NAIST-IS-DD0561020

Doctoral Dissertation

On the Minimum Weight of Simple Full-length Array LDPC Codes

Kenji Sugiyama

September 2, 2008

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Kenji Sugiyama

Thesis Committee:

Professor Hiroyuki Seki	(Supervisor)
Professor Minoru Ito	(Co-supervisor)
Professor Toru Fujiwara	(Osaka University)
Associate Professor Yuichi Kaji	(Co-supervisor)

On the Minimum Weight of Simple Full-length Array LDPC Codes*

Kenji Sugiyama

Abstract

Low-density parity check (LDPC) codes is a class of linear block codes introduced by Gallager in 1962 for error correction over communication channels. LDPC codes show excellent performance by using an iterative message-passing decoding algorithm, and are regarded as one of the most promising codes. Indeed LDPC codes have been adopted as one of the standard error-control coding techniques in some communication systems such as DVB-S2, IEEE802.16e, and 10Gbps Ethernet, but the substantial performance of LDPC codes is not fully clarified yet. The minimum weight of a linear code has strong relationship to the performance of the code. Much efforts have been devoted to analyze the minimum weights of LDPC codes, but it is difficult to compute the exact minimum weight of long and randomly constructed LDPC codes. To the author's knowledge, no efficient algorithm has ever been found which can compute the exact minimum weight of arbitrary LDPC codes. On the other hand, the computation of the minimum weight becomes easier if the LDPC code under investigation has certain mathematical structures. For some classes of algebraically constructed LDPC codes, the minimum weight of the code has been studied eagerly, by making use of the mathematical structure of the code. This dissertation investigates the minimum weight of Simple Full-length Array LDPC(SFA-LDPC) codes which are algebraically constructed from a family of array codes. Moderate lengths and high rates SFA-LDPC codes are recognized to achieve similar performance as randomly constructed LDPC codes on the additive white Gaussian noise channel.

*Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0561020, September 2, 2008.

The first part of this dissertation is to develop a procedure to check if there is a codeword with specified weight in a SFA-LDPC code. Obviously the problem is solvable by testing all the binary vectors with a specified codelength and a specified weight, but such an approach will suffer from huge computational complexity caused by a large number of vectors. On the other hand, a purely algebraic approach will be too complicated if the code parameters are not small. To tackle the problem, the author combined these two approaches; the mathematical structure of the code is used to narrow the search space of the succeeding computer search. Thus the investigated approach can be regarded as a hybrid of the algebraic and the computer-oriented approaches. For example, the SFA-LDPC codes are invariant under a doubly transitive group of affine permutations, which gives significant constraint on the positions of nonzero components in a codeword. By making use of this kind of properties, possible position patterns of nonzero components in a codeword of a specified weight are classified into small number of subsets. Computer search is then employed to test if each subset of patterns really contains a codeword with the specified weight. The proposed procedure revealed exact minimum weights of some SFA-LDPC codes, which were not known previously.

The second part of this dissertation gives analytical upper-bound limits of the minimum weights of some classes of SFA-LDPC codes. An SFA-LDPC code is defined according to two integer parameters, but the experimental results obtained in the former half of the dissertation suggest that the minimum weight is independent of one of the parameters. From the results, a strong conjecture arises that the minimum weights of SFA-LDPC codes with column weight four and five are upper bounded by 10 and 12, respectively. This conjecture is positively proved in the second part of the dissertation. By combining the results with previously known lower-bound limits, we can conclude that the minimum weights of the SFA-LDPC codes with column weight four is exactly 10 and those of column weight five is 10 or 12.

Keywords:

coding theory, error correcting code, LDPC code, SFA-LDPC code, minimum weight, minimum distance

Simple Full-length アレー型 LDPC 符号の最小重みに関する研究*

杉山 憲司

内容梗概

誤り訂正符号は雑音のある通信路を介して受信された受信シンボルに誤りが発生しているか否かを検査し，もし誤りが検出されれば決められた手続きにしたがって送信語を推定する符号化技術である．受信語に誤りが発生しているか否かを検出するためには，通信路上の雑音によって送信語のいくつかのビットが反転しても他の送信語と同じになってしまわないことが必要である．相異なる二つの符号語で最も近いものの距離を最小距離という．ここで符号語間の距離とは相異なるビットの数を示す．符号の最小距離は，その符号の性能を評価する上で非常に重要なパラメータである．

低密度パリティ検査 (Low Density Parity Check, LDPC) 符号は疎なパリティ検査行列で定義された線形符号で，1962 年に Gallager によって提案された．疎なパリティ検査行列とは，行列のほとんどの成分が 0 であり，1 の成分が相対的に少ない行列である．LDPC 符号は，Shannon 限界に迫る高い誤り訂正能力を持つとして注目されており，ディジタル衛星放送規格 DVB-S2 への採用や IEEE802.16e, IEEE802.3(10G BASE-T) などで実用化されている．しかし，LDPC 符号の性能は完全に定量的な評価がなされているわけではなく，例えば最小距離の評価は定性的な性能評価（統計的手法による性能評価）方法を用いることがほとんどであった．これは，LDPC 符号のパリティ検査行列が従来，計算機を用いてランダムに生成されることが多かったことに起因する．一方，近年擬似巡回 LDPC 符号のように代数的符号の構成法を応用した LDPC 符号の構成法が提案され，この代数的構造を利用した符号の性能評価方法が提案されるなど，定量的な性能評価を行う研究も行われている．

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文，NAIST-IS-DD0561020, 2008 年 9 月 2 日.

Simple full-length アレー型 LDPC (SFA-LDPC) 符号は，代数的に構成されたLDPC符号の部分クラスであり，単純で整った数学的構造を持つ．本論文では，このSFA-LDPC符号の最小重みについて議論する．SFA-LDPC符号の最小重みは, Yang, Mittelholzer らにより代数的なアプローチから研究がなされてきたが，真の最小重みは，小さなパラメータで規定された符号のクラスについて知られているのみである．パラメータが小さくない符号については，その最小重みの下界，上界が議論されているが，上下界の間には大きな隔たりがあり，正確な最小重みの解明にはいたっていない．純粋に代数的なアプローチだけに頼ったのでは，符号を規定するパラメータが大きくなるに従い，仔細に検討すべき符号語のパターンが爆発的に増加するため，最小重みを特定することは困難であると予測される．

そこで，本論文の前半ではSFA-LDPC符号の最小重みの上界値を検査するアルゴリズムを提案する．ある符号の中に特定の重みを持つ符号語が存在するか否かを検査するには，その特定の重みを持つ可能性がある符号ベクトルの全てを検査することにより可能であるが，このようなアプローチは検査すべき符号ベクトルの数が膨大になるため現実的ではない．本論文のアプローチは，代数的なアプローチに計算機を援用し，上記の代数的な手続きを自動化するものである．この方式により，最小重みが未知であったいくつかのSFA-LDPC符号について，その正確な値を明らかにすることができた．

本論文の後半では検査行列の列重みが4のとき，および5のときのSFA-LDPC符号の最小重みの上界の評価の代数的証明を与える．前半の結果を詳細に分析することにより，SFA-LDPC符号を規定する2つのパラメータのうちのひとつ (パリティ検査符号の列重みを制御するパラメータ) を固定した場合，もう片方のパラメータを変えてもその符号のなかに必ず含まれる符号語のパターンを特定できる．そのパターンを代数的に証明することにより，最小重みの上限の評価を示す．この最小重みの上限の評価手法を用い，パリティ検査行列の列重みが4のとき，および5のときのSFA-LDPC符号の最小重みの上限が，それぞれ10, 12で与えられることを示した．これらは現在までに知られている最小重みの上限の評価を大幅に改善した評価である．既知の下界とあわせることにより，列重みが4のときの最小重みは真に10，列重みが5のときの最小重みは10または12であることがいえる．

キーワード

符号理論, 誤り訂正符号, LDPC符号, SFA-LDPC符号, 最小重み, 最小距離

List of Publications

Journal Paper

(1) K. Sugiyama and Y. Kaji, "On the Minimum Weight of Simple Full-length Array LDPC Codes", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E91-A, No.6, pp. 1502–1508, June, 2008.

International Conferences (Reviewed)

(2) K. Sugiyama and Y. Kaji, "On the Minimum Weight of Simple Full-length Array LDPC Codes", Proceedings of the IEEE International Symposium on Information Theory, Nice, France, June 2007.

(3) K. Sugiyama and Y. Kaji, "A Minimum Weight Test for a Certain Subclass of Array LDPC Codes", Proceedings of International Symposium on Information Theory and Its Applications, Seoul, Korea, pp. 366–371, November 2006.

Workshops

(4) K. Sugiyama and Y. Kaji, "On the Evaluation of Upper Bound of the Minimum Weight of Simple Full-length Array LDPC Codes", Proceedings of 2007 Symposium on Information Theory and Its Applications, November 2007.

(5) K. Sugiyama and Y. Kaji, "On the Evaluation of Upper Bounds of the Minimum Weights of SFA-LDPC Codes using Mathematical Structure of Supports of Minimum Weight Codewords", Proceedings of LDPC codes Workshop 2007, August 2007 (in Japanese).

(6) K. Sugiyama and Y. Kaji, "An Algebraic Derivation of the Minimum Weight of Simple Full-length Array-LDPC Code with Certain Parameters", IEICE Technical Report, IT2006-79, pp. 105-110, March 2007.

(7) K. Sugiyama and Y. Kaji, "On the Minimum Weight of a Certain Subclass of Array LDPC Codes", Proceedings of 2006 Symposium on Information Theory and Its Applications, pp. 449–452, November 2006.

(8) K. Sugiyama and Y. Kaji, "A Minimum Weight Test for a Certain Subclass of Array LDPC Codes", IEICE Technical Report, IT2006-2, pp. 7–11, May 2006.

(9) K. Sugiyama and T. Kohnosu, "On the Minimum Distance of Array-type LDPC Codes", Proceedings of 2004 Symposium on Information Theory and Its Applications, pp. 1–4, December 2004.

Technical Report

(10) K. Sugiyama and Y. Kaji, "A Minimum Weight Test Procedure for a Certain Subclass of Array LDPC Codes", NAIST Technical Report, No. NAIST-IS-TR2008001, Nara Institute of Science and Technology, January 2008.

Acknowledgements

During the course of this work, I have received help from many people. I would like to thank all of them. First and foremost, Professor Hiroyuki Seki of Nara Institute of Science and Technology gave me great research direction, which greatly helped me improve the quality of this study. I am grateful to the members of the dissertation committee: Professor Minoru Ito of Nara Institute of Science and Technology and Professor Toru Fujiwara of Osaka University for their helpful suggestions and comments in reviewing this dissertation. My deepest appreciation goes to Associate Professor Yuichi Kaji of Nara Institute of Science and Technology for his continuous support of the work. Without his efforts, the result of my doctoral dissertation would have been far less satisfactory. Finally, I would like to thank all the members of Seki Laboratory for giving me much encouragement and help.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Error Control and Low-density Parity Check Codes

In the modern information society, people communicate each other over computer networks such as the Internet. Media that are used to convey information are called *communication channels*, or simply *channels*. A computer network is a network of channels that connect computers and convey digital data. The "communication" discussed in this dissertation is defined as a transmission of digital data over a channel. There are many different communication channels but the channels are usually noisy; transmitted data may be changed and/or lost by noise which occurs on the channel. Thus, to realize high-quality network communication, it is necessary to provide error-free communication systems. An *error correcting code* is one of the most essential technologies for implementing reliable communication system. Modern error correcting codes can increase reliability of communications with less drawback than simple and naive techniques such as the increasment of transmission energy and re-transmission of damaged data.

In 1948, Shannon published a historical paper[12] which discusses the significance and the limit of error correcting codes. Figure 1.1 shows the model of digital communication system considered by Shannon. Certain information is generated at an information source and the information is provided to an *encoder* as a sequence of symbols. The encoder transforms, or *encodes*, the sequence into

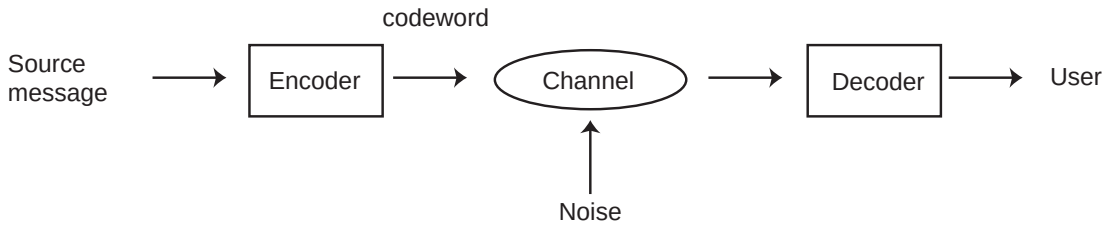

Figure 1.1. The digital communication system

a *codeword*, and transmits the codeword over a channel. The channel is noisy in general, and the transmitted codeword can be modified by the noise. At the recipient end, the *decoder* tries to detect and correct errors to estimate the transmitted codeword. A *code* which is the set of codewords must be designed so that the degradation caused by the noise is as small as possible. A number of researchers devoted their efforts for this sake, and the studies now constitute a significant portion of the Information Theory.

The importance of *linear codes* was found in the very beginning of the study of the Information Theory. A code is said to be linear if its codewords constitute a *linear space*. A number of fine mathematical properties of linear spaces make linear codes useful. For example, a linear space has a basis, and an arbitrary element in the space is represented as a linear sum of elements in the basis. This property contributes to realize an efficient encoder using a *generator matrix*. *Parity check equations* are derived from the generator matrix, and the matrix representation of the equations defines a *parity check matrix*. The product of the parity check matrix and a received vector is a *syndrome*, and the syndrome can be regarded as a fingerprint of the error which might be involved in the received vector. In summary, linear codes are considered as efficient, practical and useful from engineering viewpoints.

The *minimum distance* is one of the most important parameters which are to measure the ability of error correcting codes. The minimum distance is the smallest *Hamming distance* between two different codewords in the code, where Hamming distance is the number of symbol positions in which given codewords (vectors) differ. If the minimum distance of a code is d, then the code can correct up to $\lfloor (d-1)/2 \rfloor$ bit errors occurred in one transmitted codeword. Hence the minimum distance directly affects the ability of the code, and a number of efforts

have been made to construct codes with large minimum distance. In the case of linear codes, the minimum distance equals to the *minimum weight* which is defined as the smallest *Hamming weight* of non-zero codewords of the code. This property simplifies the discussion of minimum distance of linear codes.

For long years, the minimum distance has been a focal point in the design of good error correcting codes. A number of codes have been proposed in 1960's and 70's, and they were often evaluated by the minimum distance. In other words, a code was considered to be good if it has large minimum distance (and additionally, if it has a reasonably efficient decoding algorithm). The most successful approach in this direction was to make use of algebraic properties to design linear codes. Algebraic codes such as the BCH and Reed-Solomon codes were proposed in those era, survived for several decades and are still widely used in many information and communication systems today. Those algebraic codes have fine mathematical structures, though, their performance is far less than the theoretical limit shown in the Shannon's noisy channel theorem. It is believed that the performance will be improved if we consider algebraic codes with large parameters, but such large algebraic codes will suffer from massive decoding complexity, and not be practical at all.

The *low-density parity check codes*(*LDPC codes*) were introduced by Gallager in 1962, but have been forgotten for long years. An LDPC code is a linear block code whose parity check matrix is very sparse, meaning that most components of the matrix is zero. Gallager considered an iterative message passing algorithm on a bipartite graph which is defined from the parity check matrix, and showed that the algorithm works effectively when the graph is sparse. He also showed that the performance of the code, combined with the iterative decoding algorithm, improves as the code length increases. However, with the technology of that time, it was difficult to realize practical encoders and decoders for such a long LDPC code. With moderated size, the performance of LDPC codes is not as good as other algebraic codes, and actually, the minimum distance of LDPC codes is smaller than that of algebraic codes with almost the same parameters. Consequently many people considered LDPC codes impractical, and the study of LDPC codes has been suspended for long years. In late 1990's, stimulated by the study of Shannon-limit approaching *Turbo codes* and iterative decoding

algorithms, some coding theorists re-discovered the LDPC codes. The technology have developed significantly, and now we will be able to realize practical encoders and decoders for relatively large LDPC codes, and the codes achieve performance which is very close to the Shannon-limit.

After its re-discovery, a number of people have worked on the LDPC codes. For a certain time, no criteria were known for constructing good LDPC codes; people constructed a parity check matrix randomly, and verified if the obtained code shows good performance or not. Later, some coding theorists brought algebraic methods, which were once used to construct algebraic codes in 1960's, and tried to use the methods for constructing good LDPC codes. The most remarkable outcome of the study is the discovery of *Quasi-Cyclic* LDPC codes. Compared to the randomly constructed LDPC codes, we can have more control on the design of Quasi-Cyclic LDPC codes, which means that we are able to obtain powerful and manageable LDPC codes.

1.2. Code Parameter and the Minimum Weight

This dissertation discusses the minimum weights of a certain subclass of LDPC codes. An error correcting code is characterized by a number of parameters which have strong relation to the performance of the code. Roughly speaking, such parameters can be classified into two types; parameters which are oriented to the representation of the code, and parameters which are oriented to the code itself. An example of the first type of parameters is the *girth* of the code. The girth of an LDPC code is the size (length) of the shortest cycle in the Tanner graph of the code, where the Tanner graph is a bipartite graph whose incident matrix is the parity check matrix of the code. Note that a single code can be represented by many different parity check matrices, and the girth depends on the representation of the code. In the discussion of LDPC codes, the girth is considered to be a significant parameter because it affects the performance of the code under a message-passing decoding algorithm. If the girth is small, then the performance of the message-passing algorithm is far beyond that of the optimum (maximum likelihood) decoding algorithm. A typical parameter which is oriented to the code itself is the *weight distribution* of the code. The weight distribution

is the distribution of the number of codewords grouped by Hamming weight. Theoretically saying, the optimal performance of the code is fully determined by the weight distribution, but it is quite difficult in general to reveal the weight distribution in many cases. When the weight distribution is not available, we may approximate the performance by using the number of minimum weight codewords only. To determine the number of minimum weight codewords is much easier than to determine the weight distribution completely, and to compute the minimum weight is the first step to determine the number of minimum weight codewords.

However, it is difficult in general to analyze the exact minimum weight of long practical LDPC codes. It is known that the problem of computing the minimum weight of a binary linear code is *NP-hard* in general, and the corresponding decision problem is *NP-complete*[17]. The complexity can be reduced for structured codes, but the author could not find research results in that direction. Tanner discussed in [16] certain bounds on the minimum weights of LDPC codes by utilizing the Tanner graph. Hu and Fossorier proposed a probabilistic procedure to compute the minimum weight of LDPC codes by using a decoding algorithm for LDPC codes[7], and Hirotomo et al. proposed another probabilistic procedure[6] which utilizes the Stern's algorithm. To the author's knowledge, though, no efficient algorithm is known for computing the exact minimum weight of arbitrary LDPC codes. On the other hand, there are some results on the minimum weight of LDPC codes which have certain mathematical structures. MacKay showed that the minimum weight of regular-Quasi-Cyclic LDPC codes with column weight j is less than or equal to $(j+1)!$[9], and Fossorier discusses the minimum distance of Quasi-Cyclic LDPC codes from circulant permutation matrices[4]. Mittelholzer has derived in [11] some upper-bound limits of the minimum weights of certain *array LDPC codes*[2]. Array LDPC codes is a class of Quasi-Cyclic LDPC codes which are algebraically constructed from a family of array codes[1, 3]. Mittelholzer assumes some more additional conditions on the array LDPC codes, and thus it will be better to distinguish the investigated class from the class of general array LDPC codes. In this dissertation, let us name *simple full-length array LDPC codes* (*SFA-LDPC codes*) for the class of LDPC codes.

Mittelholzer showed that the minimum weight of the SFA-LDPC code with column weight 4 is 12 or less[11], which significantly improves the upper-bound

limit $(4+1)! = 120$ given by MacKay [9] for the general regular case. Mittelholzer also showed that the minimum weights of SFA-LDPC codes with column weight 5 and 6 are upper-bounded by 20 and 32, respectively[11]. The study of Mittelholzer is followed by Yang in [18]. Mittelholzer discussed the upper-bound of the minimum weight of SFA-LDPC codes, but Yang discusses its lower-bound. With very careful analysis, Yang showed that the minimum weight of the code is 10 or more if the column weight is 4 and the code length is 49 or more. Together with the Mittelholzer's upper-bound, this result implies that the minimum weight of SFA-LDPC codes with column weight 4 is either 10 or 12, because the code does not have odd-weight codewords. For the cases of column weight 5 and 6, we know that the lower-bounds of the minimum weights are 10 since the minimum weight of those codes are greater than or equal to that of corresponding SFA-LDPC codes with column weight 4. Together with the upper-bounds given by Mittelholzer, we have that the minimum weight of an SFA-LDPC code with column weight 5 is greater than or equal to 10 and less than or equal to 20. Similarly, the minimum weight of an SFA-LDPC code with column weight 6 is greater than or equal to 10 and less than or equal to 32.

1.3. SFA-LDPC codes and the Contribution of this Dissertation

An SFA-LDPC code is defined according to two integer parameters p and j. Let $C_A(p, j)$ denote the SFA-LDPC code defined by given p and j. Roughly speaking, taking p larger makes the code length longer and code rate higher, while taking j larger makes the code rate lower. Consequently it is naturally expected that taking p large, with j fixed, degrades the performance of the code in general. It is also known that the Tanner graph of an SFA-LDPC code does not contain a cycle of length four, but the graph contains many cycles of length six. Hence it is conjectured that the iterative message-passing algorithm will not pull out the full performance of the SFA-LDPC codes.

In spite of these negative conjectures, SFA-LDPC codes with large p show very good performance. Figures 1.2 and 1.3 show the block and bit error performances of $C_A(p, 4)$ for several p (The figures are on pp.9–10). The horizontal axis of each

Table 1.1. Code parameters of Figures 1.2, 1.3

p	code length	information bits	rate
13	169	120	0.710
17	289	224	0.775
19	361	288	0.798
23	529	440	0.832
29	841	728	0.866
31	961	840	0.874

graph indicates E_B/N_0, the *signal energy per bit to noise power spectral density ratio* (*SNR*), and the vertical axis shows the *block error rate* (bit error rate, in the Figure 1.3) in the logarithmic scale. Table 1.1 shows the code length, the number of information bits and the code rate for each value of p. Note that the code rate increases as the value of p increases. The decoding algorithm is the standard sum-product decoding algorithm with twenty iterations at maximum. For low SNR, shorter codes, which have smaller code rate, show better performance than longer codes. This seems obvious if we consider the code rate and the performance of the code. On the other hand, for SNR more than 4.0dB, the curves cross and longer codes show better performance than shorter codes. It is difficult to explain this phenomenon in terms of the rate and the girth of the code. To reveal the mechanism behind this strange behavior of SFA-LDPC codes, the weight distribution of SFA-LDPC codes will be a great help. However, as we have seen in the previous section, even the minimum distance has not yet been determined for SFA-LDPC codes. The main objective of this dissertation is to clarify the minimum weight of SFA-LDPC codes, and try to reveal the mechanism behind the results shown in Figures 1.2 and 1.3.

The basic approach considered in this study is to make use of computer search combined with the mathematical analysis taken by Mittelholzer and Yang. The approach given by Yang is powerful, but the analysis becomes too complicated if the code parameters are not small. To tackle the problem, the author first combined Yang's algebraic analysis together with computer search. Starting from Yang's work, the author developed some notions related to the minimum weight codewords of $C_A(p, j)$. The notion can be used to restrict possible support sets of

the code, and contributes to prune hopeless computation in the computer search. The author implemented the idea as a computer program, and computed the exact minimum weights of $C_A(p, j)$ with moderate parameters. The results will be presented in Chapter 3 in detail. From the results, the minimum weight of $C_A(p, 4)$ was discovered as 10, and that of $C_A(p, 5)$ was 12 as far as the program examined.

The semi-experimental results in the former half of the dissertation suggest the conjecture described below, and the latter half of the dissertation is devoted to proving the conjecture positively. The exact minimum weights were revealed for several SFA-LDPC codes, and the author found that $C_A(p, 4)$ is 10 for all prime numbers between 11 and 79. The minimum weight of $C_A(p, 4)$ was not known for $p > 79$, but according to the semi-experimental results, it seems that $C_A(p, 4)$ is 10 for all $p \geq 13$. To show this conjecture analytically, minimum weight codewords are collected by modifying the computer program developed in the former half of the dissertation, and classified according to a certain criteria. As the result of the classification, it can be shown that each of $C_A(p, 4)$ with $p \geq 11$ contains a special codeword, and the special codewords share a structure which is independent of p. By carefully analyzing the structure, it can be shown that $C_A(p, 4)$ always contains a codeword with weight 10 for any $p > 7$. Similarly, it is shown that $C_A(p, 5)$ always contains a codeword with weight 12. Detailed discussion for the above proof will be given in Chapter 4 of the dissertation. By combining the results with previously known lower-bound limits, we can conclude that the minimum weights of the SFA-LDPC codes with column weight four is exactly 10 and those of column weight five is 10 or 12.

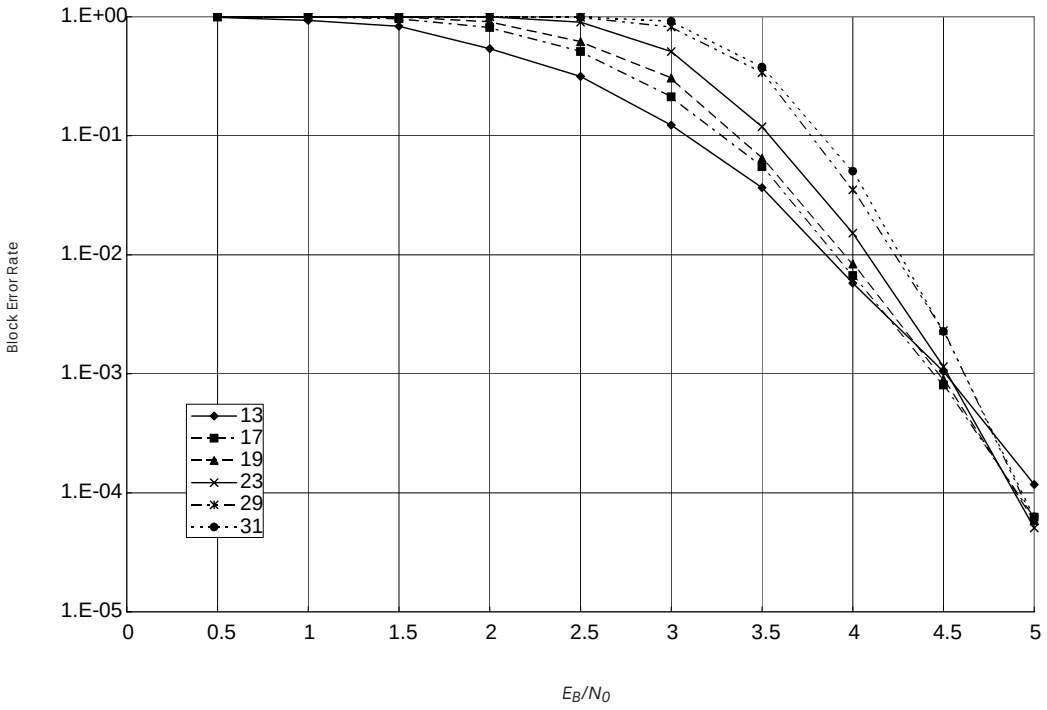

Figure 1.2. Block error rate of SFA-LDPC codes with column weight 4

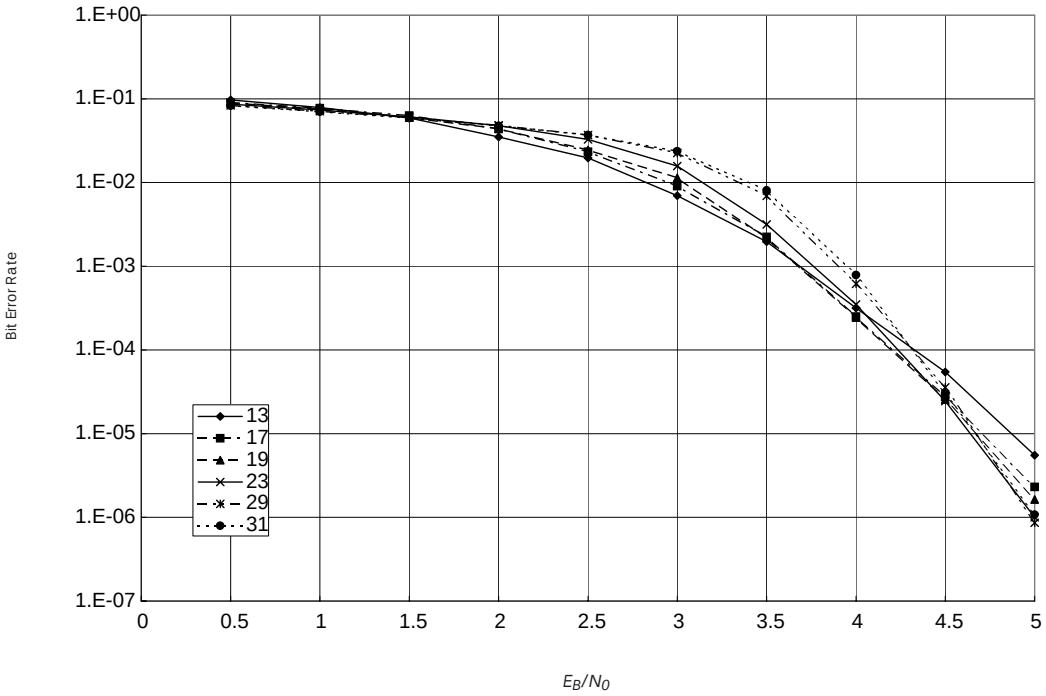

Figure 1.3. Bit error rate of SFA-LDPC codes with column weight 4

Chapter 2

Preliminaries and Basic Observations

In this chapter, the definition of the considered class of LDPC codes and some properties of the code are reviewed. These properties are required to discuss the minimum weight test procedure that will be dealt in Chapter 3 and to give the proof of the minimum weights of some classes of LDPC codes that will be dealt in Chapter 4.

2.1. Simple Full-length Array LDPC Codes

Let p be a prime number, and k and j be integers satisfying $k, j \leq p$. The binary *array LDPC code* $C_A(p, j, k)$ is a null space of the $pj \times pk$ binary matrix

$$H_A(p,j,k) = \begin{bmatrix} I & I & I & \cdots & I \\ I & P & P^2 & \cdots & P^{k-1} \\ I & P^2 & P^4 & \cdots & P^{2(k-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & P^{j-1} & P^{(j-1)\cdot 2} & \cdots & P^{(j-1)(k-1)} \end{bmatrix}$$

where I is the $p \times p$ identity matrix and P is a cyclic shift matrix defined by

$$P = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}.$$

Every row and column of P^n contains a single 1 with 0s everywhere else. If we write $P^n = [a_{l,i}]$ $(1 \le l, i \le p)$,

$$\begin{cases} a_{l,i} = 1 & (l \equiv i + n \pmod{p}), \\ a_{l,i} = 0 & (\text{otherwise}). \end{cases}$$

The matrix $H_A(p, j, k)$ is the parity check matrix of the code $C_A(p, j, k)$.

Now let us consider a special case such that $k = p$, and call this special array LDPC code a *simple full-length array LDPC code* (*SFA-LDPC code* for short). The "array LDPC codes" discussed in [11] and [18] are indeed this SFA-LDPC code. For simplicity, $C_A(p, j, p)$ and $H_A(p, j, p)$ are respectively written as $C_A(p, j)$ and $H_A(p, j)$. Let $d(p, j)$ denote the minimum distance of $C_A(p, j)$. The *rate* of $C_A(p, j)$ is $1 - \dfrac{pj - j + 1}{p^2}$, because the *rank* of $H_A(p, j)$ is $pj - j + 1$. (The rate of a code is the ratio of the number of information bits to the total code length. Besides, the rank of the parity check matrix determins the number of rows of the matrix that are linearly independent and this number equals to the number of parity check bits in a codeword. The total code length is the sum of the number of information bits and the number of parity bits.)

It can be easily shown that column vectors in $H_A(p, j)$ are all different. It is also obvious from the definition that a column vector of $H_A(p, j)$ can be decomposed into j subsequences each of which has length p and weight one. That is, if we write $H_A(p, j) = [h_{l,i}]$ $(1 \le i \le p^2$ and $1 \le l \le pj)$, then the weight of the subsequence

$$\boldsymbol{h}_i^r = \begin{pmatrix} h_{p(r-1)+1,i} \\ h_{p(r-1)+2,i} \\ \vdots \\ h_{p(r-1)+p,i} \end{pmatrix}$$

is exactly one for any $1 \leq i \leq p^2$ and $1 \leq r \leq j$. Because each subsequence of any column vector of $H_A(p,j)$ has exactly weight one, every codeword in $C_A(p,j)$ has an even weight (since the sum of odd column vectors can not be zero in the modulus of 2).

For a vector $\boldsymbol{v}$ with weight one, let $\phi(\boldsymbol{v})$ denote the position of the nonzero component in $\boldsymbol{v}$ where the position of the first component of $\boldsymbol{v}$ is regarded as zero. For example, $\phi((0,1,0,0)^T) = 1$ and $\phi((0,0,0,1)^T) = 3$. The value of ϕ is not defined for a vector whose weight is not one. Extend this notation ϕ to a column vector

$$\boldsymbol{h}_i = \left(\begin{array}{c} \boldsymbol{h}_i^1 \\ \hline \boldsymbol{h}_i^2 \\ \hline \vdots \\ \boldsymbol{h}_i^j \end{array} \right) = \left(\begin{array}{c} h_{1,i} \\ \vdots \\ \hline h_{p,i} \\ \hline h_{p+1,i} \\ \vdots \\ \hline h_{2p,i} \\ \hline \vdots \\ h_{pj,i} \end{array} \right)$$

of $H_A(p,j)$ in such a way that

$$\phi(\boldsymbol{h}_i) = \left(\begin{array}{c} \phi(\boldsymbol{h}_i^1) \\ \vdots \\ \phi(\boldsymbol{h}_i^j) \end{array} \right)$$

and also extend ϕ to the matrix $H_A(p,j)$ as

$$\phi(H_A(p,j)) = [\phi(\boldsymbol{h}_1), \ldots, \phi(\boldsymbol{h}_{p^2})].$$

For example, the parity check matrix of $C_A(3,2)$

$$H_A(3,2) = \left[\begin{array}{ccccccccc} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{array} \right]$$

is written by the ϕ-representation as

$$\phi(H_A(3,2)) = \begin{bmatrix} 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 1 & 2 & 1 & 2 & 0 & 2 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

Note that $H_A(p,j)$ and $\phi(H_A(p,j))$ are freely interchangeable. In other words, ϕ is just to represent $H_A(p,j)$ in a compact manner.

Lemma 2.1 *For an integer i with $1 \le i \le p^2$, let k and k' be integers satisfying $i = pk + k'$ and $1 \le k' \le p$. The i-th column vector of $\phi(H_A(p,j))$ can be written as*

$$(k'-1, k'-1+k, \ldots, k'-1+(j-1)k)^T \pmod{p}.$$

Proof. $H_A(p,j)$ has $j \times p$ blocks and each block has size $p \times p$. Now let $\boldsymbol{h}_i^r$ be the r-th component of i-th column vector of $\phi(H_A(p,j))$, where $1 \le r \le j$ and $1 \le i \le p^2$. Two parameters k and k' are obtained by

$$\begin{cases} k = \lfloor \dfrac{i-1}{p} \rfloor, \\ k' = i - pk. \end{cases}$$

Thus, from the construction of $H_A(p,j)$, $\phi(\boldsymbol{h}_i^1)$ is $k'-1$, since $\boldsymbol{h}_i^1$ is the k'-th column of I, where I is the $p \times p$ identity matrix. The i-th column vector of the parity check matrix is in the $(k+2)$-th column-block (because $0 \le k < p$). Hence, $\phi(\boldsymbol{h}_i^r)$ is $(k'-1)+(r-1)k$, since $\boldsymbol{h}_i^r$ is the k'-th column vector of $P^{(r-1)(k+1)}$. ∎

Lemma 2.1 means that components in a column vector in $\phi(H_A(p,j))$ constitute an arithmetic sequence. The next lemma implies that two different column vectors in $\phi(H_A(p,j))$ cannot have two or more common components in common positions.

Lemma 2.2 *Consider two different columns $\phi(h_{i_1})$ and $\phi(\boldsymbol{h}_{i_2})$ in $\phi(H_A(p,j))$ (thus $i_1 \ne i_2$ and $1 \le i_1, i_2 \le p^2$). The vectors $\phi(h_{i_1})$ and $\phi(\boldsymbol{h}_{i_2})$ have no two components in common, that is, if the j_1-th components of $\phi(h_{i_1})$ and $\phi(\boldsymbol{h}_{i_2})$ are the same, then, for any other j_2 with $j_1 \ne j_2$ and $1 \le j_2 \le j$, the j_2-th components of $\phi(\boldsymbol{h}_{i_1})$ and $\phi(\boldsymbol{h}_{i_2})$ cannot be the same.*

Proof. Let k_1, k_1', k_2 and k_2' be integers satisfying $i_1 = pk_1 + k_1'$ and $i_2 = pk_2 + k_2'$. If there is an integer j_2 in the proposition of the lemma, then

$$\begin{aligned} k_1' - 1 + (j_1 - 1)k_1 &\equiv k_2' - 1 + (j_1 - 1)k_2 \bmod p, \\ k_1' - 1 + (j_2 - 1)k_1 &\equiv k_2' - 1 + (j_2 - 1)k_2 \bmod p, \end{aligned}$$

from Lemma 2.1. The equations reduce to $(k_1 - k_2)(j_1 - j_2) \equiv 0 \bmod p$, but this cannot happen because both of $(k_1 - k_2)$ and $(j_1 - j_2)$ are relatively prime to p, a contradiction. ∎

Definition 2.3 *A collection of integers $q_1, q_2, \ldots, q_n$ is said to satisfy the* cancel-out condition *if no integer appears odd times in $q_1, q_2, \ldots, q_n$.*

Lemma 2.4 *Let $\boldsymbol{v}_1, \ldots, v_n$ be binary vectors with length p and weight one. We have that $v_1 + \cdots + \boldsymbol{v}_n = 0 \bmod 2$ if and only if the collection $\phi(\boldsymbol{v}_1), \ldots, \phi(\boldsymbol{v}_n)$ satisfies the cancel-out condition.*

Proof. It is obvious from the fact that the sum is taken in the modulus of two and from Definition 2.3. ∎

2.2. Support Matrices

Let $\boldsymbol{v} = (v_1, \ldots, v_{p^2})$ be a binary vector of length p^2. The vector v is a codeword of $C_A(p, j)$ if and only if $H_A(p, j)\boldsymbol{v}^T = \sum_{\substack{1 \le i \le p^2 \\ v_i = 1}} h_i = \boldsymbol{0}$. Thus, $\boldsymbol{v} \in C_A(p, j)$ if and only if the sum of all column vectors in

$$\{\boldsymbol{h}_i | 1 \le i \le p^2, v_i = 1\}$$

equals to zero in the modulus of two. Now define

$$\mathrm{supp}(\boldsymbol{v}) = \{\phi(h_i) | 1 \le i \le p^2, v_i = 1\},$$

and call it the *support*[1] of v. The support contains the column vectors of $\phi(H_A(p, j))$ which correspond to nonzero components of $\boldsymbol{v}$. If the weight of $\boldsymbol{v}$ is w, then

[1]The word "support" is often used to denote the set of positions of nonzero components in a vector, but the word is used in slightly different manner in this dissertation.

supp($\boldsymbol{v}$) contains exactly w column vectors because all column vectors in $H_A(p, j)$ are different. Order these w column vectors in an arbitrary way, and construct a matrix $S_v = [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_w]$. This matrix is called a *support matrix* of v. By applying Lemma 2.4 to each row of $S_{\boldsymbol{v}}$, we have the following corollary.

Corollary 2.5 *The vector $\boldsymbol{v}$ is a codeword of $C_A(p, j)$ if and only if the cancel-out condition holds for any row of $S_{\boldsymbol{v}}$. In this case let us say that $S_{\boldsymbol{v}}$ satisfies the* cancel-out condition.

For example, $\boldsymbol{v}_1 = 100101001$ is a correct codeword of $C_A(3, 2)$ but $\boldsymbol{v}_2 = 100101000$ is not. Their support matrices are

$$S_{\boldsymbol{v}_1} = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad S_{\boldsymbol{v}_2} = \begin{bmatrix} 0 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix}$$

from $\phi(H_A(3, 2))$ (given in (2.1)). The matrix $S_{\boldsymbol{v}_1}$ satisfies the cancel-out condition, but $S_{\boldsymbol{v}_2}$ does not. (2 occurs only once in the first row, and 1 occurs only once in the second row).

It is easily understood that $C_A(p, j)$ contains a codeword of weight w if and only if $\phi(H_A(p, j))$ has a sub-matrix which has w columns and satisfies the cancel-out condition.

2.3. Known Results

In this section, known results on the minimum weights of SFA-LDPC codes are summarized.

For the case $j = 2$ We can easily show that $d(p, 2) = 4$ for any prime number $p \geq 3$.

For the case $j = 3$ Yang et al. has shown that $d(p, 3) = 6$[18]

For the case $j = 4$ Mittelholzer showed that $d(p, 4)$ is 12 or less[11]. Yang showed that $d(p, 4) = 8$ for $p = 5$ and $p = 7$, and that $d(p, 4)$ is 10 or more if $p > 7$[18]. Therefore $d(p, 4)$ with $p > 7$ is either 10 or 12.

For the case $j = 5$ Mittelholzer showed that $d(p, 5)$ is 20 or less[11]. The lower-bound of $d(p, 5)$ with $p > 7$ is 10 or more, which is obtained from Yang's result for $j = 4$, and a simple observation that $d(p, j_1) \geq d(p, j_2)$ if $j_1 \geq j_2$.

For the case $j = 6$ Mittelholzer showed that $d(p, 6)$ is 32 or less[11]. The lower-bound of $d(p, 6)$ with $p > 7$ is 10 or more, as in the case $j = 5$.

2.4. Additional Remarks

This chapter gave the definition of SFA-LDPC codes and some properties of SFA-LDPC codes. The construction of array LDPC codes has an algebraic structure analogous to Reed-Solomon codes. SFA-LDPC code has maximum code-length among all array LDPC codes with same p and j. Thanks to this property, SFA-LDPC codes are invariant under a doubly transitive group of affine permutations[18, Lem2]. On the other hand, general array LDPC codes are not invariant under this group of permutations in general. For example, suppose that $p = 5, j = 3$ and $k = 4$ (Figure 2.1). The parity check matrix of this (general) array LDPC code has 3×4 blocks while that of SFA-LDPC codes has 3×5 blocks. If the 7th column vector of the parity check matrices (the left boxed vector in figure 2.1) is mapped by an affine permutation of the form $\phi(\boldsymbol{v}_7) \mapsto 2\phi(v_7) + \phi(v_{11})$, the image is not included by the parity check matrix of the (general) array LDPC code, but included by the parity check matrix of the SFA-LDPC code. Because

$$\phi(\boldsymbol{v}_7) = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \phi(\boldsymbol{v}_{11}) = \begin{pmatrix} 0 \\ 2 \\ 4 \end{pmatrix};$$

$$2\phi(\boldsymbol{v}_7) + \phi(v_{11}) = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} + \begin{pmatrix} 0 \\ 2 \\ 4 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} = \phi(\boldsymbol{v}_{23})$$

in the modulus of 5 (This image $(2, 1, 0)^T$ is the right boxed vector in figure 2.1). The parity check matrix of the SFA-LDPC codes has no lack columns and thus closed by double transitive group of affine column permutation. This property gives significant constraint on the positions of nonzero components in a codeword that if $C_A(p, j)$ contains a codeword with weight w, then there exists a codeword

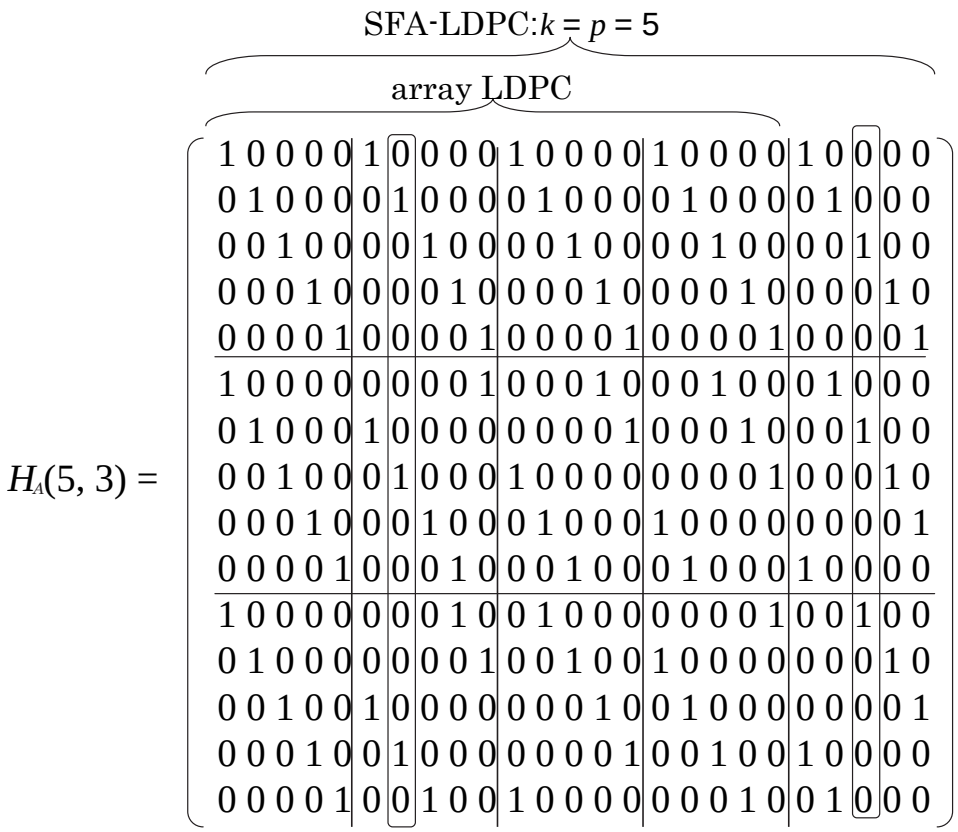

Figure 2.1. $C_A(p, j)$ is closed by doubly transitive group of affine permutations

whose weight is w and its first symbol is nonzero (i.e. one). This is because if there exists a codeword $\boldsymbol{v}$ whose weight is w, then we can always make a mapping Φ such that the first symbol of $\Phi(v)$ is nonzero[18, Lem2]. Thus, one can show that the upper-bound limit of $C_A(p, j)$ is w by the above property that there exists a codeword with weight w and its first symbol is nonzero. This property will be often used in the following chapters.

Chapter 3

Procedure for the Weight Test

In this section, the weight test procedure is presented. First, certain classes of support matrices are considered, and then, the classes are checked if they have a support matrix which fulfills the cancel-out condition or not.

3.1. Constraints on Positions of Zeros

Let us consider conditions and properties of the check and support matrices of SFA-LDPC codes.

Yang has shown that $C_A(p, j)$ is closed by doubly transitive group of affine permutations[18, Lem2]. Using this property, one can show that if $C_A(p, j)$ contains a codeword with weight w, then there exists a codeword whose weight is w and its first symbol is nonzero. Let v be such a codeword, $S_{\boldsymbol{v}} = [s_{r,i}]$ be the support matrix of $\boldsymbol{v}$, and $\boldsymbol{s}_i$ denote the i-th column vector of $S_{\boldsymbol{v}}$. From the above property and the definition of $H_A(p, j)$, the first column of $S_{\boldsymbol{v}}$ is an all-zero column vector, that is, $s_{1,1} = \cdots = s_{j,1} = 0$. (To make the following discussion simpler, let the indices of $s_{r,i}$ start from 1.)

From Lemma 2.2 and the fact that $\boldsymbol{s}_1 = \boldsymbol{0}$, a column vector $\boldsymbol{s}_i$ with $2 \leq i \leq w$ contains at most one zero (refer this property as P1). On the other hand, Corollary 2.5 assures that $S_{\boldsymbol{v}}$ must satisfy the cancel-out condition and therefore it is necessary that each row of $S_{\boldsymbol{v}}$ must contain even number of zeros. Because the first column $\boldsymbol{s}_1$ has zeros on every row, each row must contain at least one zero between the second to the w-th column positions to "cancel" the zero on the

first column (refer this property as P2). Note that satisfying the both properties P1 and P2 is a necessary condition for $\boldsymbol{v}$ to be a correct codeword.

Now, suppose that $s_{1,k} = s_{1,k+1}$ for any odd k with $1 \leq k < w$. Then, $s_{1,2} = 0$ and $s_{r,2} = (r-1)s_{2,2}$ for $2 \leq r \leq j$ from Lemma 2.1. Therefore $S_{\boldsymbol{v}}$ can be written as;

$$\left[\begin{array}{cc|cc|c|cc} 0 & 0 & e_1 & e_1 & \cdots & e_{\frac{w}{2}-1} & e_{\frac{w}{2}-1} \\ \hline 0 & s_{2,2} & s_{2,3} & s_{2,4} & \cdots & s_{2,w-1} & s_{2,w} \\ 0 & 2s_{2,2} & s_{3,3} & s_{3,4} & \cdots & s_{3,w-1} & s_{3,w} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & (j-1)s_{2,2} & s_{j,3} & s_{j,4} & \cdots & s_{j,w-1} & s_{j,w} \end{array}\right], \tag{3.1}$$

where e_i with $1 \leq i \leq \dfrac{w}{2} - 1$ satisfies $0 \leq e_i \leq p-1$. In the following, let us consider *collocations* of zeros (distribution of positions of zeros) in support matrices which satisfy the properties P1 and P2. Recall that $S_{\boldsymbol{v}}$ is a set of column-vectors. If we restrict several positions in a support matrix to have zero, then we can define a class of support matrices which satisfy the given restriction on the positions of zeros.

3.2. Collocation of Zeros in a Support Matrix

To deal with a class of support matrices, the notion of a *zero locater* is introduced. A *zero locater for a weight w vector* is a w-tuple $\boldsymbol{z} = (z_1, \ldots, z_w)$ satisfying the following conditions.

- $z_i \in \{1, \ldots, j\} \cup \{\boldsymbol{o}, \square\}$ for $1 \leq i \leq w$ where $\boldsymbol{o}$ and $\square$ are special symbols, and
- no integer appears more than once in $\boldsymbol{z}$.

Intuitively, a zero locater is to give constraints on the positions of zeros in a support matrix. If z_i with $1 \leq i \leq w$ is an integer, say k, then it says that the i-th column vector of a support matrix must have zero at the k-th row (No constraints are given for other row positions of the vector). If $z_i = \boldsymbol{o}$, then the i-th column vector of a support matrix must be a zero vector, and if $z_i = \square$, then there is no constraint (on the positions of zeros) concerning the i-th column

vector. In this chapter, support matrices are considered to be in the form (3.1), thus the first two components are always assumed that $z_1 = \boldsymbol{o}$ and $z_2 = 1$.

Let us say that a support matrix $S_{\boldsymbol{v}} = [s_{r,i}]$ of a vector $\boldsymbol{v}$ (with weight w) *fulfills* the zero locater z if $S_{\boldsymbol{v}}$ is of the form (3.1) and the (z_i, i) component of $S_{\boldsymbol{v}}$ is zero for all z_i with $z_i \neq \boldsymbol{o}, \square$. Let us also say that a vector v *fulfills* the zero locater z if at least one of the support matrices of $\boldsymbol{v}$ fulfills z (remind that the order of column vectors in a support matrix can be changed). Let $V(z)$ denote the set of vectors (of weight w) which fulfill the zero locater z. An example of these notions will be given in Section 3.4.

The following lemmas can be shown easily because we can change the order of column vectors in a support matrix.

Lemma 3.1 *If*

$$\begin{aligned} z &= (\ldots, z_{2k+1}, z_{2k+2}, \ldots), \\ z' &= (\ldots, z_{2k+2}, z_{2k+1}, \ldots), \end{aligned}$$

(the $(2k+1)$-th component and the $(2k+2)$-th component are exchanged), then $V(z) = V(z')$.

Lemma 3.2 *If*

$$\begin{aligned} z &= (\ldots, z_{2k+1}, z_{2k+2}, \ldots, z_{2k'+1}, z_{2k'+2}, \ldots), \\ z' &= (\ldots, z_{2k'+1}, z_{2k'+2}, \ldots, z_{2k+1}, z_{2k+2}, \ldots), \end{aligned}$$

(two couples of column vectors are exchanged), then $V(z) = V(z')$.

The following lemma associates zero locaters and the properties P1 and P2.

Lemma 3.3 *If the support matrix of a vector $\boldsymbol{v}$ satisfies the properties P1 and P2, then there is a zero locater which $\boldsymbol{v}$ fulfills. That is, for a certain zero locater $z = (z_1, \ldots, z_w)$, $\boldsymbol{v} \in V(z)$.*

Remind that the properties P1 and P2 are necessary conditions for a vector to be a codeword. Thus we have the following corollary.

Corollary 3.4 *For a codeword $\boldsymbol{v}$, there is a zero locater $\boldsymbol{z} = (\boldsymbol{o}, 1, z_3, z_4, \ldots, z_w)$ satisfying $\boldsymbol{v} \in V(z)$.*

3.3. Procedure for the Weight Test

Summarizing the above discussion, we can consider the following procedure to determine if there is a codeword of weight w.

1. Enumerate all of zero locaters of the form $\boldsymbol{z} = (\boldsymbol{o}, 1, z_3, z_4, \ldots, z_w)$.
2. Find support matrices (choices of column vectors of $\phi(H_A(p, j))$) which fulfill the zero locater.
3. Verify if the support matrix satisfies the cancel-out condition.

If the above procedure can find a support matrix which satisfies the cancel-out condition, then there exists a codeword of weight w. If it cannot find such a matrix, then the code does not contain codewords of weight w.

3.3.1 Enumerate all of zero locaters

To realize the procedure shown above, all of zero locaters must be enumerated. This can be done by the following recursive procedure. To simplify the description, we consider that obtained zero locaters are included in the set T, which is used as a "global variable" in the procedure. At the beginning, T is initialized as an empty set. Zero locaters which are newly found during the execution of the procedure are added to T. When the procedure complete the computation, T contains all zero locaters.

procedure *zero-loc*(z, R, i): z is a zero locater, $R \subseteq \{1, \ldots, j\}$ is a set of row positions which are not used in z, and i is a column position.

1. Let r be the smallest integer in R, and let $R \leftarrow R \setminus \{r\}$.
2. Let z' be the zero locater which is obtained from z by replacing the i-th component in z with r.
3. If $R = \emptyset$, then include z' in T and return (to the upper recursion level)
4. If $i + 2 > w$, then return with T unchanged.
5. Execute **zero-loc**(z', R, $i + 2$).

6. For each $r' \in R$ do the following (a) and (b)

   (a) Define $z'_{r'}$ as the zero locater which is obtained from z' by replacing the $(i+1)$-th component in z' with r'.

   (b) If $R \setminus \{r'\} = \emptyset$, then include $z'_{r'}$ in T and return. Otherwise execute **zero-loc**$(z'_{r'}, R \setminus \{r'\}, i+2)$.

7. return

end procedure

Let $z_{\Box} = (\boldsymbol{o}, 1, \Box, \ldots, \Box)$. By executing **zero-loc**$(z_{\Box}, \{2, \ldots, j\}, 3)$, all of zero locaters of the form $z = (\boldsymbol{o}, 1, 2, z_4, \ldots, z_w)$ are included in T.

3.3.2 Constructing Support Matrices

For a obtained zero locater $z = (\boldsymbol{o}, 1, z_3, \ldots, z_w)$, support matrices which fulfill z can be computed by the following procedure. The collocation of zeros in a class of support matrices is uniquely determined by fixing a zero locater. Let $S_{\mathcal{V}} = [s_{r,k}]$ be the support matrix which is going to be constructed by the procedure, where $1 \leq r \leq j$ and $1 \leq k \leq w$. Recall that $\boldsymbol{s}_1 = \boldsymbol{0}$ and the first row was composed so that the row fulfills the cancel-out condition.

From the construction of the parity check matrix of SFA-LDPC code, there are p column vectors in $\phi(H_A(p,j))$ such that its r-th element is zero. Let $A_0(r)$ be the set of column of $\phi(H_A(p,j))$ such that

$$A_0(r) = \{k | 1 \leq k \leq p^2, h_{r,k} = 0\}$$

and $A(r)$ be the set such that $A(r) = A_0(r) \setminus \{1\}$. For instance, if $p = 3, j = 2$, then the set the set $A_0(2)$ is $\{1, 6, 8\}$ and the set $A(2)$ is $\{6, 8\}$ since the matrix $\phi(H_A(3,2))$ is given by

$$\begin{bmatrix} 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 1 & 2 & 1 & 2 & 0 & 2 & 0 & 1 \end{bmatrix}.$$

procedure *weight test*

for $r = 2, 3, \ldots, j$

1. compute an integer $i \in [3, w]$ such that $z_i = r$. From the construction of the zero locater, there exists such an integer uniquely.

2. The set of candidate column vectors for $\boldsymbol{s}_i$, the i-th column of $S_{\boldsymbol{v}}$, is $A(r)$. If the first row of $\boldsymbol{s}_i$ is already restricted, then the candidate of $\boldsymbol{s}_i$ is determined uniquely.

   Enumerate all candidates in the back-track manner and perform the following operations for each assignment.

3. For the rest of column positions of $S_{\boldsymbol{v}}$, perform an exhaustive search for column vectors.
   During the exhaustive search, it is possible to use some criteria to abandon hopeless computation. For example, assume that $w = 6$ and the procedure has determined four column vectors $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_4$ of $S_{\boldsymbol{v}}$. If, for example, $s_{2,1} = 0$, $s_{2,2} = 1$, $s_{2,3} = 2$ and $s_{2,4} = 3$, then the procedure can abandon the current search trial because it is impossible to choose two column vectors and make the second row of $S_{\boldsymbol{v}}$ satisfy the cancel-out condition since at least four more vectors are needed to make the second row satisfy the cancel-out condition. By making use of this kind of criteria, we can skip hopeless computation and make the procedure much more efficient than straightforward search algorithms.

4. For each of constructed support matrix, check if the cancel-out condition holds or not. If the condition holds, then the code $C_A(p, j)$ contains a codeword of weight w.

end for

end procedure

3.4. Example

Consider $C_A(5, 3)$ code whose parity check matrix (ϕ-represented) is given in Figure. 3.1. First, consider if $C_A(5, 3)$ contains a codeword of weight four. If

$$\phi(H_A(5,3)) = \left[\begin{array}{cccccccccccccccccc} 0&1&2&3&4&0&1&2&3&4&0&1&2&3&4&0&1&2 \\ 0&1&2&3&4&1&2&3&4&0&2&3&4&0&1&3&4&0 \\ 0&1&2&3&4&2&3&4&0&1&4&0&1&2&3&1&2&3 \\ 3&4&0&1&2&3&4 & & & & & & & & & & & \\ 1&2&4&0&1&2&3 & & & & & & & & & & & \\ 4&0&3&4&0&1&2 & & & & & & & & & & & \end{array}\right]$$

Figure 3.1. ϕ representation of the parity check matrix $H_A(5,3)$

there exists such a codeword, then the support matrix of the minimum weight codeword must be written as

$$S = \begin{bmatrix} 0 & 0 & e_1 & e_1 \\ 0 & s_{2,2} & 0 & s_{2,4} \\ 0 & s_{3,2} & s_{3,3} & 0 \end{bmatrix}. \tag{3.2}$$

Vectors whose support matrices are in this form belong to $V(z)$ where z is a zero locater defined as $z = (\boldsymbol{o}, 1, 2, 3)$. If e_1 in (3.2) is chosen to be 1, then the third and the fourth columns of S are determined uniquely and S must be written as

$$S = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & s_{2,2} & 0 & 3 \\ 0 & s_{3,2} & 4 & 0 \end{bmatrix}.$$

To make this matrix satisfy the cancel-out condition, the second column must be $(0,3,4)^T$, but such a column is not in $H_A(5,3)$. Thus the choice of $e_1 = 1$ is faulty. We can see easily that even if we choose e_1 different from 1, it is not possible to make S satisfy the cancel-out condition. Therefore, $C_A(5,3)$ does not contain codewords of weight four.

Next, consider if $C_A(5,3)$ contains a codeword of weight six. In this case, we need to consider two zero locaters $z_1 = (o, 1, 2, 3, \square, \square)$ and $z_2 = (o, 1, 2, \square, 3, \square)$ which intuitively correspond to

$$S_1 = \begin{bmatrix} 0 & 0 & e_1 & e_1 & e_2 & e_2 \\ 0 & s_{2,2} & 0 & s_{2,4} & s_{2,5} & s_{2,6} \\ 0 & s_{3,2} & s_{3,3} & 0 & s_{3,5} & s_{3,6} \end{bmatrix}, \tag{3.3}$$

and

$$S_2 = \begin{bmatrix} 0 & 0 & e_1 & e_1 & e_2 & e_2 \\ 0 & s_{2,2} & 0 & s_{2,4} & s_{2,5} & s_{2,6} \\ 0 & s_{3,2} & s_{3,3} & s_{3,4} & 0 & s_{3,6} \end{bmatrix}, \tag{3.4}$$

respectively.

Choosing $e_1 = 1$ in S_1 results in

$$S_1 = \begin{bmatrix} 0 & 0 & 1 & 1 & e_2 & e_2 \\ 0 & s_{2,2} & 0 & 3 & s_{2,5} & s_{2,6} \\ 0 & s_{3,2} & 4 & 0 & s_{3,5} & s_{3,6} \end{bmatrix},$$

we have four choices for the second column vector because $H_A(5,3)$ have four unused column vectors whose first component is zero. If we choose $e_2 = 0$ or $e_2 = 1$, then four out of six components in the second row of S_1 must have different values, and S_1 cannot satisfy the cancel-out condition. Thus e_1 must be either of 2 or 3 or 4. For each choice of e_i, we have ${}_5C_2$ choices for the third and the fourth columns in S_1. Thus, there are $4 \times 3 \times {}_5C_2 = 120$ patterns for the choice of column vectors for S_1 above. Computer search showed that none of the 120 patterns satisfy the cancel-out condition, and therefore the choice of $e_1 = 1$ is faulty. In a similar way, we can verify that there is no support matrix which is in the pattern of (3.3) and satisfy the cancel-out condition.

For support matrices in the pattern of (3.4), choose $e_1 = 1$ and $e_2 = 3$ for example. This uniquely determines the third and the fifth columns of S_2, and allows 4 choices each for the fourth and the sixth columns. Now consider to choose the two columns so that

$$S_2 = \begin{bmatrix} 0 & 0 & 1 & 1 & 3 & 3 \\ 0 & s_{2,2} & 0 & 4 & 4 & 1 \\ 0 & s_{3,2} & 4 & 2 & 0 & 4 \end{bmatrix}.$$

If we can choose so that $s_{2,2} = 1$ and $s_{3,2} = 2$, then S_2 satisfies the cancel-out condition. Indeed, the sixth column in $\phi(H_A(5,3))$ is exactly $(0,1,2)^T$. Thus we found a support matrix which satisfies the cancel-out condition. The codeword which corresponds to the support matrix

$$S_2 = \begin{bmatrix} 0 & 0 & 1 & 1 & 3 & 3 \\ 0 & 1 & 0 & 4 & 4 & 1 \\ 0 & 2 & 4 & 2 & 0 & 4 \end{bmatrix}$$

Table 3.1. Minimum weights of $C_A(p, j)$

p	$j = 4$	$j = 5$	$j = 6$	$j = 7$
5	8*	–	–	–
7	8*	12	12	–
11	10	10	16	≥ 16
13	10	12	14	N/A
17	10	12	N/A	N/A
19	10	12	N/A	N/A
23	10	N/A	N/A	N/A
⋮	⋮	⋮	⋮	⋮
79	10	N/A	N/A	N/A

"N/A" denotes that the results are not available due to large computational time.

is 10000 10010 00000 01010 01000.

3.5. Minimum Weights Found by the Procedure

Table 3.1 shows the minimum weights of $C_A(p, j)$ computed by the proposed procedure. The values marked with * indicate the previously known exact minimum weight. Yang has shown that the minimum weight of $C_A(p, 4)$ is 10 or more if $p > 7$[18], and the proposed procedure could verify that the minimum weight of $C_A(p, 4)$ is exactly 10 for all prime numbers between 11 and 79. From this result, we can naturally conjecture that $C_A(p, 4)$ with $p > 7$ contains a codeword with weight 10, independently of the parameter p. Similarly, we can conjecture that $C_A(p, 5)$ with $p > 11$ contains a codeword with weight 12. These conjectures are the subjects of Chapter 4.

3.6. Amount of Calculation

In this section, the complexity of the proposed procedure is considered. For simplicity's sake, let us fix a zero locater z who has no pairs of elements z_{2i+1} and

z_{2i+2} without $\square$, where $0 < i < \frac{w}{2}$. For example, if $j = 3$, there are two zero locaters $\boldsymbol{z}_1 = (\boldsymbol{o}, 1, 2, 3, \square, \square)$ and $\boldsymbol{z}_2 = (\boldsymbol{o}, 1, 2, \square, 3, \square)$. Since $\boldsymbol{z}_1 = (o, 1, 2, 3, \square, \square)$ has a pair of elements $z_3 = 2$ and $z_4 = 3$ which have no $\square$, we choose $\boldsymbol{z}_2$ in the following discussion.

First, the column vectors of the matrix $S_{\boldsymbol{v}}$ are reordered so that $s_{1,2} = 0, s_{2,3} = 0, \ldots, s_{j,j+1} = 0$. Note that the first column of $S_{\boldsymbol{v}}$ is determined uniquely. Consider, for example, how many vectors in $\phi(H_A(p, j))$ can come to the second column position of $S_{\boldsymbol{v}}$. The second column of $S_{\boldsymbol{v}}$ must have 0 at the first row. From the definition of $H_A(p, j)$, we can easily understand that $\phi(H_A(p, j))$ contains p column vectors which have 0 at the first row. Among the p column vectors, the all-zero vector has been used as the first column of $S_{\boldsymbol{v}}$, and thus we have $p-1$ vectors which can be used as the second column of $S_{\boldsymbol{v}}$. With similar discussions, one can show that there are $(p-1)^j$ possible patterns for the leftmost $j+1$ column vectors of $S_{\boldsymbol{v}}$. Note that $(p-1)^j$ is much smaller than $\binom{p^2}{j}$ which is the number of patterns obtained by naive choices of j column vectors. Each elements in the first row of $s_3, \ldots, s_{j+1}$ must be canceled out. Thus we need to take $j-1$ column vectors from $\phi(H_A(p, j))$ so that the first elements of the taken $j-1$ column vectors cancel out the first row of $\boldsymbol{s}_3, \ldots, \boldsymbol{s}_{j+1}$ (note that the first elements of $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ cancel each other). The number of cases to take these $j-1$ column vectors is $(p-1)^{j-1}$. Thus the number of cases taking the leftmost $2j$ columns is $(p-1)^{2j-1}$. Next, we need to fill in the rest of $w-2j$ columns. The proposed procedure execute exhaustive search for these columns. Since the support matrix is of the form (3.1), the first elements of these columns are pairwise equal. Thus the number of cases of the exhaustive search is given by $\binom{p^2-2j}{\frac{w-2j}{2}}(p-1)^{(w-2j)/2}$. (Note that $w-2j$ is an even value because SFA-LDPC codes are even-weighted codes, and that $\binom{p^2-2j}{\frac{w-2j}{2}}(p-1)^{(w-2j)/2}$ is much smaller than $\binom{p^2-2j}{w-2j}$ which is the number of patterns obtained by naive choices).

Summarizing above discussion, the size of the search space of the proposed approach is given by $(p-1)^{j+w/2-1}\binom{p^2-2j}{\frac{w-2j}{2}}$. This is the worst amount of calculation since the proposed procedure abandon hopeless computation. On the other hand, the size of the search space of checking all combination of weight

Table 3.2. Comparison of the calculation amount of searching a codeword with weight w

Parameters			Size of the search space (Number of cases)		Advantage
p	j	w	Proposed approach (A)	Brute force approach (B)	Rate (A/B)
13	4	10	6.92270×10^{10}	3.99240×10^{15}	1.73397×10^{-5}
79	4	10	8.53992×10^{18}	2.45271×10^{31}	3.48183×10^{-13}
13	5	12	9.84486×10^{12}	7.59826×10^{17}	1.29567×10^{-5}
13	6	14	1.39983×10^{15}	1.02251×10^{20}	1.36901×10^{-5}

w out of all vector length p^2 is $\binom{p^2}{w}$. Table 3.2 shows the comparison of the calculation amount between the proposed approach and brute force approach for searching a codeword with weight w in $C_A(p, j)$. The size of the search space of the proposed approach is much less than that of brute force approach.

Let us consider the case such that some of the first elements of $\boldsymbol{s}_3, \ldots, \boldsymbol{s}_{j+1}$ cancel out each other. In other words, the chosen zero locater has pairs of elements z_{2i+1} and z_{2i+2} without $\square$, where $0 < i < \frac{w}{2}$. Since the support matrix is of the form (3.1), the $2i+1$-th and the $2i+2$-th column vector of $S_{\boldsymbol{v}}$ have the same first element e_i. An example of this case is the zero locater $\boldsymbol{z}_1$ in the above instance. $\boldsymbol{z}_1 = (\boldsymbol{o}, 1, 2, 3, \square, \square)$ has a pair of elements $z_3 = 2$ and $z_4 = 3$, which have no $\square$. In this case, the number of cases to take the leftmost $j+1$ column vectors is $(p-1)^{j-k}$, where k is the number of pairs of column vectors with same first element. For example, if we have chosen the $2i+1$-th column vector, then the $2i+2$-th column vector has a zero at the z_{2i+2}-th row and the value of its first row is e_i, which is determined uniquely. Each elements in the first row of $\boldsymbol{s}_3, \ldots, \boldsymbol{s}_{j+1}$ must be canceled out. Since there exists k pairs of columns such that the first elements of them are equal, we need to take $j-1-2k$ column vectors from $\phi(H_A(p, j))$ so that the first row of the leftmost $2j-2k$ columns satisfies the cancel-out condition. Thus the number of cases taking the leftmost $2j-2k$ columns is $(p-1)^{2j-1-3k}$. Therefore the size of the search space of the proposed approach is given by $(p-1)^{j+w/2-1-2k}\binom{p^2-2j+2k}{\frac{w-2j+2k}{2}}$, which is the worst amount of calculation.

If $S_{\boldsymbol{v}}$ has multiple zero locaters, then the size of the search space is the sum

of each search spaces. For instance, if $j = 3$, the number of the zero locater is 2. Thus if the testing weight is 6, then the size of the search space is obtained by the following calculation;

$$(p-1)^5 \binom{p^2-6}{0} + (p-1)^3 \binom{p^2-4}{1}$$

which gives us 2.17×10^5 as the size of the search space in the case of $p = 11$.

The number of zero locaters increases as j increases. For example, if $j = 4$, the number of zero locaters is 4, if $j = 5$, the number of those is 13, and if $j = 6$, the number of those is 41 (The instances of these are shown in Appendix A). Hence the amount of calculation is increased as j increases. Besides, the advantage of the SFA-LDPC code is its high rate property, thus we can assume that SFA-LDPC codes are rarely employed with rather large j.

3.7. Concluding Remarks

In this chapter, a procedure to test if a SFA-LDPC code contains codewords of specified weight was proposed. The problem might be solvable by a naive procedure which tests all the combinations of column vectors in a parity check matrix, but such an approach will suffer from huge computational complexity caused by the number of combinations of column vectors. On the other hand, it is also possible to take a purely algebraic approach as in [18]. The problem of this approach is that the theoretical analysis becomes too complicated if the code parameter is not small.

Actually, when the case of $p = 79$ and $j = 4$, the code length of $C_A(p, j)$ is 6241; and the number of cases that must be test is $\binom{6241}{10}$ if one takes naive procedure (tests all the combinations of column vectors). But, $\binom{6241}{10}$ is a number of order 10^{31}, hence, it is obviously impractical to test all the combination by a computer. Though, as shown in table 3.1, the procedure given in this chapter obtained a codeword with weight 10 of the code $C_A(79, 4)$.

The procedure considered in this dissertation can be regarded as a hybrid approach of the algebraic and the computer-oriented approaches. Some algebraic analysis in [18] is replaced by a computer search in the proposed study, which allows us to use the basic idea of [18] for codes with bigger parameters. For

example, by using the proposed technique together with the help by computer search, we obtained some exact minimum weights of $C_A(p, j)$, which were not shown in previous works.

Chapter 4

New Upper-bound Limits on the Minimum Weights of SFA-LDPC Codes

The results given in the previous chapter suggest a natural conjecture; the minimum weights of $C_A(p,4)$ and $C_A(p,5)$ are upper-bounded by 10 and 12, respectively. In this chapter, it is shown that this conjecture really holds. Together with the previously known results, now it is clarified that the minimum weight of $C_A(p,4)$ is exactly 10 for $p > 7$, and that the minimum weight of $C_A(p,5)$ is either 10 or 12.

4.1. Support Matrices in Normal Form

To discuss the minimum weights of SFA-LDPC codes, the author investigates supports that are in a special form. Assume that an SFA-LDPC code $C_A(p,j)$ has a codeword $\boldsymbol{v}$ which has a nonzero (i.e. one) component at the first symbol position. Since the first column vector $\phi(h_1)$ of $\phi(H_A(p,j))$ is an all-zero vector of length j, supp(v) must contain an all-zero vector of length j since $\boldsymbol{v} = (1,\ldots)$. On the other hand, Corollary 2.5 implies that j zeros in the all-zero vector $\phi(h_1)$ must be canceled-out by other column vectors in the support. Since a non-zero vector cancels at most one zero in the all-zero vector (due to Lemma 2.2), supp$(\boldsymbol{v})$ must contain another j column vectors each of which cancels one of zeros in the

$$\begin{bmatrix}
0 & 0 & -s_{3,3} & -2s_{4,4} & \cdots & -(j-1)s_{j-1,j+1} & s_{1,j+2} & \cdots & s_{1,w} \\
0 & s_{2,2} & 0 & -s_{4,4} & \cdots & -(j-2)s_{j-1,j+1} & s_{2,j+2} & \cdots & s_{2,w} \\
0 & 2s_{2,2} & s_{3,3} & 0 & \cdots & -(j-3)s_{j-1,j+1} & s_{3,j+2} & \cdots & s_{3,w} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
0 & (j-1)s_{2,2} & (j-2)s_{3,3} & (j-3)s_{4,4} & \cdots & 0 & s_{j,j+2} & \cdots & s_{j,w}
\end{bmatrix}$$

Figure 4.1. A normal form support matrix of a codeword with weight w

all-zero vector. Let $S_{\boldsymbol{v}} = [s_{r,i}]$ be $\mathrm{supp}(\boldsymbol{v})$ and $s_{r,i}$ denote the r-th component of the i-th column vector of $S_{\boldsymbol{v}}$. Since the order of the column vectors is not essential for the discussion of the cancel-out condition, we can reorder the column vectors of the matrix so that $s_{1,2} = 0, s_{2,3} = 0, \ldots, s_{j,j+1} = 0$ (see the entries of zeros in Figure 4.1). By using the property that the components in a column vector of $\phi(H_A(p,j))$ constitute arithmetic sequences (Lemma 2.1), the second to the $(j+1)$-th column vectors of $S_{\boldsymbol{v}}$ are fully specified as

$$\begin{aligned}
s_{r,2} &= (r-1)s_{2,2}, \\
s_{r,3} &= (r-2)s_{3,3}, \\
&\vdots \\
s_{r,j} &= (r-j+1)s_{j,j}, \\
s_{r,j+1} &= (r-j)s_{j-1,j+1},
\end{aligned}$$

where $1 \leq r \leq j$ and the equations are under the modulus of p. In the last equation, a "virtual"[1] component $s_{j-1,j+1}$ was introduced to make the condition clearer.

Figure 4.1 depicts the support matrix $S_{\boldsymbol{v}}$ which satisfies these conditions. Let us call this form of support matrix a *normal form* support matrix.

[1]The $(j-1)$-th component of the $(j+1)$-th column of normal form support matrix $S_{\boldsymbol{v}}$ is not $s_{j-1,j+1}$ itself but $-s_{j-1,j+1}$. In this sense, this component is expressed as "virtual." If we introduced a virtual component $s_{j+1,j+1}$, then $s_{j-1,j+1} = -s_{j+1,j+1}$.

4.2. The Upper Bound of $d(p, 4)$

In this section, the results in the previous chapter are extended to general SFA-LDPC codes $C_A(p, 4)$, and it is shown that $d(p, 4)$ is 10 or less for any $p > 7$.

4.2.1 Overview of the discussion

The discussion in this section is developed according to the following scenario: By investigating the experimental results analytically, we can discover a universal structure that can be found independent of the prime number p. The structure is further studied, and we can derive conditions for $H_A(p, 4)$ to have a support matrix that has 10 columns and satisfies the cancel-out condition. The discussion completes by showing that the derived conditions always hold for $H_A(p, 4)$ with an arbitrary $p > 7$.

4.2.2 Support matrices and their common structure

By using a computer program, the author has generated minimum weight codewords of $C_A(p, 4)$ for prime numbers p between 11 and 79. Among minimum weight codewords, the author restricts himself to those that have nonzero symbols at the first positions, compute their support matrices, and reorder the column vectors of the matrices so that the matrices become the normal form. Since the minimum weights of $C_A(p, 4)$ is ten for all primes between 11 and 79, the support matrices in the normal form are thus written as

$$\begin{bmatrix} 0 & 0 & s_{1,3} & s_{1,4} & s_{1,5} & s_{1,6} & s_{1,7} & s_{1,8} & s_{1,9} & s_{1,10} \\ 0 & s_{2,1} & 0 & s_{2,4} & s_{2,5} & s_{2,6} & s_{2,7} & s_{2,8} & s_{2,9} & s_{2,10} \\ 0 & s_{3,1} & s_{3,2} & 0 & s_{3,5} & s_{3,6} & s_{3,7} & s_{3,8} & s_{3,9} & s_{3,10} \\ 0 & s_{4,1} & s_{4,2} & s_{4,3} & 0 & s_{4,6} & s_{4,7} & s_{4,8} & s_{4,9} & s_{4,10} \end{bmatrix}.$$

As discussed in the end of Chapter 2, every code $C_A(p, 4)$ contains several minimum weight codewords with nonzero symbols at the first positions, and hence we have several support matrices in the normal form for each $C_A(p, 4)$. The author classified the support matrices according to the cancel-out patterns of components in the matrices, and found that an identical cancel-out pattern appears in different

choices of the prime number p. For example, $\phi(H_A(11,4))$, $\phi(H_A(13,4))$ and $\phi(H_A(17,4))$ respectively have the following support matrices;

$$\begin{bmatrix} 0 & 0 & 10 & 9 & 5 & 5 & 9 & 10 & 3 & 3 \\ 0 & 3 & 0 & 10 & 7 & 3 & 9 & 7 & 9 & 10 \\ 0 & 6 & 1 & 0 & 9 & 1 & 9 & 4 & 4 & 6 \\ 0 & 9 & 2 & 1 & 0 & 10 & 9 & 1 & 10 & 2 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 0 & 12 & 11 & 6 & 6 & 11 & 12 & 4 & 4 \\ 0 & 10 & 0 & 12 & 4 & 10 & 0 & 4 & 0 & 12 \\ 0 & 7 & 1 & 0 & 2 & 1 & 2 & 9 & 9 & 7 \\ 0 & 4 & 2 & 1 & 0 & 5 & 4 & 1 & 5 & 2 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 0 & 16 & 15 & 8 & 8 & 15 & 16 & 6 & 6 \\ 0 & 13 & 0 & 16 & 11 & 13 & 6 & 11 & 6 & 16 \\ 0 & 9 & 1 & 0 & 14 & 1 & 14 & 6 & 6 & 9 \\ 0 & 5 & 2 & 1 & 0 & 6 & 5 & 1 & 6 & 2 \end{bmatrix}.$$

The components in the matrices vary in the above three examples, but we can see that the matrices satisfy the cancel-out pattern depicted in

$$\begin{bmatrix} 0 & 0 & s_{1,3} & s_{1,4} & s_{1,5} & s_{1,5} & s_{1,4} & s_{1,3} & s_{1,9} & s_{1,9} \\ 0 & s_{2,1} & 0 & s_{2,4} & s_{2,5} & s_{2,1} & s_{2,7} & s_{2,5} & s_{2,7} & s_{2,4} \\ 0 & s_{3,1} & 1 & 0 & s_{3,5} & 1 & s_{3,5} & s_{3,8} & s_{3,8} & s_{3,1} \\ 0 & s_{4,1} & 2 & 1 & 0 & s_{4,6} & s_{4,1} & 1 & s_{4,6} & 2 \end{bmatrix}. \tag{4.1}$$

Remind the property that the components in a column vector of $\phi(H_A(p,j))$ constitute arithmetic sequences (Lemma 2.1), and we can easily understand that the common differences (parameter k in Lemma 2.1) of the second, third, fourth and fifth columns of the matrix (4.1) are $s_{2,1}$, 1, 1 and $-s_{3,5} \pmod p$, respectively. By using this property, the matrix (4.1) can be expressed as

$$\begin{bmatrix} 0 & 0 & -1 & -2 & 3s_{3,5} & 3s_{3,5} & -2 & -1 & s_{1,9} & s_{1,9} \\ 0 & s_{2,1} & 0 & -1 & 2s_{3,5} & s_{2,1} & s_{2,7} & 2s_{3,5} & s_{2,7} & -1 \\ 0 & 2s_{2,1} & 1 & 0 & s_{3,5} & 1 & s_{3,5} & s_{3,8} & s_{3,8} & 2s_{2,1} \\ 0 & 3s_{2,1} & 2 & 1 & 0 & s_{4,6} & 3s_{2,1} & 1 & s_{4,6} & 2 \end{bmatrix}, \tag{4.2}$$

where the components are under the modulus of p.

4.2.3 Conditions on the matrix components

Let d_k with $k = 6, \ldots, 10$ be common differences of the sequences in the k-th column vector of the matrix (4.2). From the relation among components in the rightmost (the tenth) column of the matrix, we have $2 \equiv -1 + 2d_{10} \bmod p$. This implies that d_{10} must satisfy

$$2d_{10} \equiv 3 \bmod p. \tag{4.3}$$

The values of $s_{1,9}$ and $2s_{2,1}$ are also represented by using d_{10} as $s_{1,9} \equiv -1 - d_{10}$ and $2s_{2,1} \equiv -1 + d_{10}$. Multiply both sides of the equations by two and we have

$$2s_{1,9} \equiv -2 - 2d_{10} \equiv -5 \bmod p, \tag{4.4}$$

$$4s_{2,1} \equiv -2 + 2d_{10} \equiv 1 \bmod p. \tag{4.5}$$

Now let us turn our attention to the sixth column of the matrix. Because $1 - d_6 \equiv s_{2,1}$, we have $4 - 4d_6 \equiv 4s_{2,1} \equiv 1$ and hence

$$4d_6 \equiv 4 - 1 \equiv 3 \bmod p. \tag{4.6}$$

As for the eighth column, $1 \equiv -1 + 3d_8$ and thus

$$3d_8 \equiv 1 + 1 \equiv 2 \bmod p. \tag{4.7}$$

We also have the relation $2s_{3,5} \equiv -1 + d_8$ from the sequence in the eighth column. Multiply both sides of the relation by three,

$$6s_{3,5} \equiv -3 + 3d_8 \equiv -1 \bmod p. \tag{4.8}$$

In the seventh column, we have $s_{3,5} \equiv -2 + 2d_7$. Multiply the relation by six, and

$$12d_7 \equiv 12 + 6s_{3,5} \equiv 11 \bmod p. \tag{4.9}$$

Finally, consider the relation $s_{2,7} \equiv s_{1,9} + d_9$ in the ninth column, $s_{2,7} \equiv -2 + d_7$ in the seventh column and the conditions (4.4) and (4.9), and we can show that

$$12d_9 \equiv 17 \bmod p. \tag{4.10}$$

Other unknown variables $s_{2,7}$, $s_{3,8}$ and $s_{4,6}$ are uniquely determined once the above variables are determined, and hence relations (4.3) through (4.10) completely state the condition with which the matrix (4.2) satisfies the cancel-out condition.

Note that, if p is a prime number greater than seven, then each of the above relation has a unique solution, since every coefficient of the variable in the relation is relatively prime to p. Consequently, we can determine a unique matrix that is shown in the form of matrix (4.2), and satisfies the cancel-out condition. If we may use multiplicative inverses, then the matrix (4.2) is written as

$$\begin{bmatrix} 0 & 0 & -1 & -2 & -2^{-1} & -2^{-1} & -2 & -1 & -5\cdot 2^{-1} & -5\cdot 2^{-1} \\ 0 & 4^{-1} & 0 & -1 & -3^{-1} & 4^{-1} & -13\cdot 12^{-1} & -3^{-1} & -13\cdot 12^{-1} & -1 \\ 0 & 2^{-1} & 1 & 0 & -6^{-1} & 1 & -6^{-1} & 3^{-1} & 3^{-1} & 2^{-1} \\ 0 & 3\cdot 4^{-1} & 2 & 1 & 0 & 7\cdot 4^{-1} & 3\cdot 4^{-1} & 1 & 7\cdot 4^{-1} & 2 \end{bmatrix}$$

under the above conditions. For the sake of readability, some components in the matrix are written in a reduced form; for example $1+3\cdot 4^{-1}$ is written as $7\cdot 4^{-1}$ instead of $1+3\cdot 4^{-1}$ itself. Finally, remind the property that the parity check matrix $H_A(p,j)$ has p^2 different column vectors, and hence an arbitrary column vector which is represented as an arithmetic sequence belongs to $H_A(p,j)$ as a column vector. Consequently, $H_A(p,j)$ with $p>7$ contains the above described unique matrix.

Summarizing the above discussion, we have the following theorem. The proof is almost a rephrase of the above discussion, but given here for the sake of readability.

Theorem 4.1 *$d(p,4)\leq 10$ for any prime p with $p>7$.*

Proof. In this proof, all calculations are taken under the modulus of p. Since p is a prime number greater than three, $ax=b$ has a unique solution if a is a composite number of 2 and 3. This property will be used later.

Consider a support matrix

$$S=[s_{r,i}] = \begin{bmatrix} 0 & 0 & -b & -2c & -3d & -3d & -2c & -b & e & e \\ 0 & a & 0 & -c & -2d & -3d+k_1 & -2c+k_2 & -b+k_3 & e+k_4 & e+k_5 \\ 0 & 2a & b & 0 & -d & -3d+2k_1 & -2c+2k_2 & -b+2k_3 & e+2k_4 & e+2k_5 \\ 0 & 3a & 2b & c & 0 & -3d+3k_1 & -2c+3k_2 & -b+3k_3 & e+3k_4 & e+3k_5 \end{bmatrix},$$

where all elements are under the modulus of p. This matrix is an instance of the matrix given as Figure 4.1, with $j=4$ and right five column vectors are

represented by using Lemma 2.1. The first components of the sixth, seventh and eighth columns are chosen so that they cancel the first components of the fifth, fourth and third columns, respectively.

For this matrix S, choose parameters to satisfy

$$4a = 1, \tag{4.11}$$
$$b = 1, \tag{4.12}$$
$$c = 1, \tag{4.13}$$
$$6d = 1, \tag{4.14}$$
$$2e = -5, \tag{4.15}$$
$$4k_1 = 3, \tag{4.16}$$
$$12k_2 = 11, \tag{4.17}$$
$$3k_3 = 2, \tag{4.18}$$
$$12k_4 = 17, \tag{4.19}$$
$$2k_5 = 3. \tag{4.20}$$

Remark that the coefficients of the parameters are composites of 2 and 3, and all the parameters are determined uniquely. We now see that S satisfies the cancel-out condition under this choice of parameters. It is obvious that the first row of S fulfills the cancel-out condition and thus we consider the second row:

- $s_{2,1} = 0$ is obviously canceled by $s_{2,3} = 0$.
- $s_{2,2} = a$ is canceled by $s_{2,6} = -3d + k_1$ because $4s_{2,2} = 4a = 1 = -12d + 4k_1 = 4s_{2,6}$ (we used equations (4.11), (4.14) and (4.16)).
- $s_{2,4} = -c$ is canceled by $s_{2,10} = e + k_5$ because $2s_{2,4} = -2c = -2 = 2e + 2k_5 = 2s_{2,10}$ (equations (4.13), (4.15) and (4.20)).
- $s_{2,5} = -2d$ is canceled by $s_{2,8} = -b + k_3$ because $-3s_{2,5} = 6d = 1 = 3b - 3k_3 = -3s_{2,8}$ (equations (4.12), (4.14) and (4.18)).
- $s_{2,7} = -2c + k_2$ is canceled by $s_{2,9} = e + k_4$ because $12s_{2,7} = -24c + 12k_2 = -13 = 12e + 12k_4 = 12s_{2,9}$ (equations (4.13), (4.15), (4.17) and (4.19)).

We have seen that the second row also fulfills the cancel-out condition. For the third row:

- $s_{3,1} = 0$ is obviously canceled by $s_{3,4} = 0$.
- $s_{3,2} = 2a$ is canceled by $s_{3,10} = e + 2k_5$ because $2s_{3,2} = 4a = 1 = 2e + 4k_5 = 2s_{3,10}$ (equations (4.11), (4.15) and (4.20)).
- $s_{3,3} = b$ is canceled by $s_{3,6} = -3d + 2k_1$ because $6s_{3,3} = 6b = 6 = -18d + 12k_1 = 6s_{3,6}$ (equations (4.12), (4.14) and (4.16)).
- $s_{3,5} = -d$ is canceled by $s_{3,7} = -2c + 2k_2$ because $6s_{3,5} = -6d = -1 = -12c + 12k_2 = 6s_{3,7}$ (equations (4.13), (4.14) and (4.17)).
- $s_{3,8} = -b + 2k_3$ is canceled by $s_{3,9} = e + 2k_4$ because $6s_{3,8} = -6b + 12k_3 = 2 = 6e + 12k_4 = 6s_{3,9}$ (equations (4.12), (4.15), (4.18) and (4.19)).

For the fourth row:

- $s_{4,1} = 0$ is obviously canceled by $s_{4,5} = 0$.
- $s_{4,2} = 3a$ is canceled by $s_{4,7} = -2c + 3k_2$ because $4s_{4,2} = 12a = 3 = -8c + 12k_2 = 4s_{4,7}$ (equations (4.11), (4.13) and (4.17)).
- $s_{4,3} = 2b$ is canceled by $s_{4,10} = e + 3k_5$ because $2s_{4,3} = 4b = 4 = 2e + 6k_5 = 2s_{4,10}$ (equations (4.12), (4.15) and (4.20)).
- $s_{4,4} = c$ is canceled by $s_{4,8} = -b+3k_3$ because $s_{4,4} = c = 1 = -b+3k_3 = s_{4,8}$ (equations (4.12), (4.13) and (4.18)).
- $s_{4,6} = -3d+3k_1$ is canceled by $s_{4,9} = e+3k_4$ because $4s_{4,6} = -12d+12k_1 = 7 = 4e + 12k_4 = 4s_{4,9}$ (equations (4.14), (4.15), (4.16) and (4.19)).

We have seen that the cancel-out condition is fulfilled in every row of S. It is also verified that all column vectors of S are valid column vectors of $H_A(p, 4)$. If the ten column vectors in S are all different, then S is a support matrix of a codeword with weight ten. If different columns in S coincide by the above instantiation, then S corresponds to a codeword whose weight is less than ten. In either case, S

corresponds to a codeword whose weight is ten or less. This implies that $C_A(p,4)$ contains a codeword whose weight is less than or equal to 10. ∎

Combine this theorem with Yang's lower-bound limit $d(p,4) \geq 10$ for $p > 7$, and we can fully clarify the minimum weights of SFA-LDPC codes with $j = 4$. The results, together with already known results, are summarized in the following corollary.

Corollary 4.2 *$d(p,4) = 8$ for $p = 5$ and 7, and $d(p,4) = 10$ for any prime $p > 7$.*

4.3. The Upper Bound of $d(p,5)$

Mittelholzer showed that the upper-bound of $d(p,5)$ is 20[11]. The author improves this upper-bound as follows.

Theorem 4.3 *$d(p,5) \leq 12$ for any prime p with $p > 7$.*

Proof. Let us consider a support matrix

$$
\begin{aligned}
S =& [s_{r,i}] \\
=& \begin{bmatrix}
0 & 0 & -b & -2c & -3d & -4e & -4e & -3d & -2c & -b \\
0 & a & 0 & -c & -2d & -3e & -4e+k_1 & -3d+k_2 & -2c+k_3 & -b+k_4 \\
0 & 2a & b & 0 & -d & -2e & -4e+2k_1 & -3d+2k_2 & -2c+2k_3 & -b+2k_4 \\
0 & 3a & 2b & c & 0 & -e & -4e+3k_1 & -3d+3k_2 & -2c+3k_3 & -b+3k_4 \\
0 & 4a & 3b & 2c & d & 0 & -4e+4k_1 & -3d+3k_2 & -2c+3k_3 & -b+3k_4
\end{bmatrix} \\
& \begin{bmatrix}
f & f \\
f+k_5 & f+k_6 \\
f+2k_5 & f+2k_6 \\
f+3k_5 & f+3k_6 \\
f+3k_5 & f+3k_6
\end{bmatrix}.
\end{aligned}
$$

This matrix is an instance of the matrix given as Figure 4.1, with $j = 5$. Choose parameters to satisfy

$$
\begin{gathered}
6a = 1, \\
b = 1, \\
6c = 11, \\
d = 1, \\
6e = 1, \\
3f = -8, \\
6k_1 = 5, \\
3k_2 = 5, \\
3k_3 = 5, \\
2k_4 = 1, \\
6k_5 = 5, \\
3k_6 = 4.
\end{gathered}
$$

It is obvious that the first row of S fulfills the cancel-out condition.
For the second row of the support matrix S:

- $s_{2,1}$ is obviously canceled by $s_{2,3}$.
- $6s_{2,2} = 6a = 1 = -4 \cdot 1 + 5 = -4(6e) + 6k_1 = 6(-4e + k_1) = 6s_{2,7}$.
- $6s_{2,4} = -6c = -11 = 2(-8) + 5 = 2 \cdot 3f + 6k_5 = 6(f + k_5) = 6s_{2,11}$.
- $3s_{2,5} = -6d = -6 = -11 + 5 = -6c + 3k_3 = 3(-2c + k_3) = 3s_{2,9}$.
- $2s_{2,6} = -6e = -1 = -2 + 1 = -2b + 2k_4 = 2(-b + k_4) = 2s_{2,10}$.
- $3s_{2,8} = -9d + 3k_2 = -9 + 5 = -4 = -8 + 4 = 3f + 3k_6 = 3(f + k_6) = 3s_{2,12}$.

For the third row:

- $s_{3,1}$ is obviously canceled by $s_{3,4}$.
- $3s_{3,2} = 6a = 1 = -9 + 2 \cdot 5 = -9d + 2 \cdot 3k_2 = 3(-3d + 2k_2) = 3s_{3,8}$.

- $3s_{3,3} = 3b = 3 = -2 + 5 = -2 \cdot 6e + 6k_1 = 3(-4e + 2k_2) = 3s_{3,7}$.
- $3s_{3,5} = -3d = -3 = -8 + 5 = 3f + 6k_5 = 3(f + 2k_5) = 3s_{3,11}$.
- $3s_{3,6} = -6e = -1 = -11 + 2 \cdot 5 = -6c + 2 \cdot 3k_3 = 3(-2c + 2k_3) = 3s_{3,9}$.
- $3s_{3,10} = -3b + 3 \cdot 2k_4 = -3 + 3 = 0 = -8 + 2 \cdot 4 = 3f + 2 \cdot 3k_6 = 3(f + 2k_6) = 3s_{3,12}$.

For the fourth row:

- $s_{4,1}$ is obviously canceled by $s_{4,5}$.
- $6s_{4,2} = 6 \cdot 3a = 3 \cdot 6a = 3 = -6 + 9 \cdot 1 = -6b + 9 \cdot 2k_4 = 6(-b + 3k_4) = 6s_{4,10}$.
- $s_{4,3} = 2b = 2 = -3 + 5 = -3d + 3k_2 = s_{4,8}$.
- $6s_{4,4} = 6c = 11 = -4 + 15 = -4 \cdot 6e + 3 \cdot 6k_1 = 6(-4e + 3k_1) = 6s_{4,7}$.
- $6s_{4,6} = -6e = -1 = -16 + 15 = 2(-8) + 3 \cdot 5 = 2 \cdot 3f + 3 \cdot 6k_5 = 6(f + 3k_5) = 6s_{4,11}$.
- $3s_{4,9} = 3(-2c + 3k_3) = -6c + 3 \cdot 3k_3 = -11 + 3 \cdot 5 = 4 = -8 + 3 \cdot 4 = 3f + 3 \cdot 3k_6 = 3(f + 3k_6) = 3s_{4,12}$.

For the fifth row:

- $s_{5,1}$ is obviously canceled by $s_{5,6}$.
- $6s_{5,2} = 4 \cdot 6a = 4 = -16 + 20 = 2 \cdot 3f + 4 \cdot 6k_5 = 6(f + 3k_5) = 6s_{5,11}$.
- $3s_{5,3} = 9b = 9 = -11 + 20 = -6c + 4 \cdot 3k_3 = 3(-2c + 4k_3) = 3s_{5,9}$.
- $3s_{5,4} = 6c = 11 = -9 + 20 = -9d + 4 \cdot 3k_2 = 3(-3d + 4k_2) = 3s_{5,8}$.
- $s_{5,5} = d = 1 = -1 + 2 \cdot 1 = -b + 4k_4 = s_{5,10}$.
- $3s_{5,7} = -2 \cdot 6e + 2 \cdot 6k_1 = -2 + 10 = 8 = -8 + 16 = 3f + 4 \cdot 3k_6 = 3(f + 4k_6) = 3s_{5,12}$.

Thus, this matrix S satisfies the cancel-out condition. This implies that $C_A(p,5)$ contains a codeword whose weight is less than or equal to 12. ∎

Together with Yang's lower bound, we obtain the following corollary.

Corollary 4.4 *$10 \leq d(p,5) \leq 12$ for $p > 7$.*

Proof. It follows from Yang's lower-bound that $d(p,5) \geq 10$ for $p > 7$, because $d(p,j_1) \geq d(p,j_2)$ for any $j_1 \geq j_2$. ∎

Chapter 5

Conclusion

The minimum weights of SFA-LDPC codes $C_A(p, j)$ have been studied in this dissertation. The problem has been investigated by Mittelholzer[11] and Yang et al.[18], and this dissertation gives an answer to the problem for the case $j = 4$, namely the minimum weight of $C_A(p, 4)$ is exactly 10 for $p > 7$. For $j = 5$ case, now we know that the minimum weight of $C_A(p, 5)$ is 12 or less, which significantly improves the upper-bound limit of 20 shown by Mittelholzer. To obtain these results, the author took the following approach: First, for certain j and w, the author systematically searched codewords in $C_A(p, j)$ with weight w by a computer program (Chapter 3). The results of the computer search suggested a common mathematical structure shared by the codewords, which were independent of the choice of the parameter p. Finally, the author was successful in showing that $C_A(p, j)$ always contains a codeword with weight w or less when $j = 4, 5$ (Chapter 4). This approach is applicable to the case with $j = 6$ or more, but one may face difficulty in generating sufficient number of support matrices from which one can find a useful structure. The author is not sure if the proposed approach is effective for $C_A(p, j)$ with $j \geq 7$, though, it is remarked that SFA-LDPC codes with large j are less important than those with small j. In a practical viewpoint, SFA-LDPC codes are useful especially at high code rate. Therefore it is expected that people who use the SFA-LDPC codes will choose parameters so that the code has sufficiently high code rate. Increasing the value of j decreases the code rate, and such parameter choice will be unlikely in practical applications.

Now we come back to the performance of SFA-LDPC codes. The basic motivation of this study was to clarify the reason why high rate SFA-LDPC codes show good performance. It was shown by this study that, for the cases $j = 4$ and $j = 5$, the parameter p does not affect the minimum weight of the code. If we fix the parameter j and increase the value of the prime number p, then the code length of $C_A(p, j)$ increases while the minimum weight of the code stays unchanged. This result suggests the minimum distance itself is not the key factor which affects the performance of the code. Instead, the following observation and discussion are obtained from the result.

- The iterative decoding algorithm is so powerful that it often corrects errors even if the distance between the transmitted codeword x and the received vector y, denoted $d(x, y)$, is beyond the correctable distance $c = \lfloor (d_{\min} - 1)/2 \rfloor$, where $d_{\min}$ is the minimum distance of the code. Note that, if $d(x, y) \leq c$, then y is "guaranteed" to be decoded correctly. If $d(x, y) > c$, then there is no such guarantee, but a decoder tries to estimate x as precisely as possible. A powerful decoder recovers several errors which are beyond the correctable distance, regardless of the minimum distance. In this case the performance is strongly affected by the total structure of the code. The minimum distance is a parameter which symbolizes a local structure around a codeword, and gives little affect on the performance of powerful decoding algorithm.

- The minimum distance does not change even if we increase p, but the fraction of the number of minimum weight codewords in the total number of codewords is expected to become smaller as we increase p. This conjecture well matches with the expected weight distributions of random code ensemble. If we choose a parity check matrix by random sampling and the obtained code has length N and rate R, then the number of codewords with weight w is expected to be

$$A(w) = 2^{N(H_2(w/N)-(1-R))},$$

  where $H_2(x)$ is a binary entropy function $H_2(x) = -x \log \frac{1}{x} - (1-x) \log \frac{1}{1-x}$[10]. The total number of codewords is 2^{NR} and

The fraction of $A(w)$ in 2^{NR}, the total number of codewords, is thus

$$2^{N(H_2(w/N)-1)}.$$

If we let w a constant value and make N large, then the fraction goes to $2^{-N} \to 0$.

At writing this dissertation, the second discussion above is just a conjecture. More detailed analysis of the weight distribution of SFA-LDPC codes is expected as the next step of this study. We can evaluate the codes more precisely by using various bounds such as union bounds if we know the weight distributions. The union bound gives the upper bound of the performance of maximum likelihood decoding. This dissertation gives some upper bounds of the minimum weight of SFA-LDPC codes, and this is regarded as the first step of research of the weight distribution of the codes.

References

[1] M. Blaum and R. M. Roth, "New Array Codes for Multiple Phased Burst Correction", IEEE Transactions on Information Theory, Vol. 39, No. 1, pp. 66–77, January 1993.

[2] J. L. Fan, "Array Codes as Low-density Parity-check Codes", Proceedings of second International Symposium on Turbo Codes and Related Topics, pp. 543–546, Brest, France, September 2000.

[3] P. G. Farrell, "A Survey of Array Error Control Codes", European Transactions on Telecommunications, Vol. 3, No. 5, pp. 441–454, 1992.

[4] M. P. C. Fossorier, "Quasi-Cyclic Low Density Parity Check Codes from Circulant Permutation Matrices", IEEE Transactions on Information Theory, Vol. 50, No. 8, pp. 1788–1793, August 2004.

[5] R. G. Gallager, "Low-density Parity-Check Codes", IRE Transactions on Information Theory, Vol. IT-8, pp. 21–28, January 1962

[6] M. Hirotomo, M. Mohri, and M. Morii, "A Probabilistic Computation Method for the Weight Distribution of Low-Density Parity-Check Codes", Proceedings of IEEE International Symposium on Information Theory and Its Application, Adelaide, Australia, pp. 2166–2170, September 2005.

[7] X. Y. Hu and M. P. C. Fossorier, "On the Computation of the Minimum Distance of Low-Density Parity-Check Codes", Proceedings of the IEEE International Conference on Communications, Paris, Vol. 2, pp. 767–771, June 2004.

[8] D. J. C. MacKay and R.M. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes", Electronics Letters, Vol. 32, No. 18, pp. 1645-1655, August 1996. Reprinted in Electronics Letters, Vol. 33, No. 6, pp. 457-458, March 1997.

[9] D. J. C. MacKay and M. Davey, "Evaluation of Gallager Codes for Short Block Length and High Rate Applications", Proceedings of IMA workshop

on Codes, Systems and Graphical Models 1999, August 1999. Edited by B. Marcus and J. Rosenthal, volume 123 of IMA Volumes in Mathematics and its Applications, pp. 113–130. Springer.

[10] D. J. C. MacKay, "Information Theory, Inference and Learning Algorithms", Cambridge Univ. Press, 2003.

[11] T. Mittelholzer, "Efficient Encoding and Minimum Distance Bounds of Reed-Solomon-type Array Codes", Proceedings of 2002 IEEE International Symposium on Information Theory, p. 282, Lausanne, Switzerland, June–July 2002.

[12] C. E. Shannon, "A Mathematical Theory of Communication", Bell System Technical Journal, Vol. 27, No. 3, pp. 379-423, July 1948.

[13] K. Sugiyama and Y. Kaji, "A Minimum Weight Test for a Certain Subclass of Array LDPC Codes", Proceedings of International Symposium on Information Theory and Its Applications, Seoul, Korea, pp. 366–371, November 2006.

[14] K. Sugiyama and Y. Kaji, "A Minimum Weight Test Procedure for a Certain Subclass of Array LDPC Codes", NAIST technical report, No. NAIST-IS-TR2008001, January 2008, http://isw3.naist.jp/IS/TechReport.

[15] R. M. Tanner, "A Recursive Approach to Low Complexity Codes", IEEE Transactions on Information Theory, Vol. IT-27, pp. 533–547, September 1981

[16] R. M. Tanner, "Minimum-distance Bounds by Graph Analysis", IEEE Transactions on Information Theory, Vol. 47, No. 2, pp. 808–821, December 2003.

[17] A. Vardy, "The Intractability of Computing the Minimum Distance of a Code", IEEE Transactions on Information Theory, Vol. 47, No. 2, pp. 1757–1766, November 1997.

[18] K. Yang and T. Helleseth, "On the Minimum Distance of Array Codes as LDPC Codes", IEEE Transactions on Information Theory, Vol. 49, No. 12, pp. 3268–3271, December 2003.

Appendix

A. Instances of Zero Locaters

All the instances of the zero locaters for $j = 4, 5, 6$ are shown in this appendix. In the following instances, the special symbols $\boldsymbol{o}$ and $\square$ are represented by numerals 0 and -1, respectively.

$j = 4$, $w = 8$

```
 0  1  2 -1  3 -1  4 -1
 0  1  2  3  4 -1 -1 -1
 0  1  2  4  3 -1 -1 -1
 0  1  3  4  2 -1 -1 -1
```

$j = 5$, $w = 10$

```
 0  1  2 -1  3 -1  4 -1  5 -1
 0  1  2  3  4 -1  5 -1 -1 -1
 0  1  2  3  4  5 -1 -1 -1 -1
 0  1  2  4  3 -1  5 -1 -1 -1
 0  1  2  4  3  5 -1 -1 -1 -1
 0  1  2  5  3 -1  4 -1 -1 -1
 0  1  2  5  3  4 -1 -1 -1 -1
 0  1  3  4  2 -1  5 -1 -1 -1
 0  1  3  4  2  5 -1 -1 -1 -1
 0  1  3  5  2 -1  4 -1 -1 -1
 0  1  3  5  2  4 -1 -1 -1 -1
 0  1  4  5  2 -1  3 -1 -1 -1
```

```
0  1  4  5  2  3 -1 -1 -1 -1
```

$j = 6$, $w = 12$

```
0  1  2 -1  3 -1  4 -1  5 -1  6 -1
0  1  2  3  4 -1  5 -1  6 -1 -1 -1
0  1  2  3  4  5  6 -1 -1 -1 -1 -1
0  1  2  3  4  6  5 -1 -1 -1 -1 -1
0  1  2  3  5  6  4 -1 -1 -1 -1 -1
0  1  2  4  3 -1  5 -1  6 -1 -1 -1
0  1  2  4  3  5  6 -1 -1 -1 -1 -1
0  1  2  4  3  6  5 -1 -1 -1 -1 -1
0  1  2  4  5  6  3 -1 -1 -1 -1 -1
0  1  2  5  3 -1  4 -1  6 -1 -1 -1
0  1  2  5  3  4  6 -1 -1 -1 -1 -1
0  1  2  5  3  6  4 -1 -1 -1 -1 -1
0  1  2  5  4  6  3 -1 -1 -1 -1 -1
0  1  2  6  3 -1  4 -1  5 -1 -1 -1
0  1  2  6  3  4  5 -1 -1 -1 -1 -1
0  1  2  6  3  5  4 -1 -1 -1 -1 -1
0  1  2  6  4  5  3 -1 -1 -1 -1 -1
0  1  3  4  2 -1  5 -1  6 -1 -1 -1
0  1  3  4  2  5  6 -1 -1 -1 -1 -1
0  1  3  4  2  6  5 -1 -1 -1 -1 -1
0  1  3  4  5  6  2 -1 -1 -1 -1 -1
0  1  3  5  2 -1  4 -1  6 -1 -1 -1
0  1  3  5  2  4  6 -1 -1 -1 -1 -1
0  1  3  5  2  6  4 -1 -1 -1 -1 -1
0  1  3  5  4  6  2 -1 -1 -1 -1 -1
0  1  3  6  2 -1  4 -1  5 -1 -1 -1
0  1  3  6  2  4  5 -1 -1 -1 -1 -1
0  1  3  6  2  5  4 -1 -1 -1 -1 -1
0  1  3  6  4  5  2 -1 -1 -1 -1 -1
0  1  4  5  2 -1  3 -1  6 -1 -1 -1
0  1  4  5  2  3  6 -1 -1 -1 -1 -1
```

```
0  1  4  5  2  6  3 -1 -1 -1 -1 -1
0  1  4  5  3  6  2 -1 -1 -1 -1 -1
0  1  4  6  2 -1  3 -1  5 -1 -1 -1
0  1  4  6  2  3  5 -1 -1 -1 -1 -1
0  1  4  6  2  5  3 -1 -1 -1 -1 -1
0  1  4  6  3  5  2 -1 -1 -1 -1 -1
0  1  5  6  2 -1  3 -1  4 -1 -1 -1
0  1  5  6  2  3  4 -1 -1 -1 -1 -1
0  1  5  6  2  4  3 -1 -1 -1 -1 -1
0  1  5  6  3  4  2 -1 -1 -1 -1 -1
```