) of the probabilities of the bits. The sign of the LLR indicates the most likely value of the bit and the absolute value of the LLR gives the reliability of the message. In this fashion, the generating functions of two messages, variable-to-check message and check-to-variable message, are defined as [1]:

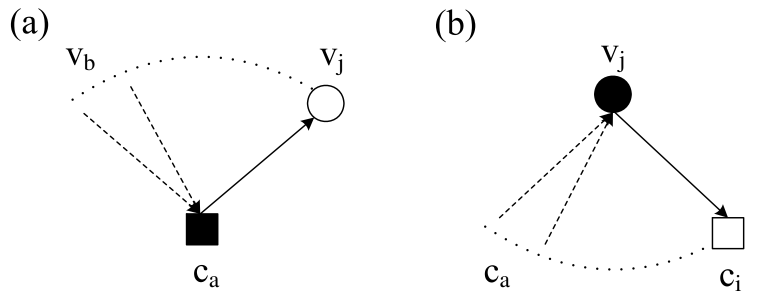


Figure 2.1: Message passing of BP decoding

$$m_{c_a \to v_j} = \log \frac{P(E_{c_a} = 0 | v_j = 0, \mathbf{r})}{P(E_{c_a} = 0 | v_j = 1, \mathbf{r})}, \tag{2.1}$$

$$m_{v_j \to c_i} = \log \frac{P(v_j = 0 | \mathbf{r}, \{E_{c_a} = 0, c_a \in N(v_j) \backslash c_i\})}{P(v_j = 1 | \mathbf{r}, \{E_{c_a} = 0, c_a \in N(v_j) \backslash c_i\})}, \tag{2.2}$$

where **r** is the received signal vector and $E_{c_a} = \sum_{v_n \in N(c_a)} v_n$ is the check equation. The iterative relation between above equation 2.1 and equation 2.2 is derived in **Appendix**.

In general, iterative procedure of BP decoding is composed of two steps. At the ith iteration of BP decoding, in the first step, all values of the check-to-variable messages are updated by using the values of the variable-to-check messages obtained at the $(i-1)$th iteration (see Fig. 2.1-(a)). In the second step, all values of the variable-to-check messages are updated by using the values of the check-to-variable messages newly obtained at the ith iteration (see Fig. 2.1-(b)). This iterative process continues until the stopping rule is satisfied.

2.1.2 Residual Belief Propagation (RBP) Algorithm

RBP algorithm is a dynamic scheduling strategy that updates first the message that has the largest ordering measure called the residual. The residual of message is the magnitude of difference between its current value and its newly computed value. For a message m_k, the residual is defined as [9], [10]:

$$r(m_k) = \|m_k^* - m_k\|, \tag{2.3}$$

where m_k^* is the newly computed m_k by some updating function.

The intuitive justification of this approach is that the differences between the messages before and after an update go to zero as loopy BP converges. By that reason, if a message has a large residual, it means that the message is located in a part of the graph that hasn't converged yet. Therefore, propagating the message which has the largest residual first should speed up the process [10].

2.1.3 RBP Decoding for LDPC Codes

RBP decoding for LDPC codes be proposed in [10] exploits the dynamic selection of the message which has the largest residual, whenever messages are updated serially. By doing so, RBP can perform decoding with very fast decoding convergence. This algorithm can be explained shortly in three steps. Following the notations in [10], let $m_{c_i \to v_j}$ be the message having the largest residual, so it is chosen for the update. In the first step, RBP updates $m_{c_i \to v_j}$, then discards the residual (otherwise set $r(m_{c_i \to v_j}) = 0$) (see Fig. 2.2-(a)). In the second step, it computes $m_{v_j \to c_a}$ with $c_a \in N(v_j) \backslash c_i$ (see

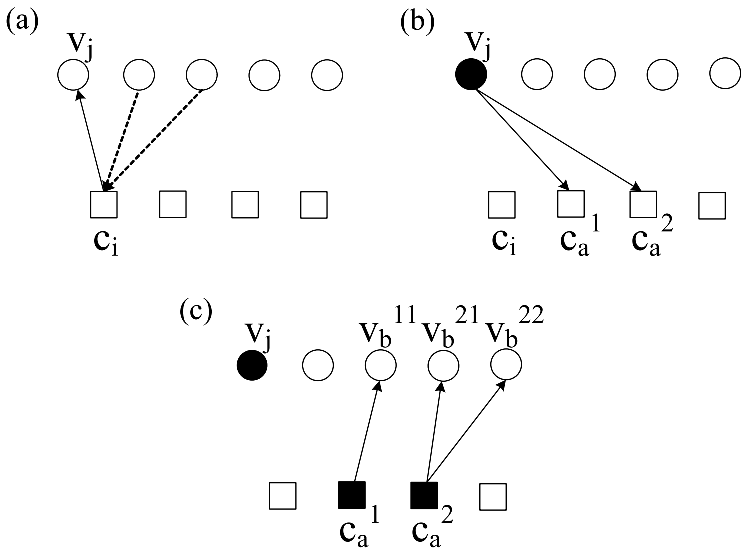


Figure 2.2: The procedure of RBP decoding for LDPC codes

Fig. 2.2-(b)). In the third step, it computes $r(m_{c_a \to v_b})$ with $v_b \in N(c_a) \backslash v_j$ and re-order the messages that are not updated yet in the order of the magnitude of residuals (see Fig. 2.2-(c)).

Notice that in this process, messages chosen for the update are discarded, and hence, all check-to-variable messages are eventually chosen only once to be updated during one iteration. An exemplary algorithm of RBP in pseudo-codes is stated in **Algorithm 1**.

Algorithm 1 RBP decoding for LDPC codes [10]

```
1: Initialize all $m_{c\rightarrow v} = 0$
2: Initialize all $m_{v_n\rightarrow c} = C_n$
3: Compute all $r(m_{c\rightarrow v})$ and generate $Q$
4: Let $m_{c_i\rightarrow v_j}$ be the first message in $Q$
5: Generate and propagate $m_{c_i\rightarrow v_j}$
6: Set $r(m_{c_i\rightarrow v_j}) = 0$ and re-order $Q$
7: for every $c_a \in N(v_j)\backslash c_i$ do
8:     Generate and propagate $m_{v_j\rightarrow c_a}$
9:     for every $v_b \in N(c_a)\backslash v_j$ do
10:        Compute $r(m_{c_a\rightarrow v_b})$ and re-order $Q$
11:    end for
12: end for
13: if Stopping rule is not satisfied then
14:    Go back to line 4;
15: end if
```

RBP decoding for LDPC codes in [10], [11] has fast decoding convergence as well as better error correcting performance than non-dynamic scheduling by solving the trapping set problem (see Fig. 2.3). However, RBP decoding for LDPC codes in [10], [11] has some margins of improvement in both error correcting performance and decoding complexity. Aspect of error correcting performance, RBP decoding generates and propagates new errors, because of its greediness, which do not appear in non-dynamic scheduled decoding [10]. Aspect of decoding complexity, We specifically try to concentrate on Lines 5 and 10 in **Algorithm 1**. When the chosen check-to-variable message is updated, $m_{c\rightarrow v}$ is unnecessarily recomputed because it has been already calculated when $r(m_{c\rightarrow v})$ is determined in the previous step. Conversely, $m_{c\rightarrow v}$ has to be computed additionally only for calculating residual values. Moreover, in Line 10, Q is re-ordered whenever residual value of each message is calculated. Here, Q denotes a set containing the information on the magnitude of all the remaining residual values.

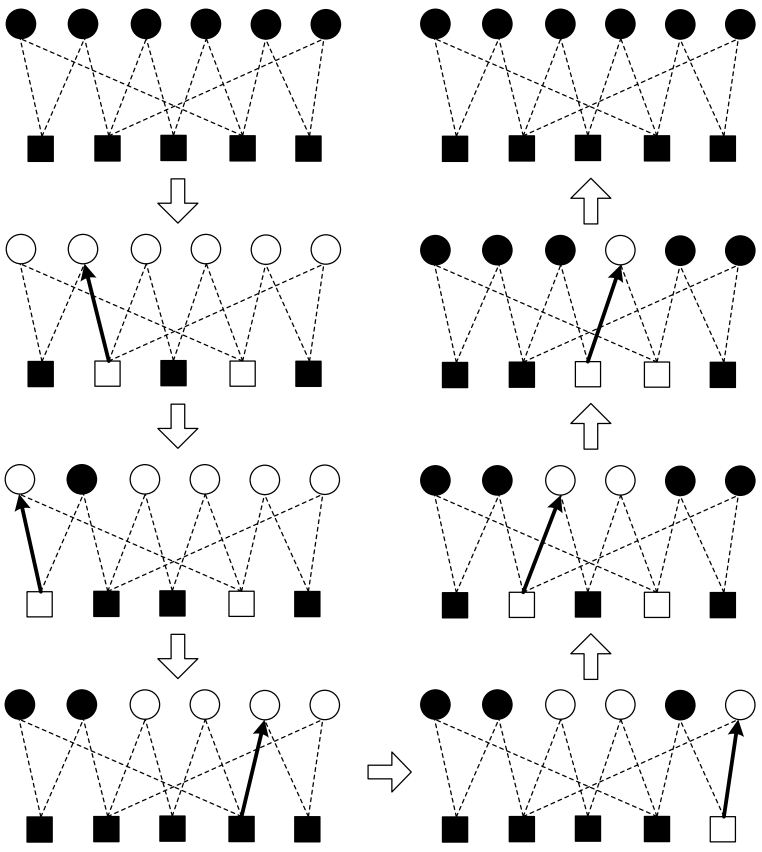


Figure 2.3: The process of solving the trapping set problem with RBP decoding; White circles and squares present variable nodes and check nodes in error respectively

2.1.4 Node-wise RBP Decoding for LDPC Codes

Node-wise RBP (N-RBP) is a less greedy algorithm than RBP for LDPC codes [10], [11]. It is suggested in order to obtain better error correcting performance for all types of errors that can be occurred when greedy scheduling strategy is used. As noted earlier, some of the greediness of RBP decoding came from the fact that it tends to propagate the chosen message based on only one check equation to less reliable variable nodes. So, N-RBP decoding simultaneously propagates all the check-to-variable messages corresponding to the chosen check node, instead of only propagating the message with the largest residual. By doing so, if the message in error is chosen with the largest residual, the error can be corrected by messages updating simultaneously to more reliable variable node. Therefore, N-RBP algorithm is less likely to generate new errors that do not occur with non-greedy algorithm.

We can also understand N-RBP decoding as a reduced complexity method of RBP decoding because it updates the chosen check node sequentially over all the check nodes instead of the chosen edge over all the edges. In this sense, N-RBP decoding is similar to LBP decoding. The main difference between LBP decoding and N-RBP decoding is the following: LBP decoding updates check nodes sequentially according to a predetermined schedule while N-RBP decoding selects the next check node to be updated based on the current state of the messages in the graph. Specifically N-RBP decoding selects the check node connected by the edge which has the largest residual. The exact algorithm is stated in **Algorithm 2**.

Algorithm 2 N-RBP decoding for LDPC codes [10]

```
1: Initialize all $m_{c \to v} = 0$
2: Initialize all $m_{v_j \to c_i} = C_j$
3: Compute all $r(m_{c \to v})$ and generate Q
4: Let $m_{c_i \to v_j}$ be the first message in Q
5: **for** every $v_k \in N(c_i)$ **do**
6:   Generate and propagate $m_{c_i \to v_k}$
7:   Set $r(m_{c_i \to v_k}) = 0$ and re-order Q
8:   **for** every $c_a \in N(v_k) \backslash c_i$ **do**
9:     Generate and propagate $m_{v_k \to c_a}$
10:    **for** every $v_b \in N(c_a) \backslash v_k$ **do**
11:      Compute $r(m_{c_a \to v_b})$ and re-order Q
12:    end **for**
13:  end **for**
14: end **for**
15: **if** Stopping rule is not satisfied **then**
16:   Go back to line 4;
17: end **if**
```

N-RBP decoding has very close error correcting performance to RBP decoding with lower decoding complexity. However, N-RBP decoding can not guarantee better performance for a small number of iterations than RBP decoding. Moreover, N-RBP decoding updates some ”waste” messages (see Fig. 2.4). In Fig. 2.4-(d), if $m_{c_i^* \to v_j^*}$ has the largest residual and c_i^* is chosen for the next update, then $m_{c_i^* \to v_k^{*1}}$ and $m_{c_i^* \to v_k^{*2}}$ are propagated to where the messages come from. So, this propagation is useless update. We call this message to ”waste” message in this thesis. In fact, there is no loss of error correcting performance whether ”waste” messages are propagated or not. Figure 2.5 confirms this fact.

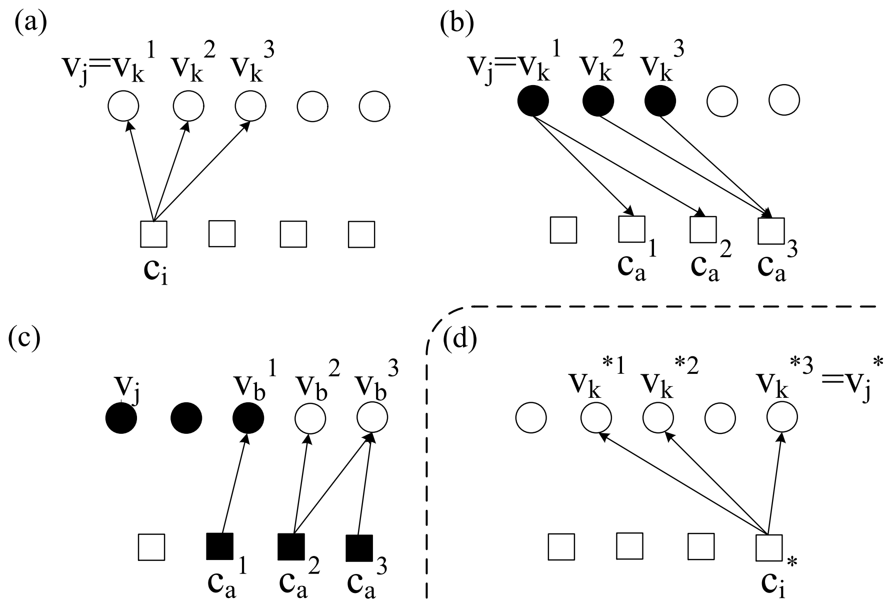

(a)
$v_j = v_k^1$ v_k^2 v_k^3
c_i
(b)
$v_j = v_k^1$ v_k^2 v_k^3
c_a^1 c_a^2 c_a^3
(c)
v_j v_b^1 v_b^2 v_b^3
c_a^1 c_a^2 c_a^3
(d)
v_k^{*1} v_k^{*2} $v_k^{*3} = v_j^*$
c_i^*

Figure 2.4: The procedure of N-RBP decoding for LDPC codes

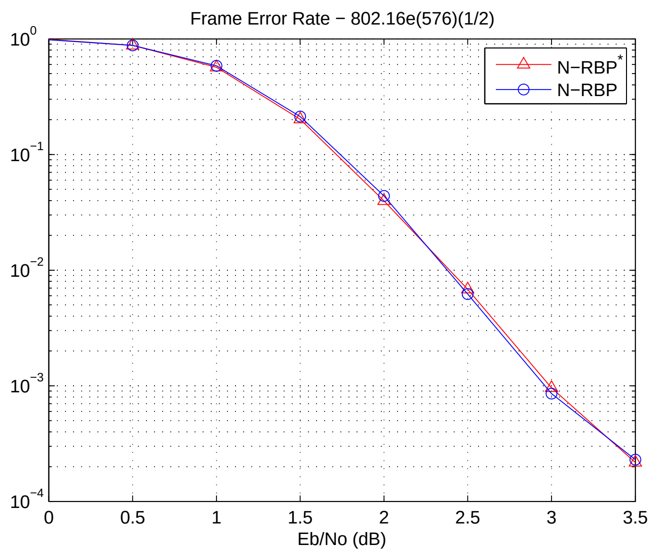


Figure 2.5: FER performance comparison of N-RBP and N-RBP* (removed waste update) decoding using IEEE 802.16e block length-576 rate-1/2 code with at most 8 iterations

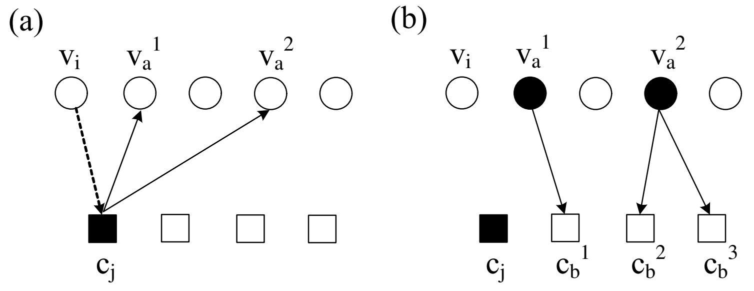


Figure 2.6: The procedure of VCRBP decoding for LDPC codes

2.2 Variable-to-Check RBP (VCRBP) Decoding for LDPC Codes

2.2.1 VCRBP Decoding for LDPC Codes

VCRBP is an advanced algorithm for overcoming the negative effect caused by the greediness of RBP. This method has not only better error correcting performance but also lower decoding complexity than RBP decoding in [10] and [11]. Main difference between VCRBP decoding and RBP decoding is in the method of calculating residuals: VCRBP decoding calculates the residual from the difference of variable-to-check message values before and after update, in other words, it sequentially updates check node corresponding to the chosen edge. On the other hand, RBP decoding calculates the residual from the difference of check-to-variable message values before and after update, in other words, it sequentially updates variable node corresponding to the chosen edge.

Figure 2.6 shows one-step reduced (than RBP) decoding procedure of VCRBP. In the first step, VCRBP decoding finds the largest $r(m_{v\rightarrow c})$ for choosing correspondent edge. If the edge corresponding to $m_{v_i\rightarrow c_j}$ is chosen, it sets $r(m_{v_i\rightarrow c_j}) = 0$, and then updates $m_{c_j\rightarrow v_a}$ for all $v_a \in N(c_j)\backslash v_i$ (see Fig. 2.6-(a)). In the second step, it updates

$m_{v_a \to c_b}$ for all $c_b \in N(v_a)\backslash c_j$ for calculating the correspondent residual $r(m_{v_a \to c_b})$ (see Fig. 2.6-(b)). Actually, every $r(m_{v \to c})$ has the same value for any $c \in N(v)$, and also $r(m_{v \to c}) = r(m_{c^* \to v})$ where $c \in N(v)\backslash c^*$ and c^* is the check node connected chosen edge. Hence, we can also calculate residual $r(m_{c_j \to v_a})$ instead of $r(m_{v_a \to c_b})$ over all the $c_b \in N(v_a)\backslash c_j$, and then only choose an edge corresponding to the largest residual. This change can reduce the decoding complexity compared with RBP decoding [10]. The exact process is explained in **Algorithm A**.

Algorithm A VCRBP decoding for LDPC codes

```
1:  Initialize all $m_{c \to v} = 0$
2:  Initialize all $m_{v_n \to c} = C_n$
3:  Find the largest $r(m_{v \to c})$
4:  Let $m_{v_i \to c_j}$ be chosen
5:  Set $r(m_{v_i \to c_j}) = 0$
6:  for every $v_a \in N(c_j)\backslash v_i$ do
7:      Generate and propagate $m_{c_j \to v_a}$
8:      for every $c_b \in N(v_a)\backslash c_j$ do
9:          Generate and propagate $m_{v_a \to c_b}$
10:         Compute $r(m_{v_a \to c_b})$
11:     end for
12: end for
13: if Stopping rule is not satisfied then
14:     Go back to line 3;
15: end if
```

In error correcting performance aspect, VCRBP decoding guarantees better performance than RBP decoding. RBP decoding tends to propagate the chosen message which has the largest residual based on only one check equation to the less reliable variable nodes. This greediness of RBP decoding can generate new errors that need a large number of message updates to be corrected. However, VCRBP decoding propagates the chosen message which has the largest residual based on all corresponding check equations except only one. So, VCRBP is a less greedy algorithm because it uses more reliable message for the residual than RBP. Moreover, VCRBP decoding solves the trapping set problem more effectively than RBP decoding by updating the check node (based on the largest residual) and then simultaneously all the variable nodes representing the check equation (see Fig. 2.7). These facts result in much faster convergence as well as better performance than RBP decoding. Figure 2.8 compares the FER performance of BP, LBP, RBP, and VCRBP decoding using IEEE 802.16e block length-576 rate-1/2 code with at most 8 iterations. Two dynamic scheduled methods, RBP and VCRBP, outperform non-dynamic methods, BP and LBP in the high SNR as well as low SNR. Moreover, we can see that VCRBP decoding outperforms RBP decoding about 0.3 dB at an FER of 10^{-4}.

From now on, we consider error correcting performance of various decoding methods with only same number of iterations. However, it does not mean same decoding complexity even if they have same number of iterations. So, we additionally analyze and simulate about error correcting performance of BP, RBP and VCRBP with real same decoding complexity. Table 2.1 shows their approximation of decoding complexity. Each brief numerical expression denotes the number of computing $m_{c\to v}$ during one iteration. Here, d_v is average degree of variable nodes and d_c is average degree of check nodes. The result is reasonable because computing $m_{c\to v}$ takes almost part in decoding complexity of LDPC codes. Figure 2.9 and Figure 2.10 show that error correcting performance of BP, RBP and VCRBP with approximately same decoding complexity. We can see the fact that RBP does not guarantee better performance than BP with same decoding complexity. However, proposed VCRBP has significantly better performance with same decoding complexity as well as same number of iterations.

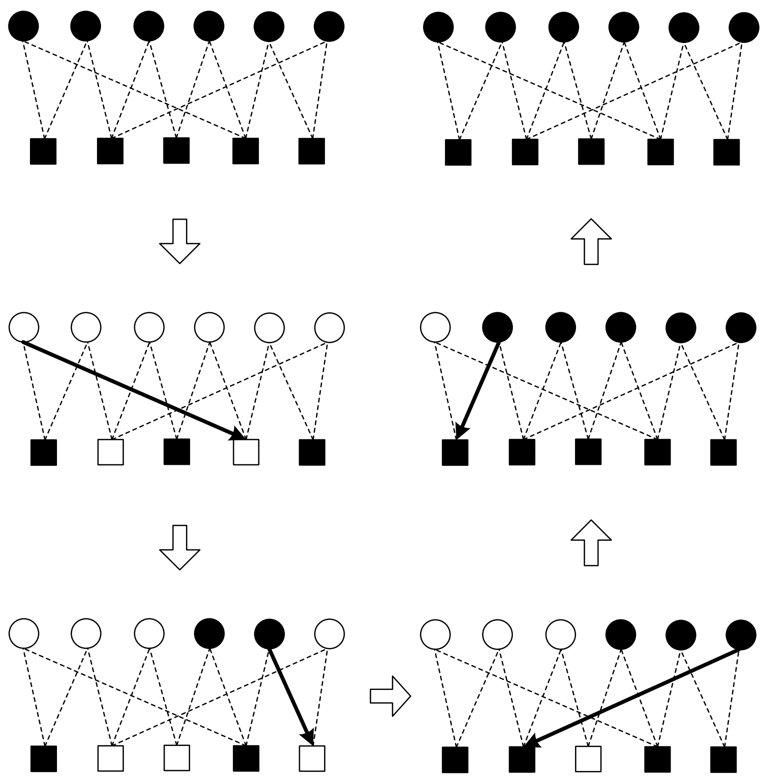


Figure 2.7: The process of solving the trapping set problem with VCRBP decoding; White circles and squares present variable nodes and check nodes in error respectively

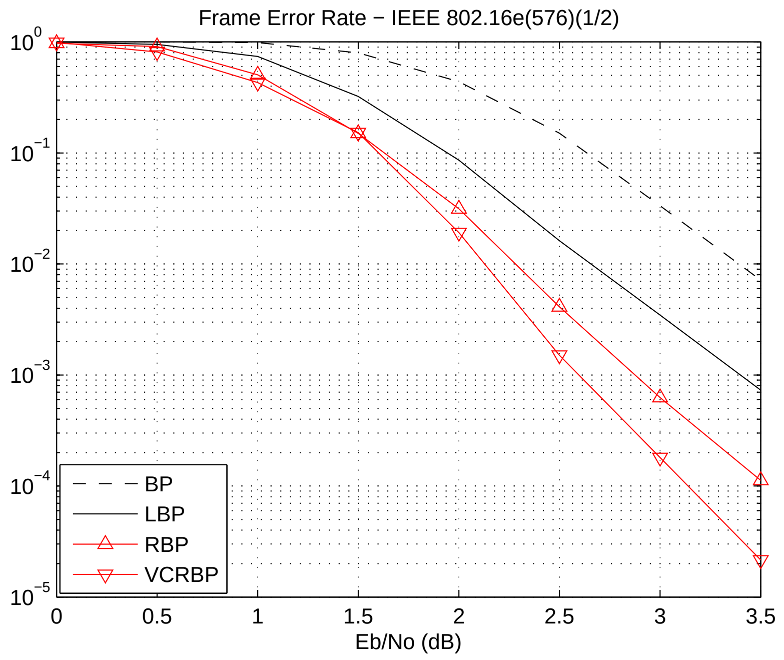


Figure 2.8: FER performance comparison of BP, LBP, RBP and VCRBP decoding using IEEE 802.16e block length-576 rate-1/2 code with at most 8 iterations

Table 2.1: Approximation of decoding complexity

	BP	**RBP**	**VCRBP**
The number of computing an $m_{c\to v}$ in one iteration	1	$(d_v-1)(d_c-1)+1$	d_c-1

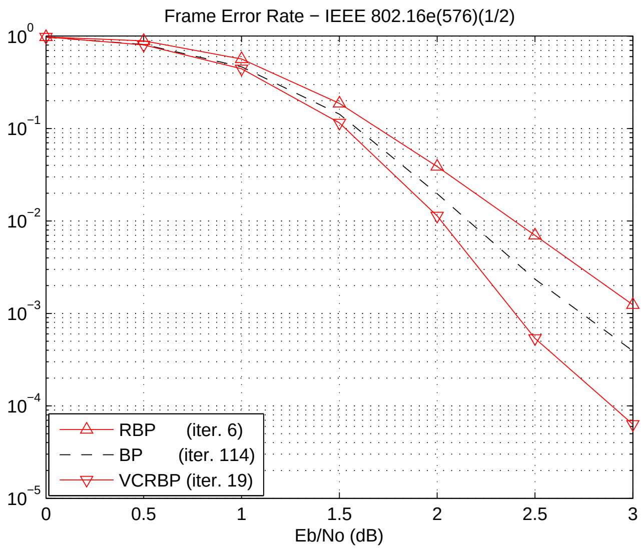


Figure 2.9: FER performance comparison of BP, RBP and VCRBP decoding using IEEE 802.16e block length-576 rate-1/2 code with maximum 114, 6, and 19 iterations, separately

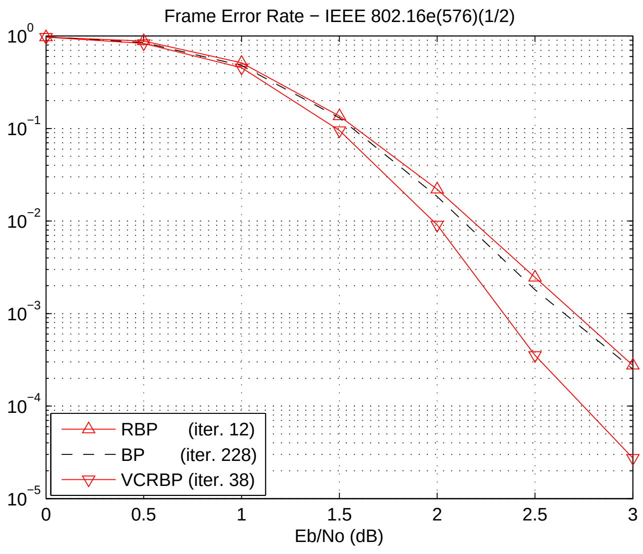


Figure 2.10: FER performance comparison of BP, RBP and VCRBP decoding using IEEE 802.16e block length-576 rate-1/2 code with maximum 228, 12, and 38 iterations, separately

2.2.2 Node-wise VCRBP Decoding for LDPC Codes

As a complexity-reduced version of VCRBP, we propose node-wise VCRBP (N-VCRBP) decoding for LDPC codes. N-VCRBP decoding is similar to SBP decoding. Main difference between SBP decoding and N-VCRBP decoding is the following: SBP decoding updates variable nodes sequentially according to a predetermined schedule while N-VCRBP decoding selects the next variable node to be updated based on residual. Specifically N-VCRBP decoding selects the variable node connected by the edge which has the largest residual. Detailed procedure of N-VCRBP decoding is presented in **Algorithm B**.

Algorithm B N-VCRBP decoding for LDPC codes

```
1:  Initialize all $m_{c\to v} = 0$
2:  Initialize all $m_{v_n\to c} = C_n$
3:  Find the largest $r(m_{v\to c})$
4:  Let $m_{v_i\to c_j}$ be chosen
5:  for every $c_k \in N(v_i)(andFlag(c_k) = 0)$ do
6:    Set $r(m_{v_i\to c_k}) = 0$
7:    for every $v_a \in N(c_k)\backslash v_i$ do
8:      Generate and propagate $m_{c_k\to v_a}$
9:      for every $c_b \in N(v_a)\backslash c_j$ do
10:       Generate and propagate $m_{v_a\to c_b}$
11:       Compute $r(m_{v_a\to c_b})$
12:     end for
13:   end for
14: end for
15: if Stopping rule is not satisfied then
16:   Go back to line 3;
17: end if
```

N-VCRBP decoding can remove edges from candidate set to avoid updating ”waste” messages. At this time, we use the parameter named ”flag”. In the previous update procedure, if c_k^1 and c_k^2 are updated then flag values of them are set to one (the others are set to zero) (see Fig. 2.11-(a), (b)). In the next update procedure, if v_i^* is chosen then $m_{v_i^* \to c_k^{*1}}$ and $m_{v_i^* \to c_k^{*2}}$ are not propagated and only flag values of them are set to zero, but on the other hand $m_{v_i^* \to c_k^{*3}}$ and $m_{v_i^* \to c_k^{*4}}$ are propagated and flag values of them are set to one (At this time, the others are set to zero) (see Fig. 2.11-(c)). By doing so, we can reduce the decoding complexity without any loss of the error correcting performance by avoiding the ”waste” updates. However, if the above situation doesn’t happen frequently, the use of ”flag” can be a burden. In that case, it can be better not to use the ”flag”.

Figure 2.12 shows that N-VCRBP has very close performance with lower complexity compared to VCRBP. N-VCRBP is also significantly better than N-RBP in both error correcting performance and decoding complexity. These facts make N-VCRBP more attractive than its counterparts.

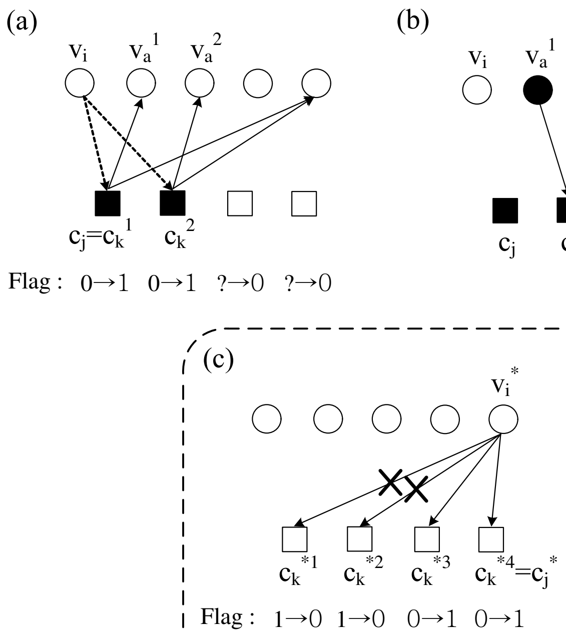


Figure 2.11: The procedure of N-VCRBP decoding for LDPC codes

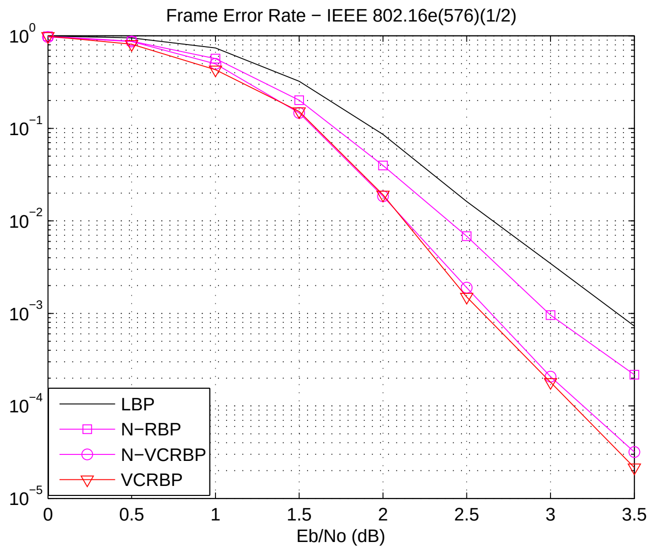


Figure 2.12: FER performance comparison of LBP, N-RBP, N-VCRBP and VCRBP decoding using IEEE 802.16e block length-576 rate-1/2 code with at most 8 iterations

2.3 Reduced Complexity Methods of RBP/VCRBP Decoding for LDPC Codes

2.3.1 Forced Convergence-based RBP/VCRBP Decoding for LDPC Codes

Forced convergence takes another approach to reduce decoding complexity by exploiting the fact that a large number of variable nodes converge to a strong belief after very few iterations, i.e., these bits have already been reliably decoded and we can skip updating their messages in subsequent iterations. This fact is already considered in [12], [13], and [14]. However, they used the magnitude of message itself in order to determine that the message converge or not. Moreover, they fix the bits to 0 or 1 in subsequent iterations. By doing so, if the fixed bit is in error, then overall codeword gets seriously negative effect. Therefore, we propose the residual of messages for the decision measure of convergence. If a residual of message is smaller than r_{th}, we can consider the message converged and then skip updating the message in the current iteration (not all subsequent iterations). Here, r_{th} is a adaptively valued constant. By using this parameter, we can reduce decoding complexity significantly. **Algorithm C** presents FC-VCRBP decoding in detail.

Algorithm C FC-VCRBP decoding for LDPC codes

```
1:  Initialize all $m_{c\to v} = 0$
2:  Initialize all $m_{v_n\to c} = C_n$
3:  Find the largest $r(m_{v\to c}) > r_{th}$
4:  Let $m_{v_i\to c_j}$ be chosen
5:  Set $r(m_{v_i\to c_j}) = 0$
6:  for every $v_a \in N(c_j)\backslash v_i$ do
7:    Generate and propagate $m_{c_j\to v_a}$
8:    for every $c_b \in N(v_a)\backslash c_j$ do
9:      Generate and propagate $m_{v_a\to c_b}$
10:     Compute $r(m_{v_a\to c_b})$
11:   end for
12: end for
13: if Stopping rule is not satisfied then
14:   Go back to line 3;
15: end if
```

However, r_{th} is an experimental value. So, to estimate how much it can be increased to achieve the goal performance is not easy. Therefore, we also suggest an approximate version of FC-RBP/VCRBP. Figure 2.13 shows that there are many edges have just very small residuals (right side of the line across the graph). They will contribute little to overall decoding convergence. For that reason, we can achieve the goal performance with lower decoding complexity by using only a part of the whole edges which has enough large residuals. For example, we can achieve the performance of RBP decoding with VCRBP decoding using only half of overall edges which have relatively large residual value. Moreover, we can control this edge percentage according to the number of iterations. In Fig. 2.14, we can see the performance of FC-VCRBP according to percentage of used edges. As mentioned above, we can achieve the goal performance with lower decoding complexity by controlling this ratio.

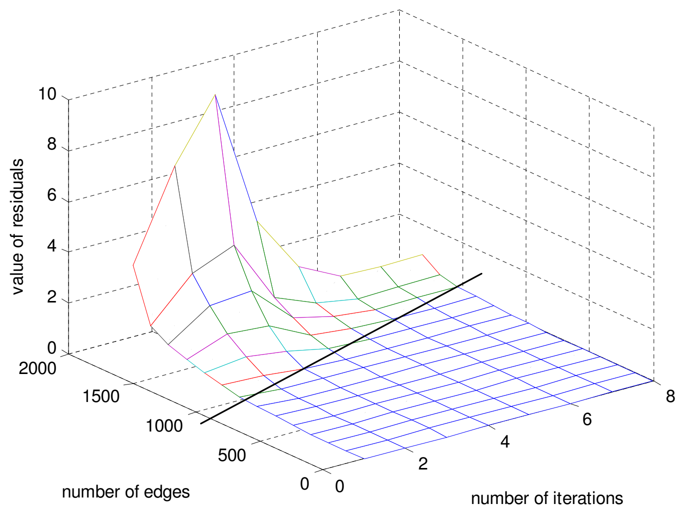


Figure 2.13: Average residual value distribution of VCRBP decoding with at most 8 iterations

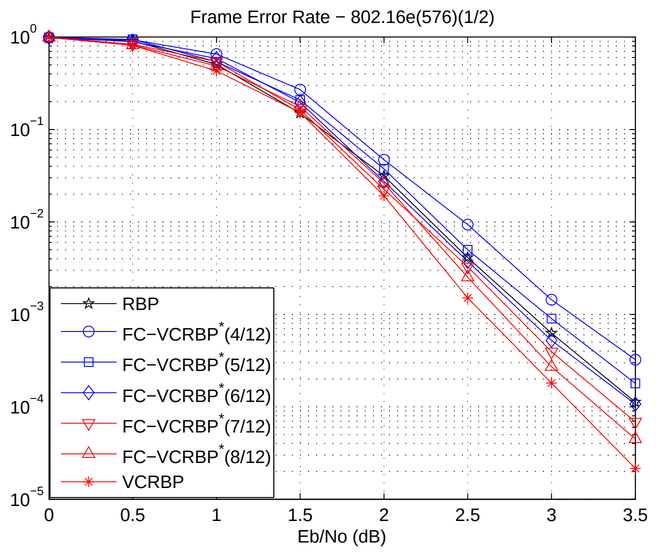


Figure 2.14: FER performance comparison of RBP, FC-VCRBP*(4/12), FC-VCRBP*(5/12), FC-VCRBP*(6/12), FC-VCRBP*(7/12), FC-VCRBP*(8/12) and VCRBP decoding with at most 8 iterations

2.3.2 Sign-based RBP/VCRBP Decoding for LDPC Codes

We just considered RBP/N-RBP, VCRBP/N-VCRBP and FC-RBP/VCRBP decoding for LDPC codes in previous sections. They have some remarkable performance gain in comparison with non-dynamic scheduling methods, but the complexity of calculating and comparing of all the residuals in these algorithms could be a serious problem. So, we propose a new dynamic scheduling method named sign-based RBP/VCRBP (S-RBP/VCRBP) for reducing the complexity. For simplifying algorithm, we define a new measure s as follows:

$$s(m_{v \to c}) = sign(m_k^*) \times sign(m_k). \tag{2.4}$$

where if m_k has a positive value then let $sign(m_k) = 1$, otherwise if m_k has a negative value then let $sign(m_k) = -1$.

If $s = -1$ at some edge then it means that sign reverse of message happen after an update at that edge. Otherwise, if sign of message at some edge before and after update do not change then $s = 1$ at that edge. The fact that change of sign happen at some edge means that its residual value changes large amount with high reliability. So, S-RBP/VCRBP chooses an edge corresponding to negative value of s. If no edge have negative value of s, S-RBP/VCRBP chooses an edge based on predetermined order so that every edge is chosen only once during one iteration.

The measure s is a very simple concept to approximate residual belief propagation algorithm. S-RBP/VCRBP does not need to calculate all residual values as well as to compare all residual values each other. Therefore, S-RBP/VCRBP is very useful method maintaining merit of RBP/VCRBP decoding and reducing their decoding complexity.

Algorithm D presents S-VCRBP decoding in detail.

```
Algorithm D  S-VCRBP decoding for LDPC codes
1:  Initialize all $m_{c \to v} = 0$
2:  Initialize all $m_{v_n \to c} = C_n$
3:  Find a $s(m_{v \to c}) = -1$
4:  Let $m_{v_i \to c_j}$ be chosen
5:  Set $s(m_{v_i \to c_j}) = 0$
6:  for every $v_a \in N(c_j) \backslash v_i$ do
7:    Generate and propagate $m_{c_j \to v_a}$
8:    for every $c_b \in N(v_a) \backslash c_j$ do
9:      Generate and propagate $m_{v_a \to c_b}$
10:     Compute $s(m_{v_a \to c_b})$
11:   end for
12: end for
13: if Stopping rule is not satisfied then
14:   Go back to line 3;
15: end if
```

Fig. 2.13 shows that S-RBP/VCRBP has close performance with remarkable diminishing complexity compared to RBP/VCRBP. We would like to note that S-VCRBP outperforms RBP as well as S-RBP with very low decoding complexity.

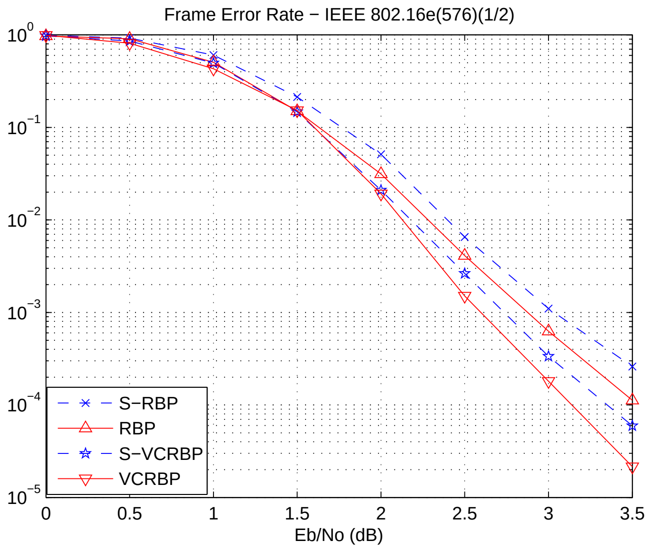


Figure 2.15: FER performance comparison of S-RBP, RBP, S-VCRBP and VCRBP using IEEE 802.16e block length-576 rate-1/2 code with at most 8 iterations

2.4 Simulation Results

This chapter presents error correcting performance comparison of BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP and VCRBP for various types of LDPC codes. QC-LDPC code mother parity check matrices designed in IEEE 802.16e standard [15], progressive edge growth (PEG) algorithm [18], modified improved PEG (MIPEG) algorithm [20], and proposed maximizing stopping set MIPEG (MS-MIPEG) is used for code construction for the simulation. We consider AWGN channel for all the simulations. The code lengths are 576, 1152, 2304, and the code rates are $1/2$ and $3/4B$.

Figure 2.20 shows error correcting performance of different scheduling strategies using IEEE 802.16e block length-576 rate-1/2 code with only 8 iterations. We can see that VCRBP, N-VCRBP, and S-VCRBP decoding have significantly better performance than RBP, N-RBP, and S-RBP decoding. Especially, we would like to note that VCRBP decoding has a gain of about 0.3dB over RBP decoding at an FER of 10^{-4}. N-VCRBP decoding has some distinctive performance gain over LBP decoding with only 8 iterations. Moreover, N-VCRBP decoding shows better performance with lower complexity compared with RBP decoding as well as N-RBP decoding.

We have checked also error correcting performances of all these codes with maximum of 50 iterations at 2.5dB in Fig. 2.17. It shows the behavior of the error correcting performance as the number of iterations is increasing up to 50. We can see that VCRBP, N-VCRBP, and S-VCRBP decoding not only converge much faster in the early stage of iterations but also maintain the performance gap in the long run of iterations.

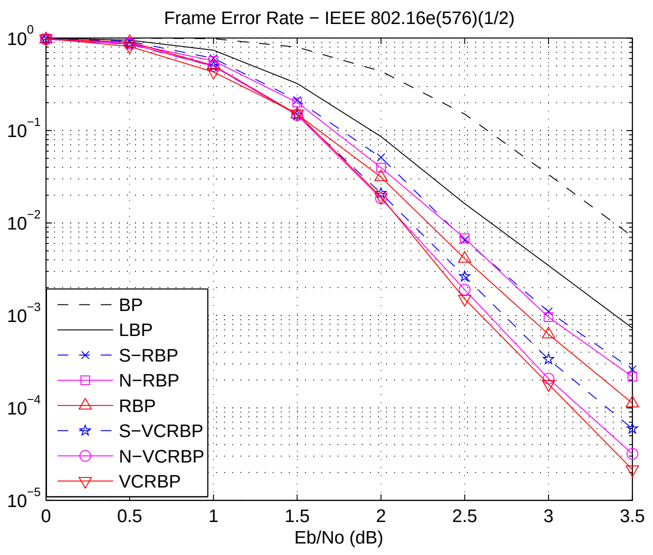


Figure 2.16: FER performance comparison of BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP, and VCRBP decoding using IEEE 802.16e block length-576 rate-1/2 code with at most 8 iterations

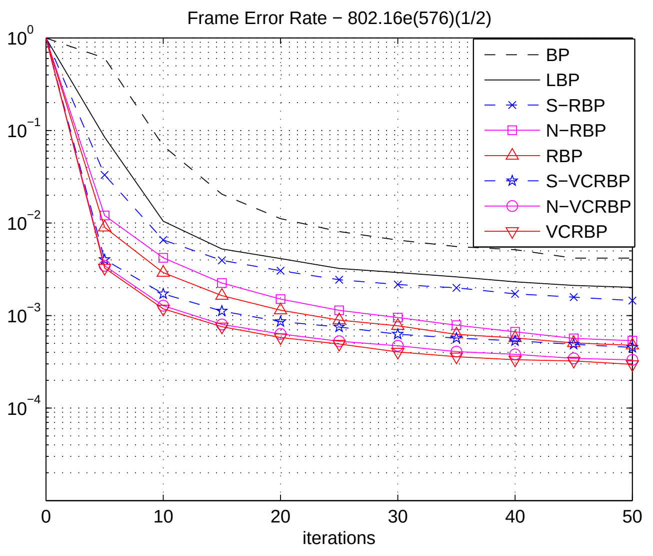


Figure 2.17: FER performance comparison of BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP, and VCRBP decoding using IEEE 802.16e block length-576 rate-1/2 code with up to 50 iterations at 2.5dB

Table 2.2 shows the average number of oscillation bits and average number of error bits [16] of different scheduling strategies using IEEE 802.16e standard parity check matrix with block length 576, rate $1/2$, and maximum 50 iterations at 2.5dB. They present that VCRBP and N-VCRBP have better error performance than RBP and N-RBP. Especially, VCRBP methods decode more accurate and fast than RBP methods. This is visualized in Fig. 2.18 and Fig. 2.19.

Table 2.2: Average number of error bits and average number of oscillation bits according to the number of iterations of RBP and VCRBP decoding

	RBP decoding		VCRBP decoding	
the number of iterations	Average number of error bits	Average number of oscillation bits	Average number of error bits	Average number of oscillation bits
1	5.187317	0	1.414365	0
2	0.708191	5.184689	0.3439	1.488789
3	0.503044	0.893145	0.186151	0.372995
4	0.455417	0.66536	0.118466	0.194583
5	5.187317	0.516024	0.090872	0.124994
6	0.708191	0.339984	0.073382	0.094086
7	0.503044	0.23462	0.060578	0.073725
8	0.455417	0.190851	0.052855	0.060909
9	0.317599	0.162827	0.045528	0.05124
10	0.208996	0.13541	0.041379	0.043655
11	0.165597	0.116002	0.037407	0.039185
12	0.139292	0.103163	0.035569	0.037234
13	0.124618	0.090794	0.032459	0.033344
14	0.102856	0.08067	0.030659	0.029794
15	0.091666	0.073122	0.029806	0.027908
16	0.082086	0.064688	0.026401	0.024783
17	0.074172	0.054249	0.026826	0.023609
18	0.066399	0.050566	0.025806	0.024185
19	0.0613	0.049894	0.025809	0.023464
20	0.054087	0.046153	0.024578	0.022835
21	0.047174	0.043923	0.022731	0.020726
22	0.046784	0.04073	0.022085	0.019133
23	0.046061	0.037703	0.020342	0.017751
24	0.041232	0.036312	0.020402	0.016773
25	0.040518	0.034796	0.019001	0.016622

	RBP decoding		VCRBP decoding	
the number of iterations	Average number of error bits	Average number of oscillation bits	Average number of error bits	Average number of oscillation bits
26	0.037175	0.03184	0.018139	0.015101
27	0.036021	0.029697	0.018595	0.015249
28	0.034082	0.029107	0.018828	0.01626
29	0.033883	0.029128	0.016672	0.01445
30	0.02973	0.027716	0.016933	0.013084
31	0.029182	0.029822	0.015863	0.013011
32	0.028734	0.029066	0.015114	0.011897
33	0.028422	0.025939	0.016414	0.012517
34	0.02704	0.022393	0.015709	0.013515
35	0.02919	0.022759	0.015029	0.012074
36	0.026691	0.025437	0.014805	0.011863
37	0.024295	0.023917	0.01502	0.011967
38	0.02199	0.02248	0.013666	0.011079
39	0.024137	0.021023	0.013864	0.010415
40	0.025931	0.020467	0.013679	0.010774
41	0.02324	0.022281	0.012741	0.00965
42	0.022426	0.021467	0.012835	0.009159
43	0.020969	0.019669	0.012593	0.009641
44	0.020799	0.019358	0.012801	0.009788
45	0.02295	0.019034	0.012064	0.009606
46	0.02084	0.017747	0.012218	0.008652
47	0.019624	0.016223	0.012703	0.00936
48	0.020031	0.016738	0.011982	0.009005
49	0.018793	0.019042	0.011293	0.008224
50	0.01734	0.017763	0.012007	0.00818

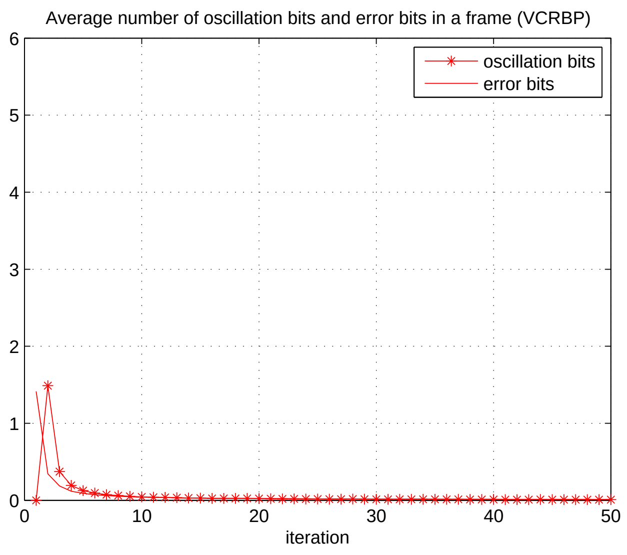


Figure 2.18: Average number of oscillation bits and error bits in a frame of VCRBP decoding using IEEE 802.16e block length-576 rate-1/2 code with at most 50 iterations at 3dB

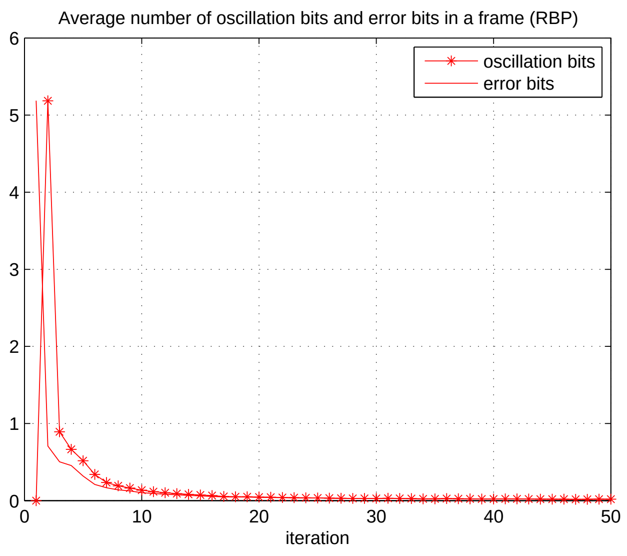


Figure 2.19: Average number of oscillation bits and error bits in a frame of RBP decoding using IEEE 802.16e block length-576 rate-1/2 code with at most 50 iterations at 3dB

For observing the performance gain of proposed decoding methods for various types of LDPC codes, we apply proposed algorithms to mother parity check matrices designed in IEEE 802.16e, block lengths are 576, 1152, 2304, and code rates are $1/2$ and $3/4B$. Table 2.3 shows the local girth distribution of them. Here, the local girth is the smallest cycle including each variable node. Fig. 2.20-Fig. 2.25 compare the error correcting performances using BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP and VCRBP. They show that VCRBP, N-VCRBP, and S-VCRBP have significantly better performance than RBP, N-RBP, and S-RBP for all the simulation codes. Especially, proposed methods show more notable performance gain with shorter block length codes and a small number of iterations.

Table 2.3: Local girth distribution of LDPC codes for simulation

	4-cycles	**6-cycles**	**8-cycles**
IEEE 802.16e, block length-576, rate-1/2	168	336	72
IEEE 802.16e, block length-1152, rate-1/2	96	672	384
IEEE 802.16e, block length-2304, rate-1/2		384	1920
IEEE 802.16e, block length-576, rate-3/4B	360	216	
IEEE 802.16e, block length-1152, rate-3/4B	432	720	
IEEE 802.16e, block length-2304, rate-3/4B		2304	

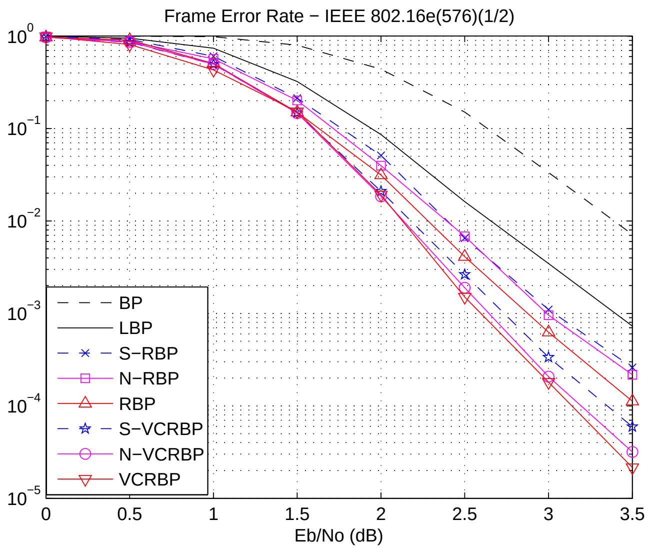


Figure 2.20: FER performance comparison of BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP, and VCRBP decoding using IEEE 802.16e block length-576 rate-1/2 code with at most 8 iterations

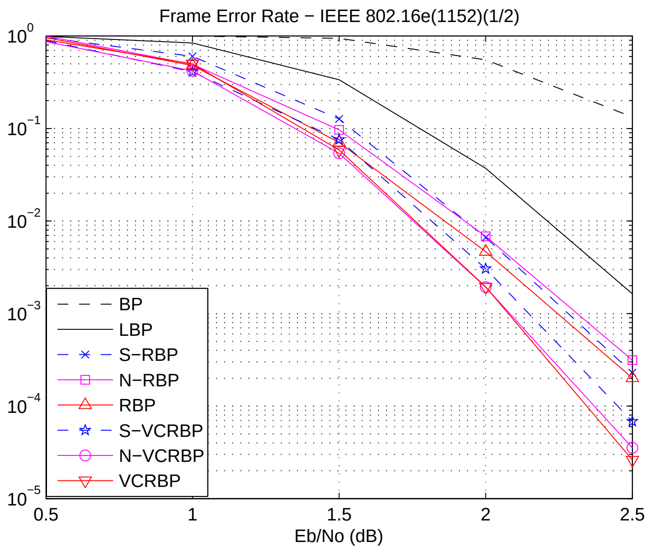


Figure 2.21: FER performance comparison of BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP, and VCRBP decoding using IEEE 802.16e block length-1152 rate-1/2 code with at most 8 iterations

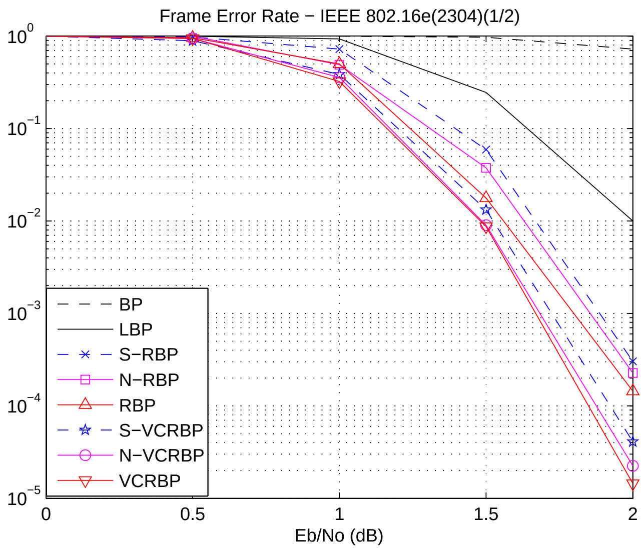


Figure 2.22: FER performance comparison of BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP, and VCRBP decoding using IEEE 802.16e block length-2304 rate-1/2 code with at most 8 iterations

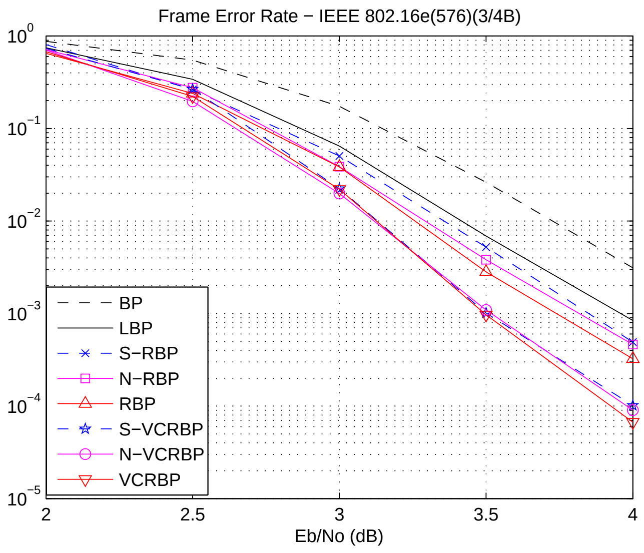


Figure 2.23: FER performance comparison of BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP, and VCRBP decoding using IEEE 802.16e block length-576 rate-3/4B code with at most 8 iterations

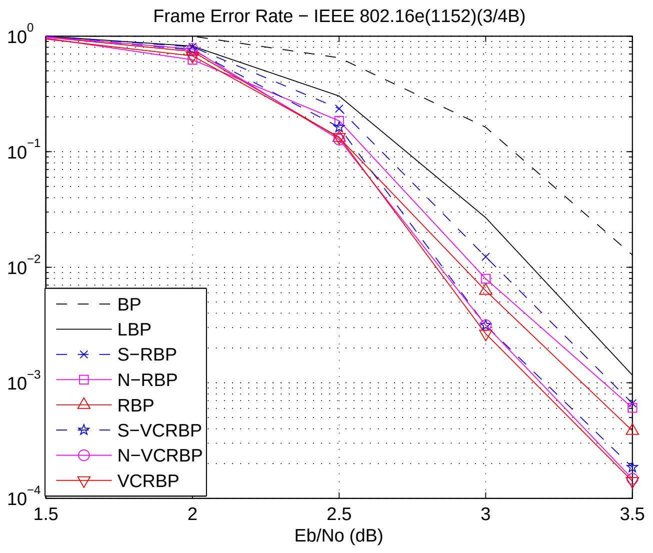


Figure 2.24: FER performance comparison of BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP, and VCRBP decoding using IEEE 802.16e block length-1152 rate-3/4B code with at most 8 iterations

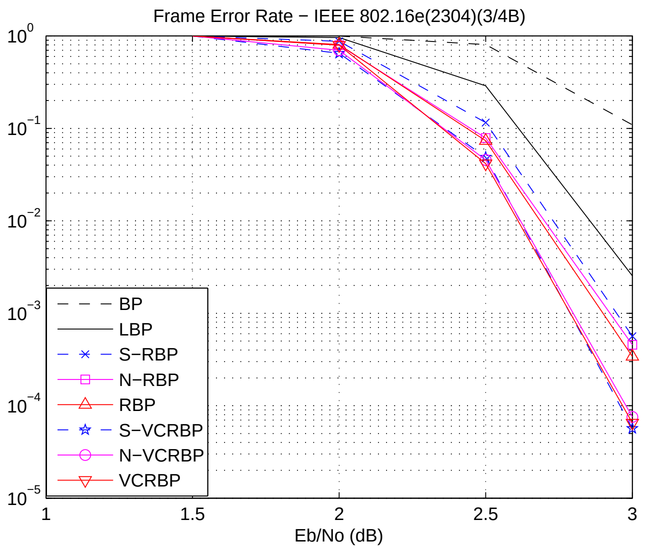


Figure 2.25: FER performance comparison of BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP, and VCRBP decoding using IEEE 802.16e block length-2304 rate-3/4B code with at most 8 iterations

As mentioned above, dynamic scheduling algorithms, RBP, VCRBP and its modifications, solve the trapping set problem caused by small stopping sets in the LDPC codes graph [21]. So, we simulate that if we use code construction method removing small stopping sets then the performance gain is maintained or not. For removing small stopping sets, we use the stopping set detection algorithm in [22], [23]. However, the code designed only by the stopping set detection often shows the degraded bit error performance in low SNR region because a large number of connected small cycles constituting a large stopping set degrades a message-passing decoding performance [23]. Therefore, we consider the cycle characters, local girth, extrinsic message degree (EMD), and approximate cycle EMD (ACE) [17], of the code graph together with the stopping set detection for code construction. Table 2.4 shows the criterion for code construction. In Table 2.5, we can see proposed algorithm reduces random selection for code construction. Moreover, Table 2.6 and Table 2.7 show that proposed code construction method is better than both progressive edge growth (PEG) algorithm and modified improved PEG (MIPEG) algorithm both of cycle and stopping set distribution character on the code graph. Fig. 2.26-Fig. 2.28 confirm the fact that proposed VCRBP, N-VCRBP, and S-VCRBP maintain the performance gain with good designed codes which are removed small stopping sets. However, VCRBP and RBP decoding have very close performance after very large iterations (see Fig. 2.29). It is reasonable because RBP decoding also solves almost trapping set errors after enough large iterations.

Table 2.4: Criterion for code construction

	PEG	MIPEG	Proposed
Construction Method	Tree spreading	Tree spreading	Tree spreading
Criterion (1st)	Largest girth	Largest girth	Largest girth
Criterion (2nd)	Lowest degree	Lowest degree	Lowest degree
Criterion (3rd)	Random	ACE	ACE
Criterion (4th)	Random	EMD	EMD
Criterion (5th)	Random	More variables	More variables
Criterion (6th)	Random	Random	Largest stopping set
Criterion (7th)	Random	Random	Random

Table 2.5: The number of random-selection for each algorithm

	Cases (Ratio)
Total edges	1304
Random selection after PEG algorithm	1017 (0.779908)
Random selection after IPEG algorithm	543 (0.416411)
Random selection after EMD criterion	83 (0.0636503)
Random selection after Variable number criterion	33 (0.0253067)
Random selection after proposed algorithm	3 (0.00230061)

Table 2.6: Local girth distribution of LDPC codes for simulation

	4-cycles	**6-cycles**	**8-cycles**
PEG, block length-576, rate-1/2		295	281
MIPEG, block length-576, rate-1/2		169	407
Proposed MS-MIPEG, block length-576, rate-1/2		160	416

Table 2.7: Stopping set radius distribution of LDPC codes for simulation

	radius 4	**radius 5**	**radius 6**
PEG, block length-576, rate-1/2	4	271	301
MIPEG, block length-576, rate-1/2		271	305
Proposed MS-MIPEG, block length-576, rate-1/2		270	306

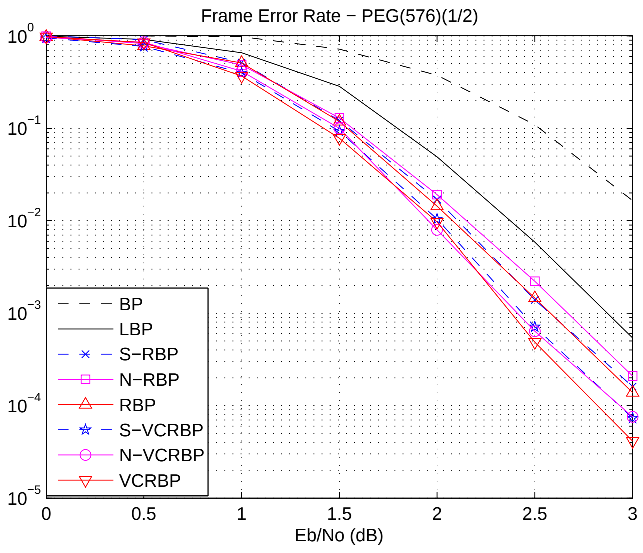


Figure 2.26: FER performance comparison of BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP, and VCRBP decoding using PEG block length-576 rate-1/2 code with at most 8 iterations

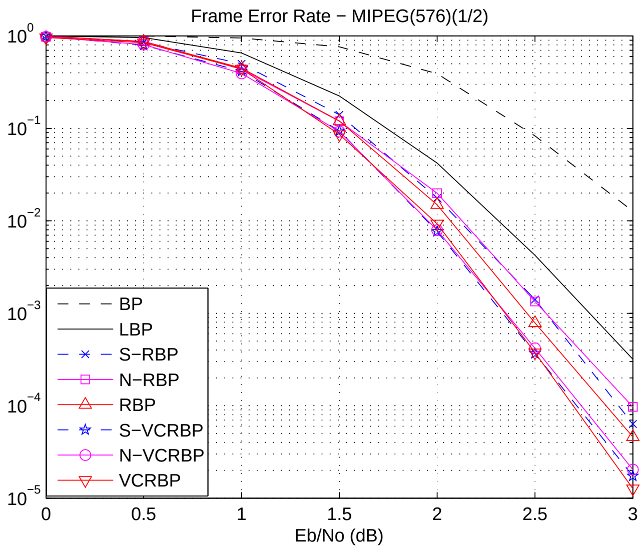


Figure 2.27: FER performance comparison of BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP, and VCRBP decoding using MIPEG block length-576 rate-1/2 code with at most 8 iterations

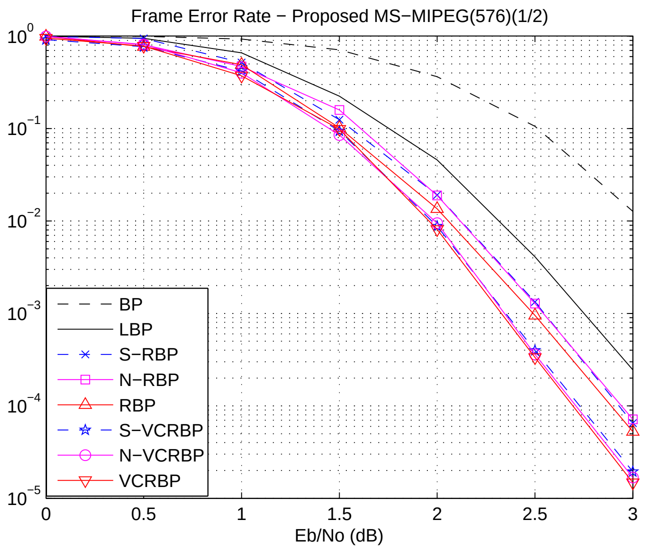


Figure 2.28: FER performance comparison of BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP, and VCRBP decoding using proposed MS-MIPEG block length-576 rate-1/2 code with at most 8 iterations

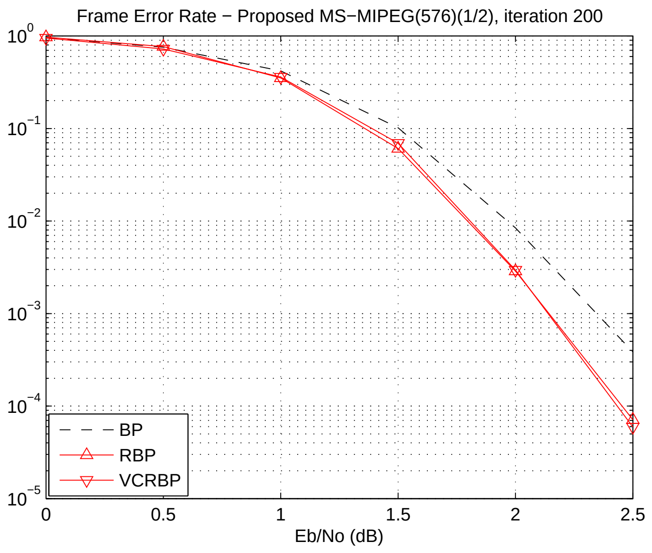


Figure 2.29: FER performance comparison of BP, LBP, S-RBP, N-RBP, RBP, S-VCRBP, N-VCRBP, and VCRBP decoding using proposed MS-MIPEG block length-576 rate-1/2 code with at most 200 iterations

2.5 Conclusions

We propose VCRBP decoding for LDPC codes. Simulation shows that (1) VCRBP decoding makes LDPC decoding converge very fast in terms of the number of iterations; and (2) it performs similarly better after sufficiently many iterations. The complexity reduction can be seen easily from the comparison of Algorithm 1 and Algorithm A.

We also propose a complexity-reduced version of VCRBP decoding, N-VCRBP decoding. N-VCRBP decoding has very close performance to VCRBP decoding with significantly lower decoding complexity.

For further reducing the complexity, we present two reduced complexity methods for both RBP and VCRBP decoding, named FC-RBP/VCRBP and S-RBP/VCRBP decoding. Both of them significantly reduce the decoding complexity with only negligible deterioration in error correcting performance.

All the proposed methods have remarkable performance gain compared to existing methods. Therefore, VCRBP, N-VCRBP and two reduced complexity methods, FC-RBP/VCRBP and S-RBP/VCRBP decoding, would be one of remedies for future communication system allowing very high data rates.

Chapter 3

Joint LDPC Codes for Multi-User Relay Channel

3.1 Decode-and-Forward Relay Channel

The decode-and-forward strategy [37] works by allowing the relay to decode the source message and then forward a bin index (partition index of codebook) of the source message to facilitate the ultimate decoding at the destination.

For more practical channel model, we consider such as concepts of time-division half-duplex relay protocol, gaussian degraded relay channel, and parity forwarding relay channel.

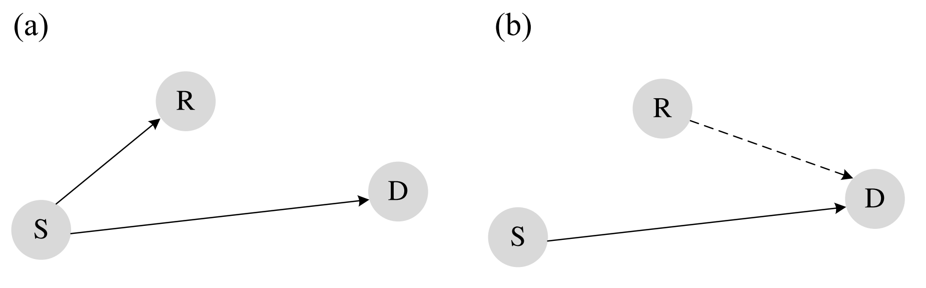


Figure 3.1: Decode-and-forward relay channel; (a) First time slot; (b) Second time slot.

3.1.1 Time-Division Half-Duplex Relay Protocol

Time-division half-duplex relay protocol [38] requires two time slots, because the relay cannot transmit and receive simultaneously. In the first time slot, the source (S) transmits message to both the relay (R) and the destination (D). In the second time slot, both S and R transmit message to D. (see Figure 3.1)

3.1.2 Gaussian Degraded Relay Channel

In gaussian degraded relay channel [39], the source divides its total power P into a fraction αP for transmitting new codeword ω_i and a fraction of $(1-\alpha)P$ for cooperatively transmitting the bin index s_i of the previous codeword ω_{i-1}. And the source transmits both ω_i and s_i to relay and destination in the ith time slot. The relay decodes received message corresponding to ω_i transmitted by source, and then computes s_{i+1}. The destination decodes received message corresponding to ω_i using s_{i+1} transmitted from both source and relay in the $i+1$st time slot and ω_i transmitted from source in the ith time slot [30].

3.1.3 Parity Forwarding Relay Channel

According to [40], [41], binning can be realized via parity generation. Parity bits are interpreted as a bin index because the set of codewords satisfying a particular parity forms a coset, and the set of all cosets form a partition of the entire codebook (see Figure 3.2). Thus, decode-and-forward is equivalent to parity forwarding.

Applying parity forwarding model in Gaussian degraded relay channel, we assume that ω_i is a codeword (information I and parity P_1) encoded by channel coding like

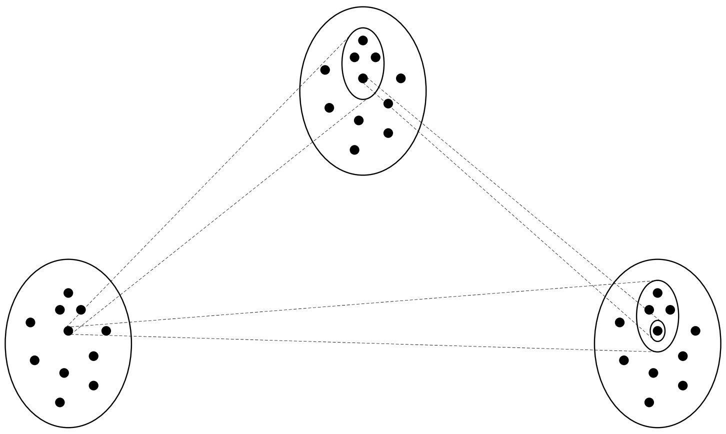


Figure 3.2: Concept of binning.

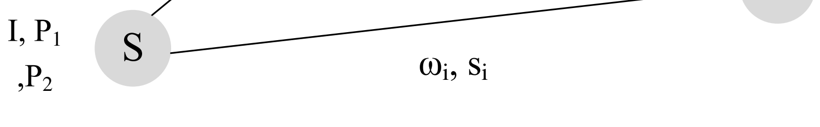


Figure 3.3: Parity forwarding gaussian degraded relay channel (ith time slot)

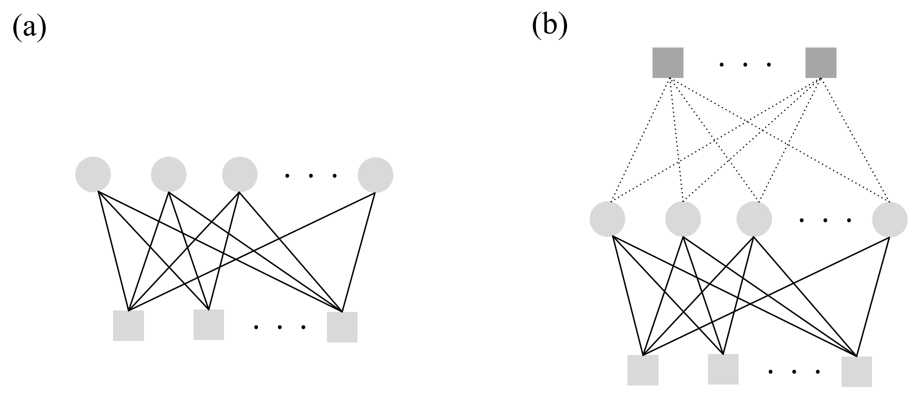


Figure 3.4: LDPC code graph; (a) conventional LDPC code graph; (b) bilayer LDPC code graph.

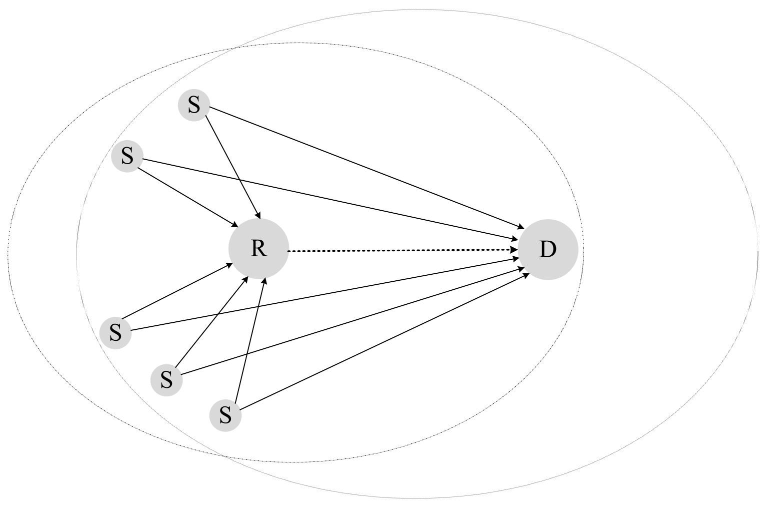
LDPC code and that s_i is an additional parity (P_2) corresponding ω_{i-1} (see Figure 3.3).

3.2 Joint LDPC Codes for Multi-User Relay Channel

3.2.1 Joint LDPC Codes Model

A conventional LDPC code is defined over a bipartite graph consisting of variable nodes and and check nodes. In Figure 3.4 (a), variable nodes and check nodes are visualized by circles and squares respectively. A variable node is connected to a check node by an edge.

As shown in Figure 3.4 (b), bilayer LDPC code graph is composed of lower and upper subgraphs. A lower part represents the code designed for the source-relay channel. The upper part is so designed that the overall graph guarantees successful decoding at the destination. However, in the multi-user case, bilayer LDPC code system optimizes each user code individually for the own channel.



Source	Relay
$\{I, P_1\}_{S\to R,\, S\to D}$	$\{P_2\}_{R\to D}$

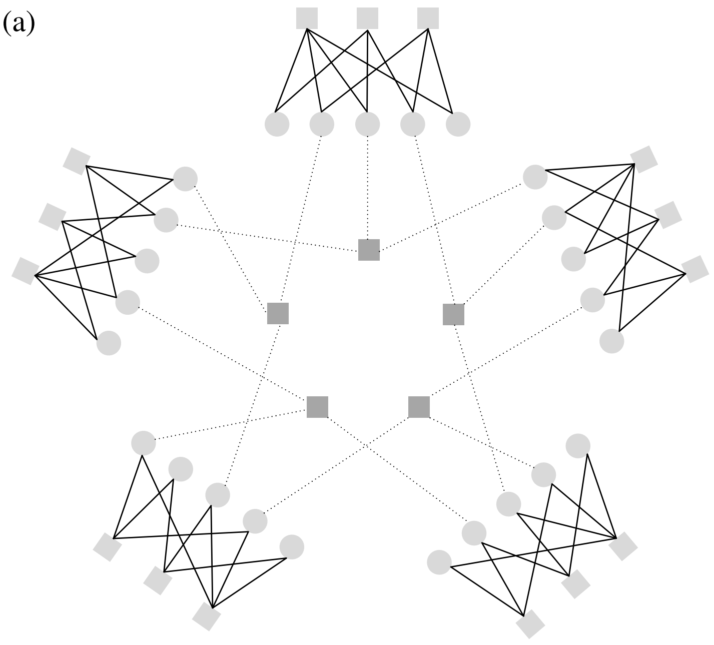
Figure 3.5: Joint LDPC code system for multi-user relay channel.

We now propose joint LDPC code to obtain some cooperative gain. Joint LDPC code is composed with two layers: the outer layer optimized for the channel between each user and relay; and the inner layer is optimized for the overall channel between users and the destination. In the joint LDPC code system, each user transmits simultaneously its data (the information I and parity P_1) to the destination D. And the relay optimally operates incoming data from each user and sends operated message P_2 to destination additionally (see Figure 3.5).

A simple model of a joint LDPC code graph is described in Figure 3.6. Overall graph of a joint LDPC code has doubly layered (inner and outer) subgraphs. The inner subgraph is added to the outer subgraph optimized to channel between source and relay, by adding extra parity bits optimized to overall channel. Hence, a variable node is connected to two types of edges: the outer edges, connecting a variable node to outer check nodes; and the inner edges, connecting a variable node to inner check nodes. Thus, the degree of a variable node can be defined by the two tuple (i, j) if the variable node is connected to i outer edges and j inner edges. We assume regular check degrees for check nodes in the outer and inner graphs, respectively. The outer check degree d_c denotes the number of edges connected to check nodes in the outer subgraph. Likewise, the inner check degree d'_c equals to the number of edges connected to the inner check node. The rate of edge which is connected to the variable node with outer degree i and inner degree j is given by $\lambda_{i,j}$, and a parameter η denotes the percentage of outer edges in overall graph. Note that $\sum_{i\geq 2, j\geq 0} \lambda_{i,j} = 1$, and $0 < \eta < 1$.

In such a method, the source encodes its data using the outer graph. The relay decodes the received codeword from the source, and computes extra parity bits using the inner graph for the source codeword. In Figure 3.2, these extra parity bits constrain the source codeword to a coset, which is of a smaller size. (In other words, the destination effectively observes a source code with a lower rate.) The destination decodes the received codeword by message passing algorithm over the entire graph. For this method to work, the degree distribution of a joint LDPC code should be optimized so that the outer layer code is capacity approaching at a higher SNR (at the relay), and the overall graph is capacity approaching at a lower SNR (at the destination). To optimize the degree distribution, a modified version of density evolution should be used to analyze the performance of a given joint LDPC code. Using joint LDPC code density evolution, the degree distribution of the code can be achieved through an optimization procedure based on iterative linear programming.

(a)



(b)

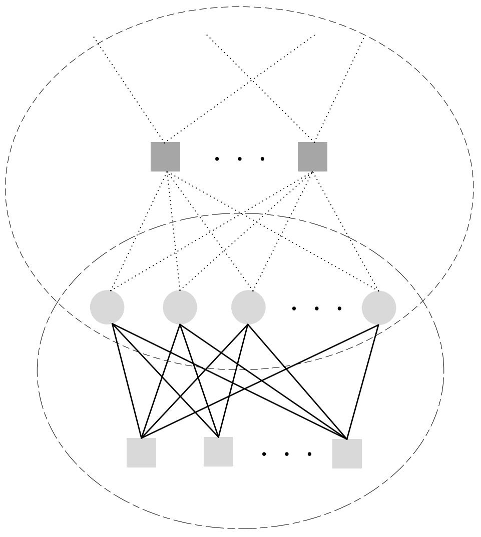


Figure 3.6: Joint LDPC code model; (a) joint code graph; (b) individual code graph.

3.2.2 Joint LDPC Codes Density Evolution

Joint LDPC code density evolution is obtained similarly as bilayer LDPC code density evolution [29], [30]. Let p^l denote the message probability density function at the input of outer check nodes, and q^l is defined similarly as the message probability density function at the input of inner check nodes at the beginning of the lth decoding iteration. The message probability density functions after check update can be computed using the conventional density evolution check update as described in [42], [43] as follows:

$$\begin{aligned} p'^l &= (\oplus^{d_c-1} p^l) \oplus (\oplus^{d'_c} q^l), \quad d_c > 1, \\ q'^l &= (\oplus^{d_c} p^l) \oplus (\oplus^{d'_c-1} q^l), \quad d'_c > 1, \end{aligned} \tag{3.1}$$

and for $d'_c = 1$,

$$\begin{aligned} p'^l &= (\oplus^{d_c-1} p^l) \oplus (q^l), \quad d_c > 1, \\ q'^l &= (\oplus^{d_c} p^l), \end{aligned} \tag{3.2}$$

where $\oplus$ denotes the check density-update operation.

For log-likelihood message-passing decoding, the density evolution update at a degree $(i,\, j)$ variable node can be computed from p'^l and q'^l to obtain the message probability density functions, $p^{l+1}_{i,j}$ and $q^{l+1}_{i,j}$, as follows:

$$\begin{aligned} p^{l+1}_{i,j} &= (\otimes^{i-1} p'^l) \otimes (\otimes^{j} q'^l) \otimes p_c, \quad i \geq 2, j \geq 0, \\ q^{l+1}_{i,j} &= (\otimes^{i} p'^l) \otimes (\otimes^{j-1} q'^l) \otimes p_c, \quad i \geq 2, j \geq 1, \end{aligned} \tag{3.3}$$

where p_c denotes the probability density function of the log-likelihood ratio received over the channel, and $\otimes^i$ denotes convolution of order i. The input message probability density functions to outer and inner check nodes, at the beginning of the $(l+1)$st iteration can be computed as follows:

$$\begin{aligned} p^{l+1} &= \sum_{i\geq 2, j\geq 0} \frac{i}{i+j}\lambda_{i,j} p_{i,j}^{l+1}, \\ q^{l+1} &= \sum_{i\geq 2, j\geq 1} \frac{j}{i+j}\lambda_{i,j} q_{i,j}^{l+1}. \end{aligned} \tag{3.4}$$

Similar to the error profile functions for a conventional LDPC code, the outer graph degree $(i,\ j)$ error profile function, $e_{i,j}^1(p^l, q^l)$ and the inner graph degree $(i,\ j)$ error profile function, $e_{i,j}^2(p^l, q^l)$ is defined as the message error probability after one density evolution iteration for input message probability density functions p^l and q^l. Let $e(p^{l+1}, q^{l+1})$ denote the overall message error probability in the multi-user bilayer graph corresponding to the message probability density functions p^{l+1} and q^{l+1}. The overall message error probability at the beginning of the $(l+1)$'st decoding iteration, $e(p^{l+1}, q^{l+1})$, can be computed as a linear combination of $e_{i,j}^1(p^l, q^l)$ and $e_{i,j}^2(p^l, q^l)$ as follows:

$$\begin{aligned} & e(p^{l+1}, q^{l+1}) \\ &= \sum_{i\geq 2, j\geq 0} \lambda_{i,j}\left(\frac{i}{i+j} e_{i,j}^1(p^l, q^l) + \frac{j}{i+j} e_{i,j}^2(p^l, q^l)\right). \end{aligned} \tag{3.5}$$

3.2.3 Joint LDPC Codes Optimization

The design of a joint LDPC code is finding an optimal variable nodes degree distribution $\lambda_{i,j}$. In other words, it is to jointly optimize both outer subgraphs for each user and the overall graph to achieve the highest overall rate. In this paper, we fix d_c, d'_c, and each outer subgraph to be a capacity-approaching LDPC code at the channel between source and relay. Then we search for variable nodes degree distribution, $\lambda_{i,j}$, that is a capacity-approaching LDPC code at the overall channel.

This approximate linear programming update for optimizing $\lambda_{i,j}$ can be formulated by maximizing R iteratively as follows:

$$\begin{aligned}
\max_{\lambda_{i,j}} \; & R = 1 - \frac{\sum_{u\geq 1}\sum_{i\geq 2} \rho_{i(u)}/i(u)}{\sum_{u\geq 1}\sum_{i\geq 2} \lambda_{i(u)}/i(u)}, \\
\text{s.t.} & \sum_{i\geq 2, j\geq 0} \lambda_{i,j}\left(\frac{i}{i+j} e^1_{i,j}(p^l, q^l) + \frac{j}{i+j} e^2_{i,j}(p^l, q^l)\right) \\
& < \mu_h(\eta p + (1-\eta)q),
\end{aligned} \tag{3.6}$$

$$\begin{aligned}
& \sum_{i(u)\geq 2} \lambda_{i(u)} e_{i(u)}(p) < \mu_h p, \\
& \text{where } \lambda_{i(u)} = \frac{1}{\eta(u)} \sum_{j(u)\geq 0} \frac{i(u)\cdot \lambda_{i(u),j(u)}}{i(u)+j(u)},
\end{aligned} \tag{3.7}$$

$$err^{l_{max}}(u) \leq err_{th}, \tag{3.8}$$

where u is the number of users, h is the number of optimization iterations, and l is the number of decoding iterations. The coefficient $0 < \mu_h < 1$ is slightly increased toward 1 at each optimization iteration. The error profile functions $e^1_{i,j}(p^l, q^l)$, $e^2_{i,j}(p^l, q^l)$ and

$e(p^l, q^l)$ are recomputed at the end of each optimization iteration using bilayer density evolution, given the new $\lambda_{i,j}$.

This optimization process is very similar to bilayer LDPC code optimization [29], [30], but in the multi-user system, we have to consider two more points. Each user code is optimized for the channel between the user and the relay in (7). And after decoding of the overall graph, error probability of the each user data has to be less than the value of threshold in (8).

3.3 Joint LDPC Codes for Ad-hoc Networks

The works of [33], [34], [35], [36] employ the so-called adaptive network coded cooperation (ANCC) method for the special case of ad-hoc networks (multi-source and one-destination). Using the protocol, low-density generator-matrix (LDGM) or lower-triangular LDPC (LT-LDPC) structure network coding is applicable. However, in the LDGM case, 4-cycle problem due to random selection exists, and in the LT-LDPC case, user delay problem exist. For one of remedies, here we propose the application model of joint LDPC code to the case of ad-hoc networks. In the first time slot, each source broadcasts its message. In the second time slot, the relay selects some messages from its received messages using optimized rule, computes their check sum, and transmits the check sum messages to the destination. This protocol needs no information about extra parity check matrix for the selection at the neighborhood node in every time as well as resolve pre-raised problems.

3.4 Simulation

In this section, the performance of a joint LDPC code is simulated. Ideally, a joint LDPC code should be optimized for both the channel between each user and the relay and the overall channel. Here, for simplicity, we design a joint LDPC code with fixed rates, $R_{SR} = 2/3$ (source-relay), $R_{RD} = 1/2$ (relay-destination), corresponding two layer of the code graph for AWGN channel. We assume that twelve users transmit data simultaneously with no interference, and the block length of each user code is 96, the block length of overall code is 2304. For convenience, we use sub-optimal block LDPC code designed in IEEE 802.16e standard for optimizing overall graph [15].

Figure 3.7 shows the bit error rate (BER) performance of this model at the destination, assuming power separation factor $\alpha = 1$ and the number of maximum decoding iterations is 8 with standard belief propagation decoding algorithm. Note that there is a gap of about 1.0 dB to bilayer LDPC code at a BER of 10^{-6} for the overall channel. The frame error rate (FER) curve graph in Fig. 3.8 also shows the performance improvement.

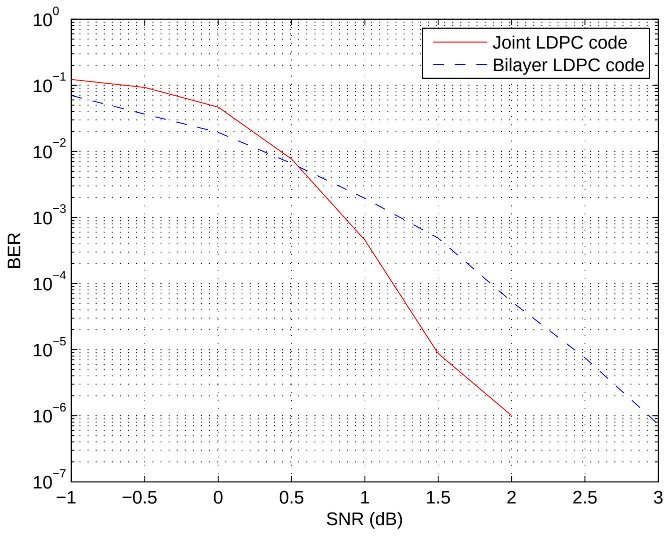


Figure 3.7: BER performance comparison of joint LDPC code (solid line) and bilayer LDPC code (dashed line)

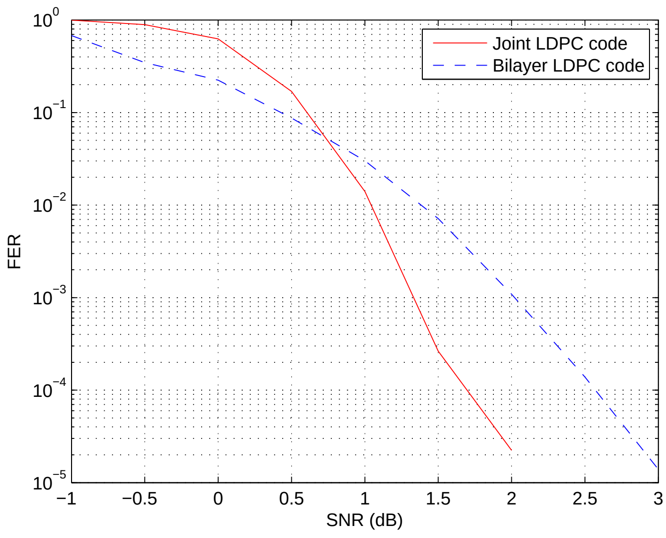


Figure 3.8: FER performance comparison of joint LDPC code (solid line) and bilayer LDPC code (dashed line)

3.5 Conclusion

The problem of wireless user cooperation in basic relay channel has been extensively studied, but theoretical and optimal solutions are still lacking. In this thesis, we show that the joint LDPC code brings remarkable performance gain by user cooperation (about 1.0 dB at 10^{-6}). Moreover, joint LDPC codes guarantee achieved service quality by controlling the distribution of edges corresponding to each user data at the relay.

On the other hand, the price we have to pay for this improvement are (1) the relay must be much more intelligent than before and (2) overall code length decoded simultaneously at the destination becomes much larger than before.

Chapter 4

Conclusion

We propose VCRBP decoding for LDPC codes for fast convergence, better error correcting performance and lower complexity. We also propose complexity-reduced methods of VCRBP that are N-VCRBP, FC-VCRBP, and S-VCRBP. All the proposed methods outperform RBP as well as BP with same decoding complexity. Especially, they are more effective with short block length codes and a small number of iterations.

Moreover, we can use VCRBP concept for punctured LDPC codes. In the punctured LDPC codes, the scheduled decoding using puncturing pattern (in recoverability order) outperforms conventional (random) scheduled decoding and conventional BP decoding [44]. So, VCRBP methods are very effective with random punctured LDPC codes which can not use puncturing pattern to decode the codeword because the node which has a big residual value is likely to have the highest level of recoverability.

On the other hand, we propose joint LDPC codes for multi-user relay channel. The joint LDPC codes concept is a channel-network coding technique which is a combinative method of Forward Error Correction (FEC) and Network coding (NC). We can obtain the improvement of error correcting performance by only using more intelligent relay for user cooperation.

As a future work, a quantification of trade-off between decoding complexity and error correcting performance is needed. In the VCRBP algorithm, the complexity reduction of computing and comparing the residual would be also a topic of future research. In the joint LDPC codes concept, we can also consider a channel-network coding model design for multi-relay channel for a further work.

Appendix

Derivation of message generating functions

In this appendix, we present a brief derivation of generating functions of two messages, variable-to-check message and check-to-variable message. For log likelihood decoding, variable-to-check message can be expressed as

$$\begin{aligned}
m_{v_j \to c_i} &= \log \frac{P(v_j = 0|\mathbf{r}, \{E_{c_a} = 0, c_a \in N(v_j)\backslash c_i\})}{P(v_j = 1|\mathbf{r}, \{E_{c_a} = 0, c_a \in N(v_j)\backslash c_i\})} \\
&= \log \frac{P(v_j = 0|r_j, \{r_n, n \neq j\}, \{E_{c_a} = 0, c_a \in N(j)\backslash c_i\})}{P(v_j = 1|r_j, \{r_n, n \neq j\}, \{E_{c_a} = 0, c_a \in N(j)\backslash c_i\})} \\
&= \log \frac{P(r_j|v_j = 0)}{P(r_j|v_j = 1)} + \log \frac{P(v_j = 0|\{r_n, n \neq j\}, \{E_{c_a} = 0, c_a \in N(v_j)\backslash c_i\})}{P(v_j = 1|\{r_n, n \neq j\}, \{E_{c_a} = 0, c_a \in N(v_j)\backslash c_i\})} \\
&= C_{v_j} + \log \frac{\prod_{c_a \in N(v_j)\backslash c_i} P(E_{c_a} = 0|v_j = 0, \{r_n, n \neq j\})}{\prod_{c_a \in N(v_j)\backslash c_i} P(E_{c_a} = 0|v_j = 1, \{r_n, n \neq j\})} \\
&= C_{v_j} + \sum_{c_a \in N(v_j)\backslash c_i} \log \frac{P(E_{c_a} = 0|v_j = 0, \{r_n, n \neq j\})}{P(E_{c_a} = 0|v_j = 1, \{r_n, n \neq j\})}, \qquad \text{(A.1)}
\end{aligned}$$

where **r** is the received signal vector, the check equation $E_{c_a} = \sum_{v_n \in N(c_a)} v_n$.

Let us define check-to-variable message as following

$$m_{c_a \to v_j} = \log \frac{P(E_{c_a} = 0|v_j = 0, \{r_n, n \neq j\})}{P(E_{c_a} = 0|v_j = 1, \{r_n, n \neq j\})}. \qquad \text{(A.2)}$$

Intuitively, we can write the following equation. (We can also prove this by using simple induction.)

$$
\begin{aligned}
&P(E_{c_a}=0|v_j=0,\{r_n,n\neq j\})-P(E_{c_a}=0|v_j=1,\{r_n,n\neq j\})\\
&=\prod_{v_b\in N(c_a)\backslash v_j}[P(v_b=0|\mathbf{r},\{E_{c_a}=0,c_a\in N(v_j)\backslash c_i\})\\
&\qquad\quad -P(v_b=1|\mathbf{r},\{E_{c_a}=0,c_a\in N(v_j)\backslash c_i\})]. \qquad \text{(A.3)}
\end{aligned}
$$

Let $p=P(E_{c_a}=0|v_j=0,\{r_n,n\neq j\})$. From (A.2), we can write

$$
m_{c_a\to v_j}=\log\frac{p}{1-p}\Rightarrow p=\frac{\exp(m_{c_a\to v_j})}{\exp(m_{c_a\to v_j})+1}=\frac{1}{2}(1+\tanh(\frac{m_{c_a\to v_j}}{2})) \qquad \text{(A.4)}
$$

$$
1-p=\frac{1}{2}(1-\tanh(\frac{m_{c_a\to v_j}}{2})). \qquad \text{(A.5)}
$$

Similarly, let $q=P(v_b=0|\mathbf{r},\{E_{c_c}=0,c_c\in N(v_b)\backslash c_a\})$ then

$$
m_{v_b\to c_a}=\log\frac{q}{1-q}\Rightarrow q=\frac{\exp(m_{v_b\to c_a})}{\exp(m_{v_b\to c_a})+1}=\frac{1}{2}(1+\tanh(\frac{m_{v_b\to c_a}}{2})) \qquad \text{(A.6)}
$$

$$
1-q=\frac{1}{2}(1-\tanh(\frac{m_{v_b\to c_a}}{2})). \qquad \text{(A.7)}
$$

Using (A.3), (A.4), (A.5), (A.6), and (A.7) we can write

$$
\begin{aligned}
&\tanh(\frac{m_{c_a\to v_j}}{2})=\prod_{v_b\in N(c_a)\backslash v_j}\tanh(\frac{m_{v_b\to c_a}}{2})\\
&\Rightarrow m_{c_a\to v_j}=2\,\mathrm{arctanh}\prod_{v_b\in N(c_a)\backslash v_j}\tanh(\frac{m_{v_b\to c_a}}{2}). \qquad \text{(A.8)}
\end{aligned}
$$

Bibliography

[1] R. G. Gallager, *Low-Density Parity-Check Codes.* Cambridge, MA: MIT Press, 1963.

[2] J. Zhang and M. Fossorier, ”Shuffled belief propagation decoding,” *IEEE Trans. on Comm.*, pp. 53:209-213, Feb. 2005.

[3] M. Rovini, F. Rossi, P. Ciao, N. LInsalata, and L. Fanucci, ”Layered Decoding of Non-Layered LDPC Codes,” *In Proc. 9th EUROMICRO Conference on Digital System Design*, pp. 537-544, August, 2006.

[4] Zhang, J., Wang, Y., Fossorier, M., and Yedidia, J.S., ”Replica Shuffled Iterative Decoding”, *IEEE International Symposium on Information Theory*, pp. 454-458, Sep. 2005

[5] Junsheng Han, Siegel, P.H., Lee, P., and Lin jia-ru, ”Improvement of Shuffled Iterative Decoding,” *IEEE Information Theory Workshop 2006*, Chengdu, China, Oct. 2006

[6] P. Radosavljevic, A. de Baynast, and J.R. Cavallaro, ”Optimized message passing

schedules for LDPC decoding," *In 39th Asilomar Conference on Signals, Systems and Computers*, pp. 591-595, Nov. 2005.

[7] Zhiyong He, Sebastien Roy, and Paul Fortier, "Lowering error floor of LDPC codes using a joint row-column decoding algorithm", *ICC-2007*, Glasgow, Ecosse, june 2007.

[8] Oren Golov and Ofer Amrani, "Edge-based Scheduled BP in LDPC Codes," *ISIT2007*, Nice, France, June 2007

[9] G. Elidan, I. McGraw, and D. Koller, "Residual belief propagation: informed scheduling for asynchronous message passing," *In Proc. 22nd Conference on Uncertainty in Artificial Intelligence*, MIT, Cambridge, MA, July 2006.

[10] A. I. Vila Casado, M. Griot, and R. D. Wesel, "Informed Dynamic Scheduling for Belief-Propagation Decoding of LDPC Codes," *In Proc. IEEE ICC 2007*, Glasgow, Scotland, June 2007.

[11] A. I. Vila Casado, M. Griot, and R. D. Wesel, "Improving LDPC Decoders via Informed Dynamic Scheduling," *IEEE Information Theory Workshop 2007*, Lake Tahoe, CA, Sep. 2007.

[12] E. Zimmermann, P. Pattisapu, P. K. Bora, and G. Fettweis, "Reduced Complexity LDPC Decoding using Forced Convergence," *in Proceedings of the 7th International Symposium on Wireless Personal Multimedia Communications (WPMC04)*, Abano Terme, Italy, pp. 12-15, Sep. 2004.

[13] E. Zimmermann, W. Rave, and G. Fettweis, ”Forced Convergence Decoding of LDPC Codes - EXIT Chart Analysis and Combination with Node Complexity Reduction Techniques,” *in Proceedings of the 11th European Wireless Conference (EW’2005)*, Nicosia, Cyprus, pp. 11-13, April 2005.

[14] E. Zimmermann, P. Pattisapu, and G. Fettweis, ”Bit-Flipping Post-Processing for Forced Convergence Decoding of LDPC Codes,” *in Proceedings of the 13th European Signal Processing Conference (EUSIPCO’05)*, Antalya, Turkey, pp. 04-08, Sep. 2005.

[15] IEEE C802.16e-05/0066r3, ”LDPC coding for OFDMA PHY”.

[16] Satoshi GOUNAI, and Tomoaki OHTSUKI, ”Decoding Algorithms Based on Oscillation for Low-Density Parity Check Codes,” *IEICE Trans. fundamentals*, vol. E88-A, No. 8, Aug. 2005.

[17] T. Tian, C. R. Jones, J. D. Villasenor , and R. D. Wesel, ”Construction of Irregular LDPC Codes with Low Error Floors,” *IEEE Int. Conf. Comm.*, vol. 5, pp 3125-3129, May 2003.

[18] X. Y. Hu, E. Eleftheriou, D. M. Arnold, ”Progressive Edge-Growth Tanner Graphs,” *IEEE GLOBECOM’01.*, vol. 2, pp. 995-1001, Nov. 2001.

[19] H. Xiao and A. H. Banihashemi, ”Improved Progressive-Edge-Growth (PEG) Construction of Irregular LDPC codes,” *IEEE Comm. Letter*, vol.8, pp. 715-717, Dec. 2004.

[20] Sung-Ha Kim, Joon-Sung Kim, Dae-Son Kim, and Hong-Yeop Song, ”LDPC Code Construction with Low Error Floor Based on the IPEG Algorithm,” *IEEE Communications Letters*, vol. 11, no. 7, pp. 607-609, July 2007.

[21] T.J. Richardson, ”Error Floors of LDPC Codes,” *Proc. Allerton Conference on Communication, Control, and Computing*, pp. 1426-1435, 2003.

[22] S.H. Lee, K.S. Kim, Y.H. Kim, and J.Y. Ahn, ”A cycle search algorithm based on a message-passing for the design of good LDPC codes,” *IEICE Trans. fundamentals*, vol. E88-A, pp. 1599-1604, Jun 2005.

[23] S.H. Lee, K.S. Kim, J.K.Kwon, Y.H. Kim, and J.Y.Ahn, ”Design of an LDPC code with low error floor,” *Proc. IEEE Inter. Symp. Inform. Theory (ISIT)*, pp. 990-994, Adelaide, Australia, Sep. 2005.

[24] Kevin R. Jacobson and Witold A. Krzymien, ”Realistic Throughput of Cellular Multi-Hop Relay Networks With Spatial Reuse,” *WPMC 2006, San Diego, CA*, pp. 966-970, Sep. 2006.

[25] Bo Dong, Lin Xie, Peiliang Qiu, and Qinru Qiu, ”LDPC-based Distributed Space Time Cooperative Systems with Non-regenerative Relays,” *Systems and Computers, Conference Record of the Thirty-Ninth Asilomar Conference on*, pp. 1419-1423, Nov. 2005.

[26] Taehyuk Kang, and Volkan Rodoplu, ”Algorithms for the MIMO Single Relay Channel,” *IEEE Trans. on Wireless Commun.*, vol. 6, no. 5, May 2007.

[27] J. Kim, N. Kim, H. Kim, and H. Park, ”A New Transmission Scheme using Turbo Codes in Relay Channels,” *JCCI.* May 2007.

[28] Arnab Chakrabarti, Alexandre de Baynast, Ashutosh Sabharwal, and Behnaam Aazhang, ”Low Density Parity Check Codes for the Relay Channel,” *IEEE Journal on Selected Areas in Commun.*, vol. 25, no. 2, Feb. 2007.

[29] P. Razaghi and Wei Yu, ”Bilayer LDPC codes for the relay channels,” *in Proc. of IEEE Int. Conf. Commun.(ICC)*, nol. 4, pp. 1678-1682, June 2006.

[30] P. Razaghi and Wei Yu, ”Bilayer low-density parity-check codes for decode-and-forward in relay channels,” *Submitted to IEEE Trans. Inf. Theory*, 2006.

[31] C. Hausl and J. Hagenauer, ”Iterative Network and Channel Decoding for the Two-Way Relay Channel”, *In Proc. IEEE International Conference on Communications (ICC 2006)*, Istanbul, Turkey, June 2006.

[32] C. Hausl, F. Schrechenbach, and I. Oikonomidis, ”Iterative network and channel decoding on a tanner graph,” *in Proceedings of the 39th Annual Allerton Conference on Communication, Control and Computing*, 2005.

[33] Xingkai Bao and Jing Li, ”Matching Code-on-Graph with Network-on-Graph: Adaptive Network Coding for Wireless Relay Networks,” *Proc. Allerton Conf. on Commun., Control and Computing IL*, Sep. 2005.

[34] Xingkai Bao and Jing Li, ”On the Outage Properties of Adaptive Network Coded Cooperation (ANCC) in Large Wireless Networks,” *Proceeding of IEEE Interna-*

tional Conference on Accoustic, Speech, and Signal Processing (ICASSP), Toulouse, France, May 2006.

[35] X. Bao and J. Li. ”An information theoretic analysis for adaptive-network-coded-cooperation (ANCC) in wireless relay networks,” *In Proc. IEEE Int. Symp. Information Theory (ISIT)*, Seattle, Washington, July 2006.

[36] Xingkai Bao and Jing Li, ”A Unified Channel-Network Coding Treatment for User Cooperation in Wireless Ad-Hoc Networks,” *ISIT 2006*, Seattle, USA, July 2006.

[37] Peyman Razaghi, and Wei Yu, ”A Structured Generalization of Decode-and-Forward Strategies for the Multiple-Relay Network,” *IEEE International Symposium on Information Theory*, Nice, France, June 2007.

[38] M. A. Khojastepour, A. Sabharwal, and B. Aazhang, ”On capacity of Gaussian ’cheap’ relay channel,” *in Proceedings of IEEE 2003 Global Communications Conference (Globecom-2003)*, 2003.

[39] T. M. Cover and A. A. E. Gamal, ”Capacity theorems for the relay channels,” *IEEE Trans. Inform. Theory*, vol. 25, no. 5, pp. 572-584, Sep. 1979.

[40] P. Razaghi and W. Yu, ”Parity forwarding for multiple-relay networks,” *in Proc. of IEEE Int. Symp. Inform. Theory*, 2006, pp. 1678-1682.

[41] R. Zamir, S. Shamai, and U. Erez, ”Nested linear/lattice codes for structured multiterminal binning,” *IEEE Trans. Inform. Theory*, vol. 48, no. 6, pp. 1250-1276, June 2002.

[42] T, Richardson and R. Urbanke, ”The capacity of low-density parity-check codes under message-passing decoding,” *IEEE Trans. Inform. Theory*, vol. 2, pp. 599-618, Feb. 2001.

[43] T.J. Richardson, M.A. Shokrollahi, R.L. Urbanke, ”Design of capacity-approaching irregular low-density parity-check codes,” *IEEE Trans. on Inf. Theory*, pp. 619-637, Feb. 2001.

[44] J. Ha, D. Klinc, J. Kwon, and S. W. McLaughlin, ”Layered BP Decoding for Rate-Compatible Punctured LDPC Codes,” *IEEE Commun. Letter*, vol. 11, no. 5, pp. 440-442, May 2007.

국 문 요 약

LDPC 부호를 위한 VCRBP 알고리즘

본 논문에서는, LDPC 부호의 복호를 위한 동적 스케줄링 기법인 VCRBP 알고리즘을 제안한다. 제안한 방식은 업데이트 전후의 변화량 (레지듀얼) 이 가장 큰 비트-체크 메시지를 먼저 업데이트 함으로써 기존 RBP 방식에 비해 탁월한 성능 개선을 보인다. 뿐만 아니라, VCRBP는 RBP에서의 불필요한 정렬 과정과 추가적인 계산 과정을 제거함으로써 복잡도를 현저하게 줄였다. 또한, 가장 큰 레지듀얼에 대응되는 비트 노드로부터 나가는 모든 메시지들을 동시에 업데이트 하는 N-VCRBP를 제안한다. N-VCRBP는 낮은 복잡도로 VCRBP에 근접한 성능을 보인다. 복잡도를 더욱 낮추기 위하여, RBP와 VCRBP에 모두 적용 가능한 FC-RBP/VCRBP와 S-RBP/VCRBP를 제안한다. FC-RBP/VCRBP는 이미 수렴이 일어난 메시지를 현재 반복 과정에서 업데이트를 생략 함으로써 복잡도를 줄이고, S-RBP/VCRBP는 레지듀얼 값을 계산하는 대신 부호의 반전이 일어나는 메시지를 먼저 업데이트 하여 복잡도를 줄인다. 모의실험 결과, 제안하는 모든 알고리즘들이 기존 방식들에 비하여 두드러진 성능 개선을 보임을 확인하였다. 특히 길이가 짧은 부호에서 그리고 낮은 반복 횟수에서 더욱 효과가 크게 나타났다.

두 번째 주제는 다중-사용자 릴레이를 위한 결합 LDPC 부호의 설계이다. 결합 LDPC 부호는 높은 오류 정정 성능과 서비스 질을 보장하는 채널-네트워크 부호 방식이다. 기존보다 지능형 릴레이를 사용함으로써 셀 환경에서뿐만 아니라 애드혹 네트워크 환경에서 역시 효과적으로 적용될 수 있다.

핵심되는 말 : LDPC 부호, RBP 알고리즘, 빠른 수렴 복호, 동적 스케줄링 복호, 채널-네트워크 부호, 협력적인 릴레이