Low-Complexity Distributed Source Coding

by

Haishan Wang

A thesis submitted to the
Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of
Master of Applied Science
in Electrical and Computer Engineering

The Ottawa-Carleton Institute for Electrical and Computer Engineering (OCIECE)
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada
May 2011

Your file Votre référence
ISBN: 978-0-494-81709-4
Our file Notre référence
ISBN: 978-0-494-81709-4

The undersigned hereby recommends to
the Faculty of Graduate and postdoctoral Affairs
acceptance of the thesis

Low-Complexity Distributed Source Coding

Submitted by Haishan Wang, B. Eng.
in partial fulfillment of the requirements for the degree of
Master of Applied Science in Electrical and Computer Engineering

---

Thesis Supervisor
Professor Amir H. Banihashemi

Chair, Department of Systems and Computer Engineering
Professor Howard M. Schwartz

Carleton University
May 2011

Abstract

We investigate the low-complexity distributed source coding (DSC) using Low-Density Parity-Check (LDPC) codes in this thesis. We consider a DSC scheme where the compressed source is represented by parity bits and the decoding is performed with the information of the second source as side information. For this scheme, we design a half-regular LDPC code, and apply the min-sum algorithm and its modifications for the recovery of the compressed source. The application of half-regular LDPC codes and (modified) min-sum algorithms reduces the complexity compared to the application of LDPC codes with irregular degree distributions and the decoding algorithm of belief propagation (BP).

We demonstrate that normalized and offset min-sum algorithms both perform very well in the context of DSC and that for short block lengths, their performance with our simple half-regular LDPC code is comparable with the best existing scheme which uses BP and irregular degree distributions. Simulation results on quantized versions of modified min-sum algorithms also indicate that with only 4 quantization bits and with the proper selection of the correction factor, the quantized algorithms perform close to the floating-point versions.

Acknowledgements

I would like to thank my parents, Yongde Wang and Xiumei Zhao, as well as my wife Yanfei Gao, for their love and endless support throughout my research period.

I would like to acknowledge and express my deepest gratitude to my supervisor, Professor Amir H. Banihashemi, for his invaluable assistance, support, guidance and continuous encouragement in my research work, without which this research project would not have been possible.

Many thanks also go to friends and colleagues at the research group and laboratory, especially Hua Xiao and Jinshi Qiu, for their help and suggestions.

Haishan Wang

May 07, 2011
Ottawa, Canada

Table of Contents

List of Figures and Tables

List of Acronyms

BER	Bit Error Rate
BSC	Binary Symmetric Channel
BP	Belief Propagation
DE	Density Evolution
DISCUS	Distributed Source Coding Using Syndromes
DSC	Distribute Source Coding
DISCUP	Distributed Source Coding Using Parity
LDPC	Low-Density Parity-Check
LLR	Log-Likelihood Ratio
MS	Min Sum algorithm

List of Symbols

X, Y	two correlated sources
$P_{X,Y}(x,y)$	the probability density function of sources X and Y
$H(X), H(Y)$	entropies of the two sources
R_X, R_Y	compressed rates for two sources
$H(X \mid Y)$	the entropy of X conditional on Y
p	the crossover probability of a BSC
R	the LDPC code rate
$\mathbf{H}$	the parity-check Matrix
$(\lambda(x), \rho(x))$	the degree distribution pair for the LDPC code
n , m	the variable node and the check node
$Z_{mn}^{(i)}$	the message sent from variable node n to check node m
$L_{mn}^{(i)}$	the message sent from check node m to variable node n
$M(n)$	the set of neighboring check nodes for variable node n
$N(m)$.	the set of neighboring variable nodes for check node m
q	the quantization bits
α	the normalization factor
μ	the offset factor

Chapter 1 Introduction

1.1 Motivation and Objectives

In recent years, wireless sensor networks have created a lot of attention from researchers. A wireless sensor network consists of hundreds to several thousands of small sensor nodes that are scattered throughout a region and are able to take the measurements of environmental variables. Generally, sensor networks are designed to support specific applications, such as detection of biological agents and toxic chemicals, measurement of temperature in a region, real-time area video surveillance, etc. Sensor nodes are very tiny and have limited energy resources. Since the sensor networks sometimes are deployed in inaccessible environments, usually sensor nodes are designed to work throughout their mission without replacing the batteries. This limited amount of energy has a significant impact on all aspects of a wireless sensor network, from the amount of information that the node can process, to the volume of wireless communication it can carry across large distances. To reduce the transmission energy usage of sensor nodes becomes the main goal when we design a sensor network.

Distributed source coding (DSC) is one of the enabling methods for sensor networks to reduce the transmission energy usage [1], [2]. To get the better performance for code compression, various error control codes have been applied to DSC. LDPC codes are one of them and they have better performance than others [3]-[5].

Belief Propagation (BP) is the decoding algorithm for LDPC codes which can have the closest performance to the channel capacity. However, the complexity of BP is also much higher than other decoding algorithms. Our goal is to apply low-complexity decoding algorithms on DSC to achieve the compatible performance. Under this goal, we reviewed two popular methods in distributed source coding which are syndrome approach and parity approach. We choose to use parity approach in this thesis since we can take advantage of knowing channel information from our channel model. We also have designed LDPC codes based on our channel model, given modified Min-Sum (MS) algorithms as decoding algorithm, and compared the performance of our designed system to the results from the previous research.

1.2 Thesis Contributions

In this thesis, we first review pertinent literature describing existing distributed source coding schemes. Then we study syndrome approach and parity approach for distributed source coding. The major contributions of this thesis include:

1. Design of the half-regular Low-Density Parity-Check (LDPC) codes for DSC using parity approach;
2. Applying the min-sum decoding algorithm on binary symmetric channel in the context of source coding for correlated sources;
3. Comparing the performances of designed LDPC codes to non-uniform irregular LDPC codes, uniform irregular LDPC codes and DISCUS using LDPC codes;

4. Applying the modified min-sum decoding algorithm in the context of distributed source coding;
5. Optimizing the clipping threshold for quantized MS algorithm.
6. Studying the performance of modified MS algorithm with quantized messages using designed LDPC codes.

1.3 Thesis Organization

In Chapter 2, we present the background and literature review related to the thesis work. First of all, we introduce how distributed source coding is introduced and explain Slepian-Wolf coding Theorem. Slepian-Wolf theorem gives the theoretic bounds on the independent compression of two correlated sources. Secondly, we give a brief review of LDPC codes and their popular decoding algorithm. Finally, we discuss the application of LDPC codes to DSC, where there are two different approaches, syndrome approach and parity approach. The thesis focuses on parity approach.

In Chapter 3, we focus on low complexity decoding algorithms for DSC. The problem is formulated first, and then we represent the correlation model for the channel. After that, we design LDPC codes based on the parity approach. Then we analyze messages passed between nodes. In the end, we discuss the improved min-sum algorithms.

In Chapter 4, the simulation results are given. We compare the performance among various decoding algorithms and also compare our results with previous performance

from published literature. The results show that even with much simpler LDPC codes and low complexity decoding algorithms, we still get a compatible performance.

In Chapter 5, we conclude the thesis and give a few suggestions for future work.

Chapter 2 Background and Literature Review

2.1 Distributed Source Coding

As we mentioned in Chapter 1, one of the technologies that is commonly used to reduce the transmission energy usage in wireless sensor networks is distributed source coding (DSC). Distributed source coding is referred to as the compression of multiple correlated sensor outputs that do not communicate with each other. These sensors send their compressed outputs to a receiver or central point for joint decoding, which can reconstruct outputs correctly [1]. The transmission of compressed outputs consumes less energy, which achieves the goal for designing a sensor network.

To explain distributed source coding, we will start from discrete sources with joint encoding. After that, we will discuss the Slepian-Wolf coding theorem.

2.1.1 Discrete Sources with Joint Encoding

We consider discrete sources are not connected to each other physically. For example, X and Y are correlated discrete independent and identically distributed (*i.i.d.*) sources. X_i and Y_i, $i=1,\cdots,\infty$ are a sequence of drawings from the sources X and Y, respectively. For lossless compression, we want to get $\hat{X}=X$ and $\hat{Y}=Y$ after decompression. If the sources X and Y are jointly encoded, which means each encoder has the value of both X and Y, we know from Shannon's source coding theory that a

rate given by the joint entropy $H(X,Y)$ of X and Y is sufficient. Figure 2-1 shows that two encoders collaborate and sources X and Y are jointly encoded.

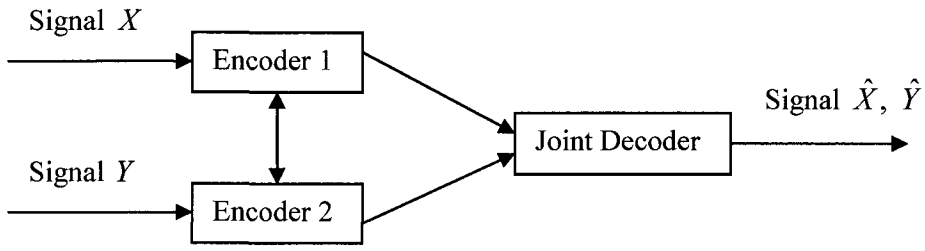


Figure 2-1 Joint encoding of X and Y

Since $H(X,Y) = H(Y) + H(X \mid Y)$, we could first compress Y into $H(Y)$ bits per sample. Then based on the full information of Y at both the encoder and the decoder, we compress X into $H(X \mid Y)$ bits per sample. Thus, we jointly compress the sources X and Y to a rate that equals to joint entropy $H(X,Y)$. After the decompression at the decoder, we can completely reconstruct both X and Y.

2.1.2 Slepian-Wolf Coding Theorem

Joint entropy $H(X,Y)$ of X and Y is sufficient to reconstruct both X and Y after jointly decoding when sources X and Y are jointly encoded. But for a distributed source coding problem, what if the sources X and Y are separately encoded for some reasons? Figure 2-2 shows the case where the encoders do not collaborate and sources X and Y are encoded independently.

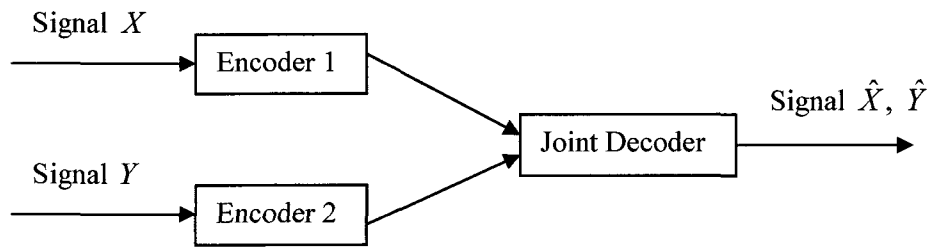


Figure 2-2 Distributed encoding of X and Y

If we encode X and Y independently with entropy $H(X)$ and $H(Y)$, we have compression rate $R = H(X) + H(Y)$, which is in general greater than joint entropy $H(X,Y)$. However, Slepian and Wolf [1] showed that compression rate $R = H(X,Y)$ is sufficient to reconstruct information successfully even for separate encoding of correlated sources. The Slepian-Wolf coding theorem says that with the joint distribution $p(x,y)$ of the discrete sources (X,Y), there are the information theoretic bounds on the independent compression of two correlated sources. The achievable compression rate region is given by:

$$R_1 \geq H(X \mid Y),\ R_2 \geq H(Y \mid X),\ R_1 + R_2 \geq H(X,Y),$$

which is shown in Figure 2-3. Any compression rate located in the achievable rate region can ensure successful reconstruction of sources through joint decoding. We will focus on compression of X with side information Y at the Joint Decoder.

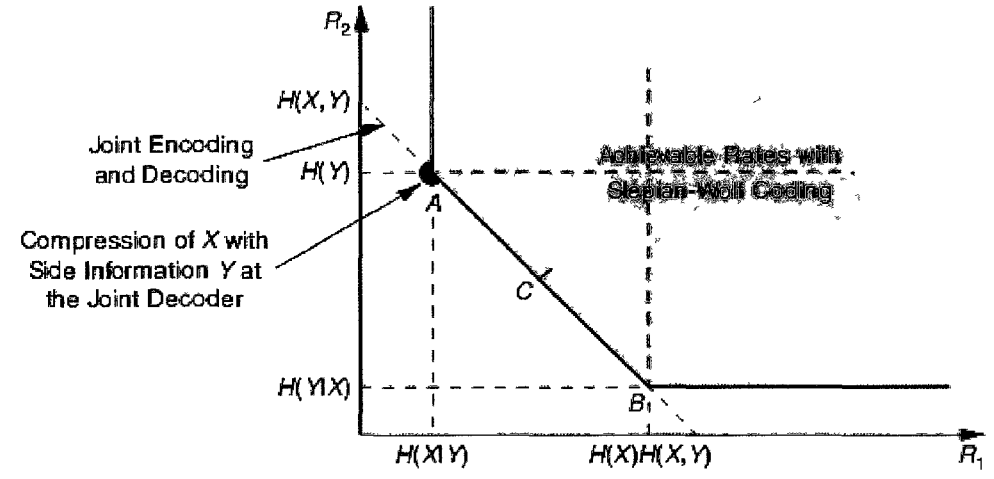


Figure 2-3 The Slepian-Wolf rate region for two sources [2]

Although Slepian and Wolf proved the Slepian-Wolf theorem based on random binning, the proof of the theorem is asymptotic and non-constructive. Just as Shannon's channel coding theorem does not show us how to construct a practical code; the Slepian-Wolf theorem does not either. Based on the Slepian-Wolf theorem, the linear codes have been researched to achieve the Slepian-Wolf bound since the early 1980s by constructing a practical Slepian-Wolf code [6]. Most recently, frameworks that can reach the Slepian-Wolf bound, using turbo codes and low-density parity-check (LDPC) codes, have been developed. We will discuss the details of code constructions later.

2.2 Low-density parity-check (LDPC) codes

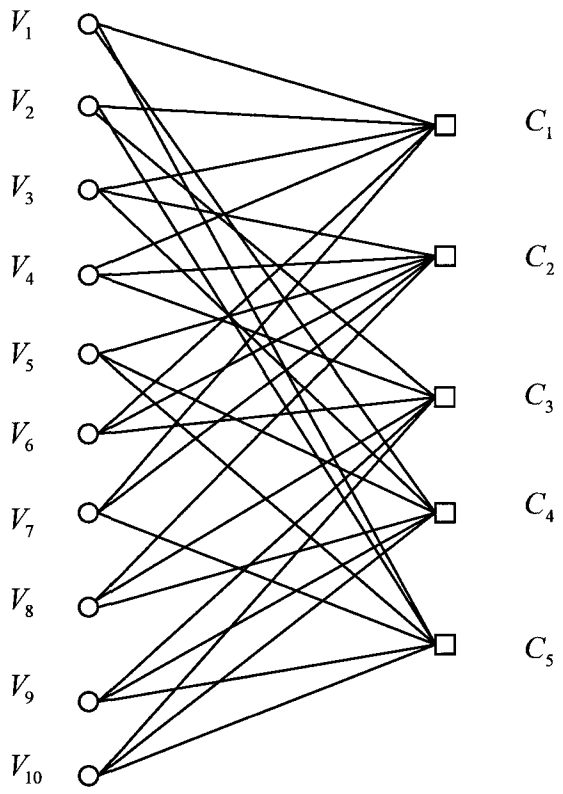
Low-density parity-check (LDPC) codes were originally proposed in the 1960s by Robert Gallager [3] and they have been strongly promoted and used to solve the channel capacity problems recently [4], [5]. LDPC codes have a pseudo-random structure and the

iterative decoding algorithms are used to decode them. The structure and algorithms are easy to implement and are also near-optimal, which makes many researchers choose LDPC codes for different applications.

LDPC codes are linear block codes, best described by their parity-check matrix **H** and the associated Tanner graph, shown in Figure 2-4, which is also a bipartite graph. The parity-check matrix **H** of a binary LDPC code is sparse; that is, it consists mainly of 0s and a small number of 1s (hence low density). The Tanner graph of an LDPC is an equivalent representation of the parity-check matrix **H**. It consists of two sets of nodes. The first set consists of n variable nodes that represent the n bits of a codeword. The second set consists of m nodes, called check nodes that represent the parity constraints. The graph has edges between n variable nodes and m check nodes. The number of neighboring nodes that a node is connected to is said to be its degree. The regular LDPC codes have the same degree for all the variable nodes and check nodes, respectively, but the degree of nodes in the irregular LDPC codes vary. The Tanner graph is also the fundamental base of the message-passing algorithm, which is the most commonly used decoding algorithm for LDPC codes. Figure 2-4 is the Tanner graph for a (3, 6) regular LDPC code of length 10 and rate ½. There are 10 variable nodes and 5 check nodes.

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

(a)



(b)

Figure 2-4 (a) **H** matrix, (b) Tanner Graph

Irregular LDPC codes can be described by the degree distribution pairs ($\lambda(x)$, $\rho(x)$). Both $\lambda(x)$ and $\rho(x)$ are polynomials which have the form $\mu(x) = \sum_{i\geq 2} \mu_i x^{i-1}$, where μ_i is the fraction of edges that are connected to nodes with degree i. $\lambda(x)$ specifies the degrees of the variable nodes and $\rho(x)$ specifies the degrees of the check nodes. Given both $\lambda(x)$ and $\rho(x)$, the code rate is determined, but there are many channel codes, and thus many parity-check matrices that can be formed. They all follow the same degree distributions. Usually one such matrix is constructed randomly.

In general, any parity check matrix satisfying the degree distributions will perform well, but the presence of short cycles in the Tanner graph degrades the performance because they create dependencies among the messages and cause the decoding algorithms to perform sub-optimally. In some cases, some edges may have to be rearranged to obtain a code with better performance. Overall, an optimized irregular LDPC code is expected to have better performance than a regular one of the same codeword length and code rate.

2.2.1 LDPC Decoding Algorithms

Among a variety of decoding algorithms, belief propagation (BP) or sum-product algorithm [4], [5] is the well-known iterative message-passing algorithm, which can achieve a good decoding performance. Standard BP algorithm needs to do large amount of logarithmic, exponential, and multiplicative computations in the log-likelihood ratio (LLR) domain. These computations increase hardware complexity for the implementation.

Min-Sum (MS) [22] was introduced as an alternative method which can significantly reduce the hardware complexity by using the simple comparison and summation operations to replace the complex computations in BP, with a cost of performance degradation. Therefore, more research has been done to improve the decoding performance of the MS algorithm. Recently, improved MS algorithms such as normalized MS or offset MS using additional correction factors have been preferred for many practical applications since they offer comparable decoding performance to the BP algorithm for LDPC codes [7]-[10].

This thesis will focus on the improved MS algorithms and we will discuss the details in the later chapters.

2.3 Distributed Source Coding with LDPC codes

In the late 1990s, Distributed Source Coding Using Syndromes (DISCUS) was introduced by Pradhan and Ramchandran [11], who constructed a framework using the syndromes of linear codes to achieve the Slepian-Wolf bound. Later on, LDPC codes were commonly used in DISCUS [12]-[14]. More recently, Sartipi and Fekri gave another similar approach with LDPC codes to achieve the bound by using parity bits instead of syndromes [15]-[17].

2.3.1 Distributed Source Coding Using Syndromes (DISCUS)

Distributed Source Coding Using Syndromes is a linear block coding framework designed for distributed source coding of correlated sources. Suppose X and Y are two statistically dependent sources, which are i.i.d. equal-probable binary random variables. The relationship between sources X and Y can be fully described by the probability mass function $P_{X,Y}(x,y)$. Since this thesis focuses on the asymmetric distributed source coding scenario, based on the Slepian-Wolf theorem, the signal Y can be compressed conventionally and sent at full rate $H(Y)$, and it is recovered perfectly at the decoder. Then the signal X can be compressed as close as possible to the Slepian-Wolf bound of $H(X|Y)$. The system of DISCUS is shown in Figure 2-5, where syndromes S are sent through a lossless channel.

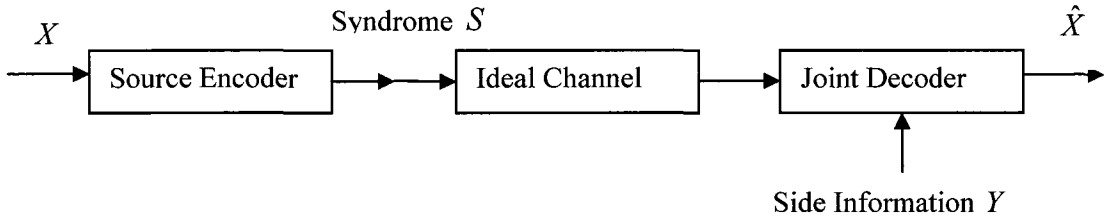


Figure 2-5 System for DISCUS with side information

The correlation between signals X and Y can be modeled as the input and output of a binary symmetric channel (BSC) with the crossover probability of $P[X \neq Y] = p$. For BSC, we also have $H(X \mid Y) = H(p) = -p\log_2 p - (1-p)\log_2(1-p)$. Figure 2-6 is equivalent to Figure 2-5.

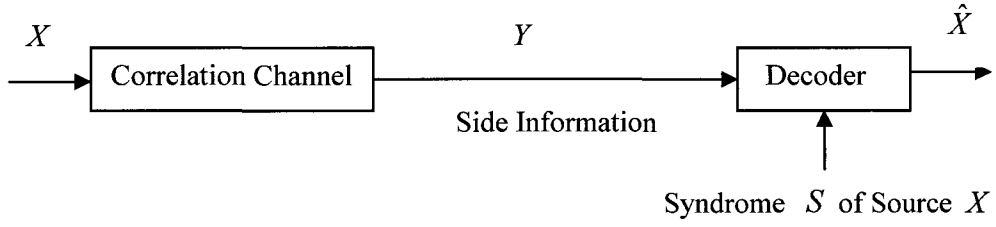


Figure 2-6 Equivalent correlation channel for DISCUS

To encode an N-bit sequence of source X, we multiply X with a given parity check matrix $\mathbf{H}$ to find the corresponding syndrome S. $\mathbf{H}$ is the parity-check matrix of an (n,k) linear binary block code. S is sent through a lossless channel. At the decoder side, we must determine the N-bit sequence of X from its $(N-K)$-bit syndrome S and the corresponding N-bit sequence of Y, which is compressed conventionally and recovered perfectly at the decoder.

In order to apply message passing algorithm or BP algorithm, we need to know the log-likelihood ratio (LLR) for all the bits. The correlation model is BSC with crossover probability of p, X is the input of the channel and Y is the output. Therefore, the LLR is equal to $\pm\log\frac{1-p}{p}$, where the sign is decided by the BSC output value Y. Then by knowing the LLR's of all bits, we can use BP algorithm to estimate the signal sequence X.

2.3.2 Distributed Source Coding Using Parity (DISCUP)

Distributed Source Coding Using Parity (DISCUP) has a framework similar to that of DISCUS. The block diagram is shown in Figure 2-7. The difference is that parity bits are sent through a lossless channel in DISCUP instead of syndromes.

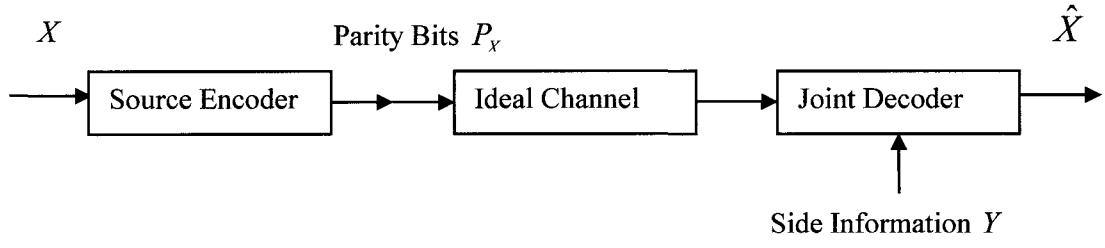


Figure 2-7 System for DISCUP with side information

To encode a K-bit sequence of source X, we use X as the input of a systematic encoder which has a generator matrix $\mathbf{G}$. Given parity check matrix $\mathbf{H}$ for an (n,k) binary linear block code, generator matrix $\mathbf{G}$ can be obtained via Gaussian Elimination with a possible column permutation to $\mathbf{H}$. The output of the encoder includes X and

$(N-K)$-bit parity sequence P_X. Only parity bits P_X will be sent through a lossless channel.

At the decoder side, the K-bit sequence Y is attached to parity bits P_X to form a codeword with noisy systematic part. This becomes the input to the decoder. The decoder will determine the original sequence X based on the noisy codeword. Since the input vector to the decoder has two parts, information bits and parity bits, it can be treated as being passed through two parallel channels, which is shown in Figure 2-8.

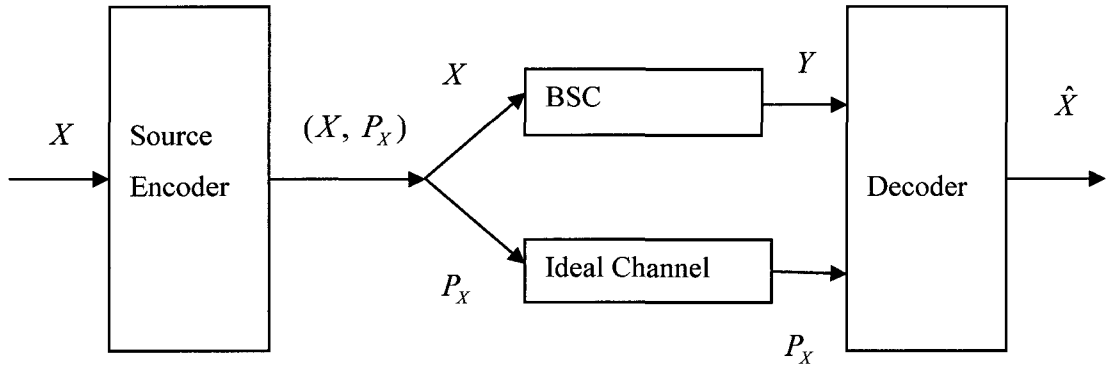


Figure 2-8 Parallel Channel Model of DISCUP

Both syndrome and parity approaches can be used for distributed source coding to approach the Slepian-Wolf bound using LDPC codes. However, the parity approach can be used to design LDPC codes which take advantage of the parallel channel model since we have the prior information as to which set of bits pass through which channel. Sartipi and Fekri in [15]-[17] designed a non-uniform LDPC code by considering the fact that

different bits in signal are passed through different channels. A non-uniform LDPC code can have two or more independent sub-polynomials for each polynomial in the degree distribution pairs. It has been shown that the gap to the theoretical Slepian-Wolf bound of non-uniform LDPC codes is 60% smaller than DISCUS using LDPC codes [18]. So in this thesis, we will focus on the parity approach.

Our goal is to design a simpler LDPC code with less variation in the degree distribution, which will make the implementation simpler, but the performance of the codes with modified min-sum decoding algorithms is still close to the irregular codes designed in [17]. The modified min-sum algorithms used in the thesis are much simpler than the BP algorithm to implement. These algorithms have not been used in the context of DSC before, and this thesis is the first one that studies the performance of these algorithms in DSC context.

We will discuss details of modified min-sum algorithms in the next chapter.

Chapter 3 Low-Complexity Distributed Source Coding

In this chapter, we will design an LDPC code for distributed source coding using parity approach and apply low-complexity LDPC decoding algorithms, which are min-sum algorithm and its modifications mentioned before. We will also compare our system performance with previous work in [17], [18].

There are four sections in this chapter. The problem is formulated first, and then we will present the correlation model to calculate the LLR for all signal bits. After that, we will design the LDPC codes based on the parity approach. Since min-sum algorithm is applied on binary symmetric channel (BSC), we will analyze messages passed between nodes. In the end, we will discuss the improved min-sum algorithms which improve the decoding performance while keeping the complexity relatively low.

3.1 Problem Formulation

Consider a system which is shown in Figure 3-1. This system has the following assumptions, which are used for the rest of this thesis.

1. X and Y are two correlated sources in the system. Both of them are i.i.d. equal-probable binary random sequences with length K. It is applicable if the two sources are quantized and already compressed with a robust binary source coding scheme.

2. The correlation between X and Y is modeled as the input and output of a binary symmetric channel (BSC) with crossover probability $P[X \neq Y] = p$.

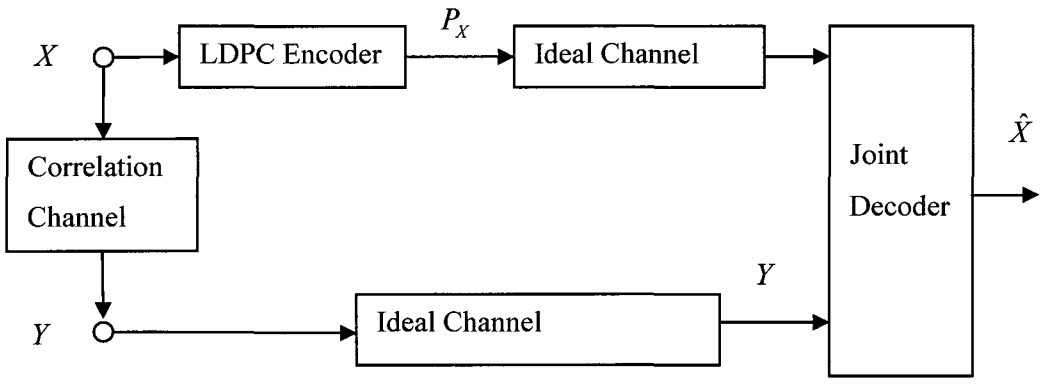


Figure 3-1 System for compression of X with the side information Y

3. R_X and R_Y are compression rates for sources X and Y.

4. Signal Y is compressed conventionally and sent at full rate $R_Y = H(Y) = 1$. It is recovered perfectly and available for the decoder.

5. Signal X is compressed with the compression rate following the Slepian-Wolf bound $R_X \geq H(X \mid Y) = H(p)$.

6. P_X is the parity bits of sequence X generated by an (n,k) LDPC code encoder, which has generator matrix $\mathbf{G}$ and the corresponding parity-check matrix $\mathbf{H}$. Both of them are in the systematic form.

7. R is the code rate for the LDPC code. Then we have

$$R_X = \frac{n-k}{k} = \frac{n}{k} - 1 = \frac{1}{R} - 1 \tag{3.1}$$

With these assumptions, we will try to get the best possible performance of the system by designing an LDPC code and applying low-complexity decoding algorithms.

3.2 LDPC Code Design

From the previous section, we know that the parity approach for DSC can be demonstrated as a system having two independent parallel channels, shown in Figure 2-8. Information bits and parity bits pass through a BSC and a noiseless channel, respectively. In order to take advantage of prior knowledge as to which set of bits passed through which channel, a non-uniform LDPC code has been designed in [15], [17]. A standard LDPC code has a degree distribution pair $(\lambda(x), \rho(x))$, where $\lambda(x)$ denotes the variable-node degree distribution and $\rho(x)$ is for the check-node. The non-uniform LDPC code has two degree distributions $\lambda^1(x)$ and $\lambda^2(x)$ for information bits and parity bits, respectively. Then the degree distribution pair becomes $((\lambda^1(x), \lambda^2(x)), \rho(x))$, where $\rho(x)$ is still for the check nodes.

Irregular degree distributions have been used in [17] for designing the non-uniform LDPC code. Since we focus on low-complexity DSC, we choose regular degree distributions to design the non-uniform LDPC code used in this thesis. Irregular LDPC codes generally have better performance than regular codes. However they come at the expense of increased complexity in the implementation. Regular LDPC codes have relatively good performance and low implementation complexity, and have applications such as for magnetic recording [24]. The check nodes and variable nodes with the same

degree equalize the delay of all the nodes and would improve the throughput of the decoder.

We thus consider the case where the information bits have the same column weight, so do parity bits. Then we have the following degree distribution pair for the designed non-uniform LDPC code,

$$\begin{aligned} \Lambda(x) &= \{\lambda^1(x), \lambda^2(x)\} = \{x^{d_1-1}, x^{d_2-1}\} \\ \rho(x) &= \{x^{d_c-1}\} \end{aligned}, \tag{3.2}$$

where d_1 is the column weight of information bits in variable nodes, and d_2 is for parity bits. d_c is the row weight of check nodes.

In order to compare the system performance with the previous work, we set the designed LDPC code rate at $R = 2/3$, although the same approach can also be used for other code rates. The compression rate R_X is related to code rate by (3.1), and we thus have $R_X = 1/2$. Then, if $\overline{d_v}$ is the average column weight of variable nodes, we have

$$\begin{aligned} & R = 1 - \frac{\overline{d_v}}{d_c} = \frac{2}{3} \text{ and } \overline{d_v} = \frac{2}{3} d_1 + \frac{1}{3} d_2 \\ & \Rightarrow d_c = 2d_1 + d_2 \end{aligned}. \tag{3.3}$$

Since parity bits pass through the noiseless channel in the parallel channel model, the average degree of the information bits must be higher than that of the parity bits. This will provide the information bits with more protection. To use the simplest LDPC code, we set $d_1 = 3, d_2 = 2$ and get $d_c = 8$. Then our designed LDPC codes have the following degree distribution pair

$$\Lambda(x) = \{\lambda^1(x), \lambda^2(x)\} = \{x^2, x^1\}$$
$$\rho(x) = \{x^7\} \quad . \tag{3.4}$$

3.3 Min-Sum Algorithm for the non-uniform LDPC codes over BSC

There are two main reasons why we choose min-sum algorithm for DSC. First, we are focusing on low-complexity DSC. Choosing MS algorithm will decrease the decoding complexity. We would also like to obtain get performance compatible with that of BP by using modified or improved MS algorithms. Second, the channel model between two sources is BSC, and while there are results on the application of MS over BSC [25], modified MS algorithms have not been studied over BSC to the best of our knowledge. In this thesis, we will examine the performance of MS and modified MS algorithms over the BSC.

3.3.1 Min-Sum Algorithm

In our framework, we have a binary (n,k) non-uniform LDPC code whose information and parity bits are transmitted over a BSC and an error-free channel, respectively. Let L_n be the log-likelihood ratio (LLR) of bit n. At iteration i, we denote $Z_{mn}^{(i)}$ and $L_{mn}^{(i)}$ as the message sent from variable node n to check node m and the message sent from check node m to variable node n, respectively [4]. We also denote the set of neighboring check nodes for variable node n as $M(n)$, and the set of neighboring variable nodes for check node m as $N(m)$. Then the check node and variable node processing in each iteration i of standard BP decoding include the following two steps.

(i) Horizontal Step - check nodes processing, for each m and each $n \in N(m)$, update

$$L_{mn}^{(i)} = 2\tanh^{-1}\left(\prod_{n' \in N(m)\backslash n} \tanh\left(\frac{Z_{mn'}^{(i-1)}}{2}\right)\right) \tag{3.5}$$

(ii) Vertical Step – variable node processing, for each n and each $m \in M(n)$, update

$$Z_{mn}^{(i)} = L_n + \sum_{m' \in M(n)\backslash m} L_{m'n}^{(i)} \tag{3.6}$$

Check node processing in BP algorithm require a large amount of logarithmic, exponential and multiplicative computations. MS algorithm simplifies the update rule in check nodes by modifying (3.5) into

$$L_{mn}^{(i)} = \prod_{n' \in N(m)\backslash n} \operatorname{sgn}\left(Z_{mn'}^{(i-1)}\right) \cdot \min_{n' \in N(m)\backslash n} \left|Z_{mn'}^{(i-1)}\right| \tag{3.7}$$

Figure 3-2 shows the graphical representation of message-passing in variable nodes and check nodes. The number of incoming messages for information nodes and parity nodes are different, $d_1 - 1$ and $d_2 - 1$, respectively. The channel LLRs are also different, which will be discussed in the next section.

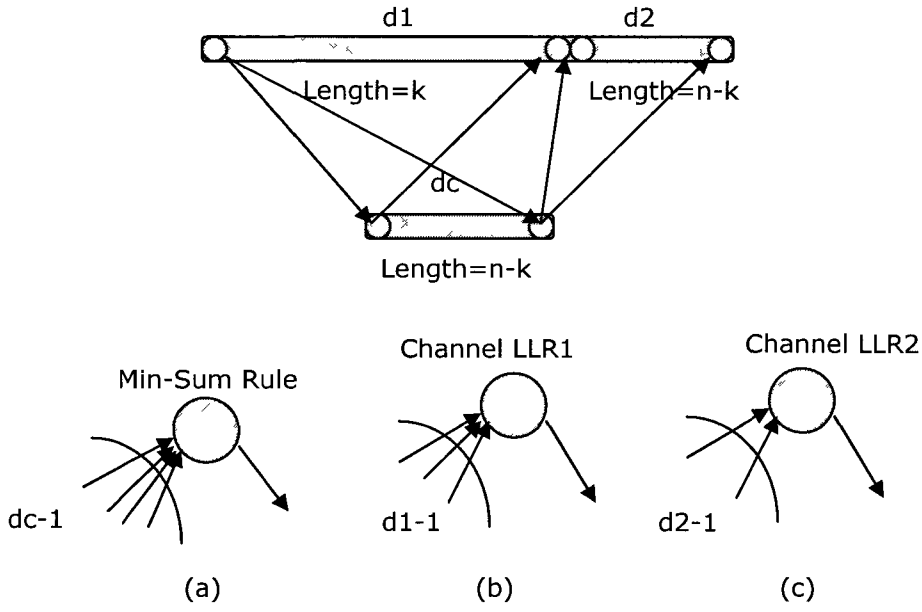


Figure 3-2 Processing in check nodes (a), information nodes (b) and parity nodes (c)

3.3.2 Parallel Channels and LLR

In order to apply MS algorithm to an LDPC code, we first need to know the LLR of every bit. The information bits of the codeword are transmitted through BSC, which is shown in Figure 3-3(a) and has crossover probability $p < 0.5$. Parity bits pass through a noiseless channel, which is shown in Figure 3-3(b).

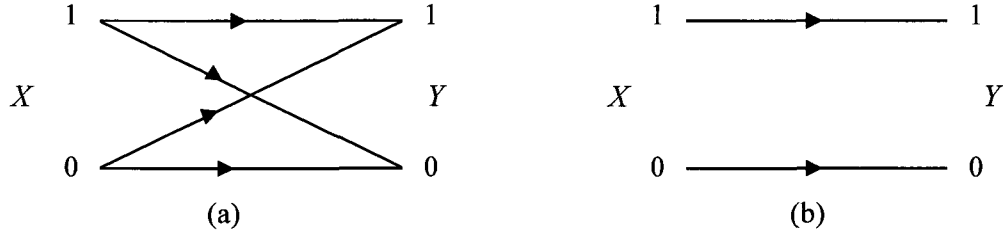


Figure 3-3 Two parallel channels for information bits and parity bits.

To simplify the analysis, the codeword is assumed to be all-zero. It is equivalent to send codeword with random bits since both the channel and the decoder are symmetric [26]. Therefore, the LLR of the information bits are

$$\begin{aligned} LLR_{y=0} &= \ln \frac{P(y=0 \mid x=0)}{P(y=0 \mid x=1)} = \ln \frac{1-p}{p} \\ LLR_{y=1} &= \ln \frac{P(y=1 \mid x=0)}{P(y=1 \mid x=1)} = \ln \frac{p}{1-p} \end{aligned} \tag{3.8}$$

Let $\ln \frac{1-p}{p} = \beta$ $(p < 0.5,\ \beta > 0)$. Then LLRs at the information bits have two values, $+\beta$ with probability $1-p$ and $-\beta$ with probability p.

Since the parity bits of the codeword are transmitted through the noiseless channel and all zero codeword is transmitted, LLR of all bits is $+\infty$ with probability 1.

3.3.3 Messages Passed between Nodes

Since we apply MS algorithm, it is possible to track the values of messages passed between variable nodes and check nodes. Suppose S is used to denote the set of message values passed between variable nodes and check nodes. Then at iteration i, $S_I^{(i)}, S_p^{(i)}$ and $S_q^{(i)}$ represent the sets of output values of information bit nodes, values of incoming and output of check nodes, respectively.

Before we start to track down all the messages passed between nodes, we first assume d_1 is an odd number for the general case. Then we study the case $d_1 = 3, d_2 = 2$ and $d_c = 8$. At the starting point of iterations, LLRs at the information bits

have two values, $+\beta$ with probability $1-p$ and $-\beta$ with probability p. LLR at the parity bits are all the same, $+\infty$.

Therefore, $S_I^{(0)} = \{+\beta, -\beta\}$ and $S_p^{(0)} = \{+\beta, -\beta, +\infty\}$ where $+\infty$ is the value from parity bit nodes. According to the MS algorithm, the output messages of check nodes also have three values $+\beta$, $-\beta$ and $+\infty$, $S_q^{(0)} = \{+\beta, -\beta, +\infty\}$.

Based on the incoming messages to variable nodes and the known channel LLR, we can find out the output messages of all the variable nodes. At the iteration 1, based on the "sum" rule, the output messages of information bit nodes could be $+\infty$ if at least one of the incoming messages is $+\infty$. If none of the incoming messages is $+\infty$ and d_1 is an odd number, the values also could be $+d_1\beta, +(d_1-2)\beta, \cdots, +\beta, -\beta, \cdots, -(d_1-2)\beta, -d_1\beta$. The output message of parity bits is always $+\infty$ since the channel LLR for these bits is $+\infty$.

Then mathematically at iteration 1,

$$\begin{aligned} S_I^{(1)} &= \{+d_1\beta, +(d_1-2)\beta, \cdots, +\beta, -\beta, \cdots, -(d_1-2)\beta, -d_1\beta, +\infty\} \\ &= \{3\beta, \beta, -\beta, -3\beta, +\infty\}, \text{ if } d_1 = 3. \\ S_p^{(1)} &= S_I^{(1)} \\ S_q^{(1)} &= S_p^{(1)} = S_I^{(1)} \end{aligned} \tag{3.9}$$

Similarly, we can get the values at iteration 2,

$$\begin{aligned} S_I^{(2)} &= \begin{Bmatrix} +(2d_1+1)\beta, +(2d_1+1-2)\beta, \cdots, +\beta, -\beta, \cdots, \\ -(2d_1+1-2)\beta, -(2d_1+1)\beta, +\infty \end{Bmatrix} \\ &= \{7\beta, 5\beta, 3\beta, \beta, -\beta, -3\beta, -5\beta, -7\beta, +\infty\}, \text{ if } d_1 = 3. \\ S_p^{(2)} &= S_I^{(2)} \\ S_q^{(2)} &= S_p^{(2)} = S_I^{(2)} \end{aligned} \tag{3.10}$$

At iteration 3, the set become even larger.

$$S_I^{(3)} = \begin{Bmatrix} +(2(2d_1+1)+1)\beta, +((2(2d_1+1)+1)-2)\beta, \cdots, +\beta, \\ -\beta, \cdots, -((2(2d_1+1)+1)-2)\beta, -(2(2d_1+1)+1)\beta, +\infty \end{Bmatrix}$$
$$= \begin{Bmatrix} 15\beta, 13\beta, 11\beta, 9\beta, 7\beta, 5\beta, 3\beta, \beta, -\beta, -3\beta, \\ -5\beta, -7\beta, -9\beta, -11\beta, -13\beta, -15\beta, +\infty \end{Bmatrix}, \text{ if } d_1 = 3 \tag{3.11}$$

We use $n^{(i)}$ to represent the number of output values of information bit nodes except the value $+\infty$ at the ith iteration. Then $n^{(0)} = 2$ and $n^{(i+1)} = \left(\left(n^{(i)} - 1\right)\left(d_1 - 1\right) + 1\right) + 1$ where $d_x = \left(n^{(i)} - 1\right)\left(d_1 - 1\right) + 1$ is the coefficient of the largest value $+d_x\beta$.

For example, for $d_1 = 3$, we have $n^{(1)} = 4$, $n^{(2)} = 8$, $n^{(9)} = 1024$, $n^{(15)} = 65534$. For $d_1 = 5$, we have $n^{(1)} = 6$, $n^{(2)} = 22$, $n^{(9)} = 349526$, $n^{(15)} = 1431655766$.

The above analysis is based on d_1 being an odd number. We also have the same situation when d_1 is an even number, and the relationship for the size of the message alphabet space is $n^{(0)} = 2$ and $n^{(i+1)} = \left(\left(n^{(i)} - 1\right)\left(d_1 - 1\right) + 1\right) + 1$, which is exactly the same as the case where d_1 is odd.

From the analysis above, we can clearly see that the number of possible output values of information bit nodes increases exponentially as the iteration i increases. In the following, we discuss the application of a quantized version of MS algorithm with $q+1$ bits. Then 2^{q+1} values, including $+\infty$, can be represented in the min-sum algorithm. If any value of the messages is beyond this range, we will truncate it to one of the end

values of the quantization range. If q is large enough, the performance would be very close to the case without quantization.

3.3.4 Quantized Min-Sum Algorithm

As discussed above, quantized min-sum algorithm using $q+1$ bits can represent 2^{q+1} values, which are in the set

$$\left\{ -(2^q-1)\beta,\ \cdots,\ -\beta,\ 0,\ \beta,\ \cdots,\ +(2^q-1)\beta,\ +\infty \right\}.$$

The messages whose values are larger than $+(2^q-1)\beta$ will be clipped to the value $+(2^q-1)\beta$, and the messages whose values are less than $-(2^q-1)\beta$ will be clipped to the value $-(2^q-1)\beta$. One thing worth mentioning is that there is no value $-\infty$ in our designed system since we assume all zero codeword is transmitted. However, we do need to represent $-\infty$ in reality.

There are two benefits for quantized min-sum algorithm. First, it is easy to implement. Instead of storing the actual values of the passed messages, we can use the integer values represented by quantization bits to represent them and have the same decoding results. Second, the performance of quantized min-sum algorithm will be very close to floating-point min-sum when q is large enough. We will give the details in the next chapter through simulation results.

3.4 Improved Min-Sum Algorithm

Comparing with standard BP (sum-product) algorithm, Min-Sum (MS) algorithm significantly decreases the complexity during decoding, but the performance of min-sum algorithm also drops compared with BP algorithm.

Analyzing BP algorithm and MS algorithm, both of them have the same variable node update rule. The difference between them is in the check node updating rule. If L_1 and L_2 represent the values $L_{mn}^{(\iota)}$ computed by BP algorithm with (3.5) and MS algorithm with (3.7) respectively, the following two statements indicated in [19] are true.

1. L_1 and L_2 have the same sign, $\text{sgn}(L_1) = \text{sgn}(L_2)$.
2. L_2 has a larger magnitude than L_1, $|L_2| \geq |L_1|$.

Based on these two statements, two improved MS algorithms are developed, which are normalized MS or offset MS [7], [8], [9], [10].

3.4.1 Normalized MS Algorithm

A normalization factor α is used in the check node processing rule. The value $L_{mn}^{(\iota)}$ is divided by the factor α by modifying (3.7) to

$$L_{mn}^{(\iota)} \leftarrow L_{mn}^{(\iota)} / \alpha \tag{3.12}$$

where $\alpha > 1$. To optimize the performance, the factor should vary for each iteration and node degree. We set the factor α here to be a constant value for simplicity.

3.4.2 Offset MS Algorithm

Offset MS is another approach to improve the performance. The check node processing rule is changed as below

$$L_{mn}^{(i)} \leftarrow \operatorname{sgn}(L_{mn}^{(i)}) \cdot \max(\left|L_{mn}^{(i)}\right| - \mu,\ 0) \tag{3.13}$$

where μ is a positive constant value. The offset MS improves the accuracy of the extrinsic messages by reducing the reliability values. All reliability values which are smaller than the constant factor are set to 0. Therefore, those values have no contribution to the following variable node processing.

3.4.3 MS Algorithm with Two-Dimensional Correction

Normalized and offset MS algorithms are considered as one dimensional correction since they only modify the check node processing rule. Two dimensional corrections for MS on irregular LDPC codes were implemented in [20] to use the degree differences.

Two dimensional normalized MS algorithm not only modifies the check node processing rule, but also modifies the variable node processing rule as below

$$Z_{mn}^{(i)} = L_n + \frac{1}{\alpha_v} \cdot \sum_{m' \in M(n) \backslash m} L_{m'n}^{(i)} \tag{3.14}$$

where α_v is the normalization factor for variable nodes.

Similarly, we can have the variable node processing rule for two dimensional offset MS algorithm as below

$$Z_{mn}^{(i)} = L_n + \operatorname{sgn}(S_{L_{m'n}^{(i)}}) \cdot \max(\left|S_{L_{m'n}^{(i)}}\right| - \mu_v, 0) \tag{3.15}$$

where $S_{L_{m'n}^{(t)}} = \sum_{m' \in M(n) \setminus m} L_{m'n}^{(t)}$, and μ_v is the offset factor for variable nodes.

The research [20] on MS algorithm with two-dimensional correction also indicates that only one dimensional correction is needed if the code is either bit-regular or check-regular. Since our designed LDPC code is half regular, we will focus on one dimensional correction, normalized and offset min-sum algorithms.

Chapter 4 Simulation Results for Low-complexity DSC

In this chapter, we give simulation results for distributed source coding of two correlated sources under different scenarios. First, we simulate distributed source coding with min-sum algorithm using floating-point messages. Then we apply the modified MS algorithm to compare the improvement in the performance. We also simulate the quantized MS on DSC which uses the quantized messages represented by quantization bits during decoding.

As we discussed in Chapter 3, all the LDPC codes used for simulation in this chapter are designed LDPC codes which are randomly chosen from the ensemble defined by the following degree distribution

$$\begin{aligned} \Lambda(x) &= \{\lambda^1(x), \lambda^2(x)\} = \{x^2, x^1\} \\ \rho(x) &= \{x^7\} \end{aligned} . \tag{4.1}$$

During the code construction, we also removed the cycles of length 4. Thus the LDPC codes are half-regular, having the variable-node degree of the information bits $d_1 = 3$, variable-node degree of the parity bits $d_2 = 2$ and the check-node degree is $d_c = 8$. The code rate R is $2/3$. We simulate with two different code block lengths, 10^6 and 10^3. The simulation programs are written by the author using C programming language.

Based on (3.1) and the Slepian-Wolf theorem, we can have the theoretical compression rate for signal X which is $R_X = \frac{1}{R} - 1 = 0.5$. The theoretical compression rate is also the

conditional entropy $H(X|Y)$ since signal Y is compressed conventionally and sent at full rate $R_Y = H(Y) = 1$. Thus we have the Slepian-Wolf theoretical limit 0.5 for these LDPC codes and the corresponding value for the crossover probability $p = 0.11$ resulting from $H(p) = H(X|Y) = 0.5$. During the simulation, since the compression rate for signal X is fixed as 0.5, we modify the crossover probability for each simulation point to get bit error rate as low as 10^{-6}.

4.1 Performance comparison of MS algorithms and its modifications with floating-point messages

We discussed distributed source coding using parity approach in the last chapter. In this section, we give the simulation results by this approach using MS and modified MS algorithms. In the simulations, we set 100 as the maximum number of iterations during decoding algorithm. For each point, either 100 frame errors or 5×10^8 bits were simulated. The simulation will stop after 5×10^8 bits were processed if the frame errors are less than 100, otherwise, it will stop when it first reaches 100 frame errors. The messages passed during decoding are floating-point values.

4.1.1 Performance of Normalized Min-Sum Algorithms

Figure 4-1 shows the performance of the normalized min-sum algorithm with various normalization factors. The LDPC code block length is 10^6. In the figures, normalized min-sum algorithms with various factors are compared with conventional min-sum. Also the performance of sum-product algorithm is given for reference. In Figure 4-1,

normalized min-sum algorithms have the better performance than conventional min-sum, and some of them are also close to the performance of sum-product, such as those with the normalization factor 1.25 and 1.55. Slepian-Wolf limit is also given in the figure, which shows there are gaps between decoding performance and the limit.

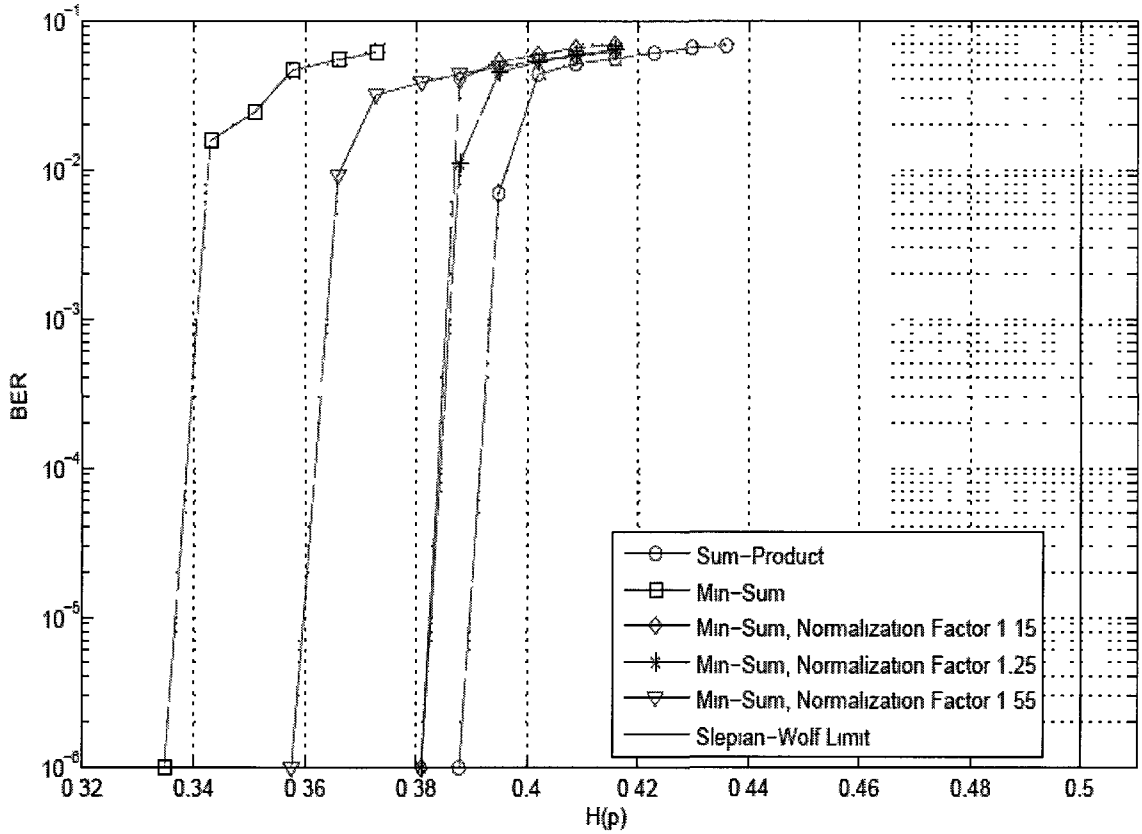


Figure 4-1 Performance of normalized min-sum algorithm with normalization factors, 1.15, 1.25 and 1.55; LDPC code block length 10^6

From Figure 4-1, we observe that different normalization factors result in different improvements in the performance. To find out the optimized factor, we run another set of simulations and the results are shown in Figure 4-2. LDPC code block length of 10^6 is

used in the simulations. It shows the effect of normalization factor on error performance of normalized min-sum algorithm for different correlation values between two sources, which are different values of the crossover probability of the BSC. The results show that the normalization factor 1.25 is optimal for different values of crossover probability p, which means that it is the optimum factor regardless of the compression rate $H(X|Y)$ for signal X.

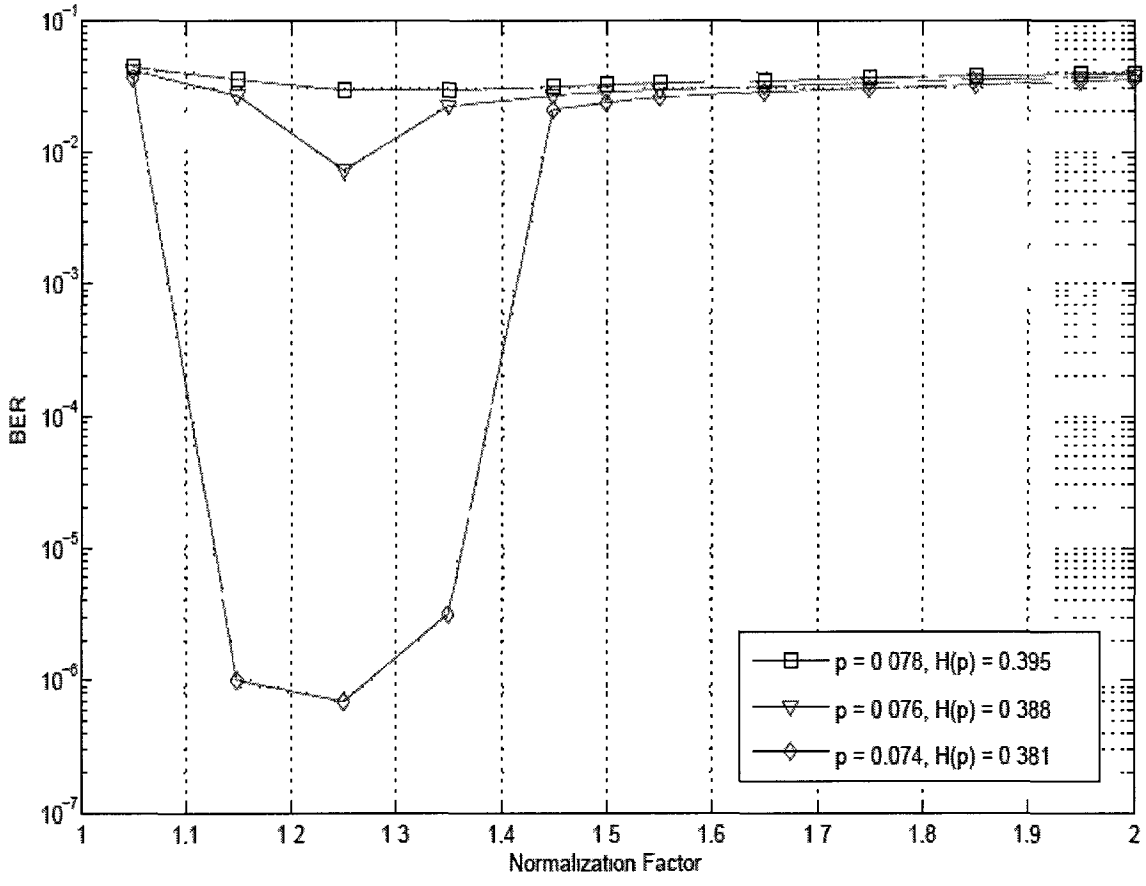


Figure 4-2 Effect of normalization factors on the error performance of normalized min-sum algorithm; LDPC code block length 10^6

We also run the same simulation experiments on an LDPC code with short block length 10^3. The results show that the same trends as those of longer block length are presented in Figure 4-3. Normalized min-sum algorithm with normalization factor 1.25 also has the best performance in the case.

In general, we observe that the performance is improved significantly when normalization factor is applied for both large and short block lengths. On short block length, normalized min-sum even has better performance than sum-product.

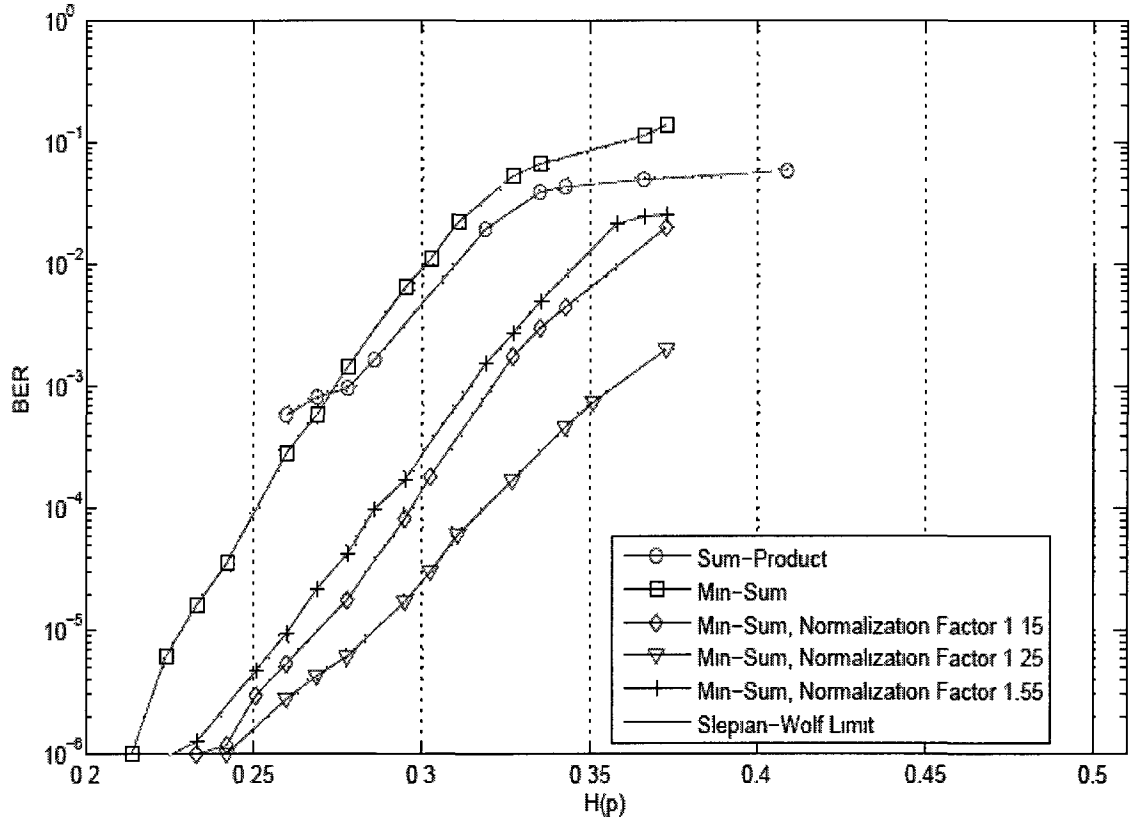


Figure 4-3 Performance of normalized min-sum algorithm with normalization factors, 1.15, 1.25 and 1.55; LDPC code block length 10^3

4.1.2 Performance of Offset Min-Sum Algorithms

Similar to the experiments for normalized min-sum algorithm, we run the same simulations on offset min-sum. Figures 4-4 and 4-5 show the performance of the offset min-sum algorithm with various offset factors. The LDPC code block length is 10^6 and 10^3, respectively. In the figures, offset min-sum algorithms with various factors are compared with conventional min-sum. Also the performance of sum-product algorithm and Slepian-Wolf limit are given for reference.

We observe that the performance is improved significantly when offset factor is applied for both large and short block lengths. For the short block length, offset min-sum even has better performance than sum-product. We also see different offset factors result in different improvements in the performance.

Figure 4-6 shows the effect of offset factor on error performance of the offset min-sum algorithm. LDPC code block length of 10^6 is used in simulations. Results in Figure 4-6 match those in Figures 4-4 and 4-5. Offset min-sum algorithm with the offset factor 0.5 has the best performance among others regardless of values of the crossover probability p.

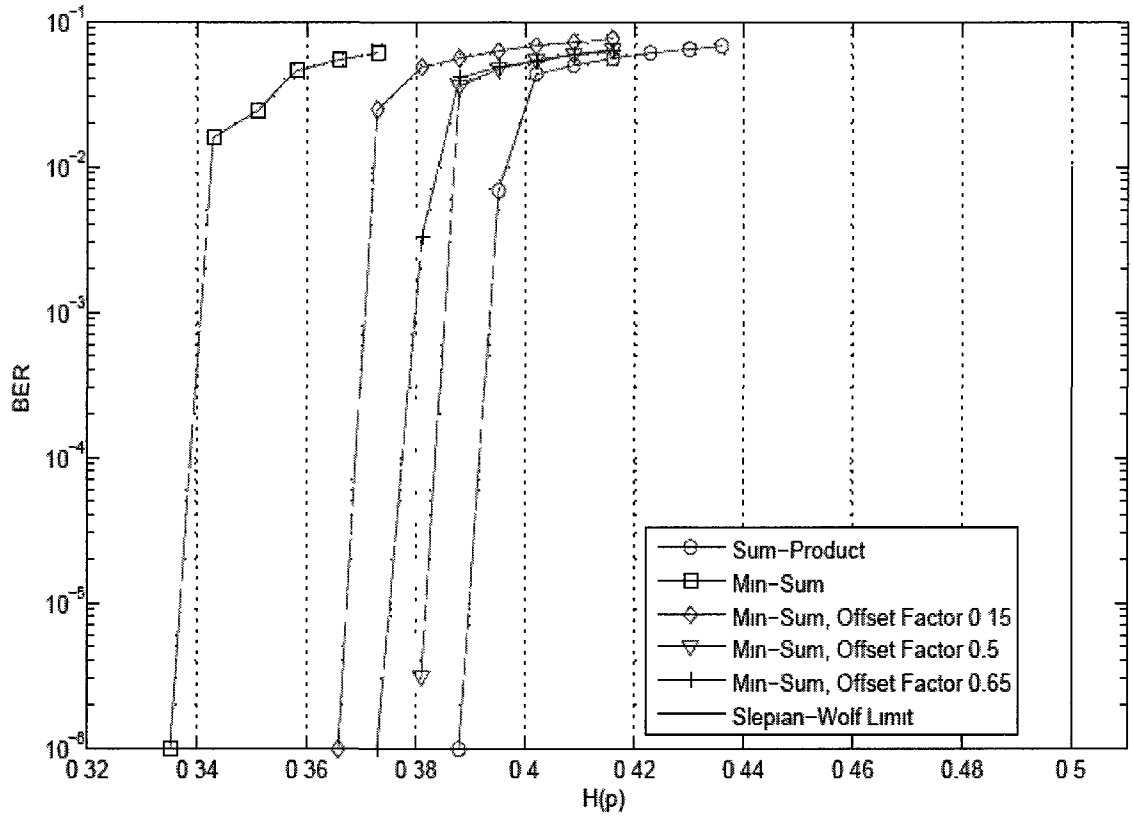


Figure 4-4 Performance of offset min-sum algorithm with offset factors 0.15, 0.5 and 0.65; LDPC code block length 10^6

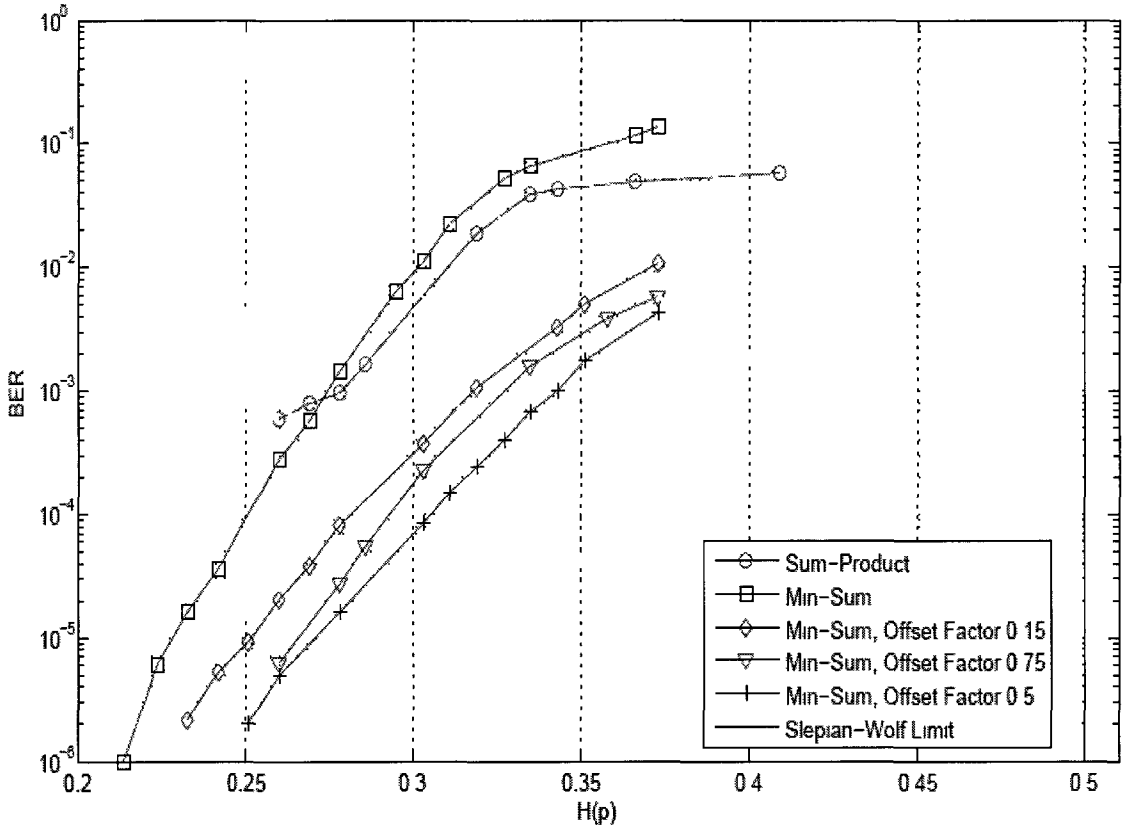


Figure 4-5 Performance of offset min-sum algorithm with offset factors 0.15, 0.5 and 0.75; LDPC code block length 10^3

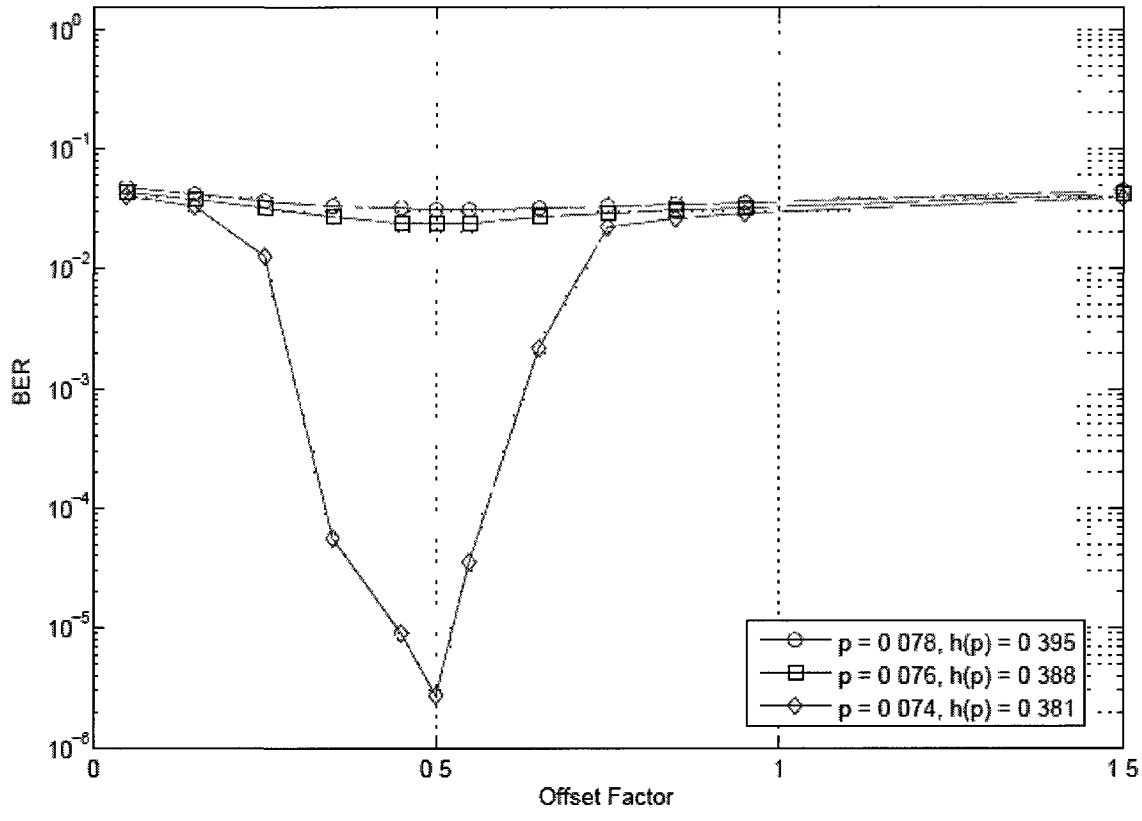


Figure 4-6 Effect of offset factor on error performance of offset min-sum algorithm; LDPC code block length10^6

4.1.3 Performance Comparison of the Proposed DSC with the Existing Schemes

Distributed source coding using parity approach has been studied by other researchers, [15]-[17], [27]. In [17], non-uniform LDPC codes have been designed for this approach. The non-uniform LDPC code designed in [17] is irregular, and has the following degree distribution

$$\Lambda(x) = \{\lambda^1(x), \lambda^2(x)\} = \{0.1784x + 0.6272x^4 + 0.1845x^6, 0.6406x + 0.0319x^6 + 0.3043x^8\}$$
$$\rho(x) = \{0.52x^{12} + 0.48x^{13}\}. \quad (4.2)$$

Comparing the degree distribution (4.2) with our designed LDPC code (4.1), one can see that our designed LDPC code is much simpler. Not only our design has regular structure for each of the information bits and parity bits, but also the average degrees of variable and check nodes are smaller in our design resulting in less computational complexity per iteration. With length 10^3, the degree distribution (4.2) has 3741 edges which is 40% more than our designed LDPC code (4.1). Since the number of operations per iteration of sum-product algorithm is proportional to the number of edges, the reduced edges will save 40% in computation time. The computational complexity per iteration is decreased more in our scheme because we use min-sum algorithm which has simpler check node update rule. Also since our designed LDPC code has smaller maximum degrees, the processing delay will be smaller and consequently the throughput will be higher.

In the paper [17], Sartipi and Fekri showed that the performance of non-uniform LDPC code with length 10^3 is 60% better than DISCUS using LDPC code with the same length. If the degree of variable nodes is chosen uniformly at random from $\lambda^1(x)$ and $\lambda^2(x)$ in

(4.2), the designed code is referred to as "uniform LDPC code" in [17] because the information bits and the parity bits have the same degree distribution. The performance of a uniform LDPC code in DSC with length 10^3 is still 15% better than DISCUS using an LDPC code with the same length. The performances of non-uniform LDPC code, uniform LDPC code and DISCUS using LDPC code are shown in Figures 4-7 and 4-8. The curves corresponding to the results of [17] in Figures 4-7 and 4-8 are copied from [17] and we did not regenerate the results. The decoding algorithm used in [17] is sum-product, or BP. For comparison, the simulation results of normalized and offset MS algorithms with optimal factors using the LDPC code described by (4-1) are also given in Figures 4-7 and 4-8, respectively. As we mentioned before, we set 100 as the maximum number of iterations in the simulations. For each simulation point, either 100 frame errors or 5×10^8 bits were simulated.

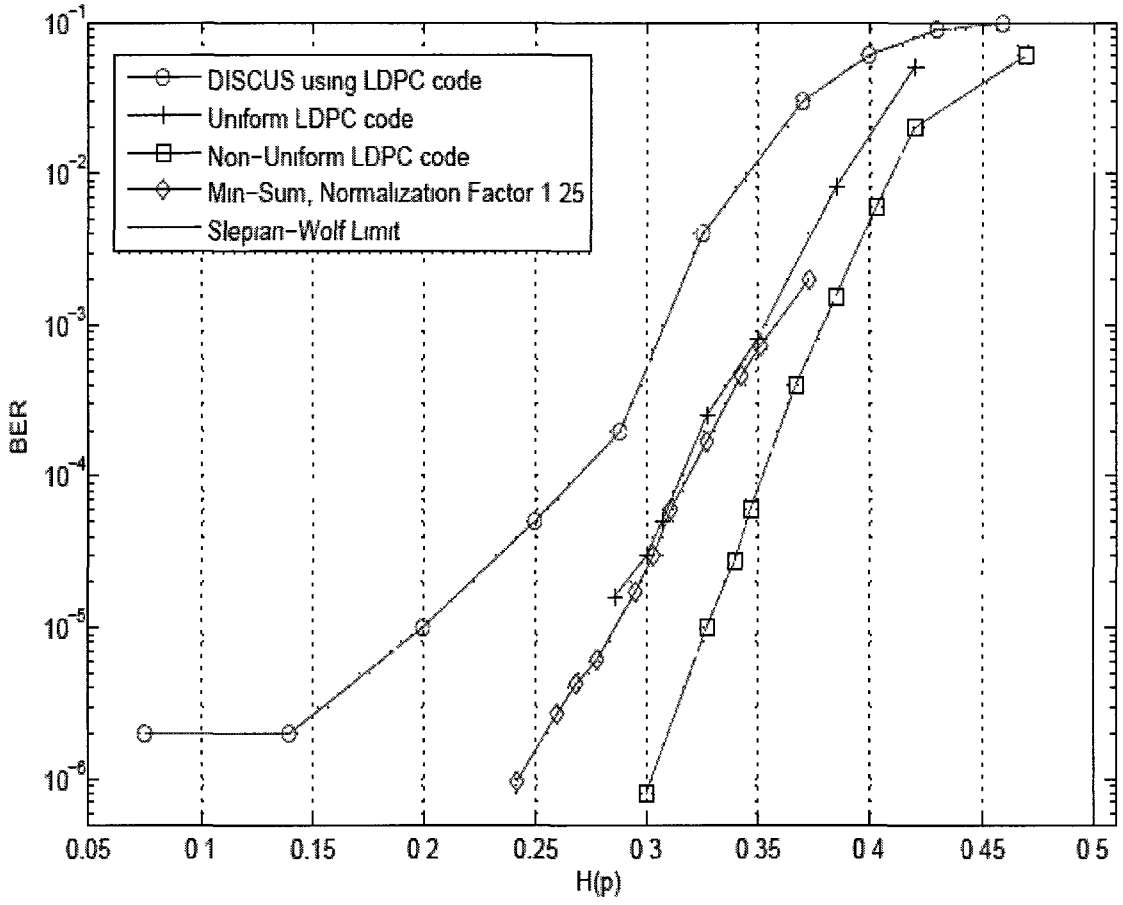


Figure 4-7 Comparison of different DSC schemes for block length n = 10^3

Figure 4-7 shows that with a lower complexity decoding algorithm (normalized min-sum with normalization factor 1.25) and simpler code structure, the proposed scheme outperforms DISCUS with an LDPC code. It also performs slightly better than the DISCUP with a uniform LDPC code. In Figure 4-8, offset min-sum with offset factor 0.5 shows a performance similar to that of uniform LDPC code.

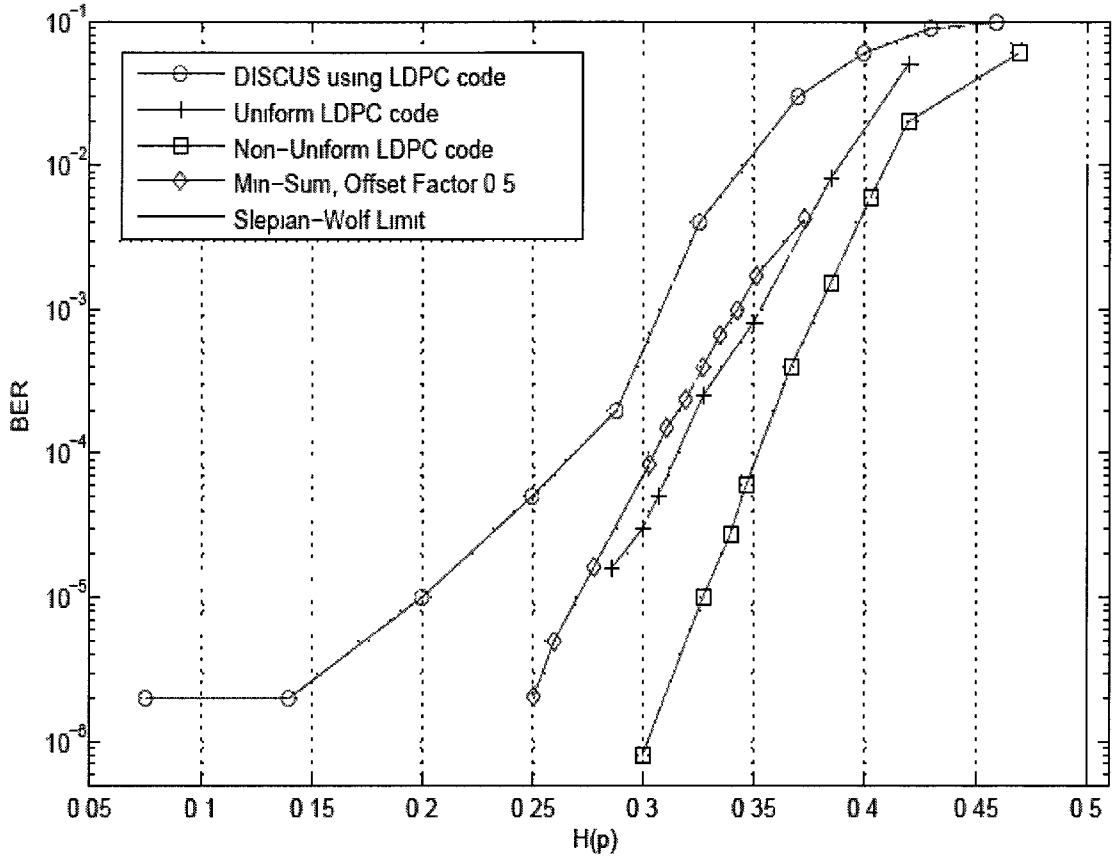


Figure 4-8 Comparison of different DSC schemes for block length n = 10^3

To show that the performance of normalized and offset MS in the DISCUP scheme improves when the code length increases, and also to compare their performance with the performances reported in [17] for other lenghts, the simulation also run with the code lengths 2000 and 4000 in addition to 1000. Table 4-1 shows the gap of the non-uniform coding schemes in [17] from the Slepian-Wolf limit. The gap is measured at BER of 10^{-5}. It also shows similar results from our simulations for normalized and offset MS

algorithms, respectively, with the designed LDPC codes. The results show that the gaps from the theoretical limit decrease as the code length increases, for both normalized and offset min-sum algorithms. They also demonstrate that for all block lengths, the proposed scheme performs only slightly inferior to that of [17] while having a significantly lower complexity.

Table 4-1 Comparison of Gaps from the Slepian-Wolf theoretical limit

Code length	1000	2000	4000
Gap from the theoretical limit (Non-uniform LDPC code) [17]	0.1726	0.1416	0.1267
Gap from the theoretical limit (Normalized min-sum, optimal factor 1.25)	0.209	0.173	0.165
Gap from the theoretical limit (Offset min-sum, optimal factor 0.5)	0.231	0.191	0.179

4.2 Performance of Quantized MS in DSC

To implement min-sum or modified min-sum algorithms using circuits, the messages passed between nodes cannot have floating-point values. Suppose each message passed between nodes is represented by $q+1$ bits, with the first bit representing the sign and the rest representing the magnitude. Then the message belongs to an alphabet of size $2^{q+1}-1$, which can be represented by $[-N,\cdots,-1,0,1,\cdots,N]$, where $N=2^q-1$. There is also one more combination of the bits left that we can use to represent $+\infty$ in our case.

From the analysis in section 3.3, we know that the magnitude of the messages passed during the min-sum algorithm on a BSC is an odd number times a value $\beta=\ln\frac{1-p}{p}$, where p is the crossover probability and $p<0.5$, or it is equal to $+\infty$. Suppose that the quantization step is β, if a message is in the range $(n\beta-\frac{\beta}{2},\ n\beta+\frac{\beta}{2})$, where n is an integer, the message is then quantized to n, if $-N\le n\le N$, to $-N$, if $n<-N$ and to N, if $n>N$.

We also study how normalization factor and offset factor impact the performance of quantized min-sum algorithm. For this, we first apply the normalization factor or the offset factor to the actual messages passed between the nodes, and then we quantize the messages by using the quantization rule described above.

In the following, the performances of quantized MS, quantized normalized MS and quantized offset MS are presented in sections 4.2.1, 4.2.2 and 4.2.3, respectively.

4.2.1 Performance of Quantized Min-Sum Algorithm

Figures 4-9, 4-10 and 4-11 show the performance of quantized min-sum algorithm. In each figure, we choose different quantization bits. The performances of conventional floating-point min-sum, sum-product and Slepian-Wolf limit are also given for comparison.

The simulation results show the performance of quantized min-sum algorithm becomes closer to conventional min-sum as quantization bits increase. In fact, the results show the performance curves are almost identical to floating-point min-sum after the number of quantization bits q is equal to or larger than 5.

With the simulation above, the clipping threshold $\pm C_{th}$ is always equal to $\pm(N\beta+\frac{\beta}{2})$. To get the optimal clipping threshold C_{th}, we run extensive simulations. The results are shown in Figures 4-12 to 4-15. The results show that the optimal clipping threshold is rather insensitive to the number of quantization bits q and to BSC crossover probability p. The optimal clipping threshold for our designed LDPC code is $C_{th} = 3.0$.

We apply the optimal clipping threshold to quantized min-sum algorithm, and the results are shown in Figure 4-16. The optimal clipping threshold improves the performance of quantized min-sum algorithm, which is slightly better than that of conventional min-sum with floating-point messages.

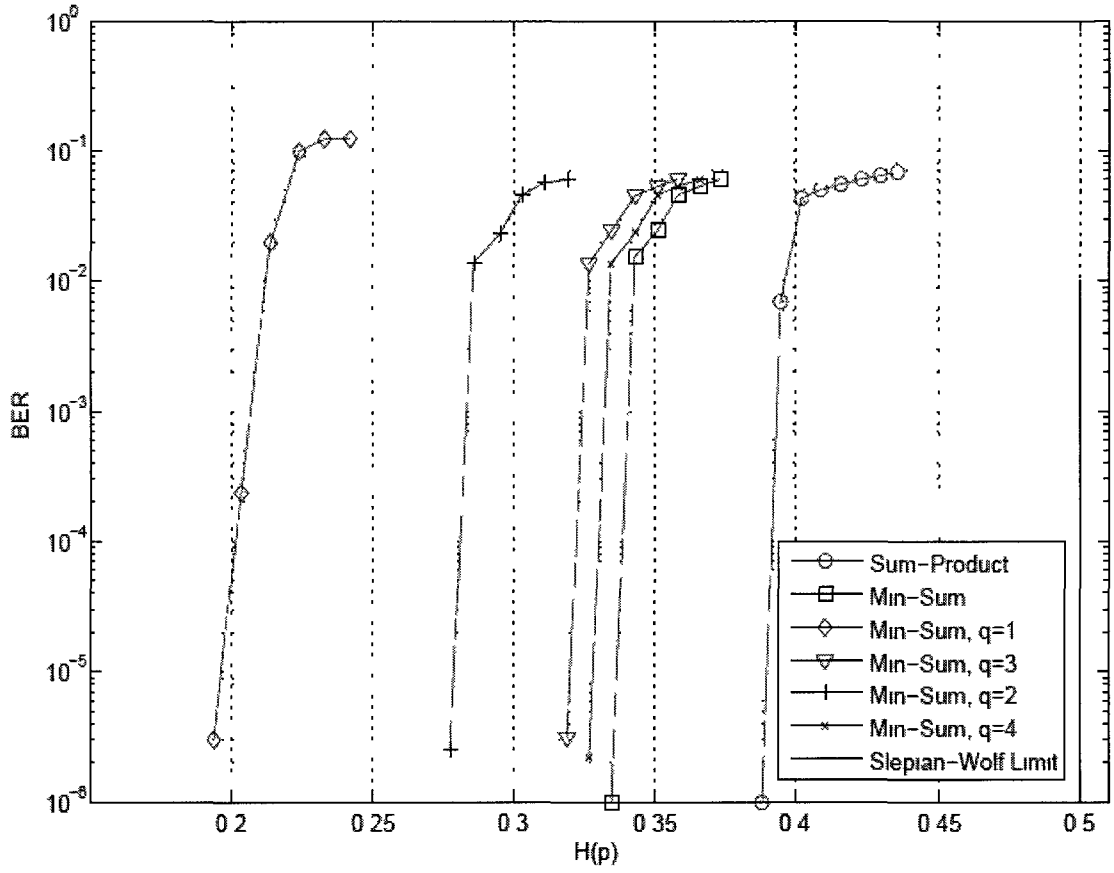


Figure 4-9 Performance of quantized min-sum (quantization bits q=1, 2, 3 and 4), with LDPC code of block length 10^6.

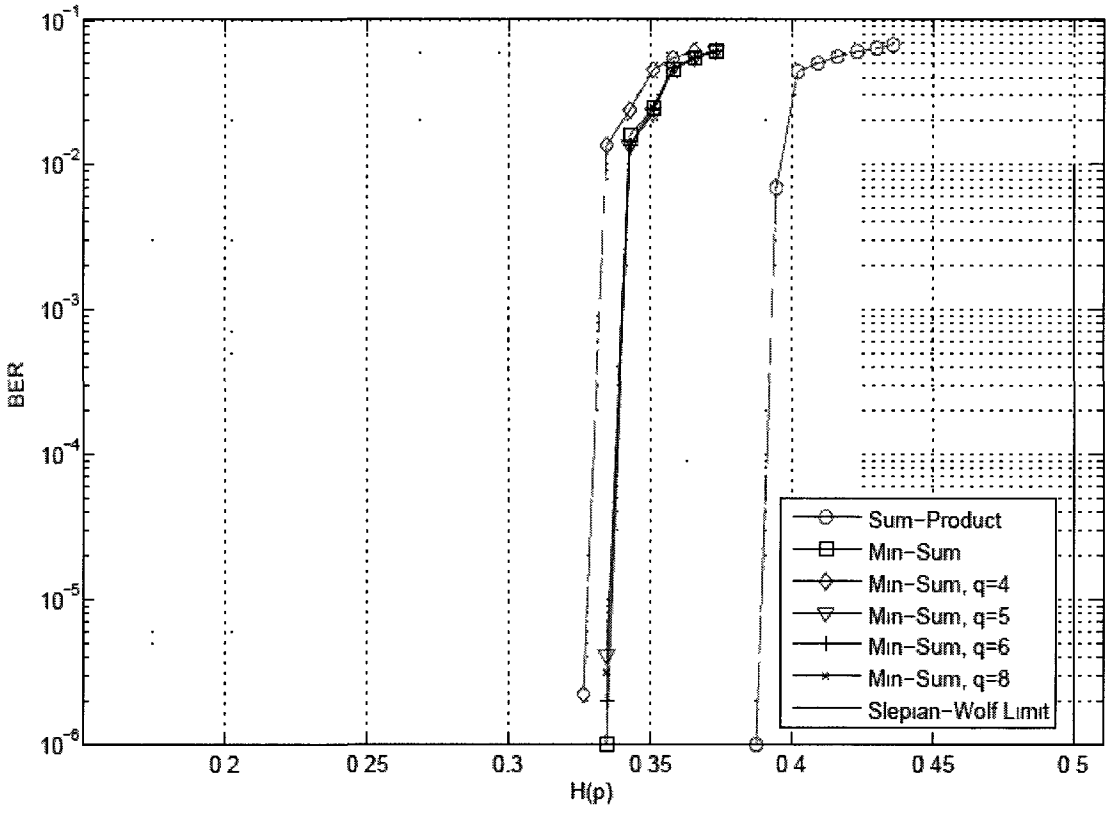


Figure 4-10 Performance of quantized min-sum (quantization bits q=4, 5, 6 and 8), with LDPC code of block length 10^6.

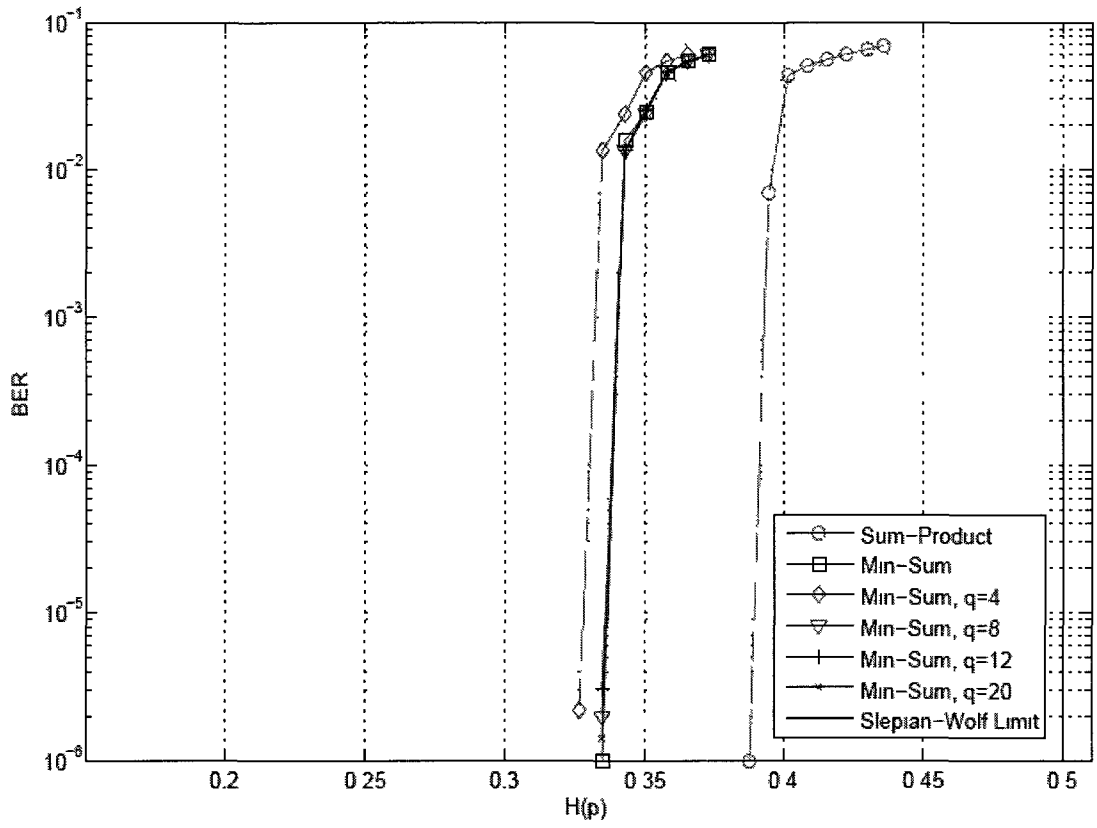


Figure 4-11 Performance of quantized min-sum (quantization bits q=4, 8, 12 and 20), with LDPC code of block length 10^6.

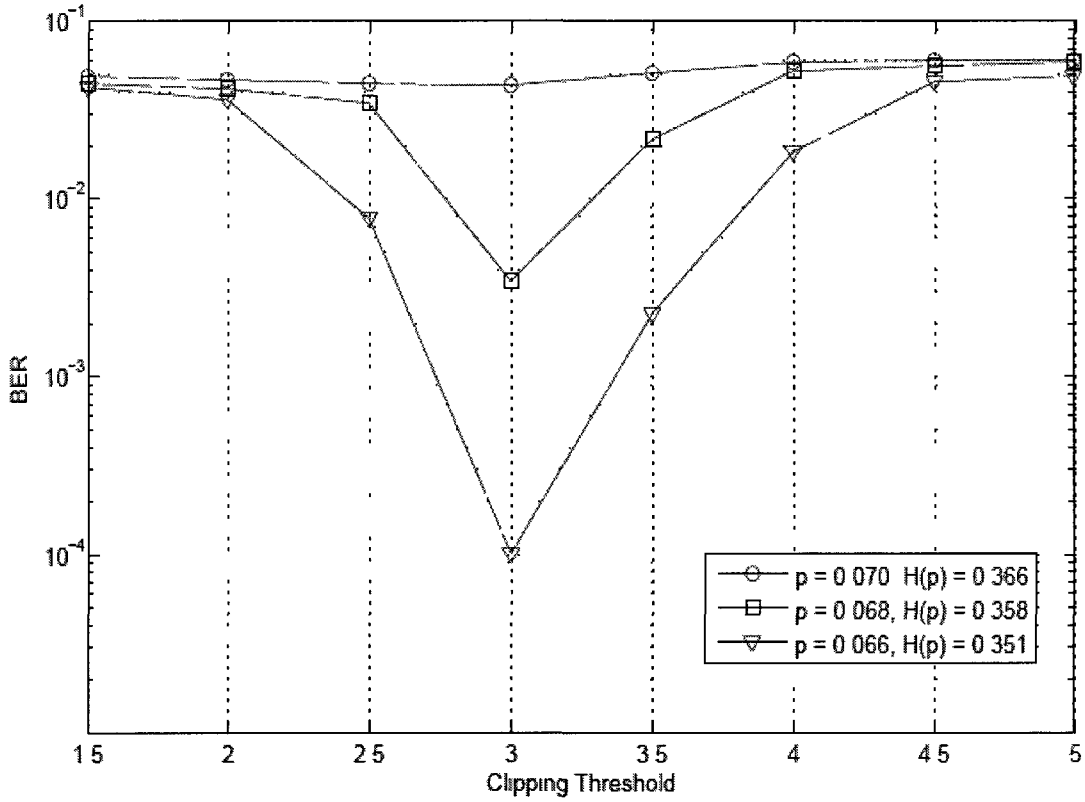


Figure 4-12 Effect of clipping threshold C_{th} on error performance of quantized min-sum with $q = 3$; LDPC code block length 10^6

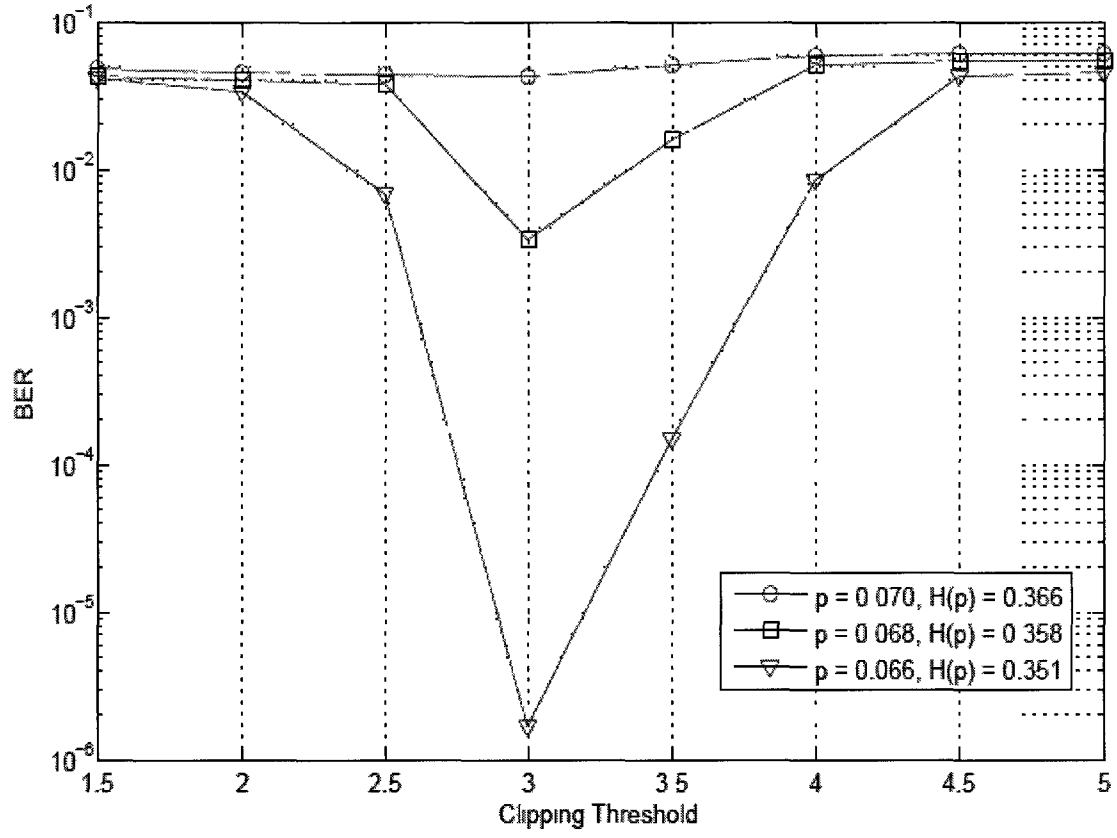


Figure 4-13 Effect of clipping threshold C_{th} on error performance of quantized min-sum with $q = 4$; LDPC code block length 10^6

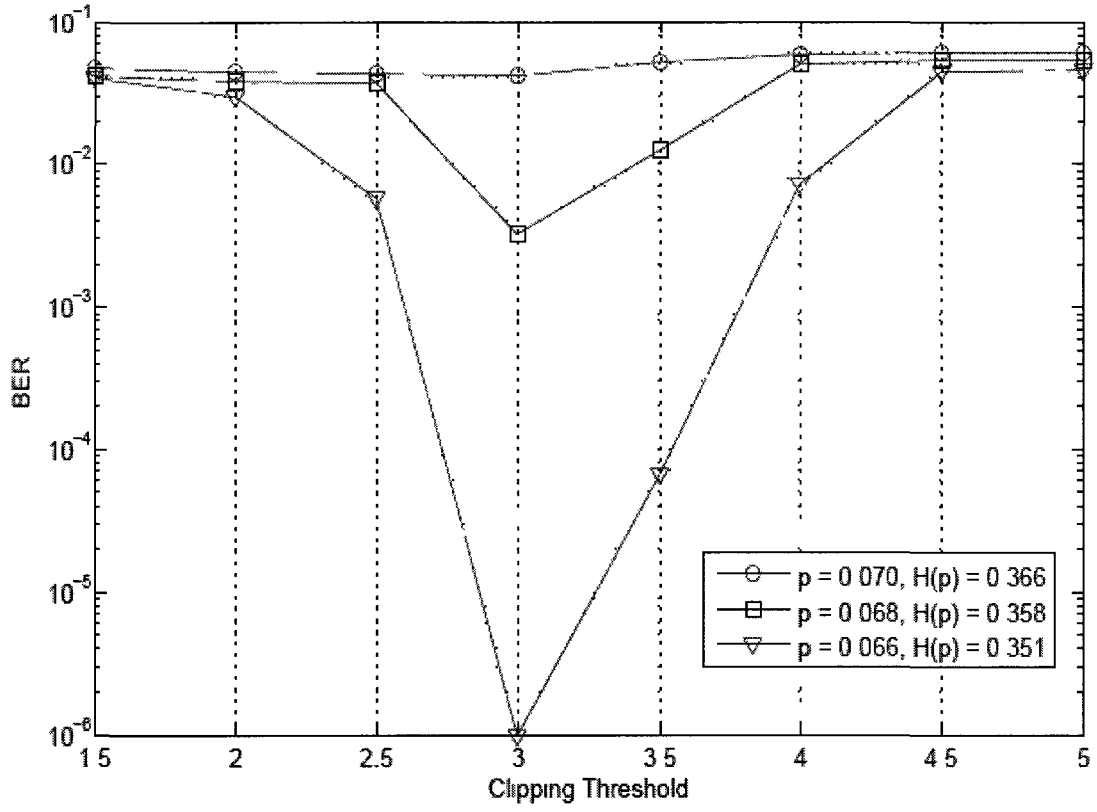


Figure 4-14 Effect of clipping threshold C_{th} on error performance of quantized min-sum with $q = 8$; LDPC code block length 10^6

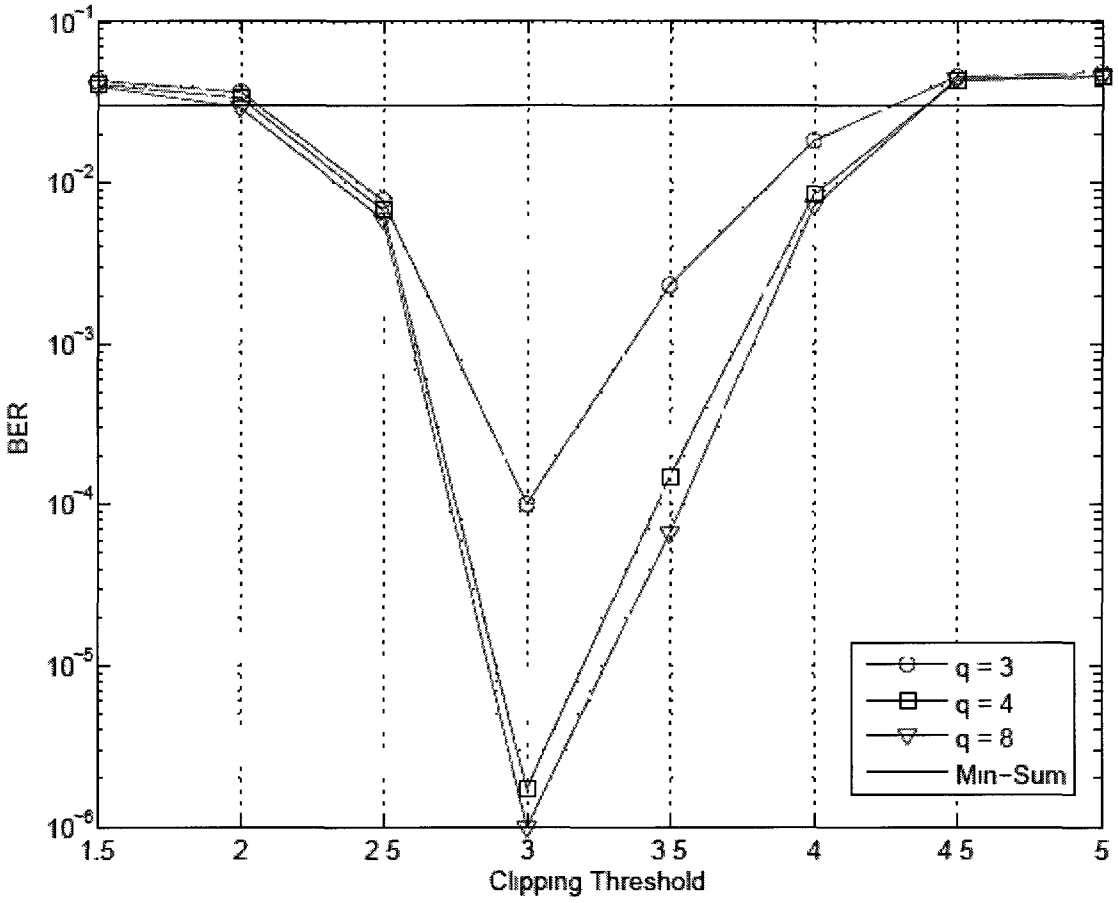


Figure 4-15 Effect of clipping threshold C_{th} on error performance of quantized min-sum at $p = 0.066$, $H(p) = 0.351$; LDPC code block length 10^6

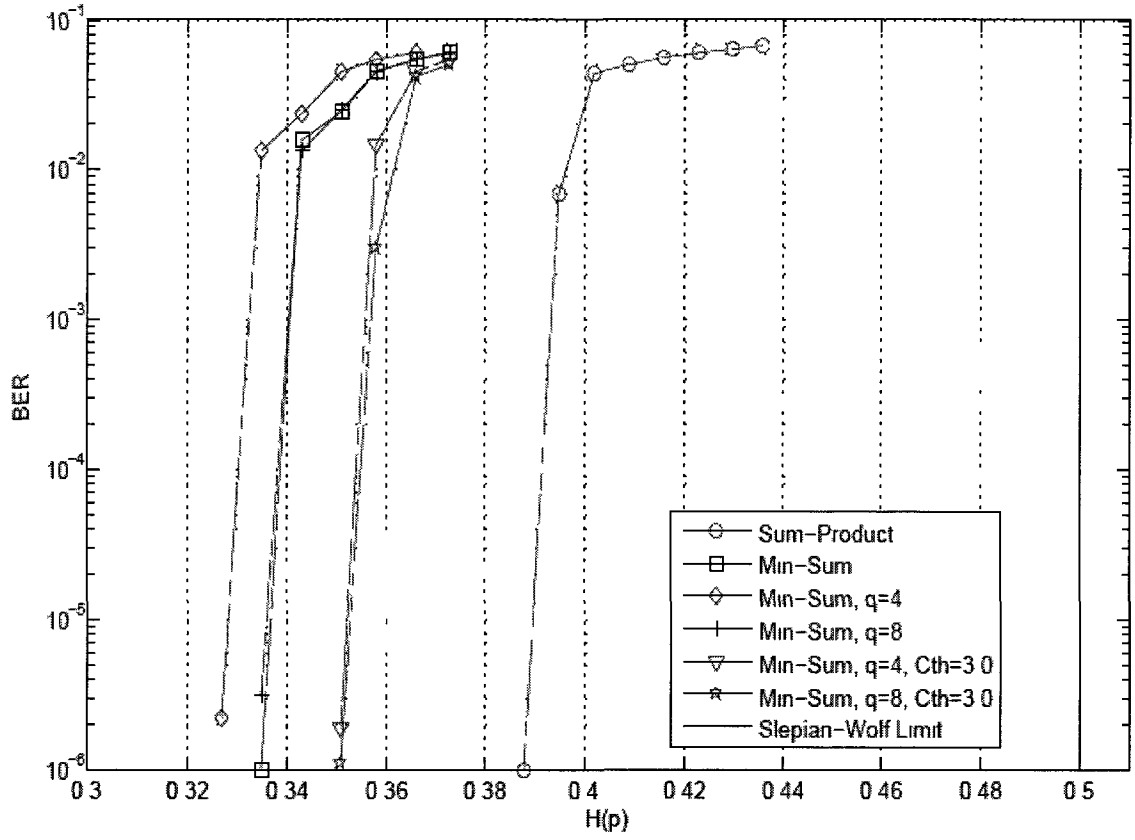


Figure 4-16 Performance of quantized min-sum with the optimal clipping threshold; LDPC code of block length 10^6

4.2.2 Performance of Quantized Normalized Min-Sum Algorithm

To improve the performance of quantized min-sum, we apply normalization factor to the decoding algorithm. The previous simulations in section 4.1.1 show that normalization factor significantly improves the performance of the conventional min-sum. The simulations in this section convey similar results which show quantized normalized min-sum for short block lengths has better performance than min-sum. The LDPC code block length is 10^3.

The performance of quantized normalized min-sum algorithms with the number of quantization bits q = 3, 4, 5 and 6 is given in Figure 4-17. These results are based on the optimal clipping threshold 3.0 which was found in Section 4.2.1, and the optimal normalization factor 1.25 which was found in Section 4.1.1. The performances of floating-point sum-product, floating-point min-sum, floating-point normalized min-sum and Slepian-Wolf limit are also given for comparison. The figure shows that based on the selected values for the normalization factor and the clipping threshold, the performance saturates at q =4, and no considerable improvement can be seen by increasing the value of q further.

Figures 4-18 and 4-19 show the similar results for the performance of quantized normalized min-sum algorithms with various normalization factors. These results along with the results of Figures 4-20, 4-21, 4-22 and 4-23 show that the performance of quantized normalized MS cannot be further improved by modifying the normalization factor, i.e., the normalization factor 1.25 is still close to optimal for quantized normalized

MS as it was for the floating-point version. One then may have to consider changing the clipping threshold to further improve the performance.

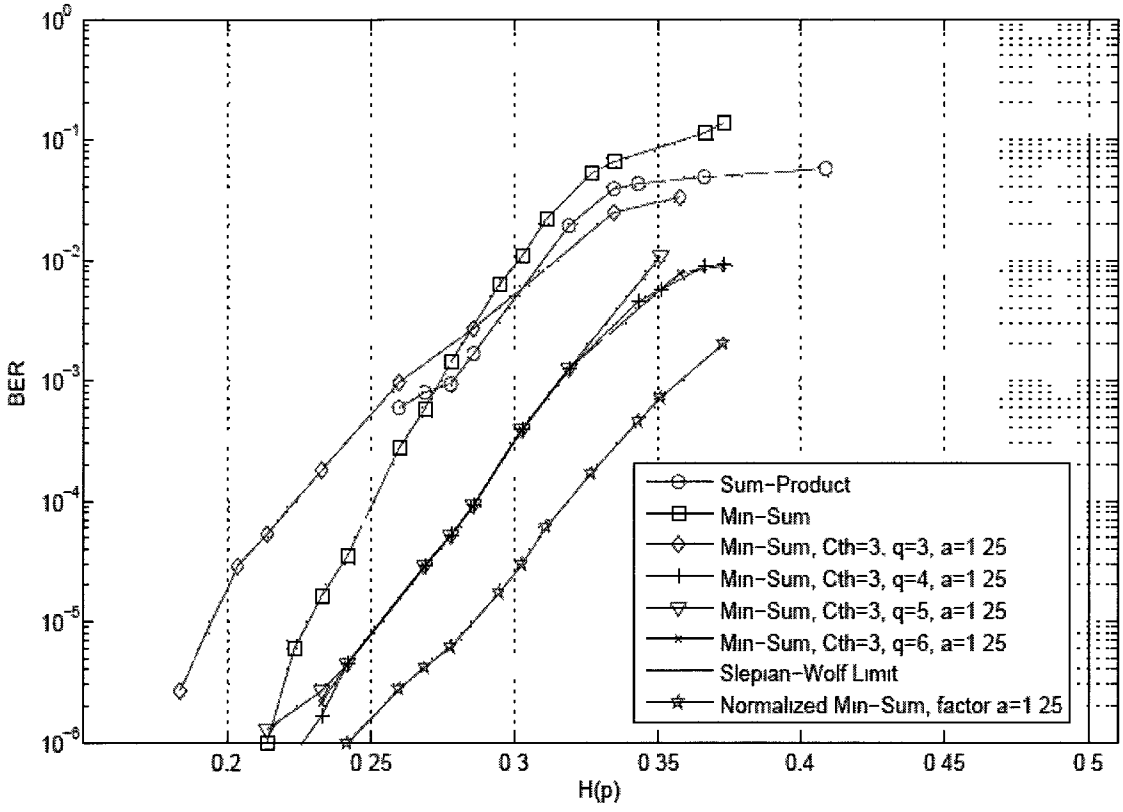


Figure 4-17 Performance of quantized normalized min-sum for normalization factor 1.25, $C_{th} = 3.0$ and for different number of quantization bits

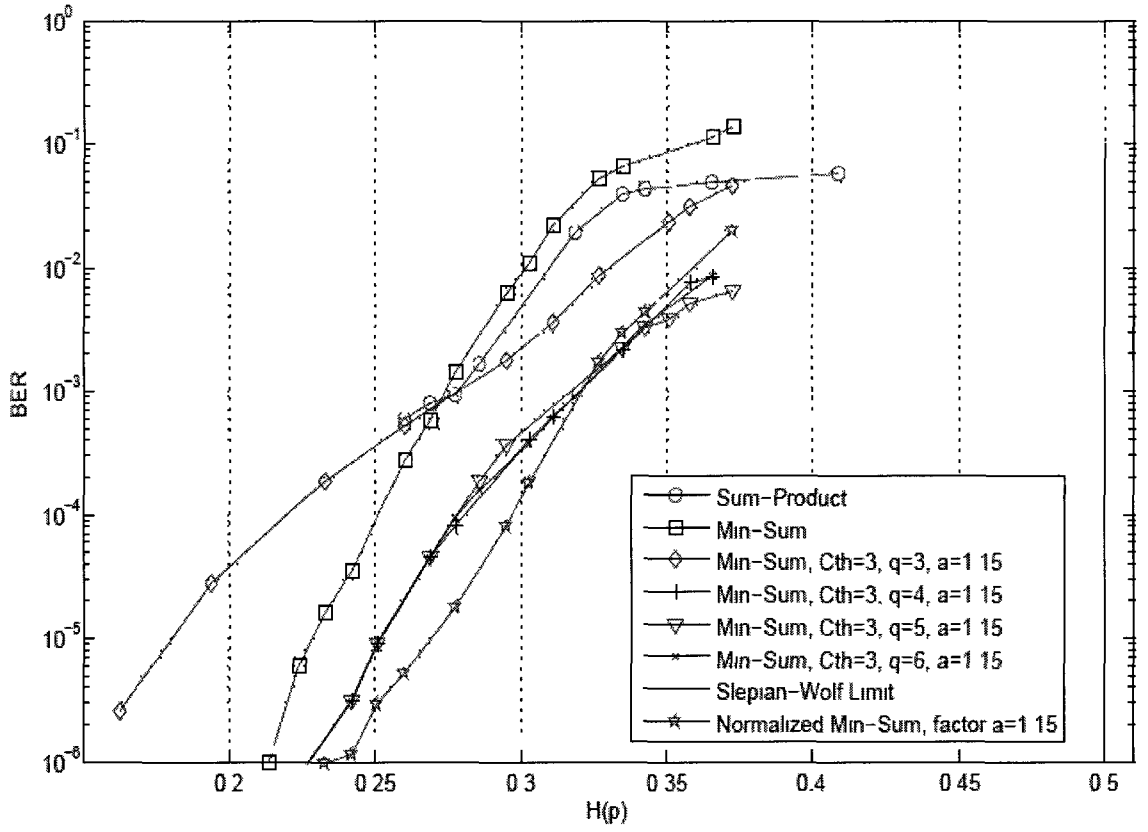


Figure 4-18 Performance of quantized normalized min-sum for normalization factor 1.15, $C_{th} = 3.0$ and for different number of quantization bits

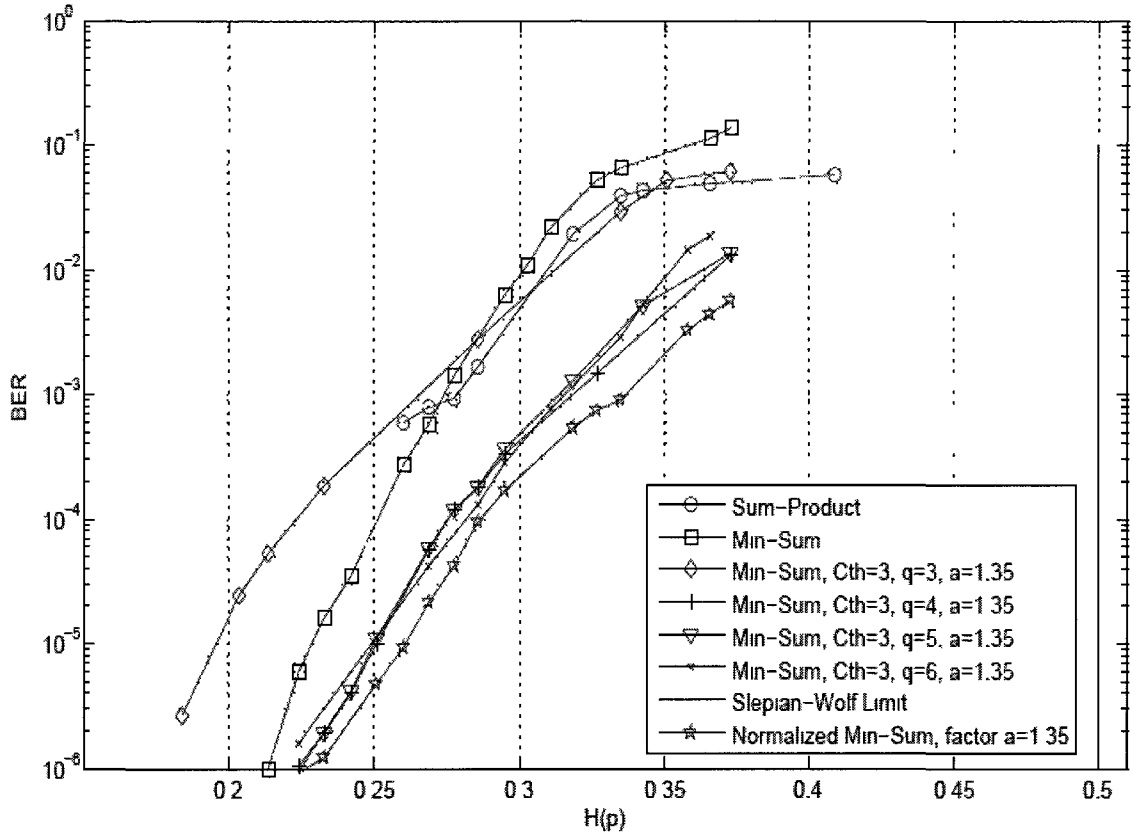


Figure 4-19 Performance of quantized normalized min-sum for normalization factor 1.35, $C_{th} = 3.0$ and for different number of quantization bits

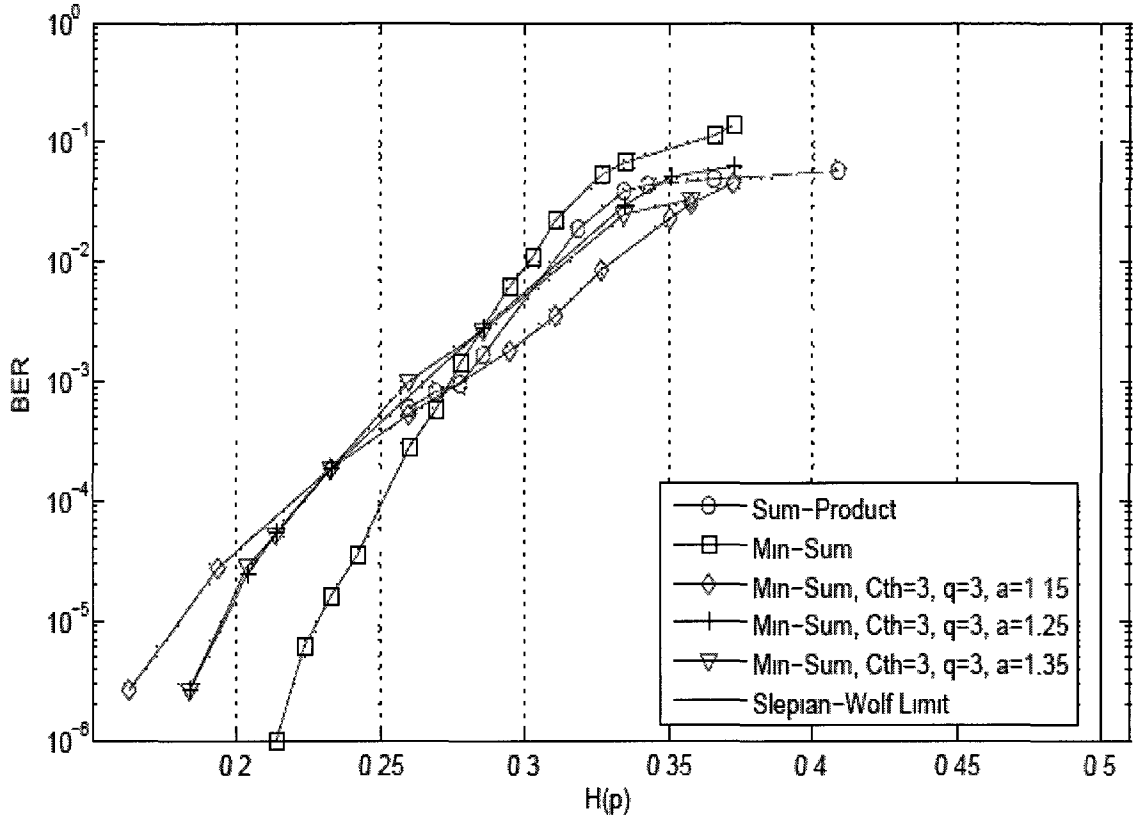


Figure 4-20 Performance of quantized normalized min-sum for 3 quantization bits and for different normalization factors

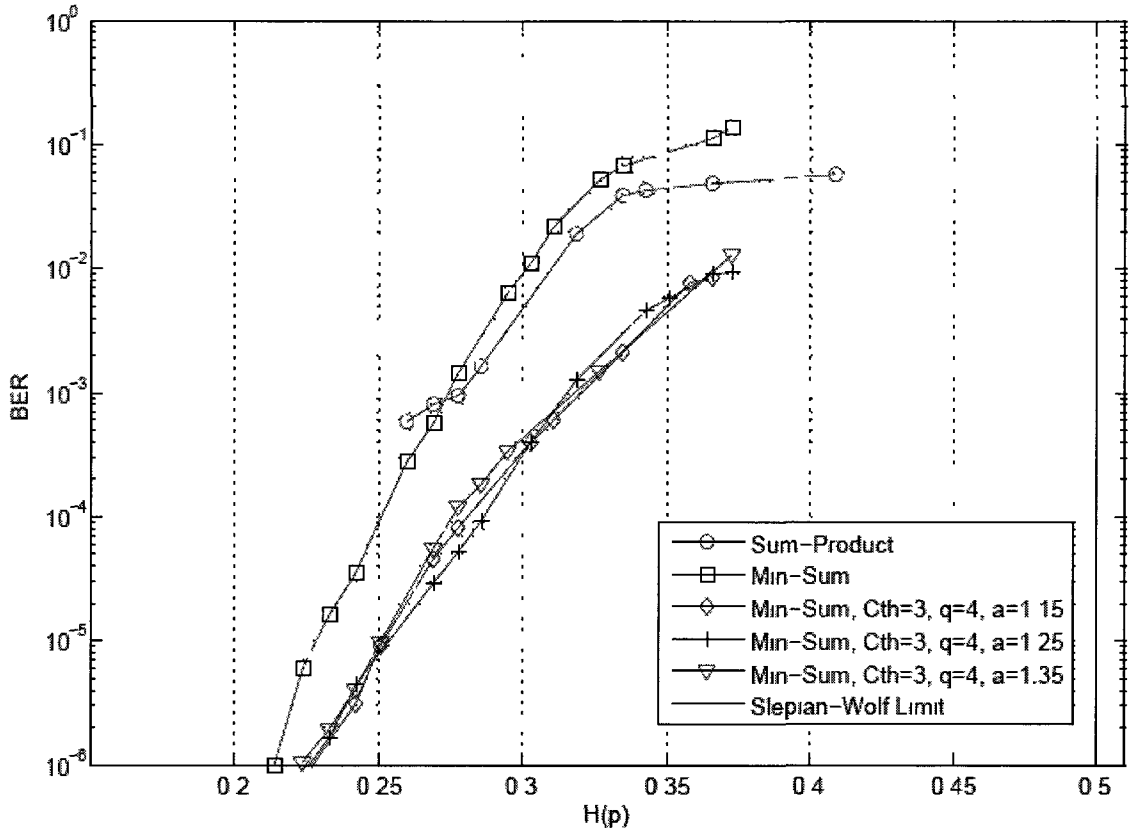


Figure 4-21 Performance of quantized normalized min-sum for 4 quantization bits and for different normalization factors

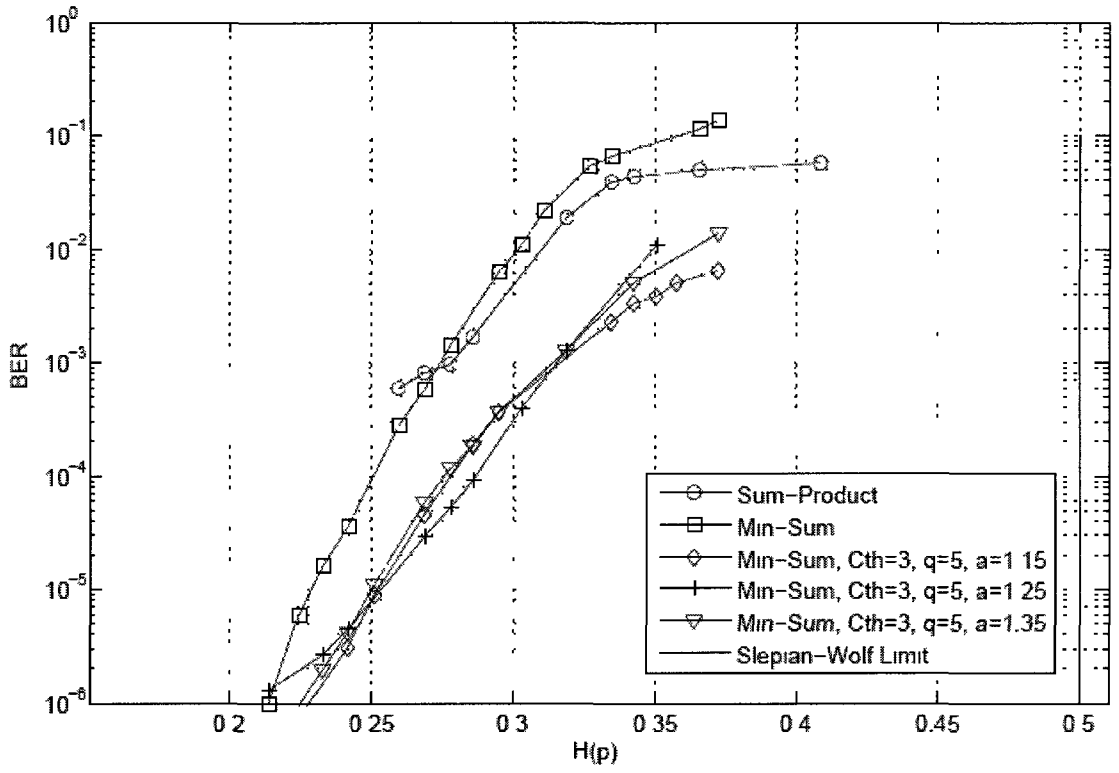


Figure 4-22 Performance of quantized normalized min-sum for 5 quantization bits and for different normalization factors

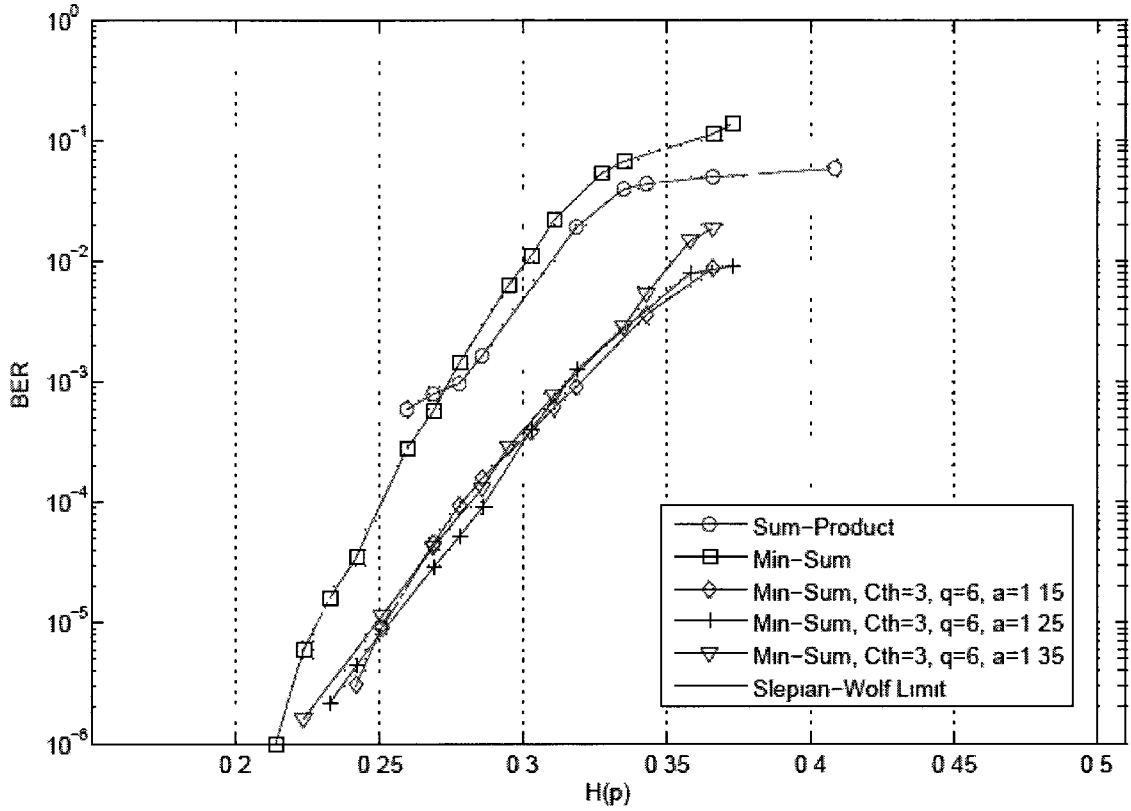


Figure 4-23 Performance of quantized normalized min-sum for 6 quantization bits and for different normalization factors

4.2.3 Performance of Quantized Offset Min-sum Algorithm

In this section, we compare the performance of quantized offset min-sum algorithm with floating-point min-sum and floating-point offset min-sum. Performance of sum-product is also given for reference. The LDPC code block length is 10^3.

Figures 4-24, 4-25 and 4-26 show the performances of offset quantized min-sum algorithms, with different offset factors and different quantization bits. The clipping threshold 3.0 is used for the decoding. The results in these figures indicate that the offset factor 0.5 is still close to optimal for quantized offset MS as it was for floating-point offset MS, and that the best performance complexity tradeoff is obtained by $q = 4$. Increasing q beyond 4 does not improve the performance in any measurable way while increasing the complexity. The performance obtained with $C_{th} = 3.0$, $q = 4$, and the offset factor of 0.5 is however still slightly away from the best floating-point performance of offset MS. The gap may be closed if one optimizes the clipping threshold for the quantized case.

Figures 4-27 to 4-30 show the same results from a different point of view. In each figure, the number of quantization bits is fixed and the value of the offset factor is changed.

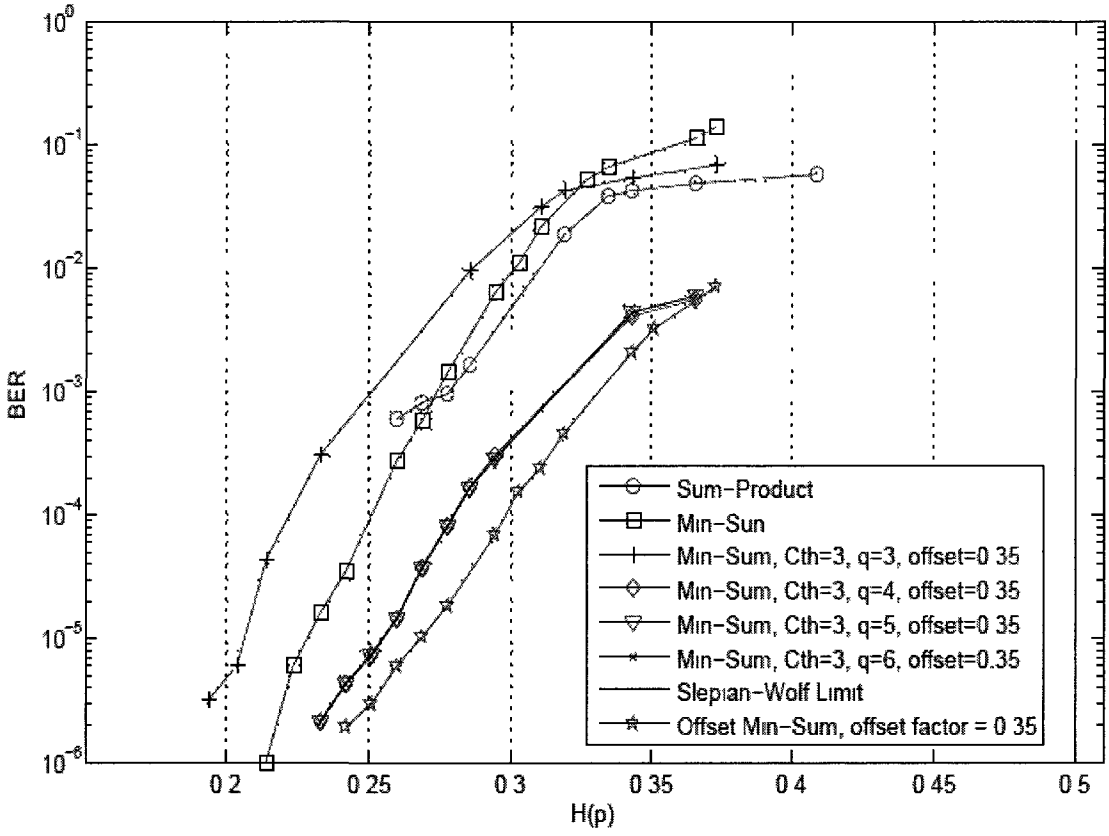


Figure 4-24 Performance of quantized offset min-sum for offset factor 0.35, $C_{th} = 3.0$ and for different number of quantization bits

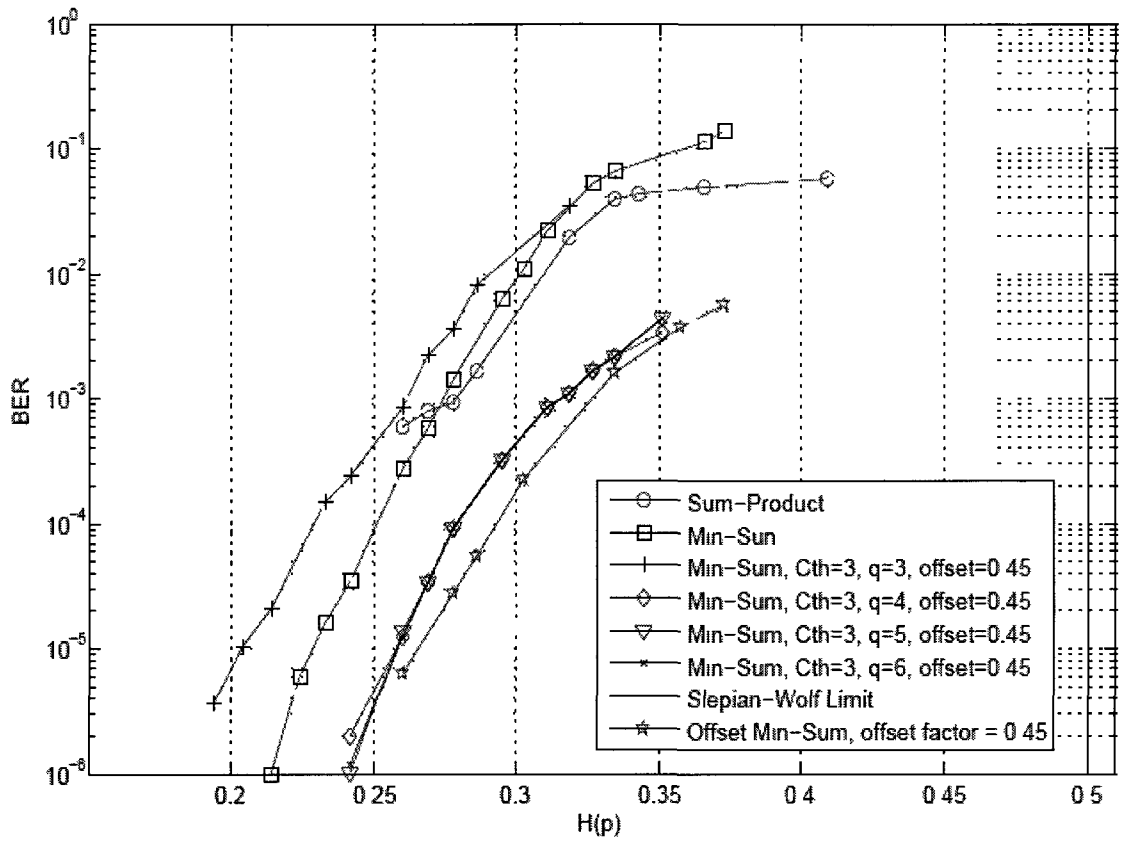


Figure 4-25 Performance of quantized offset min-sum for offset factor 0.45, $C_{th} = 3.0$ and for different number of quantization bits

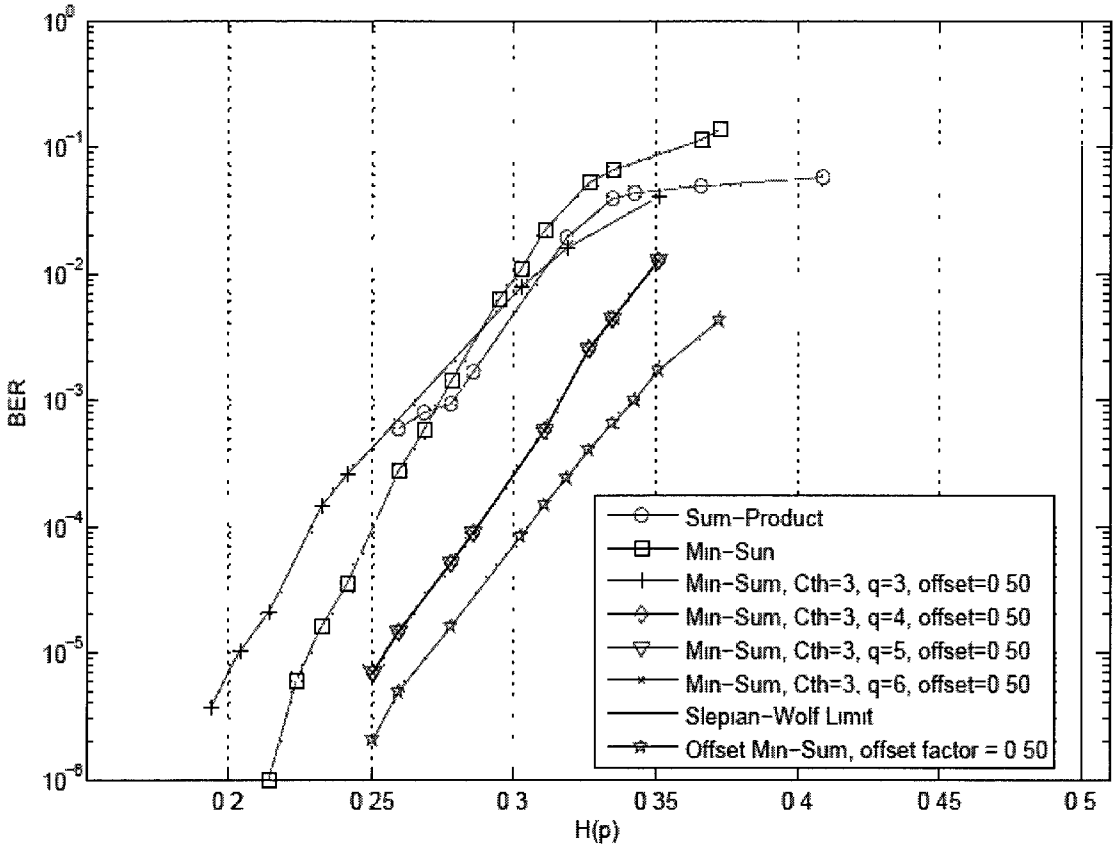


Figure 4-26 Performance of quantized offset min-sum for offset factor 0.50, $C_{th} = 3.0$ and for different number of quantization bits

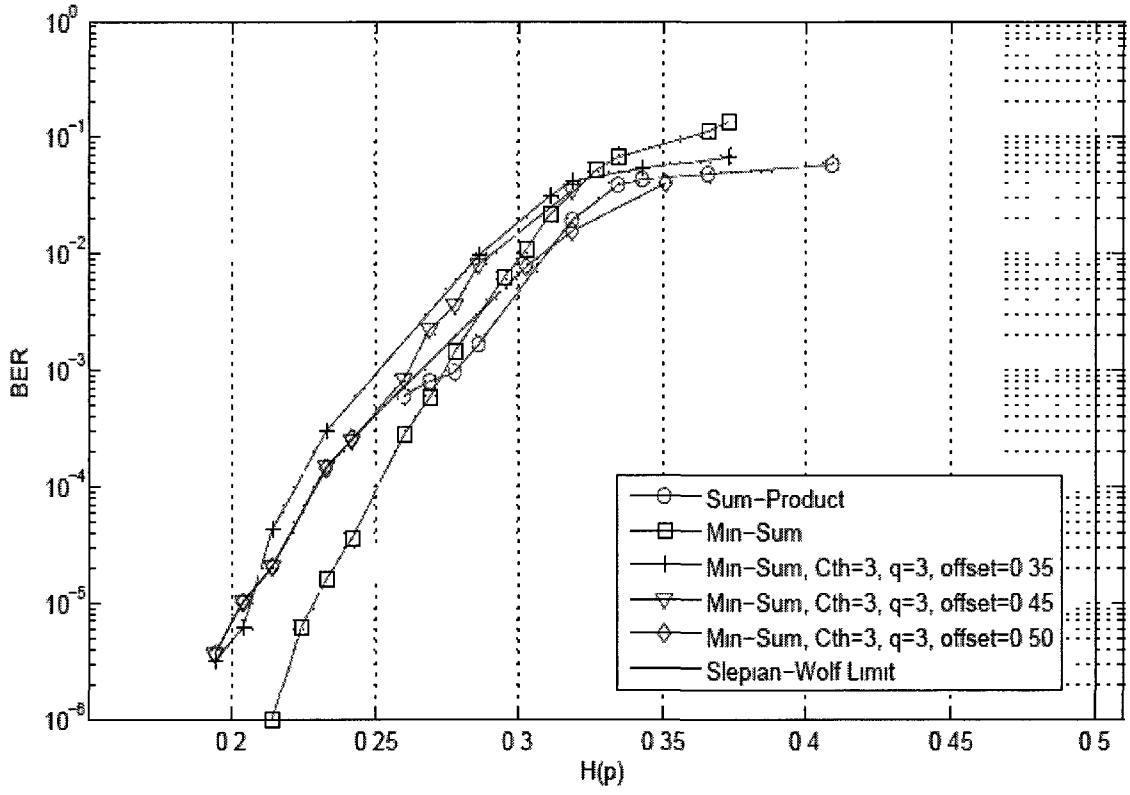


Figure 4-27 Performance of quantized offset min-sum for 3 quantization bits and for different offset factors

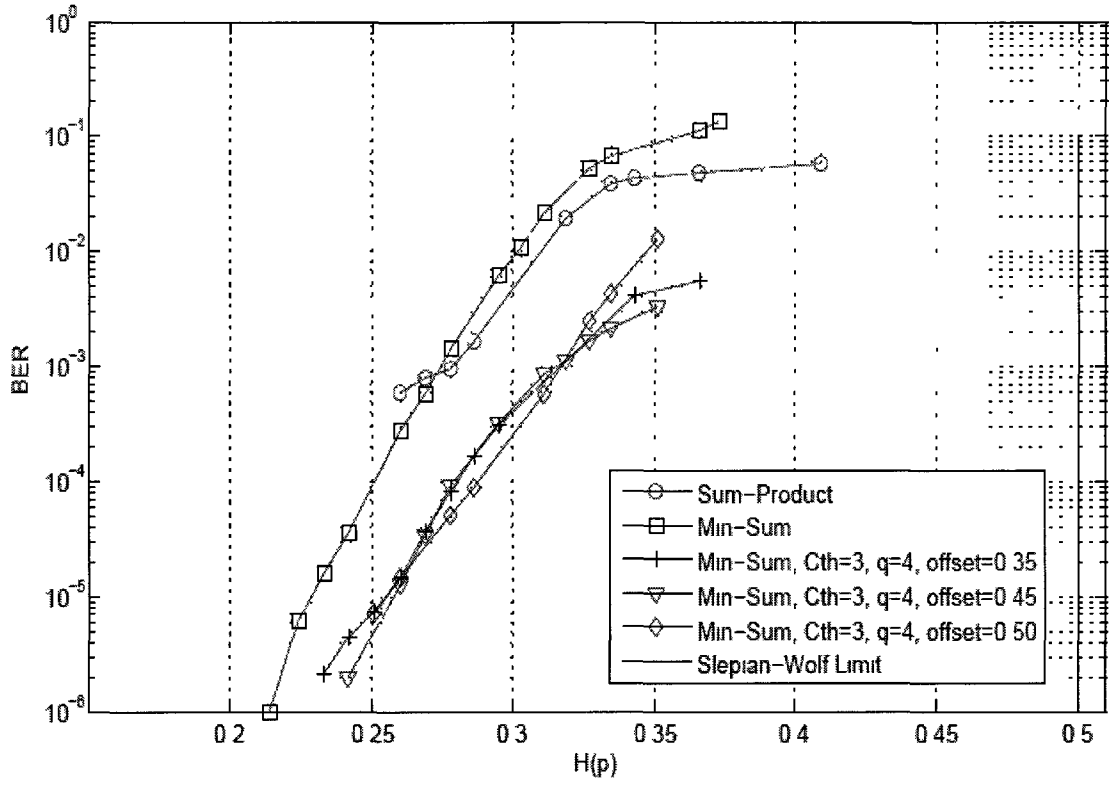


Figure 4-28 Performance of quantized offset min-sum for 4 quantization bits and for different offset factors

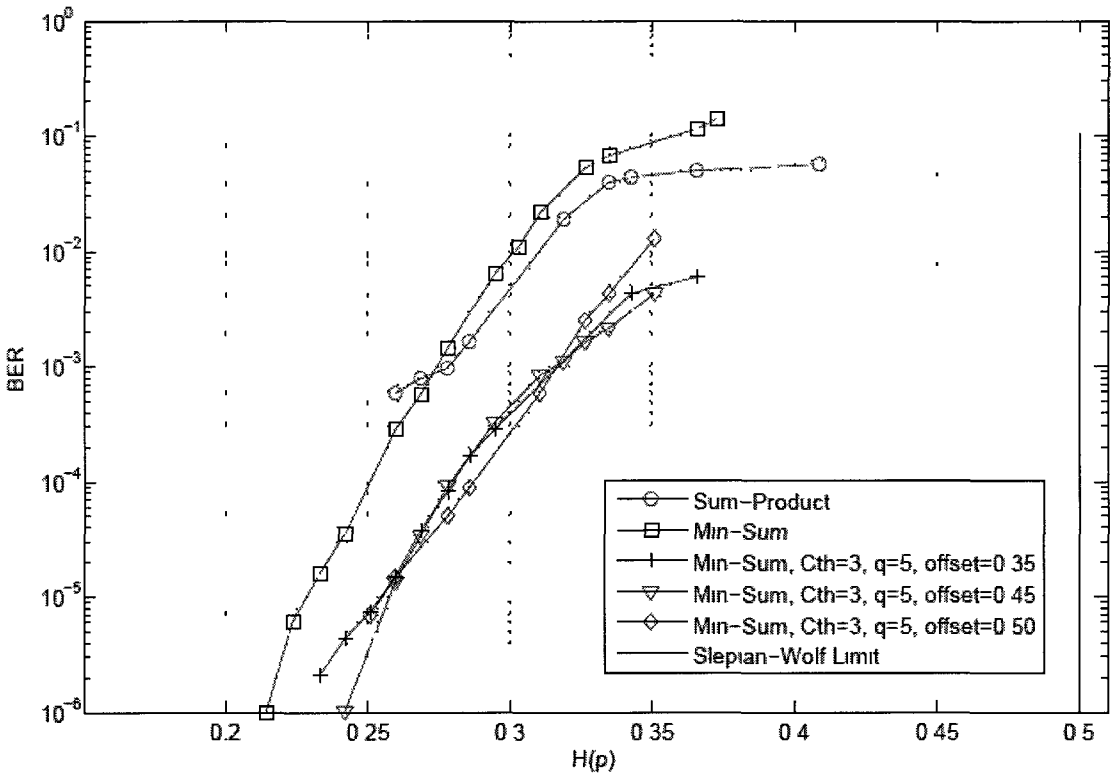


Figure 4-29 Performance of quantized offset min-sum for 5 quantization bits and for different offset factors

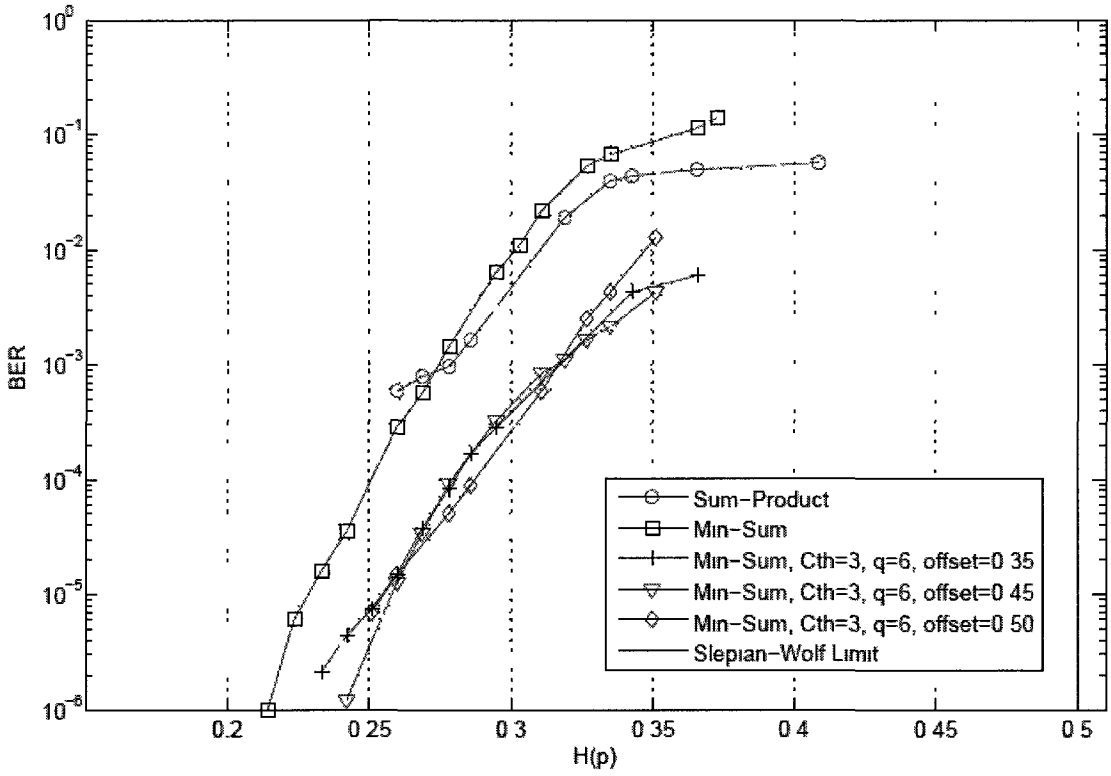


Figure 4-30 Performance of quantized offset min-sum for 6 quantization bits and for different offset factor

Chapter 5 Conclusions and Future Work

5.1 Conclusions

In this thesis, we investigated the low-complexity distributed source coding using LDPC codes. We reviewed two popular methods in distributed source coding which are syndrome approach and parity approach. We chose parity approach to build our system model.

Based on the system model which includes two the parallel channels, we designed a half-regular LDPC code, and applied the min-sum algorithm and its modifications to recover the compressed source X using the side information of the correlated source Y. The application of half-regular LDPC codes and (modified) min-sum algorithms is to reduce the complexity of the proposed DSC scheme compared to the application of LDPC codes with irregular degree distributions and the commonly used decoding algorithm of belief propagation.

We demonstrated that normalized and offset min-sum algorithms both perform very well in the context of DSC and that for short block lengths, their performance with our simple half-regular LDPC code is comparable with the best existing scheme which uses BP and irregular degree distributions. This is while the complexity of our scheme is significantly lower, partly because of the low complexity of the decoding algorithm and partly due to the simpler and sparser parity-check matrix of our LDPC code.

We also studied the performance of quantized versions of modified MS algorithms. Our results demonstrated that with only 4 quantization bits and with the proper selection of the correction factor, the quantized algorithms perform close to the floating-point versions.

5.2 Future Work

A few suggestions on future research are presented below.

1. Our designed LDPC code is half-regular and its structure is very simple. Other half-regular LDPC codes with different degree distributions can be examined.
2. It would be interesting to analyze modified min-sum algorithms in the context of DSC using the density evolution technique and compare the results with those obtained by simulations.

References

[1] D.Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, Vol. 19, No. 4, pp. 471-480, Jul. 1973.

[2] Z. Xiong, A.D. Liveris, and S. Cheng, "Distributed source coding for sensor networks", *IEEE Signal Processing Magazine*, Vol. 21, No. 5, pp. 80-94, Sep. 2004.

[3] R. G. Gallager, Low Density Parity Check Codes. PhD thesis, MIT, Cambridge, MA, 1963.

[4] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory,* Vol. 45, No. 2, pp. 399-431, 1999.

[5] T. J. Richardson, M. A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory,* Vol. 47, No, 2, pp. 619-637, February 2001.

[6] I. Csiszar, "Linear codes for sources and source networks: error exponents, universal coding," *IEEE Transactions on Information Theory,* Vol. 28, No. 4, pp. 585-592, July 1982.

[7] J. Chen and M. Fossorier, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Communication Letters,* Vol. 6, No. 5, pp. 208–210, May 2002.

[8] J. Chen and M. Fossorier, "Density evolution for BP-based decoding algorithms of LDPC codes and their quantized versions," *Proc. IEEE Globecom,* Nov. 2002, pp. 1378–1382.

[9] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X. Hu, "Reduced-Complexity Decoding of LDPC codes," *IEEE Transactions on communications,* Vol. 53, No. 8, pp. 1288-1299, August 2005.

[10] J. Zhao, F. Zarkeshvari, and A. Banihashemi, "On Implementation of Min-Sum Algorithm and Its Modifications for Decoding Low-Density Parity-Check (LDPC) Codes," *IEEE Transactions on communications,* Vol. 53, No. 4, pp. 549-554, April 2005.

[11] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): design and construction," *Proc. IEEE Data Compression Conference*, March 1999, pp. 158–167.

[12] A. D. Liveris, Z. Xiong, and C. N. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," *IEEE Communication Letters,* Vol. 6, No. 10, pp. 440–442, Oct. 2002.

[13] A. D. Liveris, Z. Xiong, and C. N. Georghiades, "Compression of binary sources with side information at the decoder using low-density parity-check codes," *Proc. IEEE Globecom,* Nov. 2002, pp. 1300–1304.

[14] D. Schonberg, K. Ramchandran, and S. S. Pradhan, "Distributed code constructions for the entire Slepian-Wolf rate region for arbitrarily correlated sources," *Proc. IEEE Data Compression Conference,* Mar. 2004, pp. 292-301.

[15] M. Sartipi and F. Fekri, "Source and channel coding in wireless sensor networks using LDPC codes," *Proc. IEEE Communications Society Conf. on Sensor Communications and Networks,* Oct. 2004, pp. 309–316.

[16] M. Sartipi and F. Fekri, "Distributed source coding in wireless sensor networks using LDPC coding: the entire Slepian-Wolf rate region," *Proc. IEEE Wireless Communications and Networking Conference,* Mar. 2005, pp. 1939–1944.

[17] M. Sartipi and F. Fekri, "Distributed Source Coding Using Short to Moderate Length Rate-Compatible LDPC Codes: The Entire Slepian-Wolf Rate Region", *IEEE Transactions on communications,* Vol. 56, No. 3, pp. 400-411, March 2008.

[18] D. Schonberg, K. Ramchandran, and S. S. Pradhan, "LDPC codes can approach the Slepian-Wolf bound for general binary sources," *Proc. 40th Annual Allerton Conference, Urbana-Champaign,* IL, Oct. 2002.

[19] J. Chen and M. Fossorier, "Near optimum universal belief propagation based decoding of LDPC codes," *IEEE Transactions on communications,* Vol. 50, No. 3, pp. 406-414, March 2002.

[20] J. Zhang, M. Fossorier, D. Gu, and J. Zhang, "Two-Dimensional Corection for Min-Sum Decoding of Irregular LDPC Codes," *IEEE Communications Letters,* Vol. 10, No. 3, pp. 180-182, March 2006.

[21] D. Oh and K. Parhi, "Min-Sum Decoder Architectures with Reduced Word Length for LDPC Codes", *IEEE Transactions on Circuits and System,* Vol. 57, No. 1, pp. 105-115, January 2010.

[22] M. Fossorier, M. Mihaljevic and H. Imai, "Reduced complexity iterative decoding of low density parity check codes based on belief propagation," *IEEE Transaction on Communication,* Vol. 47, No. 5, pp. 673-680, May 1999.

[23] A. Wyner, "Recent results in the Shannon theory," *IEEE Transactions on Information Theory,* Vol. 20, No. 1, pp. 2-10, Jan. 1974.

[24] H. Song and J. R. Cruz, "Reduced-complexity decoding of Q-ary LDPC codes for magnetic recording," *IEEE Transaction on Magnetics*, Vol. 39, No. 2, pp. 1081-1087, Mar 2003.

[25] S. K. Planjery, D. Declercq, S. K. Chilappagari and B. Vasic, "Multilevel decoders surpassing belief propagation on the binary symmetric channel," *Proceedings International Symposium on information Theory*, Jul. 2010, pp 769-774.

[26] T. Richardson and R. Urbanke. "The Capacity of Low-Density Parity Check Codes under Message-Passing Decoding," *IEEE Transactions on Information Theory,* Vol. 47, No. 2, pp. 599-618, Feb. 2001.

[27] F. Cabarcas and J. Garcia-Frias, "Approaching the Slepian-Wolf boundary using practical channel codes," *Proc. IEEE International Symposium on Information Theory*, June 2004, p. 330.